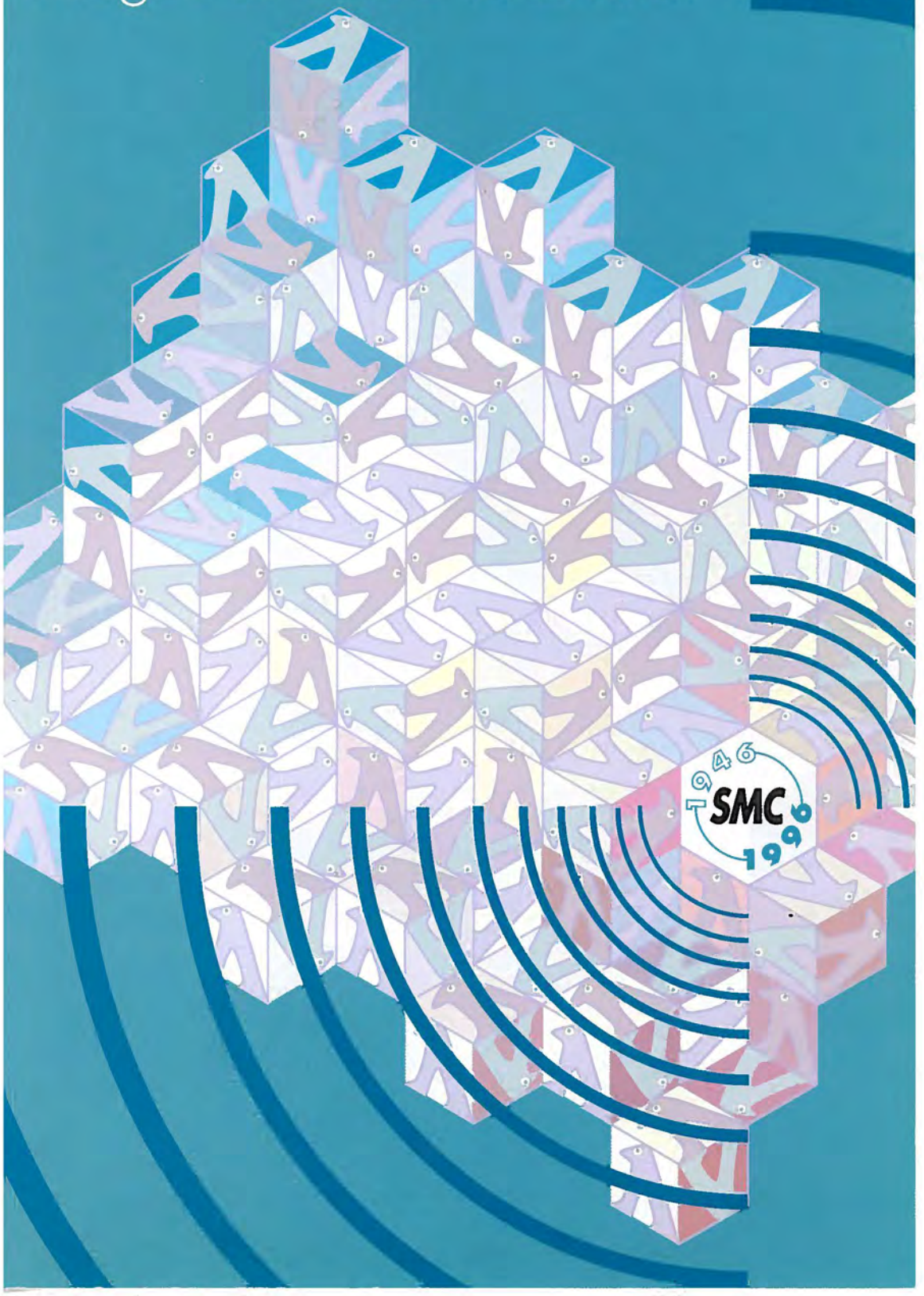
Images of **SMC** Research 1996



Images of SMC Research 1996

ISBN 90 6196 462 8

Publisher:
Stichting Mathematisch Centrum
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Printed in The Netherlands by
Drukkerij Mart.Spruijt bv
Dynamostraat 7
1014 BN Amsterdam
The Netherlands

Contents

Foreword

Dear Reader,

The book in your hands is published on the occasion of the Fiftieth Anniversary of the Netherlands Mathematics Foundation SMC (Stichting Mathematisch Centrum), the eldest of the research foundations constituting the Netherlands Organisation for Scientific Research NWO (Nederlandse Organisatie voor Wetenschappelijk Onderzoek).

The intention of this book is to project an image of SMC, for a much broader readership than usual. In fact, its authors have been promised a reader of a general scientific erudition, but without specialized knowledge of either mathematics or computer science. As a level of expertise this is not very well defined and its interpretation can be seen to vary considerably throughout the book, but in all cases the authors presuppose some degree of curiosity more than anything else. The certainty that nobody ever reads a work like this from cover to cover gave an excuse to indulge in a substantial size, which moreover offers the reader the benefit of an ample choice.

The authors are all associates of SMC of long standing, in the sense that either their work has been supported on a project basis as part of the National Mathematics Programme (Landelijke Activiteiten Wiskunde) or that they belong to the research staff of SMC's institute, the Centre for Mathematics and Computer Science CWI (Centrum voor Wiskunde en Informatica). The first part of the text consists of four essays in a general vein, all of which were contributed from outside the institute. But most of the authors of the following shorter articles, comprising the second part

of the book, have also taken SMC's anniversary, so near the turn of the century, as an opportunity for a reflection on their subdiscipline in this temporal perspective. Consequently, the reader in search of contributions of a more historical slant may be somewhat disappointed - however, what there is describes a project that in SMC's history has served as a model of its kind.

Unavoidably, such a collection not only informs about but also gives an account of work done. The book presents samples over the entire range of SMC's activities and doing so invites assessment. After all you, the intended reader, will eventually be asked to foot the bill as a taxpayer. SMC would like to convince you that at least its share of your money is well spent. However, any judgment should take into consideration the goals that were aimed at and it is here that SMC would like to add a note in the margin.

As said, SMC has supported mathematical research at the Netherlands universities for many years. The budget for this research was always relatively modest and the outcome, while never judged less than of high quality by its reviewers, has never provoked international sensation. What, then, is the significance of such an activity and, indeed, its priority in competition with other work in the NWO sphere? In a small country such as The Netherlands, mathematics is bound to have a strong international orientation - in fact, all our best mathematicians cherish affiliations abroad. This being so, it is of prime importance that there exists at least one platform before the national scientific forum where priorities within the discipline can be decided and where important new developments are identified. It is this signal function of SMC which this volume attempts to emphasize.

Another remark concerns the institute CWI, the name of which couples mathematics with another discipline - an alliance deemed by some to be in a range varying from awkward to unholy. SMC is still proud of the fact that the very first steps of computer science in this country were set under its aegis and to the present day it sets with conviction as a theme for its institute: *the synergy of mathematics and computer science*. In fact, the history of mathematics has always shown a strong interaction of this field with the major issues in society. This was the case in the first industrial revolution when mechanics and electrodynamics were among the main items on the mathematical agenda, and again in the second revolution when mathematical methods for optimization and control of deterministic and stochastic systems, both technological and managerial, were developed. Now we are living through the first phase of a third revolution triggered by the developments in information technology, already touching all spheres of human life. The main theme in this phase appears to be the control of the dynamics and complexity of information based processes, of a dimension previously unheard of. If history is any guide, the key to all this will again be in the development of new mathematical concepts. This promises work for many

years to come for applied mathematicians, and much food for thought for their colleagues of the less mundane variety. CWI is opting for an important role in this field and takes the opportunity to show its colours in the third section of this book.

In the mean time, by all this you have been distracted too long from what really was the first aim in this production: to provide good reading!

G.Y. Nieuwland
Chairman of SMC Board of Trustees

Editorial Foreword

This book was composed on the occasion of the 50th anniversary of the Stichting Mathematisch Centrum (SMC). It provides an impression of the research carried out under the umbrella of SMC. All group leaders of CWI - the research institute of SMC - and project leaders from each of the National Working Parties managed by SMC were asked to contribute by writing an essay on their research. In addition, two mathematicians and two computer scientists were asked to give their views on the future of their disciplines. This has resulted in 38 contributions, 22 of which are by CWI researchers.

The various activities were coordinated by an editorial committee consisting of W.A.M. Aspers, J.W. de Bakker, P.J.W. ten Hagen, M. Hazewinkel, P.J. van der Houwen, and H.M. Nieland. The bulk of the editorial work was carried out by H.M. Nieland and W.A.M. Aspers. Wim Aspers was desk editor. His careful reading led to several improvements. Henk Nieland took responsibility for the presentation of the contributions in such a way that also non-specialists could get at least a taste of the subject, without sacrificing scientific depth.

The book was prepared for printing by SMC's Publication department. Tobias Baanders designed the cover and the layout, and managed the illustrations; Josi Foe took care of the typesetting, and Jan Schipper performed various background actions. Frank van de Wiel provided the necessary support in electronic file handling.

Finally, we want to express gratitude to all authors for their articles.

Computers: (Ac)counting for Mathematical Proofs

A.M. Cohen

1. Introduction

Some forty-five years ago, it became apparent that one could count on computers for computing. Many engineers, physicists and others, but few mathematicians, seemed to be aware of it.

For instance, numerical analysis arose from the need outside of mathematics to perform elaborate computations. The finite element method has been conceived by engineers (cf. e.g. [10]). Many numerical results dating from before 1950 are due to physicists. Not until forty years ago, numerical analysis came alive as a part of mathematics.

History repeated itself about thirty years ago, when it became clear to physicists and computer scientists that exact (symbolic) computations could be carried out successfully on a computer. Again, it took a few years before software and computer power had been developed to the point where even mathematicians could be impressed to the extent that they wanted to use it. In 1985 that point was reached, and subsequently computer algebra was being incorporated into mathematics. Over the last decade, computer algebra has grown into a part of mathematics.

A pattern emerges, according to which computers are being used in disciplines outside of mathematics, in a way that mathematicians will learn to appreciate only several years later, when others have shown its success.

A third to follow in line with this pattern might be automatic proof verification. Except for a single, although quite significant, exception like N.G.

de Bruijn, formal proof checking was not taken seriously by mathematicians over the last twenty years. In the meantime, logicians and computer scientists have developed software tools for verifying formal proofs quite thoroughly. So much so that the use is becoming of some interest to mathematicians.

Extrapolating from the experiences with numerical analysis and computer algebra, it seems only natural to predict that, if the imminent success of these automatic proof verifiers persists, mathematicians will also count this field as a part of mathematics.

Each of these three disciplines, numerical analysis, computer algebra and proof verification, comes with field specific software. In this article I want to highlight just one aspect of this software, namely the question how the classical notion of proof should be interpreted now that, due to computers, new methods of computation and verifying proofs become available.

1.1. Numerical mathematics

The last few years we have seen an increase of interactions between computer algebra systems and numerical software. Usually the numerical software is used as a library in a computer algebra environment. In the context of proofs it is of considerable interest to explore how numerical mathematics can be used to compose both efficient and exact computations.

For example, finding the sign of a certain real algebraic number can be done by purely algebraic methods which are far more inferior in time performance than numerical methods. The optimal strategy seems to use exact arithmetic to determine the precision necessary for computations with an approximation of the algebraic number to result in the same sign as the exact computation. Numerical software with the required precision will then finish the job much more efficiently than exact arithmetic can.

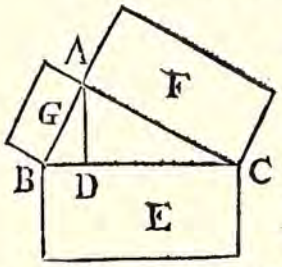
Because numerical software is not made for exact computations, but also because of my ignorance regarding the topic, I will leave out numerical mathematics from most of the discussion and concentrate on proofs in computer algebra and verification.

2. Two kinds of proofs

As opposed to the early times of computers, when their use demanded a thorough knowledge of commands, idiom, a plethora of patience and perseverance, computers are now pleasant, playful, instructive, and sometimes even efficient tools for mathematicians. The computer lends itself to all kinds of mathematical experiments, computations and visualizations, which can lead to interesting conjectures and experimental circumstantial evidence. But, next to striking claims, our strong mathematical tradition also demands proofs, and my interest is in finding out what help the computer can offer in this direction.

160 EVCLIDIS ELEMENT.

PROP. XXXI. THEOR.



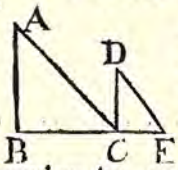
In rectangulis triangulis BAC *figura* E, *quae fit a latere* BC, *rectum angulum* A *subtendente, aequalis est eis* F + G, *quae a lateribus rectum angulum comprehendentibus fiunt, similibus & similiter descriptis.*

α. cor. 8. 6. Duc perpendicularem AD: & erit α ÷ BC,
β. 2. cor. 20. 6. BA, BD, item ÷ BC, CA, DC. Hinc β BC: BD = E: G, & BC: DC = E: F, vel inuersè DC: BC = F: E, & BD: BC = G: E. Er-
γ. 24. 5. go γ BD + DC: BC = F + G: E. Sed BD
δ. sch. 14. 5. + DC = BC: ergo δ F + G = E. Q.E.D.

Aliter.

ε. 1. cor. 20. 6. F: E = ε (CA: BC)² = ε CAq: BCq, & G: E = (AB: BC)² = ABq: BCq. Ergo γ F + G: E = CAq + ABq: BCq. Sed CAq
ζ. 47. 1. + ABq = ζ BCq. Ergo F + G = δ E.

PROP. XXXI. THEOR.



Si duo triangula ABC, DCE, *quae duo latera duobus lateribus proportionalia habent* (BA: AC = CD: DE), *componantur secundum vnum angulum ita, vt homologa latera ipsorum* BA & CD, item AC & DE, *sint parallela: reliqua trian-*

17

Figure 1. Fragment of Euclid's Elements (around 300 B.C.), the classical book based on deductive proof with the axiomatic method (J.F. Gleditsch, Leipzig, 1743. Courtesy W.J.L. Nieland).

The classical notion of proof (see also figure 1) is that of a chain of steps, each of which is understandable to our colleagues, forming a deduction of a given assertion from axioms according to a certain logic. A necessary condition for the acceptance of an assertion as a theorem is the validation of a proof of it by a number of colleagues. My starting point is not so much to overthrow this classical notion as to examine how it should be interpreted in the context of new appearances of proofs due to the surge of powerful computers.

The most obvious appearance lies in arithmetic: a proof based on a certain amount of arithmetic, so enormous that it cannot be performed by hand, will be called a *computational proof* if it can be worked out by computer. But should any series of computations be accepted as proof? To this question I will devote considerable attention.

The second appearance is in the form of *formal proofs*, which are so elaborate and precise that they can actually be read and verified by a computer program (often called the *proof checker*, the procedure is called *automated verification*). Here the question I want to address is: of what use could such a system be for mathematics or mathematicians?

3. Automated verification

I will start with formal proofs. Each step in such a proof follows from precisely indicated axioms by use of precisely indicated deduction rules. It is characteristic of the elaborate texts representing such a proof that they can be verified automatically, with existing software like LEGO.

3.1. Example

By way of illustration, I present a formal proof for the claim that 13 is a prime number.

definitions:
$\mathrm{divides}(m : \mathbf{N}, n : \mathbf{N}) = \exists\, p : \mathbf{N}.\ p * m = n.$
$\mathrm{prime}(p : \mathbf{N}) = \forall\, q : \mathbf{N}.\ \mathrm{divides}(q, p) \to (q = 1) \vee (q = p).$
axiom1: $\forall\, i, j, k, l : \mathbf{N}.\ ((i = j * k + l) \wedge (0 < l < k)) \to \neg(\mathrm{divides}(k, i)).$
axiom2: $\forall\, i : \mathbf{N}.\ (\forall\, j : \mathbf{N}.\ 1 < j \leq \mathrm{Sqrt}(i)$
$\to \neg(\mathrm{divides}(j, i))) \to \mathrm{prime}(i).$
axiom3: $\forall\, i : \mathbf{N}.\ 1 < i \leq \mathrm{Sqrt}(13) \to (i = 2) \vee (i = 3).$
axiom4: $\forall\, i : \mathbf{N}.\ (i = 2) \to (13 = 6 * i + 1) \wedge (0 < 1 < i).$
lemma5: $\forall\, i : \mathbf{N}.\ (i = 2) \to \neg(\mathrm{divides}(i, 13)).$ (axiom1, axiom4)
axiom6: $\forall\, i : \mathbf{N}.\ (i = 3) \to (13 = 4 * i + 1) \wedge (0 < 1 < i).$
lemma7: $\forall\, i : \mathbf{N}.\ (i = 3) \to \neg(\mathrm{divides}(i, 13)).$ (axiom1, axiom6)
lemma8: $\forall\, i : \mathbf{N}.\ 1 \leq i \leq \mathrm{Sqrt}(13) \to \neg(\mathrm{divides}(i, 13)).$
(axiom3, lemma5, lemma7)
conclusion: prime(13). (axiom2, lemma8)

In order to bound the size of this example, I have allowed for 1 to be a prime, and have made the following assumptions, which appear as axioms in the proof:

- If $i = j * k + l$ and $0 < l < k$, then k does not divide i (axiom1),
- It suffices for the proof of our claim to check that no natural number less than or equal to (the floor of) $\sqrt{13}$ and bigger than 1 divides 13 (axiom2),
- the only natural numbers less than or equal to (the floor of) $\sqrt{13}$ and bigger than 1 are 2 and 3 (axiom3),
- $13 = 6 * 2 + 1$ (axiom4), and
- $13 = 4 * 3 + 1$ (axiom6).

In order to increase human readability, I have not given the literal input into the proof checker LEGO, but a palatable version that still gives the flavour. After reading the text, LEGO will return a 'check mark', saying that the proof is accepted. If some step in the proof has not been derived using the specified deduction rules, the program stops and reports where it got stuck.

This example might give the impression that formal proofs consist of unwanted compilations of trivia. But, regardless how big the bulk of formalities may seem in the above example, in general it is expected to be a linear function of the length of the classical proof. Thus, when the first classical proofs can be dealt with in formal guise, much bigger ones will follow suit.

Also, there are clear indications that formal proof checkers, just like people, can work with meta-theorems, which will cater for considerable size reductions of the formal proof. (Compare with the use of macros in source texts of programs like TEX.)

In short, due to such favourable developments proof checkers may turn into serious candidates for everyday mathematical use.

3.2. The use of proof checkers

But, you may wonder, what then is the use mathematicians can make of these automated verifiers? I will list four ways in which I envisage a role for the proof checker.

1. There are indications of the practical use that automated verification may have. In computer science for instance, simple communication protocols which are being used have been proved correct (cf. [4, 8, 9]). Especially for hardware like processor chips, which are produced in large quantities, it can be economically justified to devote plenty of time, energy and money to

increase the reliability of the procedures that are being baked into the chip. Automated verification may be laborious, calling the whole production back to the factory is likely to cost excessively more. If the well-publicized mistake with the Pentium chip was indeed due to an error in the mathematical design for division, an automated verifier might have detected it before production. On the other hand, if, as other rumours have it, the mistake was due to some pieces of data falling off the blueprint when being copied, no automated verifier would have helped.

2. The mere fact that it is possible to verify a proof automatically, brings about a challenge to actually supply such a proof. If proof verification were to enable mathematicians, in exchange for somewhat greater precision and elaboration, to formalize their work in a new generation of 'Bourbakism', then this might well become the standard. Let us not forget that today's proofs are much more rigorous than those of previous eras.

3. The formal proof verifier can be of service when putting together or restructuring a complicated proof. By using the proof checker interactively, and declaring axioms all intermediate steps that haven't been proved yet and are deemed necessary (in much the same way we did in the proof that 13 is prime), we can dynamically develop a strategy for finding the proof of the full claim.

4. This interactive use can also be of significance to education. In a computer learning environment, the proof checker might offer the necessary structure for helping the student to realize a proof and for verifying intermediate results.

Presently, good human interfaces are lacking. Before the mathematician will successfully employ formal proof checkers, the link with daily mathematical use should be much more direct. One of the necessary ingredients for achieving this is 'the mathematical vernacular', as suggested by De Bruijn: a language so formal that it can be used as input for a proof checker, yet so close to everyday language that it produces human readable texts.

4. The computational proof

The characteristic property of a *computational proof* is that a lot of arithmetic is involved of the kind that is easy to automate. On the one hand, invoking such a proof implicitly acknowledges the lack of a better one: the mathematician's sense of beauty favours a short proof with little arithmetic involved over a computational proof.

On the other hand, it does not imply that the proof is necessarily a dull chain of calculations. For instance, there are much more subtle computational proofs of the claim 'n is prime' for a specified natural number n than the most obvious one, which is based on the equivalence of primality of n with the truth of the assertion:

for each natural number k between 1 and $\sqrt{n}$ we have $\gcd(n, k) = 1$.

We then let the computer determine all possible values for k and verify, for each of those values of k, that $\gcd(k, n) = 1$. If the prime number n is a 31 digit number, then this verification entails more than 10^{15} gcd computations, amounting to more than 10^8 years of computation. But present day software provides a verification of this fact in a few seconds.It is an art to develop such fast computational proofs; experience has shown that, as a byproduct of this activity, surprising mathematical theorems may emerge.

I have already asserted that a classical proof is accepted only after a number of colleagues have read it and convinced themselves of the validity of each step. The multiplication $2 * 3 = 6$ is an acceptable step, but a proof consisting of 10^{31} such steps is not. For well-known computer results such as the Four Color Theorem (cf. [1, 2]) and the non-existence of the projective plane of order 10, this is the core of the problem: the outline of the proof was known long before the computational proof was finished. The tremendous amount of dull repetitive work, far too much for an ordinary work station, gives a kind of proof that is rather reluctantly received.

Figure 2. N.G. de Bruijn originated in 1968 the idea of machine verified proofs, using the Automath languages. Courtesy Birkhaüser inc.

4.1. Projective planes of order 10

A projective plane of order 10 is a configuration consisting of a set of 111 *points* and just as many subsets of size 11 of the point set, called *lines*, such that each pair of distinct points is on exactly one line, and each pair of distinct lines meets in precisely one point.

The theorem by C.W.H. Lam, L.H. Thiel and S. Swiercz (cf. [12, 13]) says that such a projective plane does not exist. At the time they finished the non-existence proof on computer, coding theoretic arguments had led to the observation that if a projective plane of order 10 existed, it would have at least one of 66 different well-specified subconfigurations on 19 points. Very crudely, the proof of Lam c.s. consists of an exhaustive search for possible extensions of each of these 66 subconfigurations to a projective plane of order 10.

So here each error in the proof could have blocked a conceivable road to finding a projective plane. In their announcement [13], the authors are extremely careful in formulating their result. I quote:

> This note reports the result of a computer search for 19-point configurations, which, when taken together with previous results, implies that a plane of order 10 does not exist.

Part of the proof is carried out on one of the fastest supercomputers available at the time (1989): the CRAY-1A. Months of computer time have been used on that machine. Thus, the search cannot easily be repeated by a colleague (at least not as of 1995). Also, the slim chance of a shocking result does not motivate other researchers to repeat the effort. After all, the odds are high that the projective plane of order 10 does not exist indeed!

4.2. Oracles

The possibility to repeat the computational proof came up as a criterion for acceptance. Similarly to the case of a classical proof, it is of particular importance that colleagues involved in the validation procedure of a computational proof will be able to perform their verifications on their own machines with relative ease. In particular, if the result of a computation can be verified in a way that has little or (even better) nothing to do with the computation itself, and is much quicker to verify than the original computation, this condition is satisfied. A simple but typical instance of an independent verification is the factorization of large numbers like RSA-129.

Theorem (A.K. Lenstra et al.) RSA-129:

$$114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541$$
$$=$$
$$3490529510847650949147849619903898133417764638493387843990820577 *$$
$$327691329932667095499619881908344614131776429679929425397982885333$$

It took Lenstra et al. several months to factor this 129 digit number, whereas the verification of the result is a single multiplication of two large numbers. Borrowing the terminology from circles in which the study of non-deterministic polynomial time algorithms is popular, I will call this the *oracle function* of the algorithm: it delivers, in a way that is irrelevant to the user, a result that we can check ourselves for correctness. In everyday use of computer algebra systems, the oracle function is not always so blatantly prominent.

4.3. Algorithms

Of course, in a computational proof, the possibility of hardware failures also needs to be considered. As we have recently seen in the case of the pentium

chip, even the basic arithmetic of a (new) processor may be erroneous. Also, the chance of a spontaneous error in computer arithmetic, for example caused by a speck of dust, is small but not inconceivable. Lam c.s. mention a detection of such an error on the CRAY. This is another reason why repetition or, even better, repeatability of a computational proof on different processors should be a necessary condition.

But what holds for hardware and basic arithmetic, also holds for software, for the implementation of an algorithm. If we accept the result of a multiplication on computer, then why should we not accept more complicated programs? Here too, different incarnations of the algorithm will enhance the acceptability of a computational proof if they provide the same output.

4.4. The Buchberger algorithm

In order to go into somewhat greater depth regarding the use of software, I will discuss the use of one of the key results of computer algebra: the Buchberger algorithm. This algorithm takes as input a system of polynomial equations in several unknown, and outputs an equivalent system of equations, from which the solutions can be read off almost immediately (in general after use of a factorization algorithm for polynomials in a single variable). The output is often called a *Gröbner basis*. This 'normal form' for polynomial equations has many useful applications. There are implementations of the Buchberger algorithm in the systems Bergman, CoCoa, Felix, Ganith, GB, KAN, Macaulay, Maple, Mathematica, Reduce, Saclib2, Singular,...

The Buchberger algorithm is a beautiful generalization of both the gcd algorithm for polynomials in a single variable and Gauss elimination for linear equations in several unknown. But here I do not want to go into the theory of the Buchberger algorithm; that has been done quite frequently lately; see for instance [6]. I would rather deal with its use in two examples from my own experience. One of the reasons why the algorithm has become such a great success, lies in the fact that many mathematical problems can be formulated as solving polynomial equations.

5. The icosahedral group

For the first example, consider the well-known icosahedron (see also figure 3). The icosahedral group is by definition the group of all symmetries (or, if you prefer, isometries) of the icosahedron. I will show how, using the Buchberger algorithm, we can transform the purely geometric description into algebraic data, namely a matrix form for each of three reflections generating the group. This in turn can be used for an algebraic description of the points of the dodecahedron.

All reflections leaving the icosahedron invariant, are of the same type: they reflect in a hyperplane as indicated in figure 4.

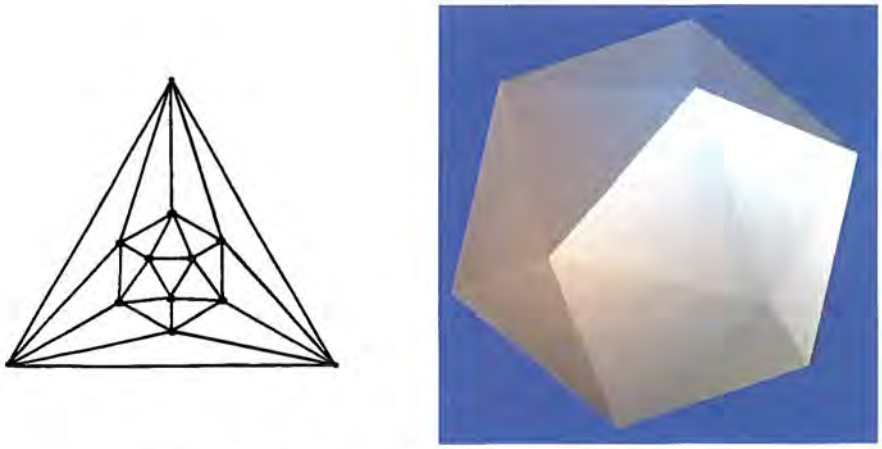


Figure 3. The icosahedron: as a graph (left) and 3D (right).

We can now choose three reflections x, y and z in such a way that the icosahedral group is generated by them. To this end, we take the angles between the various pairs of reflecting hyperplanes to be $60°$ for x and y, $90°$, for x and z, and $36°$ for y and z.

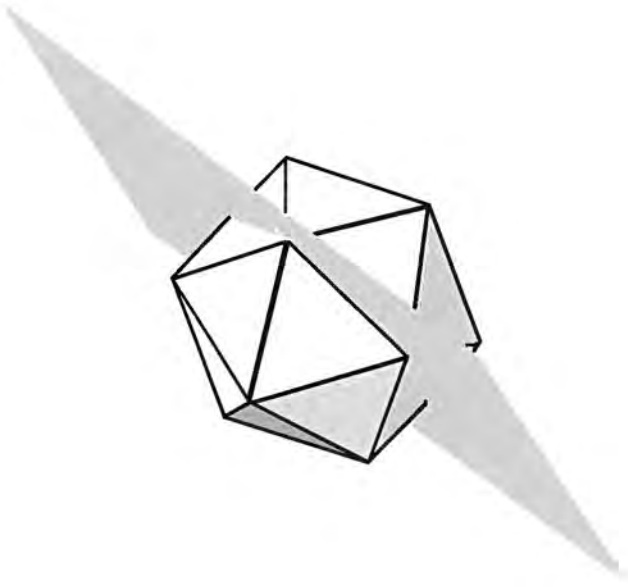


Figure 4. Reflections leaving the icosahedron invariant.

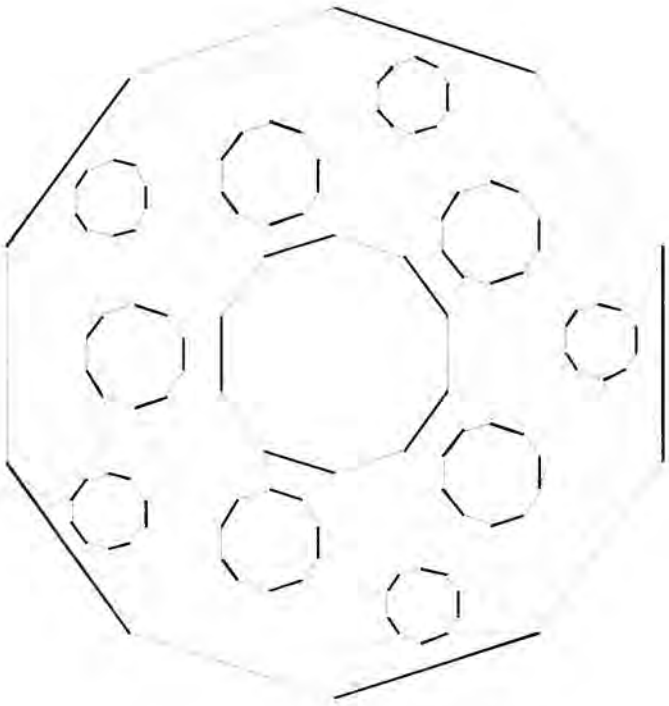


Figure 5. A Cayley graph of the icosahedral group.

We then have the following relations for x, y, and z:

$$x^2 = y^2 = z^2 = 1,$$

$$(xy)^3 = (yz)^5 = (xz)^2 = 1.$$

The first line expresses the fact that the reflections x, y, and z each have order 2. The second line holds because the product of two reflections is a rotation along twice the angle between the corresponding reflecting hyperplanes.

According to Coxeter the relations given are *defining* relations for the icosahedral group. To elaborate on this, we shall from here on view the icosahedral group as the abstract group generated by abstract elements x, y, and z subject to the relations given above. To see that this definition of the icosahedral group coincides with the former, one can invoke a procedure known as 'Todd-Coxeter' enumeration, which results in a so-called Cayley graph. From a free construction of this graph, in which each edge is labeled with one of the three reflections, figure 5 results.

Here, the labeling is as follows: a dotted line segment corresponds with x, an ordinary segment with y, and a fat segment with z.

The number of vertices of this graph is 120, the number of elements of the icosahedral group. These points can be identified (in a meaningful manner) with the elements of the group. By the way, the icosahedron itself can be recovered from this picture by fusing each of the twelve 10-gons to a point.

5.1. Embedding of the icosahedral group

But our goal is to show how, by use of the Buchberger algorithm, we can find orthogonal matrices for the elements of the icosahedral group. We can restrict ourselves to a search for matrices of the reflections x, y, z. Since they generate the whole group, every element can be written as a product of these (and all products are elements of the group).

Because x and z commute, we can choose them, without loss of generality, as follows:

$$x = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Thus, it remains to find the matrix $y = (y_{i,j})_{1 \le i,j \le 3}$. Here are some excerpts of a Maple program in which Buchberger's algorithm is called to solve the equations for the 9 coefficients of y deduced from the defining relations for the generators x, y, and z of the icosahedral group.

1. We load the linear algebra package, and input the 3 matrices x, y and z.

2. As a further preparation, we put the unknown entries of the matrix y in a list

   ```
   > vars := [y11,y12,y21,y13,y31,y22,y23,y32,y33]:
   ```

   and create the identity matrix of size 3:

   ```
   > idm := matrix(3,3,[[1,0,0],[0,1,0],[0,0,1]]):
   ```

3. We have written a routine **mkeq** that, given a matrix, puts its coefficients into a set. Using this routine, we put the equations resulting from $y^2 = 1$ into the set **eqy**.

   The expression **evalm** stands for 'evaluate as a Matrix'. When executed, it will write out the formal object **y ^ 2** as a matrix. The expression **idm** stands for the identity matrix of size 3.

   ```
   > y2 := evalm(evalm(y^2) - idm): eqy := mkeq(y2);
   ```

   $$eqy := \{y31\ y12 + y32\ y22 + y33\ y32,\ y21\ y13 + y22\ y23 + y23\ y33,$$
   $$y31\ y11 + y32\ y21 + y33\ y31,\ y21\ y11 + y22\ y21 + y23\ y31,$$

$$y11\ y13 + y12\ y23 + y13\ y33, y11\ y12 + y12\ y22 + y13\ y32,$$
$$y12\ y21 + y22^2 + y23\ y32 - 1, y13\ y31 + y23\ y32 + y33^2 - 1,$$
$$y11^2 + y12\ y21 + y13\ y31 - 1\}$$

4. Similarly for the equations coming from $(xy)^3 = 1$. Here the relation is rewritten to $xyx = yxy$ so as to keep the degree of the polynomials as low as possible. The set of equations is called **eqxy**.

```
> xyx := evalm(  x &* y &* x): yxy:= evalm( y &* x &* y):
> eqxy := mkeq(evalm(xyx - yxy)):
```

5. We continue with the equations from $(yz)^5 = 1$, calling the result **eqyz**.

6. Because y is a reflection, it has trace 1. Also, the matrix xy of order 3 should have trace 0 (the order cannot be 1 as x and y are distinct). This gives two linear equations for the coefficients of y. They can be obtained as follows:

```
> lineqs := {trace(evalm(y ) ) -1, trace(evalm(x &* y ) )};
```

$$lineqs := \{ y11 + y22 + y33 - 1, -y11 + y22 + y33 \}$$

Theoretically, these linear equations are superfluous. But experience tells us that whenever possible, linear equations should be added to reduce the complexity of the problem as much as possible.

7. The orthogonality conditions are also being translated into equations:

```
> yo := evalm( y &* transpose(y) - idm): eqo := mkeq(yo);
```

$$eqo := \{y21^2 + y22^2 + y23^2 - 1, y31^2 + y32^2 + y33^2 - 1,$$
$$y11^2 + y12^2 + y13^2 - 1, y21\ y11 + y12\ y22 + y13\ y23,$$
$$y31\ y11 + y12\ y32 + y13\ y33, y21\ y31 + y32\ y22 + y23\ y33\}$$

8. We now collect all equations found so far:

```
> eqs := eqy union eqxy union eqyz union lineqs union eqo;
```

$$eqs := \{y11 + y22 + y33 - 1, y21^2 + y22^2 + y23^2 - 1,$$
$$y31^2 + y32^2 + y33^2 - 1, y11^2 + y12^2 + y13^2 - 1,$$
$$\ldots \quad \ldots \quad \ldots$$
$$y33 + y13\ y31 - y23\ y32 - y33^2, y11 + y11^2 - y12$$
$$y21 - y13\ y31,$$
$$-y12 + y11\ y12 - y12\ y22 - y13\ y32, -y11 + y22 + y33\}$$

9. It is time to load the Gröbner basis package, and invoke the Buchberger algorithm. Here, the ordering of the variables in the list `vars` plays a role.

10. The Gröbner basis found by Maple is

$$\left\{ \begin{array}{c} 2y_{11} - 1, \\ y_{12} + 4y_{33}y_{32}y_{31} + 2y_{31}y_{32}, \\ y_{21} + 4y_{33}y_{32}y_{31} + 2y_{31}y_{32}, \\ y_{13} - y_{31}, \\ 2y_{31}^2 + y_{33} - 1, \\ 2y_{22} + 2y_{33} - 1, \\ y_{23} - y_{32}, \\ -1 + 4y_{32}^2, \\ -2y_{33} + 4y_{33}^2 - 1 \end{array} \right\}$$

Notice that the matrix y is symmetric. This we could have known in advance as it has order 2 and is orthogonal.

11. From the upper triangular structure of this Gröbner basis we can read off the general form of a solution. The last equation is quadratic in the single unknown y_{33} and so can easily be solved.

```
> gbo[9];
```

$$-2\,y33 + 4\,y33^2 - 1$$

```
> solve(");
```

$$\frac{1}{4} + \frac{1}{4}\sqrt{5}, \frac{1}{4} - \frac{1}{4}\sqrt{5}$$

```
> y33 := "[1];
```

$$y33 := \frac{1}{4} + \frac{1}{4}\sqrt{5}$$

In general, we need to factor the polynomial. Each irreducible factor then describes the algebraic numbers which the variable can take as values in an algebraic extension field. In our case the quadratic equation for y_{33} gives directly that, up to algebraic conjugates, y_{33} equals $-\cos(4\pi/5) = \frac{1+\sqrt{5}}{4}$, a familiar number in the context of the icosahedron.

12. By successively solving the equations one by one in the opposite order to which they are listed, we obtain the complete solution for y. Let me describe the next step:

```
> gbo[8];
```

$$-1 + 4\,y32^2$$

```
> solve(");
```

$$\frac{1}{2}, \frac{-1}{2}$$

```
> y32 := "[1];
```

$$y32 := \frac{1}{2}$$

13. Continuing this way, we find, up to algebraic and matrix conjugates, the unique solution:

$$y = \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} - \frac{1}{4}\sqrt{5} & -\frac{1}{4} + \frac{1}{4}\sqrt{5} \\ -\frac{1}{4} - \frac{1}{4}\sqrt{5} & \frac{1}{4} - \frac{1}{4}\sqrt{5} & \frac{1}{2} \\ -\frac{1}{4} + \frac{1}{4}\sqrt{5} & \frac{1}{2} & \frac{1}{4} + \frac{1}{4}\sqrt{5} \end{bmatrix}.$$

Thus, we have not only found a solution y, but also know that, up to certain conjugacies, the solution is unique.

5.2. Application

I will show how, as a byproduct, we can find the coordinates of the 12 vertices of an icosahedron: choose a vector h fixed by the reflections y and z (for example $h = [-\frac{1+\sqrt{5}}{2}, 1, 0]$). Then repeated application of x, y, and z to h will produce a set (a so-called orbit of the group) consisting of the 12 vertices.

$$B = \left\{ \left[1, 0, \frac{1}{2} + \frac{1}{2}\sqrt{5}\right], \left[-1, 0, \frac{1}{2} + \frac{1}{2}\sqrt{5}\right], \left[0, -\frac{1}{2} - \frac{1}{2}\sqrt{5}, 1\right], \right.$$
$$\left[0, -\frac{1}{2} - \frac{1}{2}\sqrt{5}, -1\right], \left[0, \frac{1}{2} + \frac{1}{2}\sqrt{5}, 1\right], \left[1, 0, -\frac{1}{2} - \frac{1}{2}\sqrt{5}\right],$$
$$\left[0, \frac{1}{2} + \frac{1}{2}\sqrt{5}, -1\right], \left[-\frac{1}{2} - \frac{1}{2}\sqrt{5}, -1, 0\right], \left[-\frac{1}{2} - \frac{1}{2}\sqrt{5}, 1, 0\right],$$
$$\left.\left[\frac{1}{2} + \frac{1}{2}\sqrt{5}, 1, 0\right], \left[\frac{1}{2} + \frac{1}{2}\sqrt{5}, -1, 0\right], \left[-1, 0, -\frac{1}{2} - \frac{1}{2}\sqrt{5}\right]\right\}$$

To find the edges of the icosahedron one can proceed likewise.

5.3. Conclusion

Summarizing, we have two results. In the first place, we have a concrete presentation of the icosahedron and its group. To prove the correctness of this presentation we only have to check that a system of polynomial equations has a certain solution. Here, the Buchberger algorithm played the role of an oracle for finding it.

In the second place we have found that, in a certain sense, the solution is unique. (To be more precise: up to algebraic and matrix conjugation, there is a single matrix representation of the icosahedral group in which x, y, and z become reflections.) To prove the correctness of this assertion, it may seem that verification of the full arithmetic in Buchberger's algorithm is necessary—similar to the situation of the projective plane of order 10.

But Buchberger's theory gives a tool to see the Gröbner basis as an oracle. To this end, some more a priori arithmetic is necessary (or rather, some more storing of byproducts), in the same vein as the extended Euclidean algorithm needs more (storage) than the usual gcd computation. (When determining the gcd of two polynomials f and g it is not hard to deliver two polynomials a and b such that the gcd is $af + bg$. Now the verification that a given polynomial d is the gcd is nothing but the check that $d = af + bg$ and $d|f$ and $d|g$. This is shorter than the gcd computation itself.) Just like in these two well-known examples, there exists an 'extended Buchberger algorithm', which gives as extra output a way in which the Gröbner basis elements can be written as a linear combination of the input equations. Due to the extra output, the verification that, for a given system of equations, the output is indeed a Gröbner basis, is brought back to an exercise in standard polynomial arithmetic.

It is unfortunate that most commonly used computer algebra systems do not have standard facilities for the execution of the extended Buchberger algorithm. (Macaulay and Singular have a 'lift' command that does the job.) This omission may point to a somewhat all too practical attitude with which the general purpose packages are being used: the results are being accepted in gratitude, but the validity of their outcome is not always questioned to the extent that one would need for a mathematically acceptable proof.

6. A LARGER EXAMPLE

In the example just given, the proof that the embedding exists and is unique (in a certain sense), can be given in many other ways (for instance by use of classical character theory of groups). An important reason for presenting it in the context of a Gröbner basis computation, is that it is representative of cases where no other method is available for achieving a comparable result.

6.1. Kostant's conjecture

One such instance is Kostant's conjecture, which asserts that certain finite groups occur as subgroups of certain Lie groups. In the hardest case, the Lie group involved is the one of type E_8. In order to read on, you need not know more about this Lie group than the fact that it is a 248-dimensional variety of square matrices of size 248, whose group multiplication is the ordinary matrix multiplication. Let us call this group H (abbreviating 'haystack').

According to Kostant's conjecture, this very large group should have a very tiny group as a maximal closed Lie subgroup. This tiny group is the simple group of size 113460, and is known as the fractional linear group over the field of order 61. It is usually denoted by $L(2,61)$, but here, we will denote it by N (abbreviating 'needle').

Thus we are facing the question whether the tiny group N embeds in the large group H, and if so, how.

6.2. The solution

Some ten years ago R.L. Griess Jr. and I brought the problem back to a system of polynomial equations. The method we used, although more delicate, is comparable to the approach we described for the icosahedral group: the needle N is generated by three elements u, t and w satisfying the following defining relations:

$$\begin{aligned} u^{61} = t^{30} = 1, &\qquad tut^{-1} = u^1, \\ w^2 = 1, &\qquad wtw = t^{-1}, \\ (uw)^3 = 1, &\qquad wu^2w = t^{-1}u^{-2}wu^{30}. \end{aligned}$$

The matrices for u and t were easy to determine by use of some Lie theory. The coefficients of the third matrix, w, could be described as rational functions of 9 parameters. The equations that could be directly derived from the defining relations were too large to handle. Therefore, more complicated computations were set up, which made use of projections onto t-eigenspaces.

This led to a system of 57 equations in 9 unknown. Each equation had about 9 monomials. All by itself, it is nothing particular of a problem that it can be put into a system of polynomial equations. Solving the system of equations is another ball game, though. Around 1986, at the time the larger general purpose computer algebra systems came about, I tried to solve these equations in vain (using Macaulay and Maple).

About four years ago, together with B. Lisser, I ventured another try. After having made the preparations for solving the set of polynomial equations, I found a way to dodge the polynomial equations by solving a system of more than 1000 *linear* equations in 248 unknown. This system turned out to be solvable with Gauss elimination in several computer algebra systems.

When I tried to solve the large (heavily overdetermined) system of linear equations, the diagonal that appeared due to Gauss elimination, crept on towards the last column, until it stopped at the one but last, and stayed there for all of the overdetermining equations that were to follow. At that moment I was very sure I had found the needle N in the haystack H, and a unique one at that. Nevertheless, I performed all the necessary verifications to establish that my findings were correct: the *computational proof* was clearly presented (cf. [5]).

But, in a way, I was *counting* too heavily on the convincing power of the computer. Only little later I would have to *account* for what I had done. When it came to publication, my colleagues were quite skeptical regarding the computer computations. I had to pay the price for computations that I could not do by hand: the price that the computations could not be accepted as proof. Or, to put their reactions into a milder perspective: I was asked to specify under which circumstances computer calculations are acceptable as (part of) a proof, in particular, if they can no longer be checked by any person by hand.

6.3. Computer calculations as part of a proof

Just as I have done for the icosahedron, I will address the proofs for existence and for uniqueness separately.

Existence

To verify that the three matrices u, t and w satisfy the defining relations, we only have to perform standard calculations. Once we have chosen a suitable coefficient domain for our computations, the necessary matrix multiplications can be carried out in any one of the special purpose packages GAP, MAGMA, and LiE. Each matrix multiplication will take a few seconds, but that is quite an acceptable time span, even for interactive work.

In this manner, each colleague can verify that u, t, and w generate a
32 subgroup of the group of all invertible square matrices of size 248, which is isomorphic to N. In order to finish the proof of Kostant's conjecture in this case, we still need to verify that each of u, t, and w belongs to the group H. Again, this verification does not require anything beyond standard arithmetic.

Uniqueness

Here we were lucky. Because of linearity of the system of equations eventually used, uniqueness can be either derived from the (straightforward) repetition of the Gaussian elimination or from an LUP decomposition of the original system.

If we would have had to resort to the system of 57 polynomial equations found earlier, a uniqueness proof using the extended Buchberger algorithm

Definition Let K be a field.
1. An *associative algebra* over K is a tuple $(A, +, 0, -, *, \cdot)$, such that $(A, +, 0, -, *)$ is an associative ring, $(A, +, 0, -, \cdot)$ is a vectorspace over K, and $\forall \lambda \in K, x, y \in A : \lambda \cdot (x * y) = (\lambda \cdot x) * y = x * (\lambda \cdot y)$.
2. A *Lie algebra* over K is a tuple $(A, +, 0, -, [], \cdot)$, such that $(A, +, 0, -, \cdot)$ is a vectorspace over K, $[]$ is a bilinear map, $\forall x \in A : [xx] = 0$ and $\forall x, y, z \in A : [x[yz]] + [y[zx]] + [z[xy]] = 0$.

Corollary Let $(A, +, 0, -, *, \cdot)$ be an associative algebra over K.
Define a binary map $[]$ as follows $[xy] =_{df} x * y - y * x$.
Then $(A, +, 0, -, [], \cdot)$ is a Lie algebra over K.
Proof
1. $[]$ is bilinear, because $[(\lambda \cdot x)y] = \lambda \cdot (x * y) - \lambda \cdot (y * x) = \lambda \cdot [xy] = [x(\lambda \cdot y)]$
and $[(x + y)z] = x * z + y * z - z * x - z * y = [xz] + [yz]$
and $[x(y + z)] = x * y + x * z - y * x - z * x = [xy] + [xz]$.
2. $[xx] = x * x - x * x = 0$.
3. $[x[yz]] + [y[zx]] + [z[xy]] = 0$ by computation#.

```
> noncom x,y,z;
                                   x, y, z noncom
> procedure br(a,b);a*b-b*a;
                                   procedure br
> br(x(1),br(y(1),z(1)))+br(y(1),br(z(1),x(1)))+br(z(1),br(x(1),y(1)));
                                   0
```

Computation session in Reduce to prove the Jacobi identity

Figure 6. Elaborate mathematical proofs rely increasingly on the use of Computer Algebra.

would have been desirable. But the usual version of this algorithm was already infeasible at the time. In the meantime, eight years after our first attempt, the system of 57 polynomial equations has been solved twice, first with Macaulay, later with Singular; in both cases the same (unique) solution was found, which of course coincided with the solution of the linear equations.

In summary, although matrices are involved of size larger than we can conveniently deal with by hand, the arithmetic carried out with up-to-date software is so standard, that they can be viewed as acceptable (parts of) proof. The requirement of repeatability is met.

6.4. Conclusion of the large example

In the above discussion I left out some aspects which are worth mentioning.

For instance, I referred to choosing a suitable 'coefficient domain'. Kostant's conjecture concerns Lie groups. Hence, it is formulated for the co-

Simple groups L having a central extension that can be embedded in the complex Lie group of exceptional type X_n	
X_n	L
G_2	Alt_5, Alt_6, $L(2,7)$, $L(2,8)$, $L(2,13)$, $U(3,3)$
F_4	Alt_7, Alt_8, Alt_9, $L(2,25)$, $L(2,27)$, $L(3,3)$, ${}^3D_4(2)$, $U(4,2)$, $O(7,2)$, $O^+(8,2)$
E_6	Alt_{10}, Alt_{11}, $L(2,11)$, $L(2,17)$, $L(2,19)$, $L(3,4)$, $U(4,3)$, ${}^2F_4(2)'$, M_{11}, J_2
E_7	Alt_{12}, Alt_{13}, $L(2,29)^?$, $L(2,37)$, $U(3,8)$, M_{12}
E_8	Alt_{14}, Alt_{15}, Alt_{16}, Alt_{17}, $L(2,16)$, $L(2,31)$, $L(2,41)^?$, $L(2,32)^?$, $L(2,49)^?$, $L(2,61)$, $L(3,5)$, $Sp(4,5)$, $G_2(3)$, $Sz(8)^?$

efficient domain of the complex numbers. The data on the finite group N however make it possible to realize all numbers involved as algebraic numbers. In theory, the exact arithmetic of algebraic numbers on computers is possible, and is in fact one of the major raisons d'être for computer algebra. But computations regarding square matrices of size 248 are not feasible when the coefficients are algebraic numbers of considerable size.

Therefore, we have chosen for reduction modulo a suitable prime number p. Due to some classical mathematical reasoning, it is necessary and sufficient for the embedding of N in H to solve the problem of finding matrices for u, t, and w over coefficients that are integers modulo p.

Besides simplification of the calculations, the technique of reduction modulo a prime number had another good consequence. It got J.-P. Serre interested, who recently produced a computer free proof of the embedding of N in H. He used reduction modulo the prime 61, which requires a much more intricate argument for lifting N back to H, but has the advantage that the subgroup N is known to exist (in the version of H mod 61) from the theory of groups of Lie type. By the way, Serre's proof does not give uniqueness of the embedding of N in H.

To end this example, I would like to mention that the work on Kostant's conjecture is part of a much bigger programme, namely to determine all maximal finite subgroups of the exceptional Lie groups. In this classification, only a few open problems are left. In the table above, question marks indicate which embeddings are still unpublished at the time of writing.

The table is taken from [7]; there however, the group $L(2,41)$ is erroneously left out. In the table L is always a finite simple group and G an exceptional complex simple Lie group (one of G_2, F_4, E_6, E_7, E_8). We provide a twofold interpretation of this table.

1. If L appears on the line of G in the table, then it has a finite central extension that is embeddable in G, with a possible exception for five question marks '?'.

2. If L appears neither on the line of G nor on a line above it, then no finite central extension of L embeds in $G(\mathbf{C})$.

One of the five question marks appears with the group $L(2, 41)$ of fractional linear transformations over the field of 41 elements. Very recently (April 2, 1995) Serre announced a computer-free proof of the embedding of this group in H, to which Griess and A.E.J. Ryba reacted by announcing a computational proof in the making for the embedding of $L(2, 32)$. Thus, serious mathematics sometimes has the likings of a correspondence chess game between G. Kasparov and a group of computer-chess players.

7. Integration

Now that we have gone over some of the features of the new appearances of a proof, I want to add a few words on their interaction. With regard to this topic, I have once heard a logician express the ideal of having all mathematics be verified by proof checkers. If this would imply that the computer algebra systems should account for each arithmetic step in their executions of algorithms, giving a deduction of it which can be input to a proof checker, I am not convinced it is the right goal.

First of all, it does not bring about any pragmatic help for the usual mathematical activities. Secondly, it is not feasible to make any mathematical progress on this basis, simply because proof checkers cannot perform arithmetic with ease and/or speed comparable to computer algebra systems. In the formal proof that 13 is a prime, we had to come up with axioms like $13 = 6 * 2 + 1$ and $13 = 4 * 3 + 1$ in order to keep the number of lines to reasonable length. A proof of an arithmetic equality like any of these two in LEGO comes down to writing out both hand sides as the 13-th successor of 0 within the natural numbers. So this would not be a feasible approach to proving that a certain 31 digit number is prime. (A much more realistic approach would be to prove correctness of the usual number arithmetic, recognizing strings of digits as numbers and next to use meta-theorems; but for simplicity I will overlook this possibility here, especially since eventually the arithmetic is bound to be delegated to software better suited for computations than proof checkers.)

I would rather favour the point of view where proof checkers will accept identities coming from computer algebra systems. Some experiments in this direction have shown that at least this approach is feasible: from the proof checker LEGO, an expression has been sent off for simplification to the computer algebra system REDUCE; the resulting equality between the

input expression and the output expression has been fed into LEGO as an axiom. By combining the results of computer algebra work with proofs in proof checkers in this way, a much more powerful tool for mathematics is being created than any of the two can offer individually.

8. Conclusion

Having stressed repeatability and verifiability of a computational proof, I might have given the impression that the validity of a proof would be quantifiable. To refute this, consider the following thought experiment. Of an explicitly given number n of 31 digits, say, the assertion is being made that it is a prime number. As a proof of this assertion, a long chain of computations is presented. However, an error occurs in one of these computations (so in fact the chain of computations is *not* a proof).

Now suppose five colleagues peruse this erroneous proof independently, leaving a very slim chance ϵ, say $\epsilon = 10^{-15}$, that the error remains undetected.

I have already mentioned that arithmetic on a computer is not 100% reliable. But it is quite likely that, using independent repetitions, we arrive at a likelihood of more than $1-\epsilon$ that the error in the long chain of computations is found, regardless of whether the proof is written up as a formal proof or as a computational proof.

The point I am trying to make is that there are very short probabilistic arguments that, after verification, give a likelihood of at most ϵ that the assertion is wrong. One such a probabilistic proof for the assertion that n is prime, comes from Solovay and Strassen (cf. [3]), and makes use of the following result: For $n \in \mathbf{N}$, $n > 1$, n odd:

$$n \text{ is prime}$$

$$\Leftrightarrow$$

$$\forall\, k \in \mathbf{N},\ 0 < k < n,\ \begin{cases} \gcd(n,k) = 1 & \text{and} \\ k^{\frac{n-1}{2}} \equiv \left(\frac{k}{n}\right) & (\mathrm{mod}\ n). \end{cases}$$

where $\left(\frac{k}{n}\right)$ is the Jacobi symbol.

If n is not prime, then the likelihood that the technical condition above holds for a random k between 1 and n is at most $1/2$. Hence, the likelihood that n is not prime and that the condition holds for 50 random choices of k, is at most 2^{-50}, in particular less than $\epsilon \approx 10^{-15}$. The verification of the technical condition for a single k is extremely fast.

Therefore we can, with the probability of an error which is smaller than the likelihood of a non-detected error by our five colleagues, establish that the assertion is correct by means of a relatively short computation. Still I expect that only few mathematicians will accept this probabilistic argument as a real proof.

By use of this paradox I wanted to illustrate that a reliability estimate all by itself does not *count* for the notion of proof; we shall have to take the human, esthetic standards and values into *account*.

ACKNOWLEDGMENTS

I am very grateful to H.P. Barendregt, N.G. de Bruijn, H. Elbers, F.G.M.T. Cuypers, L.J. van Gastel, H.J.M. Sterk, H.A. van der Vorst for their valuable input.

The topic of the computer influence on mathematical proofs is also dealt with in [3], and various reactions triggered by it (among which [11]).

This text is a translated version of the closing address given by the author at the 31st Nederlands Mathematisch Congres, Groningen, April 21, 1995 and will also be published in *Nieuw Archief voor Wiskunde* (March 1996).

REFERENCES

1. K. APPEL, W. HAKEN (1976). Every planar map is four-colorable. *Bull. A.M.S. 82*, 711-712.
2. K. APPEL, W. HAKEN (1989). Every planar map is four colorable. *Contemporary Math., Vol. 98*, AMS, ISBN 0-8218-5103-9.
3. L. BABAI (1994). Probably true theorems, cry wolf? *Notices AMS 41(5)*, 453-454.
4. M. BEZEM, R. BOL, J.F. GROOTE (1995). *Formalizing Process Algebraic Verifications in the Calculus of Constructions*, Report CS-9502, Dept. Math. & CS. Eindhoven University of Technology.
5. A.M. COHEN, R.L. GRIESS JR., B. LISSER (1993). The group $L(2,61)$ embeds in the Lie group of type E_8. *Comm. Algebra 21*, 1889-1907.
6. D. COX, J. LITTLE, D. O'SHEA (1992). *Ideals, Varieties. An introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Math., ISBN 3-540-97847-X, Springer-Verlag, Berlin.
7. A.M. COHEN, D.B. WALES (1993). Finite simple subgroups of semisimple complex Lie groups—a survey. To appear in *Proceedings of Groups of Lie type and their Geometries*. W.M. KANTOR (ed.), Como.
8. J.F. GROOTE, J. VAN DE POL (1993). *A Bounded Retransmission Protocol for Large Data Packets*, Preprint 100, Dept. Phil. Utrecht University.
9. L. HELMINK, M.P.A. SELLINK, F. W. VAANDRAGER (1994). *Proof-Checking a Data Link Protocol*, CWI Report CS-R9420, Amsterdam.
10. H. HRENNIKHOFF, D. MCHENRY (1990). see J.T. Oden's article in A history of scientific computing. S.G. NASH (ed.). ACM Press, New

York. 152–166. (ISBN 0-201-50814-1)

11. J. Horgan (1993). Trends in Math., The death of proofs. *Scientific American. 74.*
12. C.W.H. Lam (1991). The search for a finite projective plane of order 10. *Amer. Math. Monthly.*
13. C.W.H. Lam. L.H. Thiel, S. Swiercz (1989). The non-existence of finite projective planes of order 10. *Canad. J. Math. 41*, 1117–1123.

The Quest for Correctness

H.P. Barendregt

Die Genauigkeit, Kraft und Sicherheit dieses [mathematischen] *Denkens, die nirgends im Leben ihresgleichen hat, erfüllte ihn fast mit Schwermut.* (The precision, strength and certainty of this [mathematical] thinking, that is unequaled in life, almost pervaded him with melancholy. R. Musil: The man without qualities, Ch. 28.)

R. Musil [13]: Der Mann ohne Eigenschaften.

1. Summary

Modern society has a strong need for reliable information technology (IT). To warrant correct designs for hardware and software systems, there is a thorough methodology (specification, design based on subspecifications and composition of components, and correctness proofs). Because of the difficulty of making specifications and proofs, the success of this method has been only partial, mainly in the area of hardware design.

Presently a new technology is emerging: *computer mathematics*. It consists of the interactive building of definitions, statements and proofs, such that it can be checked automatically whether the definitions are well-formed and the proofs are correct. Hereby the human user provides the intelligence and the system does part of the craftsmanship. Some forms of computer mathematics are already of use for the design of hardware systems. After the technology has matured, it may become a tool for the development of mathematics comparable to systems of computer algebra, but with a scope

and strength that is essentially beyond. Moreover, it probably will also be useful for the design of reliable software.

2. The problem

Once upon a time, money was by convention a substitute for gold. Rather than exchanging goods with goods or exchanging goods with gold it was more practical to exchange goods with money. The gold that was available in exchange for the otherwise worthless coins or paper notes was stored in banks. The central bank of the most powerful country had its reserve of gold stored in a well-protected place: Fort Knox. Only powerful criminals or James Bond (with the aid of pretty girls) could enter such a place. About 25 years ago the link between money and gold was abandoned; since then money stands for itself. As a consequence Fort Knox lost its importance. Moreover, the flow of cash has been drastically reduced by means of several forms of electronic payment, and therefore possession has become equivalent to the right sequence of bits in some central computer of a bank. Even in a relatively peaceful country as The Netherlands such computers are stored in bunkers surrounded by a defense moat against tanks. But the James Bond of today or tomorrow will not need force or women to enter these places. Computer hacking using external connections makes the new Fort Knox vulnerable. Perhaps external connections can be limited or avoided (but what then is the use of a central computer?). There is, however, a more serious problem: the software of the central computer may be ill-designed (by accident or on purpose). Well-informed sources in the banking world admit that this is indeed a very worrisome danger. (The daily flow of money through computer networks is about US$ 10^{12}. Whereas in an average conventional bank robbery a couple of US$ 1,000 is stolen, in an average computer bank robbery one does catch US$ 1,000,000. The frequency of these successful crimes, made possible by system design flaws, is classified information.

More examples can be given of the importance of correct systems. Simple products such as electric razors carry their own microprocessors and software. The manufacturers do not want to have bugs in these systems for obvious reasons. A more striking example is the following. Well-informed sources of a large airline company have stated that 'just 24 hours of failure of our worldwide reservation system will cause bankruptcy', because of missed orders. Even more important is the correctness of systems on which the safety of people depend: for example for the control of factories or air-traffic, or for medical or military applications.

These examples all show that correct systems—both the hardware and the software—are essential for the survival of a company. It is fair to state, that in this digital era correct systems for information processing are more valuable than gold. The remarkable thing is that in spite of our techno-

Figure 1. Correct software and hardware is of vital importance in many real-life situations, for example in air traffic reservation systems. Photo's Capital Press.

logically advanced hardware, we still are by and large in the stone-age of software. By this is meant that it is very hard to produce correct software or predict the time it takes to produce it.

3. Design methodology

Part of the problem is that the needed systems are very complex. How can one correctly design a system with millions of transistors or consisting of millions of lines of program code?

3.1. The Chinese box

A proposed solution is the following. The method is well-known, but usually not explicitly described. Given the task to construct a system, one should first transform the informal requirements into a more precise (formal) specification. Then one designs the system accordingly. Finally, one proves that the design satisfies the given specification. For this the specification and the design need to be formulated in one language.

This is an ambitious programme; it works only if the following items are available with full precision:

1. an expressive but intuitively understandable specification language;
2. an expressive design language with good tools to combine modules together;
3. a tool to verify proofs of formal statements.

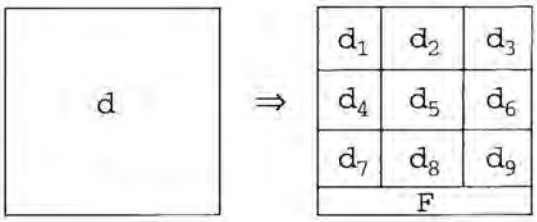


Figure 2. Design methodology.

If these three conditions are satisfied, then one can use the following construction methodology. Given a specification S, one wants to make a design d, such that one can verify

$$S(d),$$

i.e., that the program satisfies the specification. This can be done by providing

- specifications $S_1, \ldots, S_k$;
  a constructor F;
- a proof of the following statement (where the $d_1, \ldots, d_n$ are arbitrary)

$$S_1(d_1) \& \ldots \& S_k(d_k) \Rightarrow S(F(d_1, \ldots, d_k)).$$

So the problem of constructing d satisfying S is transformed into the k (easier) problems of constructing $d_1, \ldots, d_k$ satisfying $S_1, \ldots, S_k$, respectively.

If these $d_1, \ldots, d_k$ are available, then the required d can be found:

$$d = F(d_1, \ldots, d_k).$$

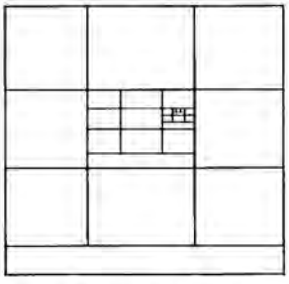


Figure 3. The Chinese box.

see figure 2. In order to find now the $d_1, \ldots, d_k$, the method is applied again to $S_1, \ldots, S_k$; and so on. In this way we obtain a 'Chinese box' containing several boxes, each in turn containing other boxes, etcetera, see figure 3. An example of this is the following. Suppose we want a music reproducing device of high quality. This can be specified by stating that the difference between the actual sound and the reproduced sound is small (according to some appropriate measure). One way to obtain such a device is to buy a CD-player, a CD, an amplifier, boxes and wires, to have a current outlet, and then make the right connections. Each of the components can be specified. Now to obtain e.g. an amplifier, one needs a transformer, integrated circuits, etcetera, and make the right connections. In order to obtain a transformer, one needs a magnet, wire, etcetera. Etcetera.

The example not only shows that designing is a refined job, but also that making precise specifications is important as well. For example the CD should contain stored music according to a fixed coding scheme, otherwise the CD player cannot be constructed.

Of course the method has to be 'well-founded': the smallest boxes should not be empty. Indeed, for the design of software the smallest boxes will have to contain programs provided by the instruction set of the processor. These processors can be constructed following a similar design methodology for hardware out of smaller boxes. In the end *Nature* provides the final step: electrons in circuits that do their work.

3.2. Partial success

The design methodology given above is very general. It can be applied to many situations, in which complicated 'objects' have to be constructed. In the area of IT it has provided a partial success for the construction of verified hardware. For more than a decade hardware design has been done according to the scheme specification/design/proof. The reason for this success is that hardware is relatively simple, being comparable to propositional logic. In this mathematical theory one can state and prove, for example, that 'B and A or not B implies A', in symbols

$$(B\&(A \vee \neg B)) \Rightarrow A. \qquad (*)$$

Here the objects of interest are easy to specify as Boolean functions. Moreover, propositional logic is decidable. That is, proofs of correct statements can be generated automatically.

This description, however, is a simplification. For two reasons the situation with hardware design is more interesting. In the first place propositional logic, in spite of being decidable, is related to problems of high complexity (NP-complete and NP-hard). Methods like Binary Decision Diagrams have been developed to overcome this difficulty, making it possible

to deal with propositions with large numbers of variables (> 100, and not just 2 as in $(*)$). Secondly, hardware is somewhat more complicated than propositional logic. This is caused on the one hand by real-time aspects of circuits, and on the other hand by multiple repetitions of patterns that can better be treated in a more powerful theory. As a result the theory and practice of hardware verification is a flourishing field (see T.F. Melham et al. [12]).

3.3. A challenge

Also in the area of software design there have been efforts to prove correctness. But here matters are essentially more difficult. Software corresponds to predicate logic, with statements like *'there exists an x such that for all y one has $x \leq y$ implies that for all y there exists an x such that $x \leq y$'*; in symbols

$$\exists x \forall y . x \leq y \Rightarrow \forall y \exists x . x \leq y.$$

The difficulty is that the so-called quantifiers 'for all' and 'there exists' range over potentially infinite sets, and therefore it comes as no surprise, that this theory is undecidable.

Hardware $\sim$ propositional logic (decidable)
Software $\sim$ predicate logic (undecidable)

Another problem is to find the right granularity. The first introduction of correctness proofs in program design, by Hoare, was in connection with imperative programs in which continually a given state (the values of the variables) is modified. By formulating a suitable property of the state and proving that it is invariant under the modifications of the program, correctness proofs can be given. At that time this method was definite progress. But the method is of a granularity that is too fine to prove that large software systems are correct.

New programming paradigms such as functional programming, (see e.g. R. Plasmeijer and M. van Eekelen [18] or D. Pountain [19]), possibly including object-oriented features, may become a good design tool to overcome this difficulty by using the methodology of the Chinese box. Software design still lacks a good language for specification and the right tools for correctness proofs. This is one of the reasons, why we still are in the software crisis. In the next section we will discuss systems of so-called *computer mathematics* that may very well change this situation.

4. Computer mathematics

Because computer mathematics (CM) is a relatively new technology, presently in its second generation of prototypes, there is not yet a standard name

for it. Alternative names are: 'systems for proof development', 'systems for theory development' and 'interactive theorem provers'. We have chosen the name 'computer mathematics' because of the analogy with systems for computer algebra (CA).

4.1. What is computer mathematics?
It is well-known that computers perform numerical computations. Since the 1960s systems for computer algebra have been developed, that can represent exactly numbers like $\sqrt{\pi}$ and perform symbolic computations quite well. (However, since several systems of CA state for example that $(\sqrt{x})^2 = x$ without requiring that $x \geq 0$, the diligent judgement of a mathematician remains necessary.)

Systems for computer mathematics go an essential step beyond this. They can deal with arbitrary mathematical notions. For example, it is possible in these systems to represent exactly a Hilbert space or more complex mathematical structures. This is possible because one can formulate statements involving quantifiers ($\forall, \exists$) and predicates. CM systems also provide support for mathematical reasoning, for example by manipulating complex expressions. One fundamental difference between equational reasoning (both numerical and symbolic) and reasoning with predicates is that in many cases the former is decidable, whereas the latter usually is not. For this reason systems of computer mathematics are interactive, whereby the mathematician takes the lead.

Before going into more detail we first want to mention three important contributions of the Greek philosopher Aristotle (384-322 B.C.) to the field of computer mathematics. He established the following.

Description of the axiomatic method
Aristotle distinguished *concepts* and *propositions*. Concepts need to be defined and propositions need to be proved. (A *proof* is a sequence of statements, such that each one is either an axiom or follows from previous ones by reason. A proof p is *proving* A, if A is the last statement in the sequence p. The discovery of proofs, attributed to Thales (625-545 B.C.), is one of the greatest inventions of humanity. They occur in various degrees of precision (depending on the refinedness of the statements and the reasoning) and are the main foundation for the reliability of science.) But in order to have a proper start one needs *primitive concepts* and primitive propositions, the so-called *axioms*. Not much later Euclid (around 300 B.C.) carefully based his famous book *Elements* on this axiomatic method.

Formulation of the quest for logic
Using the axioms, and primitive and defined concepts, one can prove propositions by means of steps motivated by intuitive reasoning. Aristotle wanted

to provide a set of explicit rules (*syllogisms*), that is powerful enough for most intuitively valid proofs. Although his teacher Plato (427-347 B.C.) was against this programme, Aristotle succeeded partially, by singling out a correct (but incomplete) set of syllogisms. But more important was, that Aristotle had the courage to state the problem of formalising reason.

Distinction between proof-checking and theorem proving

Aristotle said, that if someone showed him a proof of a statement, then he would be able to verify the correctness of that proof (and thereby of that statement relative to the axioms). If, however, he would be asked to prove a given theorem (of which a proof existed, but that was not given to him), then he would not always be able to do so.

Figure 4. G.W. Leibniz.

The philosopher and mathematician G.W. Leibniz (1646-1717) went further, by expecting more from formalisms and machines. He wanted to construct

- a language L, in which arbitrary problems could be formulated;
- a machine, that could determine the correctness of statements in L. (It is interesting that around 1700 the belief in machines was such, that Leibniz had in mind to ask as first question: 'Does God exist?')

But, as was hinted at by Aristotle (and proved later by Turing), such automated deduction is in general not possible. An actual system of computer mathematics (like LEGO or Coq) is less powerful than Leibniz would have wanted. It consists of a user interface in which the user can construct a so-called (mathematical) context:

- primitive notions, axioms and defined notions;

furthermore one can formulate

- statements;

and for some of these statements one can construct interactively

- proofs.

The computer will verify whether the definitions are well-formed and whether the proofs are correct. Such definitions and proofs need to be

given in a fully formal way, otherwise they cannot be verified mechanically. A fully formal proof is called a *proof-object*.

It is clear, how much this technology is related to the three ideas of Aristotle. His programme to find a complete system of logic, was completed by G. Frege (in 1879, more than 2200 years after the original quest), building upon work of Leibniz, Boole and Peirce. (Frege did start with the formalisation of some mathematics, but unfortunately used an inconsistent system of mathematical axioms.) Soon after, B. Russell and N. Whitehead gave a fully formalized version of small fragments of consistent mathematics (*Principia Mathematica*, 1910). This work formed the basis of the fundamental results stating that arithmetic is essentially incomplete (K. Gödel, 1931) and undecidable (A. Turing, 1936). In practice, however, Russell and Whitehead's system is not adequate for full formalization, because the system does not contain names, which causes actual theories to become unfeasibly large; moreover, there is a need for substitution instances of theorems, which in *Principia* were indicated informally.

The idea of machine verified proofs originated with the Dutch mathematician N.G. de Bruijn, who in 1968 designed for this purpose a family of languages generically called Automath (see Nederpelt et al. [14]). Inspiration for this came from the meaning of the logical connectives, as put forward by L.E.J. Brouwer and A. Heyting. The ideas are also related to work of Gentzen, Church and Howard (see figure 5, subsection 4.3). R. Boyer pointed out to me that also in McCarthy [11] automated proof-checking was considered. In fact McCarthy's paper is rather close to the present paper. An essential difference is that the use of type theory (see below) and natural deduction proofs is not discussed by him.

As was pointed out by Aristotle proof search is essentially more difficult than proof-checking. By the definition of proof, automated proof verification is always possible, while automated deduction is not. Nevertheless, for special areas of interest there are good systems for automated deduction. For example, the geometry prover of Chou and Wu (see Bundy [4], p. 393), can derive automatically Morley's theorem concerning triangles in which the three angles are trisected. But this is a statement in a decidable theory. Another example is concerned with predicate logic. Although this theory is undecidable, one can derive automatically a class of tautologies of predicate logic, that are more difficult than those used in most mathematical texts.

In spite of the remark of Aristotle that proof-checking is different from theorem proving, systems for CM usually incorporate both. The reason is that in a pure proof-checker it is very boring to write down a formal proof. On the other hand general theorem proving is impossible. Usually one needs to give a sequence of lemmas to the system, before it can prove an interesting result. So there is a spectrum between proof-checkers and theorem provers.

4.2. Why computer mathematics?
Automated proof-checking and the development of formal definitions, statements and proof-objects is important for the following reasons.

Correctness
Both mathematics and system design require correctness. In the two disciplines the problems are somewhat different.

In mathematics the body of knowledge has become very large and complicated. Proofs of complicated theorems are verified by humans in the so-called sociological way: the proof is divided into parts, that are checked by different specialists. A prime example is the result about the structure of the simple finite groups, with a proof of more than 20.000 pages. If one succeeds in generating formal proof-objects for such complicated theorems, then the computer verification will add to the acceptance of these (methodological considerations will be given below). But this is an extreme example. It is also important to have automatic proof verification for less spectacular results. If a mathematician sees a useful but unknown result, with a proof of, say, 20 pages long, then it saves time to know beforehand whether the result is correct or incorrect. Moreover, it is useful to have mathematical theorems in a verified library, because the size of the collection of results is growing so rapidly, that it has become impossible for one person to understand all of it or even just know about all of it. In this way arbitrary details can be looked up and combined.

It should be admitted that formalizing mathematics is difficult. This is not so much because of the length of proofs but because of their depth. Also standard mathematical proofs contain jumps that are not formal, but are clear to an artisan in the field. It is fair to say, that at present mathematical proofs can be verified better by a human than by a computer, because of the mentioned difficulty in formalisation. In subsections 4.3, 4.4 it is indicated how this may change.

In the field of system design (hardware and software) the problem of verification is different from that in usual mathematics. Here most proofs are long but intellectually not stimulating. This implies that machine verification is essential. As mentioned in section 2 this is done successfully for hardware design, but not yet in software design because of the lack of specification and proof tools. I expect, that when the technology of CM will have matured, it will have a strong influence on software design.

Support
Systems of CM may facilitate the construction of large proofs (both in mathematics and in system design). These constructions can be done top-down or bottom-up. One may leave some lemmas unproven or even concepts undefined; the necessary details can be filled in later. Of course even without

a system of CM this can be done. But the help of such a system consists in verifying the well-formedness of definitions and the correctness of proofs. Moreover, the systems keep a record of those details that are still left out. Another support by systems of CM consists in generating formal proofs from so-called *tactics*, to be discussed in the next subsection.

Program extraction
If a statement of the form '$\forall x \exists y \ldots$' is proved, then this often gives rise to an algorithm that finds the y in terms of x. If the proof is given formally, then the algorithm can be extracted automatically in the form of a program, see e.g. Parent [15].

Education
As it is a fact, that in several 'civilised' countries the notion of proof is not taught anymore at high-school level, it becomes necessary that university students of mathematics, science and technology get acquainted with them as soon as possible. Interactive systems for proof development will be of definite help, notably because such systems are patient. Moreover, the proofs can be found only, if one understands what one is doing.

Cultural value
Suppose, that with the support of a CM system writing verified mathematics is not much more difficult than writing an intuitively correct paper in TEX, then a new standard of precision may emerge. By building a library of verified results, mathematics may be protected against corruption in times that the subject is not cultivated anymore (as essentially happened in the Middle Ages). In Bundy [4], pp. 238-251, a dramatic but non-Utopian plea for building such a library is formulated as the *QED manifesto*. One quotation: '[building such a library is] *of significant cultural character. .... Like the great pyramids, the effort required (especially early on) may be great: but the rewards can be even more staggering than this effort*'.

Foundational interest
It is an interesting challenge to see, whether it is possible to build systems of CM, such that it does not require too much effort to construct proof-objects. In this respect De Bruijn has as thesis, that in a proper system of CM the length of a formal proof or required tactics is just a constant factor times the length of a complete intuitive proof. Experience so far is in favour of this thesis. For the first generation prototypes the factor is about 30; for the second generation that uses tactics it is about 10. Also it is of interest to study, in which class of formal systems proofs can be well represented (set theories vs. type theories, other systems).

There may be a methodological objection to the idea of computer verifica-

tion. If a mathematical statement is verified for its correctness by a computer, are we willing to believe that statement? There could be a mistake in the design of the verifying program.

This question has a satisfactory answer. If the verification is warranted by a proof-object that is made public and that is verifiable by a relatively simple method (by a program consisting of a few pages), then one can recheck the statement locally, i.e., on a PC with one's own personal proof-checker. Under these conditions of repeatability, one can trust the correctness of the statement at least as much as (or even more than) the safety of a bridge over which one is going to walk.

4.3. Formal CM systems

Systems of computer mathematics with portable proof-objects (as required by the quest for reliability discussed above) are to be done in a formal system T that should have the following properties.

- T is *adequate*: the usual mathematical concepts, statements and proofs can be expressed (in a natural way) formally in T. Adequacy requires the following particulars.

    1. Adequacy for *defining*. The system T has sufficiently rich concepts and allows the introduction of names.
    2. Adequacy for *reasoning*. The usual logical deductions occurring in mathematics are representable as proof-objects of T.
    3. Adequacy for *computing*. Symbolic (and numerical) computations, as well as equational reasoning, are possible in T.

- T is *faithful*: T is conservative in the sense that if it states that P is a proof of statement S (in context Γ), then the intuitive statement S is provable in ordinary mathematics relative to the corresponding context.

- T is *efficient* (for the machine): the verification of the well-formedness of a definition and the correctness of a proof can be verified in a feasible way.

- T is *practical* (for the human user): writing mathematics in T is not much more difficult than writing it in informal language.

Following the ideas in the languages of the Automath family, now a wide spectrum of *type theory* systems are used as formal system T. See Nederpelt et al. [14], 229-247 for a discussion. The simplest of these are called Pure Type Systems (PTSs), see Barendregt [1]. Under influence of D. Scott and P. Martin-Löf inductive types and extra reduction rules are added to the

formal systems. The resulting extensions are called Type Systems (TSs). See Martin-Löf [10] and Paulin-Mohring [16] for a description of these.

Type theories are formal systems in which there is a natural way to represent statements and proofs. In fact, it seems more natural to encode mathematics in these systems than in the more conventional set theory. The reason is, that type theory has a natural way to use many-sorted logic (in order to deal with structures like vector spaces, in which there are vectors and scalars belonging to different 'sorts'), as well as to formalize second and higher order logic (to reason about properties of propositions, or properties of properties of propositions; in this way one can formulate the notion of 'infinite'). To make a variation on a statement of Laplace, we can say that 'we do not need the hypotheses of set theory'. (Napoleon remarked to Laplace that in his work 'Mécanique Céleste' he did not mention the author of the universe. Laplace answered: 'Sire, I did not need that hypothesis'.) Moreover, set theory sometimes gives rise to unnatural questions, for example $\emptyset \in \sqrt{2}$? In type theory such questions cannot be formulated.

One important aspect of (P)TSs is that some proof steps (of definitional nature) do not need to be given explicitly. Suppose we have proved $P_1 \& P_2 \Rightarrow Q$. Now define $P \equiv P_1 \& P_2$. Then we have, of course, $P \Rightarrow Q$. In a (P)TS the same proof-object for $P_1 \& P_2 \Rightarrow Q$ works also for $P \Rightarrow Q$. This P and $P_1 \& P_2$ are said to be *definitionally equal* and share the same inhabitants (by a rule of PTSs). In TSs more equalities hold in this way. For example there exists a term Sq (for squaring), such that for a natural number like 3 one has $\mathrm{Sq}(3) = 9$ definitionally. So a proof of $A(9)$ is also a proof of $A(\mathrm{Sq}(3))$. This is the essential difference between TSs and PTSs.

Now we will discuss how the list of requirements applies to type theories.

1(a). As was already mentioned, type theories are strong enough to represent most mathematical reasoning. In addition to this adequacy, the extra data types available in TSs make representations easier in these systems. It is known also in programming that extra data types make life much easier. Surprisingly many concepts in mathematics are related to inductively defined data types (for example the notion of polynomial).

1(b). Although proofs of most tautologies (syllogisms) of predicate logic, that are needed in mathematics as deduction steps, can be found automatically by resolution methods, this does not mean that the problem of formalising logical steps is solved. The reason is that one needs names in mathematics. Now even if

$$\mathrm{name}_1 \Rightarrow \mathrm{name}_2$$

is a tautology, this is so, only after the names are replaced by the proper expression they stand for. This so-called 'unfolding' should not be done fully, because then the expressions become unfeasibly large. So the problem boils down to deciding what names have to be unfolded.

Each statement A of informal (but precise) mathematics can be translated as a formal statement (A) in logic. A mathematical context Γ consists of a set of axioms and definitions. When we write these formally, we obtain a formal context (Γ). Predicate logic is such that we can derive (A) from (Γ) exactly when A is informally provable from Γ.

Type theory goes one step further. A statement A is transformed into the type (i.e., collection)

$$[A] = \text{ the set of proofs of } A.$$

So A is provable if and only if $[A]$ is 'inhabited' by a proof p. Now a proof of $A \Rightarrow B$ consists (according to the Brouwer-Heyting interpretation of implication) of a function having as argument a proof of A and as value a proof of B. In symbols

$$[A \Rightarrow B] = [A] \to [B].$$

Similarly

$$[\forall x{\in}A.Px] = \Pi x{:}A.[Px],$$

where $\Pi x{:}A.[Px]$ is the cartesian product of the $[Px]$, because a proof of $\forall x{\in}A.Px$ consists of a function that assigns to each element $x{\in}A$ a proof of Px. Using this interpretation a proof of $\forall y{\in}A.Py \Rightarrow Py$ is $\lambda y{:}A\lambda x{:}Py.x$. Here $\lambda x{:}A.B(x)$ denotes the function that assigns to input $x{\in}A$ the output $B(x)$.

Verifying whether p is a proof of $[A]$, boils down to verifying whether in the given context the type of p is equal to $[A]$.

Figure 5. The essence of proof-checking.

1(c). Equational reasoning is not yet incorporated in a feasible way in
52 TSs. Systems of CA can produce valid equations. Several ways of doing this in CM systems are being studied: the *believing* way, in which the CM system just accepts equations produced by a CA system; the *skeptic*, in which the CA system is forced to give evidence (a proof-object) for each statement it sends to the CM system; and finally the *autarkic* way, in which the CM system learns to do equational reasoning by incorporating some verified term rewriting techniques.

2. Unfortunately faithfulness is only known for several adequate PTSs and not for the corresponding more practical TSs. It is conjectured, however, that the right TSs are faithful.

3. As to efficiency, both for the PTSs and the TSs correctness is decidable by a simple program. But the verification is in general not feasible in these systems. It is, however, an empirical fact, that if formal definitions and

proofs in these type theories come from definitions and proofs understood by a human, then the verification of correctness is feasible.

4. Even the stronger TSs are not yet practical. What is needed, is a kind of higher language (in the sense that FORTRAN and PASCAL are higher programming languages than assembler) that is convenient to express mathematics but that can be translated easily to the more low-level language of (P)TSs. Such a language is called by De Bruijn a *mathematical vernacular*.

4.4. Implementations

The first prototype CM system was the Automath proof-checker built in 1970, see Nederpelt et al. [1994], pp. 783-804. In this system the proof-object had to be constructed by hand. It required a *mathematical monk* to formalise a non-trivial part of mathematics. In the second generation prototypes the proof-objects are generated via so-called tactics. A tactic is a sequence of commands that the user can give to the system; from this a proof-object can be compiled automatically. Suppose, for example, that one has to create a formal proof for

$$\forall x, y{\in}A\ [P(x, y){\Rightarrow}Q(x)] \qquad (+)$$

(the 'goal') from a certain context. Then one 'pushes a button' and the system adds to the context that $x, y{\in}A$ and the assumption $P(x, y)$. Now the goal is to prove $Q(x)$ from the extended context. As soon as this is done the system provides a proof for (+). See figure 6 for an example of tactics. Not shown are the answers of the system after each statement made by the user. These answers consists of new (simpler) goals—as described above—so that the human does not get lost while designing the proof of (+).

These tactics do not constitute a vernacular because they are close to the syntactical structure of the formal proof, rather than to the mathematical idea of the informal proof.

4.5. Existing systems

The principal second generation prototype systems for CM with portable proof-objects are NuPrl (see Constable et al. [5]), Coq (see Dowek et al. [7]), and LEGO (see Luo et al. [9]). These systems have as extra features:

- tactics.
- term rewriting.
- (some form of) automated deduction.

Tactics make it much easier for the human user to construct a proof-object. NuPrl was the first system based on type theory using tactics. Many theorems are proved using it.

```
(*Theorem Drinkers'_principle.*)
Goal ({P:Prop} P \/ ~P) ->
     {Cafe : Type} {w:Cafe} {Drunk : Cafe -> Prop}
     Ex [x:Cafe] (Drunk x) -> {x:Cafe} Drunk x;
  intros EM ___;
  Refine EM ({x:Cafe} Drunk x);
  Intros _; Refine ExIntro; Refine w;
  intros _; Immed;
  intros; Refine EM (Ex [x:Cafe] ~(Drunk x));
  intros; Refine H1; intros; Refine ExIntro; Refine t;
  intros _; Refine H2 H3;
  intros;  Refine H;
  intros; Refine EM (Drunk x);
  intros; Immed;
  intros; Refine H1;
  Refine ExIntro; Refine x; Immed;
Save Drinkers'_principle;
```

Figure 6. Tactics.

Incorporation of term-rewriting makes it easier to deal with symbolic and other forms of computation. In particular the autarkic way of incorporating CA can make use of this facility.

Automated deduction based on resolution solves some of the logical steps to be made. As pointed out before, diligent use of unfolding definitions is necessary. In some versions of Coq this can be done by 'clicking' on the name. LEGO has some 'automated' unfolding, necessary for ease of use. One of the newer features of Coq is the automatic translation of a proof-object into a proof in natural language, see Coscoy et al. [6].

In figure 7 one can see for Smullyans 'Drinkers' principle' an informal proof (in 'my best mathematical style'), the proof-object, and finally a translation of that formal proof into natural language. The formal proof is obtained through an interactive session in which the user provides tactics to the machine, see figure 6.

Although the translated proofs in natural language are somewhat 'stiff', these may turn out to be useful for the construction of a vernacular. The reason is that seeing a formal proof-object does not easily lead to understanding, while a proof in natural language does. In particular this is so, when sufficiently many details that are obvious to a human are elided.

Smullyan's 'Drinkers' principle': in a room with people there always is at least one person, such that if that person starts to drink, then everybody in the room starts to drink.

R.M. Smullyan

Theorem(Drinkers' principle) Let U be a non-empty set and let Q be a predicate on U. Then

$$\exists x{\in}U.[Q(x) \to \forall y{\in}U.Q(y)].$$

Proof. We distinguish two cases.
Case 1. $\forall y.Q(y)$. Then an arbitrary $x{\in}U$ makes the implication true.
Case 2. $\neg\forall y.Q(y)$. Then $\neg Q(x_0)$ for some $x_0{\in}U$. Now take $x = x_0$, to make the implication vacuously true.

Fig. 7a.

```
Theorem Drinkers' principle
Statement
 (∀P: Prop. P ∨ ¬P) ⇒
  ∀U: Set. U ⇒ ∀Q: U ⇒ Prop. ∃x: U. (Q x) ⇒ ∀x0: U. (Q x0)
Proof
 λH1: ∀P: Prop. P ∨ ¬P. λU: Set. λH0: U. λQ: U ⇒ Prop.
 (or_ind
    ∀x0: U. (Q x0) ∨ ¬(∀x0: U. (Q x0)) ∃x: U. (Q x) ⇒ ∀x0: U. (Q x0)
    λH1': ∀x0: U. (Q x0).
    (ex_intro U λx: U. (Q x) ⇒ ∀x0: U. (Q x0) H0 λH2: (Q H0). H1')
    λH1': ¬(∀x0: U. (Q x0)).
    (or_ind
       ∃x: U. ¬(Q x) ¬(∃x: U. ¬(Q x)) ∃x: U. (Q x) ⇒ ∀x0: U. (Q x0)
       λH2: ∃x: U. ¬(Q x).
       (ex_ind
          U λx: U.¬(Q x) ∃x: U. (Q x) ⇒ ∀x0: U. (Q x0)
          λx: U. λE: ¬(Q x).
          (ex_intro
             U λx0: U. (Q x0) ⇒ ∀x1: U. (Q x1) x
             λH3: (Q x). (False_ind ∀x0: U. (Q x0) (E H3))) H2)
       λH2: ¬(∃x: U. ¬(Q x)).
       (ex_intro
          U λx: U. (Q x) ⇒ ∀x0: U. (Q x0) H0
          λH3: (Q H0). λx: U.
          (or_ind
             (Q x) ¬(Q x) (Q x) λH4: (Q x). H4
             λH4: ¬(Q x).
             (False_ind (Q x) (H2 (ex_intro U λx0: U. ¬(Q x0) x H4)))
             (H' (Q x)))) (H' ∃x: U. ¬(Q x))) (H' ∀x0: U. (Q x0))))
```

Fig. 7b.

```
Theorem Drinkers' principle
Statement
 (∀P: Prop. P ∨ ¬P) ⇒
  ∀U: Set. U ⇒ ∀Q: U ⇒ Prop. ∃x: U. (Q x) ⇒ ∀x0: U. (Q x0)
Proof
 Assume ∀P: Prop. P ∨ ¬P (H)
    Let U be a set
    Consider an arbitrary element H0 in U
       Let Q : U ⇒ Prop
       Specializing H to ∀x0: U. (Q x0), we get
       (∀x0: U. (Q x0)) ∨ ¬(∀x0: U. (Q x0))
       So we have two cases:
       a) Assume ∀x0: U. (Q x0) (H1)
             Assume (Q H0) (H2)
                We have H1
             We have proved (Q H0) ⇒ ∀x0: U. (Q x0)
             We have found an element x that verifies (Q x) ⇒ ∀x0: U. (Q x0),
             namely H0
       b) Assume ¬(∀x0: U. (Q x0)) (H1)
             Specializing H' to ∃x: U. ¬(Q x) we get
             (∃x: U. ¬(Q x)) ∨ ¬(∃x: U. ¬(Q x))
             So we have two cases:
             a) Assume ∃x: U. ¬(Q x) (H2)
                   Choose an element x in U such that ¬(Q x) (E)
                   Assume (Q x) (H3)
                      From H3 and E, we deduce a contradiction
                      So this case cannot happen
                   We have proved (Q x) ⇒ ∃x0: U. (Q x0)
                   We have found an element x0 that verifies
                   (Q x0) ⇒ ∀x1: U. (Q x1), namely x
             b) Assume ¬(∃x: U. ¬(Q x)) (H2)
                   Assume (Q H0) (H3)
                      Consider an arbitrary element x in U
                         Specializing H' to (Q x) we get (Q x) ∨ ¬(Q x)
                         So we have two cases:
                         a) Assume (Q x) (H4)
                               We have H4
                         b) Assume ¬(Q x) (H4)
                               We have H4
                               We have found an element x0 that verifies
                               ¬(Q x0), namely x
                               So, from H2, we deduce a contradiction
                               So, this case cannot happen
                         We have (Q x) in both cases
                      We have proved (Q H0) ⇒ ∀x: U. (Q x)
                      We have found an element x that verifies
                      (Q x) ⇒ ∀x0: U. (Q x0), namely H0
                   We have ∃x: U. (Q x) ⇒ ∀x0: U. (Q x0) in both cases
             We have ∃x: U. (Q x) ⇒ ∀x0: U. (Q x0) in both cases
       We have proved
       (∀P: Prop. P ∨ ¬P) ⇒
        ∀U: Set. U ⇒ ∀Q: U ⇒ Prop. ∃x: U. (Q x) ⇒ ∀x0: U. (Q x0)
```

Fig. 7c.

Figure 7. Various forms of proofs of Smullyan's 'Drinkers' principle': a. the informal proof; b. the proof-object; c. the proof-object translated back into natural language.

4.6. Related work

There are several systems for CM based on a different methodology. Not all of these have portable proof-objects, and therefore one has to believe in their design. Nevertheless, these systems are rather interesting.

A system of CM based on some form of TS, but without proof-objects is Isabelle, see Paulson [17]. This system has a good module for term rewriting, which is important for equational reasoning.

Systems of CM not based on TSs are HOL, see Gordon et al. [8], the Boyer-More theorem prover, see Boyer et al. [3], OTTER, see Wos et al. [21], and MIZAR, see Rudnicki [20]. HOL is based on higher order logic and has been used for hardware verification. The Boyer-More theorem prover is based on a formal system called *primitive recursive arithmetic* (PRA). Because this system is relatively weak – it has no quantifiers and only states universal propositions – there are more strategies for automated proof search for PRA, than for the stronger theories. OTTER is based on the resolution method and is able to find many proofs of tautologies in predicate logic used in intuitive mathematical proofs. MIZAR is based on set theory formulated in predicate logic. Many theorems have been proved in this system.

5. Conclusion

Systems for computer mathematics are very promising. Nevertheless, presently they still have some weak points. There is a need for a good vernacular to make formalising more natural; there is a need for a good way to handle symbolic computations; and finally for the TSs used one needs to prove the faithfulness for the formalisations.

I expect, that within a decade systems for CM are more mature. In particular they will include (or use) the power of systems for CA to deal with equations. Then CM will be essentially stronger than CA, because of the fact that statements can be proved. (Working with CA systems one may overlook necessary side-conditions.) Two interesting uses are probable. One in the field of interactive development of mathematics and one in the field of software design.

The interactive development of mathematics does not imply 'Death of proof' or the end of human involvement with mathematics, as some have claimed. On the contrary, both proofs and the ingenuity of the user will play an essential role in computer mathematics and its applications. Proofs are essential, because without them there is no warranted correctness; humans are essential, because otherwise proofs cannot be found.

The limited experience with CM systems has shown that the phase of defining concepts is very essential. Once sufficient experience is obtained with handling complicated notions, I expect applications to specification and correctness of software systems. A necessary condition is, that software is written in a modular way, as is possible in e.g. functional languages. Some

researchers express doubts, that the design methodology of the Chinese box will be sufficient to produce correct software. They do believe, however, in program extraction from verified proofs. In any case, precise specifications and proofs will be important.

Acknowledgements
I thank the following persons for useful information: G. Barthe, M. Bezem, R. Boyer, A. Cohen, R. Constable, H. Elbers, H. Geuvers, G. Huet, H. Meijer, R. Plasmeijer, R. Platek, R. Pollack, M. Ruys, F. Vaandrager and H. Wupper.

REFERENCES

1. H.P. BARENDREGT (1992). Typed Lambda calculi. S. ABRAMSKY ET AL. (eds.). *Handbook of Logic in Computer Science*, Oxford University Press, 117-309.
2. H.P. BARENDREGT, T. NIPKOW (eds.) (1994). Types for Proofs and Programs, Lecture Notes in Computer Science, 806, Springer, Berlin.
3. R.S. BOYER, J.S. MOORE (1988). *A computational logic*, Academic Press, New York.
4. A. BUNDY (ed.) (1994). Automated Deduction—CADE-12, Lecture Notes in Artificial Intelligence, 814, Springer, Berlin.
5. R. CONSTABLE, ET AL. (1986). *Implementing Mathematics with the NuPrl Proof Development System*, Prentice Hall, London.
6. Y. COSCOY, G. KAHN, L. THÉRY (1995). Extracting text from proofs. in: M. DEZANI-CIANCAGLINI (eds.). *Typed Lambda Calculus and Applications*, Lecture Notes in Computer Science 902, Springer, Berlin, 109-123.
7. G. DOWEK, ET AL. (1993). The Coq Proof Assistant User's Guide Version 5.8. Projet Formel, INRIA, Rocquencourt.
8. M.J.C. GORDON, T.F. MELHAM (1993). *Introduction to HOL: a theorem proving environment for higher order logic*, Cambridge University Press, Cambridge.
9. Z. LUO, R. POLLACK (1992). LEGO proof development system: User's manual. Technical Report ECS-LFCS-92-211, Computer Science Department, University of Edinburgh.
10. P. MARTIN-LÖF (1984). *Intuitionistic Type Theory*, Studies in Proof Theory, Bibliopolis, Napoli.
11. J. MCCARTHY (1962). Computer Programs for Checking Mathematical Proofs, in: *Recursive Function Theory, Proceedings of a Symposium in Pure Mathematics*, V, American Mathematical Society, Providence, RI, 219-227.

12. T.F. Melham, J. Camilleri (eds.) (1994). *Higher Order Logic Theorem Proving and its Application*, Lecture Notes in Computer Science, 859. Springer, Berlin.

13. R. Musil (1952). *Der Mann ohne Eigenschaften*. Rowohlt, Hamburg.

14. R.P. Nederpelt, J.H. Geuvers, R.C. de Vrijer (eds.) (1994). Selected Papers on Automath. Studies in Logic 133. North-Holland, Amsterdam.

15. C. Parent (1994). Certified programs in the system Coq—The program tactic, in: [2], 291-312.

16. C. Paulin-Mohring (1993). Inductive Definitions in the Sytem Coq—Rules and Properties, in: M. Bezem et al. (eds.), *Typed Lambda Calculus and Applications*, Lecture Notes in Computer Science 664, Springer, Berlin, 328-345.

17. L. Paulson (1994). *Isabelle: A generic theorem prover*. Lecture Notes in Computer Science 828, Springer, Berlin.

18. R. Plasmeijer, M. van Eekelen (1993). *Functional Programming and Parallel Graph Rewriting*, Addison Wesley, New York.

19. D. Pountain, (1994). Functional programming comes of age, in: *Byte*, 183-184.

20. P. Rudnicki (1992). An overview of the MIZAR project. Available by anonymous FTP from
mcnaik.cs.ualberta.ca as `pub/Mizar/Mizar_Over.tar.Z`.

21. L. Wos, R. Overbeek, R. Lusk, J. Boyle (1992). *Automated Reasoning: Introduction and Applications*. McGraw-Hill, New York.

Mathematical Statistics: Fringe or Frontier?

R.D. Gill

1. An overview

1.1. Introduction

This essay contains a personal view of the place of mathematical statistics in mathematics and science, with an eye to the future. Statisticians are sometimes paid to make predictions, but they are trained to be careful and hedge their bets by giving an indication of its uncertainty. I shall be careful not to be specific, as far as the future is concerned. Most of the essay is concerned with the recent past (which surely contains the seeds of the future) and is built around some anecdotal case-studies of fascinating but in various senses paradoxical developments in the field.

1.2. What is mathematical statistics?

Statistics is concerned with analysing data, especially in the presence of randomness, whether due to measurement errors, deliberate sampling from a larger population, biological or behavioural variability, or whatever (more on this later). *Mathematical statistics* is the mathematical theory of how to do this. It codifies and organizes strategies for learning from data and for drawing conclusions about the real world phenomena which generated them. It is deeply connected with, and partly grown out of, probability theory, concerned with how to calculate probabilities of outcomes in random structures. Since mathematicians tend to become fascinated with the

abstract structure in the subject before them for its own sake, modern mathematical statistics contains many deep and beautiful mathematical results whose connection with real life data analysis can be tenuous. 'Applications' may only turn up later, or the theory may help in retrospect to explain why certain practical procedures work as well as they do.

1.3. Position within mathematics

As a relatively young branch of mathematics, uncertain of its independence, its practitioners often have ambivalent feelings about their relationship with mathematics proper. And real mathematicians may not always consider statistics as 'within the fold'. A large wall poster still adorns the corridor of many a German university's mathematics department, giving a well known publishing house's schematic plan of the whole of mathematics. It took me some time to locate my own discipline in this grand scheme of things. In the middle of the picture were boxes labelled with names built of permutations of the words algebra (or algebraic), analysis (analytic), geometry (geometric), number, and theory. These boxes were connected with a dense network of arrows indicating connections in all directions.

As one moved away from the centre the connecting arrows became less dense, the names became more varied, less 'pure', less abstract. A distant corner contained the box 'mathematical statistics', connected by a single arrow *from* probability theory. This in its turn was only linked to the rest of mathematics by arrows from 'measure theory' and 'potential theory'. Only after tracing back through many links did one arrive back at the centre of mathematics.

This picture is a caricature to be sure, but it reflects a common view of statistics held both by mathematicians and by other scientists. Statistics is a bit dirty and messy, a necessary evil perhaps; but from the point of view of those 'at the heart' it is a fringe event. Renowned mathematicians have occasionally called for abolishment of the whole discipline. The inventor of the Kalman filter—a piece of mathematics without which man would not have set foot on the moon—argued that 'chance' does not exist and therefore statistics is meaningless. Outsiders delight in the sometimes bitter controversies between different schools of statistics: subjectivists versus objectivists, exploratory data analysts versus decision theorists, and so on. So is statistics a marginal activity?

A rather different picture is given by a table published a book by N.J. Higham [1]. The table gives the six most often cited papers in mathematics and computer science, statistics included. Four of the six are actually papers in statistics, published in more or less theoretical journals (Journal of the American Statistical Association, Journal of the Royal Statistical Society); the other two are papers on numerical mathematics. (One of these two—introducing the fast Fourier transform—by an author who later went on

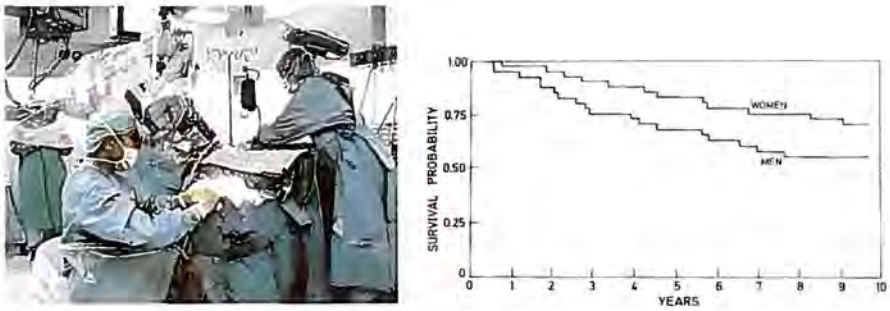


Figure 1. An important aspect of medical treatment of severe diseases like cancer is to observe survival times after treatment and draw statistical inferences from these observations. The curves drawn here (Kaplan-Meier estimates), show the survival probabilities after operation for malignant melanoma for 205 patients, stratified by sex. (Photo: courtesy Academisch Medisch Centrum Amsterdam.)

to be the founder of exploratory data analysis). Perhaps I should mention that though the cited papers were taken from mathematics and computer science, the 'citers' could have been in any discipline.

1.4. Survival analysis

Let me focus on two of these most cited papers [2, 3]. The second most often cited paper is by E.L. Kaplan and P. Meier (1958) and number four, by D.R. Cox (1972); both are concerned with survival analysis: the branch of statistics devoted to the special problems of analysing life-times, times till events, for instance the length of the disease-free period after cancer treatment in the life of cancer patients. These papers proposed new statistical techniques which became part of the standard repertoire of a huge army of cancer researchers.

But it was not just medical researchers who used these results and cited them alongside standard laboratory methods, as is their tradition. The papers just mentioned are cited enormously often by mathematical statisticians. The new techniques, developed to take account of a rather common feature of survival data, namely that many observations are *censored* (in other words, only known to exceed some value determined by the closing date of the study), relied on flashes of insight on the part of their inventors which could not be supported by then available mathematics. The special features of this kind of medical statistics has been an inspiration and a challenge to mathematicians since those key papers. Even now remarkable (for the insiders: amazing) mathematical properties are being discovered about the Kaplan-Meier survival curve estimator though it has become such a commonplace item in the statistician's toolbox that its picture has been seen on the front page of major Dutch newspapers. The work of the mathe-

maticians has also been directly used by practitioners, and survival analysis is now a large established and rich area of statistics.

The development of survival analysis did not rely solely on the then available resources of mathematical statistics. A rather abstract, recently developed and then still incomplete part of probability theory came to play a major role in understanding and developing the analysis of censored survival data: namely continuous time martingale theory and the theory of stochastic integration. I won't start to explain these terms but let me emphasize that this was pure, pure mathematics, initially as unintelligible to mathematical statisticians (let alone, applied statisticians) as it will be to most readers now.

1.5. Applications and new theory

Those landmark contributions in survival analysis also contained the seeds of major new developments in statistics going far beyond the original medical setting. The theory and practice of 'semiparametric models' had its roots there. This way of doing statistics was taken up by applied researchers in econometric modelling and in psychometrics; it has fueled intense theoretical research on building a large-sample theory for statistical inference in infinite-dimensional sample spaces.

This story shows that from an unsuspected corner of applied statistics an impulse can come which leads to redrawing the map of theoretical statistics and revitalising its connections with other areas of mathematics. Is that a once-off event? By now ancient history (the success story of the long-gone seventies and eighties)?

Figure 2. The day after Black Friday, October 13, 1989. (Photo: N. Tully-Sygma. Courtesy ABC Press.)

The answer is no. More recently—some would say, this is the success story of the present decade—financial mathematics developed deep connections with probability theory: indeed, the part of probability theory I just mentioned, the Itô stochastic calculus. The famous Black-Scholes paper [4] on how to price options was not only part of a new financial business but also part of the discovery of new deep results in probability theory. (Actually one may wonder how much recent financial catastrophies have been caused by use or mis-use of these fundamental advances.) It is easy to list many

other areas where new developments in science, technology and society have catalysed (and been effected by) revitalising activity in statistics and probability.

I would like to concentrate on two such stories. In each case the paradoxical fact is that the initial event was inimical to classical statistical theory (the abolishment crowd would lick their lips). It says something for the vitality and intrinsic need for a science of how to analyse random data that in neither case was this the end of statistics.

2. First story: the bootstrap

In 1979, B. Efron introduced into statistics his new 'bootstrap method' [5]. The essential idea of the method is to use computer simulation (in fact, a Monte-Carlo experiment, using computer generated randomness) to evaluate the accuracy of a statistical estimation procedure. 'The real world', from which random data has been obtained in order to learn about it, is replaced by an artifical 'bootstrap world' on the computer, totally under the control of the experimenter. In the bootstrap world random samples are repeatedly drawn: the variability in the estimate which is repeatedly evaluated from each new artifical dataset (an estimate of a known bootstrap world quantity) is a guide to the variability of the statistician's actual estimate which she calculated from the actually available real data in the real world.

There is a little snag in this description: the bootstrap world has to be a faithful copy of the real world in order to make the simulation experiment appropriate, but the real world is not known: to find out about it was precisely the whole purpose of the exercise. No matter, we are statisticians, so we just use our data to estimate it. Efron's audacious proposal was to do this in the most primitive way available: simply use the data points as they are as if only these values, and in precisely these proportions (each value equally likely), existed. The method then reduces to taking a sample of the same size as the original data-set *from* the original dataset (a random sample 'with replacement', so any particular data value can reappear a number of times in the new sample), recomputing the statistic of interest, repeating this procedure time and time again, and extrapolating the observed variability in the outcomes to the real world.

The method caught on like wild-fire. As the ambitions and sophistication of statisticians and the speed of their computers had increased, more and more complicated things were being done with data, and it was becoming less and less easy to use traditional means (analytic calculations in special models, or large-sample approximations) to judge the reliability of the results. Now all one had to do was leave the computer to repeat the original calculation of interest a thousand times on easily made artificial data sets, and you are done. One is then also liberated from only using methods for which explicit analytic calculations are feasible. Efron's grand

claim was that his method abolished the need for mathematical analysis in statistics, replacing it by brute force computing power, and freeing the users of statistics from the traditionally available recipes, which were becoming a straight-jacket. Now anyone can creatively decide what he or she wants to do with the data, and leave the bootstrap to furnish the standard errors, confidence intervals, significance levels or whatever.

The method was so audacious that many people a little further from statistics refused to take it seriously. How could you get more information from data by throwing in extra random variation generated by yourself on your own computer? A senior research manager at the mathematics section of a well known Dutch multinational which at the time did not employ statisticians once told me that the reason for this was that nothing new had happened in statistics for the last forty years. I mentioned the bootstrap as a counter-example: but that was such a stupid idea it only confirmed his belief. Now the same company has belatedly caught on to the fact that a lot has happened, a lot of great value in an industrial research environment, and is rapidly building up its own sizeable internal statistical consultation unit.

Interestingly the bootstrap revolution did not put mathematical statisticians out of work after all. Along with practical success stories, came case-studies where the bootstrap gave stupid answers. In any case, why should it work at all? How well does it work? Can one make it work better? An explosion of activity took place with all kinds of variants being proposed of the original easy to understand methodology, in order to make the method more reliable, more flexible, more accurate, less computer-intensive (!), and so on. Some of the mathematical tools needed to really understand why or how the bootstrap works turned out to be linked to the traditional central activities of pure probabilists, nowadays somewhat looked down on: higher order corrections to the famous central limit theorem, which says that sample averages are approximately normally distributed. Other mathematical tools were connected to fundamental advances in pure mathematics, connected to the very abstract topic of 'probability in Banach spaces'. Yet other tools were needed from the theory of asymptotic statistics.

In a sense the bootstrap liberated applied statisticians from making tedious analytic calculations but it required a deeper and more creative level of mathematical activity, namely to understand and categorise the fundamental structure of diverse statistical methods, and their relation to fundamental probabilistic limit theorems.

3. Second story: quantum statistics

My other story is a story perhaps just starting, namely a new involvement of statistics in quantum physics. Let me begin this story on what may seem a philosophical issue, namely the question of whether randomness actually

exists. This is indeed a debatable point! A tossed coin or dice, for probability theory the archetypal random experiment, is just a simple dynamical system. Small differences in the initial vertical speed and angular momentum of the coin or dice are exponentially quickly magnified to large differences in its final position. Nothing random happens at all. The randomness in the initial conditions is presumably completely deterministically explainable in similar terms. The randomness of the bootstrap samples on the computer are also not random at all: a computer slavishly carries out its instructions; a random number generator is a sophisticated but completely deterministic way of 'mixing up' initial conditions so that what comes out looks random. (Interestingly enough, the modern theory of random number generation at the same time links fundamental ideas from statistics, from number theory—the heart of pure mathematics—and from the theory of computational complexity—the heart of theoretical computer science.) Now there is one place and I believe one place only where randomness really happens in the real world, and that is at the quantum level. Quantum physics describes the completely deterministic and continuous evolution (according to Schrödinger's equation) of the state of quantum systems (systems of fundamental particles, photons, etc.). From the state at a given time can be calculated the *probability distribution* of the results of measurement of the system. To give some examples: a particular photon either does or does not pass a given polarization filter; an electron is either found or not found in a given region of space. Quantum theory tells us what the statistics would be like of many repetitions of these experiments: in a certain percentage of times, a photon passes the filter; in a certain percentage of times an electron is registered in a particular region of space. Radioactivity as measured by a Geiger-counter, showing a seemingly random series of time-points of emissions of individual particles, is another classic and nowadays even familiar example; an example where the randomness is not averaged out into the statistics of many particles but where it is still present at the macroscopic level.

Figure 3.

Now a little thought shows up a huge paradox in the theory. A measurement device (e.g., a photo-multiplyer set up to allow one to decide if a single photon does or does not go through a polarization filter) is itself just a large collection of elementary particles. The device together with the photon being measured together form a single quantum system, which develops deterministically and completely smoothly according to a huge com-

plicated Schrödinger's equation. Nothing actually 'happens', and certainly nothing happens or does not happen by chance. Similarly, a radioactive substance and a Geiger-counter together are also just one large assemblage of fundamental particles whose joint state is evolving deterministically and continuously according to the appropriate Schrödinger equation.

A traditional way out of the paradox is to suppose that things only 'happen' when a conscious observer looks at the system. Everything now becomes subjective, or circular, or an infinite regress threatens. What happens when an observer observes another observer? This is the famous Schrödinger's cat paradox, which hinges around the question of whether a cat, which is killed if and only if a certain radioactive decay takes place within a certain period of time, actually does die or not at the moment of the radioactive decay, or if this only happens when a human observer looks into the cage to see what has happened. This paradox forms part of R. Penrose's thesis (developed in his books 'The Emperor's New Mind' [6] and its sequel [7]) that human consciousness is essentially a quantum physical phenomenon and therefore artificial intelligence based on classical models of computing is impossible!

This state of affairs causes no difficultiesin practice: the theory makes predictions which so far have agreed with all empirical findings. And theory which has consequences for laboratory experiments also has consequences for everyday technology. Technological advances are leading to experiments and experiments are leading to technological advances in which truly quantum phenomena have an impact on our everyday world. 'Quantum cryptography' is a practical way of transmitting messages safe from eavesdroppers which depends on paradoxical quantum phenomena (so-called entangled states) which have fascinated philosophers and visionaries and cranks for years; right now programs for quantum computers are being designed which for instance will factor large numbers in polynomial time by sheer brute force, simultaneously trying out all the possible factorizations, coded as a quantum superposition of states. Physicists predict that within five years the first real quantum computer will have been built; it will be able to successfully factor the number 'fifteen'. This is not a joke: the first digital computer was also not of much practical use. (When the first computer in The Netherlands was demonstrated to the Minister of Science, the program it ran was a program to produce a random number. That way the minister would not notice if the computer had actually worked properly or not!)

So nowadays quantum physicists are manipulating systems of a really small number of fundamental particles. These systems exhibit random behaviour of a truly fundamental character: this is randomness which cannot be explained by recourse to hidden but deterministic variation at a lower level (this is the content of 'no-go hidden variables theorems', connected to Bell's inequalities, the Einstein-Podolsky-Rosen paradox, and so on).

One would expect that a physical theory so closely involved with probability would be well-known and well-studied by probabilists and statisticians; and that these experts would even have a big contribution to make. Strangely that is not the case. Actually just before Kolmogorov in 1933 succesfully axiomatized classical probability theory (the probability theory of dice and coin tosses, of insurance companies and casinos), John von Neumann axiomatised quantum probability. The mathematical structure was more general, and much more abstract looking than ordinary probability. Physicists (for instance R. Feynman in the famous Feynman lectures, see also [8]) and later on mathematicians have continually claimed that 'quantum probability is a different kind of probability'. The field seems far from ordinary probability and statistics, and a huge barrier has been set up between them. The physicists perhaps did not really understand the mathematical modelling involved in ordinary probability; and pure mathematicians who were able to get a feeling for the mathematical structure of quantum mechanics—and who in recent years have developed an imposing theoretical structure called quantum probability theory—probably had so little feeling for physics that they accepted the statements of famous physicists like Feynman without questioning. The many paradoxes of the field anyway are enough to make any mathematician shy of saying anything about the practical side of the subject: he or she will just prove theorems in the abstract mathematical playground which physics provides.

In my opinion quantum probability is not a different kind of probability at all. My personal opinion is that quantum *reality* is a rather different kind of reality to ordinary reality (Einstein has said: reality is weirder than we imagine; weirder than we *can* imagine), the challenge is to classical deterministic thinking rather than to probability. Moreover the field is ripe for a new involvement of statistics. Already physicists are discussing ways of learning about the state of a quantum system from the (random) results of measurements which can be made on it. A couple of books and a fast growing number of papers exists on the topic (see, e.g., [9, 10, 11]). So far it is developing independently of modern statistics. Physicists are busy reinventing classical ideas from statistics: this is not a bad thing in itself; the bad thing is that they are unaware of the tremendous advances which that science has made in the last half century.

As long as both physicists and statisticians and pure mathematicians believe that quantum probability is 'a different kind of probability' this bad state of affairs will persist. But I think there are a lot of signs that this accepted wisdom is about to be thrown aside and the result will be a tremendous enrichening both of mathematical statistics and of quantum technology.

4. Conclusion

The stories I have sketched above, and many others I could tell, show the young science of statistics vigorously growing at the interface between mathematics and society. New developments in technology and society immediately set huge challenges to applied and to theoretical statisticians: how to analyse data of growing complexity and how to answer the increasingly complex questions which society poses. These challenges reverberate into the heart of mathematics and sometimes answers are found using tools developed long ago in seemingly unrelated parts of mathematics, sometimes the challenges stimulate new fundamental advances.

It should be obvious now whether I think of statistics as a fringe or a frontier to mathematics. In my opinion it is part of the living frontier of mathematics; intensely alive; intensely unpredictable.

References

1. N.J. Higham (1993). *Handbook of Writing for the Mathematical Sciences*, SIAM, Philadelphia.
2. E.L. Kaplan, P. Meier (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*. Volume 53, 457-481.
3. D.R. Cox (1972). Regression models and life-tables. *Journal of the Royal Statistical Society*, Series B, Volume 34, 187-220.
4. F. Black, M. Scholes (1973). The princing of options and corporate liabilities. *Journal of Political Economy 81*, 639-654.
5. P. Diaconis, B. Efron (1983). Computer-intensive methods in statistics. *Scientific American*, 96-108.
6. R. Penrose (1989). *The Emperor's New Mind*. Oxford University Press.
7. R. Penrose (1994). *Shadows of the Mind*. Oxford University Press.
8. R.P. Feynman (1951). The concept of probability in quantum mechanics. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 533-541.
9. J.D.Malley, J. Hornstein (1993). Quantum statistical inference. *Statistical Science 8*, 433-457.
10. A. Holevo (1982). Probabilistic and statistical aspects of quantum theory. North-Holland, Amsterdam.
11. C.W. Helstrom (1976). Quantum detection and estimation theory. Academic Press, New York.

The Many Faces of Computer Science

H.J. Sips

1. Introduction

What constitutes the core of computer science? The answer will vary depending on who you ask. It will likely range from a branch of mathematics to an engineering discipline of constructing hardware and software systems. The fact is that techniques and systems from computer science have penetrated very deeply into other disciplines and often stimulated the development of new methods of research. Computers are probably wider used as part of a research method in other sciences than mathematics.

Computer science can be defined as the theoretical, constructive, and experimental science of information processing systems. It is a relatively new science, which has grown from a small core to a very important discipline for society in a time span of only four decades. Central in this development is the digital computer, which by its virtue of almost universal applicability as information processing medium, has placed itself amidst developments in many organizations.

With the enormous increase in the use of computers came the need to put some order in the developments and to create a solid theoretical and methodological basis on which new systems and applications can be developed. In this, computer science relies on the empirical corpus that has grown in four decades in constructing and using information processing systems [1].

In this article, we will place developments in computer science in perspec-

tive and in relation to theory, construction, and experiment as the basic constituents of computer science methodology. Of course, there can be no in-depth treatment or discussion on detailed subjects. Instead, some major developments and trends will be discussed.

2. Mathematics and computer science

In its kind, computer science is a bit of a strange science: it does not follow the traditional separation between disciplines studying artificial objects, such as mathematics, logic, and theology, and those concerned with observable objects or phenomena, such as physics or biology. In fact, computer science deals with objects from both worlds: it shares its interest in formalisms, symbolic structures and their properties with mathematics. On the other hand, it has much in common with constructive sciences such as electrical engineering when it comes to the design and realization of hardware and software systems.

Its common interest with mathematics in artificial objects is the cause that many computer science faculties have their roots in the mathematics department or be still part of them.

In this respect, the distinction between theoretical computer science and mathematics is often not very clear. This is in contrast with other disciplines such as physics, where we have theoretical physics to explain the nature of physical phenomena (often in highly mathematical terms) and mathematical physics, which is a supporting discipline for theory and experiments in physics.

2.1. Theoretical computer science versus mathematical computer science

Could the same distinction be made in computer science? Would we be able to discriminate between theoretical computer science as revealing the nature of information processing systems and mathematical computer science which is to be supportive to all branches of computer science? Let us try to make such a distinction as an experiment of thought. Complexity theory is clearly very much related to the nature of computing itself. Hence we would have no problem in classifying this field as to belong to theoretical computer science. But what about for instance Petri nets (see also figure 1) and its theory? Petri nets lend themselves equally well to describing all kinds of dynamic phenomena outside the domain of computer systems and is as such more a general mathematical modeling technique than just revealing the nature of computing. Because Petri nets are frequently applied in computer science, such a subject would then accordingly be classified as belonging to mathematical computer science. On the other hand, the Petri net model could also be considered as a model of computation serving as a semantic model for certain programming systems. From these examples it is clear that to make such a separation is far from trivial.

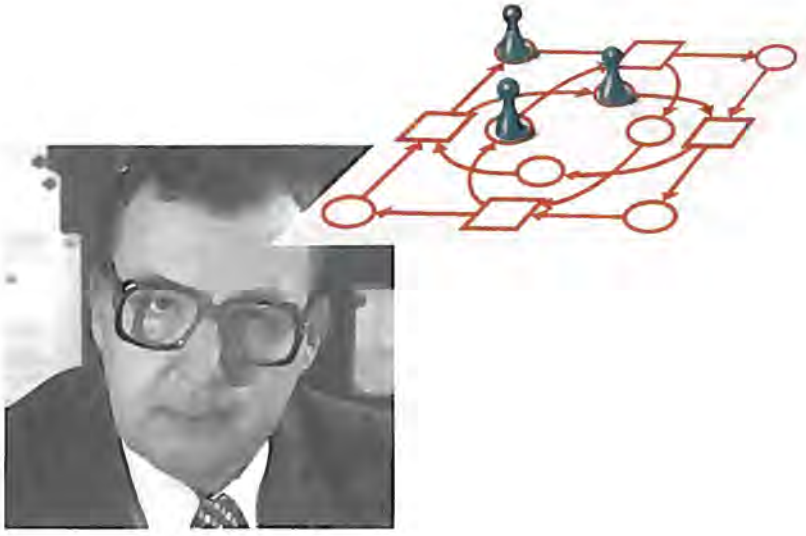


Figure 1. The German computer scientist Carl Adam Petri developed in the 1960s a general method—called Petri nets—for modelling distributed systems and processes (Photo: S. Münch, GMD).

2.2. Model and reality

Why then try to make such a distinction in the first place? The main reason is the problem we have in computer science in the distinction between model and reality [3]. It is often advocated that the application of more formal techniques in computer science will allow us to design systems at a higher level of abstraction, enable proofs of correctness, and lead to robust systems. While in principle this is true, it can only be done effectively if model and reality coincide to a considerable extent. But what is reality in computer science? It is the way we design and build systems. However, we have much freedom in doing that. There is no such thing as a reality 'out there' such as in other sciences against which a model can be validated. This also implies that we can make reality look like the model we have. Some researchers even think that the model is the basis and that reality should shape itself like that. However, current experience is that if we do this, there is a price to pay: some applications cannot be realized efficiently anymore in terms of resource usage and/or time requirements.

Another problem is that in many cases there is not really a precise notion available of the objects we use in reality. We use concepts like 'processes' and talk about 'distributed systems', but generally define them in a rather vague way. For some concepts, like processes, theories do exist, but again these are models and not reality.

2.3. Formalisms

Apart from the progress made, results obtained in the theory of computer science yet have not had a significant impact on computing practice. One of the reasons is the existing gap between model and reality as already explained. A model is necessarily an abstraction of reality. It is in the way abstractions are chosen, where things usually go wrong. Too often abstractions are made on the basis of the resulting mathematical elegance. It suits the mathematician; nicely manipulatable objects result. However, many essential features are abstracted away, leading to formalisms which cannot really be applied in practical cases. Nice examples are theories of communicating processes. The first theories only allowed synchronous communication between processes. This could not hold: asynchronous communication is very essential in many real-life systems and must be part of any theory of processes.

Another reason for the low impact of formal techniques is the highly developed (mathematical) skills that are needed to use them. Most software developers designing actual systems are not acquainted with formal techniques and reasoning. One cannot expect to educate enough people to master these methods and obtain the necessary mathematical skills. The only way would be to bring these techniques down to a form understandable to the average system designer and supported by user-friendly tools.

To overcome the current problems, research in theory should be more directed towards diminishing the gap between model and reality and less towards the (mathematical) art of modelling.

3. Constructing computer systems

The constructive part of computer science deals with methods and tools to construct hardware and software systems. The hardware side is concerned with the construction of memories, CPU's, and interconnects. This field is conceptually relatively mature in the sense that we know how to construct computer systems. (See also figure 2.) The progress in terms of capacity and speed is currently merely of a technological nature.

This does not mean that no progress has been made. The production of hardware components has become a highly industrialized process. The enormous investments required to develop a new generation has forced a certain standardization of hardware components. These developments make the construction of computer systems from basic components relatively easy.

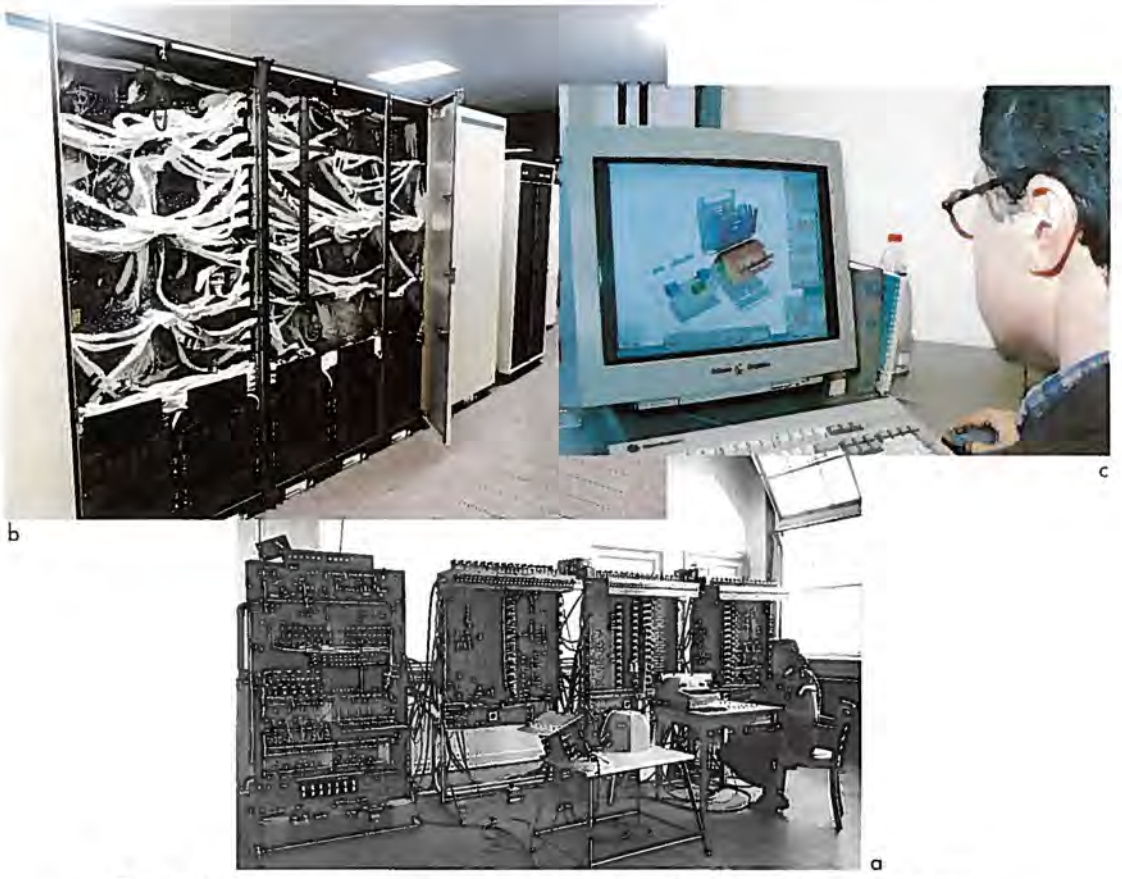


Figure 2. Contrary to software, hardware technology has considerably matured over the years: (a) the ARRA computer developed at CWI (1952), (b) a CDC Cyber 995 mainframe (1980's), and (c) a high performance graphics workstation (1990's).

On the software side, technology is much less mature. The process of software development is still dominated by much detailed hand-crafted work and, even worse, development time is not diminishing at a pace required to deliver in time robust software systems with good performance. Software development time is now becoming the major critical factor in bringing new products to the market.

This problem has been recognized for some years now and is referred to as the software crisis. Basically, two approaches have been proposed to solve the problem. On the one hand, raising the level of abstraction of programming languages and systems would give programmers a more powerful way to express their applications and leave many of the implementation details to smart compilers. The ultimate goal is to be able to automatically

generate code from precise specifications.

Another approach is to reuse code. Much too frequently, programmers implement the same functions and algorithms all over again and do not use programs that already have implemented the required features.

3.1. Programming languages

The two proposed solutions would indeed help to solve the software crisis. However, things have not developed along these lines. There is still a strong base of third generation imperative programming languages, which is not likely to disappear very soon. The original goal of a single powerful programming language for all purposes has not been achieved. On the contrary, powerful programming languages, like ALGOL, have not survived for various reasons. On the other hand, a proliferation of languages has also not occurred. We even see a development towards a smaller set of languages due to the enormous price pressure on software caused by the success of the personal computer. Good quality compilers for programming languages on personal computers can only be provided at low cost when there is a very large user base.

Thus far, higher level languages such as functional languages have not had their expected (by some) breakthrough. Partly this is caused by the lack of commercially available, efficient implementations and partly by the lack of user acceptance of the different model of computation that comes with the use of such languages. Object-oriented features on the other hand seem to find their way into the world of programming languages, not as fully fledged new programming languages, but more as add-on's to existing languages like C, COBOL, and Ada. It is not clear whether this popularity is due to the fact that objects provide an easy mechanism to create abstract data types or that features such as inheritance are favoured. The latter concept is certainly more difficult to handle, since it relies on the modelling capabilities of software designers, and when applied incorrectly, can easily lead to bad software designs.

Will higher level programming languages be accepted in the near future? It must be said that at the moment their future as general programming language is not bright. However, for prototyping purposes or as a language tailored to a specific domain, concepts found in these languages might be very useful. Systems like MatLab or various script languages show that domain specific high level programming systems do satisfy user needs. Also, the popularity of spreadsheets shows that a different programming paradigm can be attractive for specific applications, but the added value must be very clear.

3.2. Software reuse

Complementary to the use of powerful languages, software reuse has the potential to speed up program development. By applying software reuse techniques, certain parts of a program are composed from a number of well-engineered and documented and frequently used code fragments. Although an appealing idea, the problem of software reuse in part turns out to be an organizational problem. One can only apply this technique if reuse software models are accepted on a wide scale and reuse libraries are standardized. For specific fields this has long been current practice (e.g. numerical libraries), but in other domains the sheer effort seems to discourage any real progress.

Reuse on a larger grain size level has more of a chance, meaning reusing larger software components to construct new applications. For example, a spelling checker could be reused in various editors or word processing systems. The investment question is in that case a lot simpler: either use an existing piece of software or completely do the coding yourself. The remaining question is the interface problem and a possibly not completely matching functionality.

Related is the recent interest in so-called coordination languages [4]. Coordination languages in effect form a binding component between several pieces of (existing) software. The coordination language (or system) takes care of the proper interaction between the various software components. It is advocated that applications consisting of software objects written in different programming languages can be realized faster and more flexible. As an example, consider an application working according to a client/server model. Client/server interaction could be programmed in a coordination language, while the actual code for the client and server processing is written in another language. Also in distributed systems a coordination approach to system design will often be necessary, as local systems will be implemented by using different programming languages.

3.3. Conclusion

From the above arguments, one might conclude that no real progress in software construction has been made in recent years. This is too negative a conclusion. We have seen computers change from large unfriendly mastodons to user-friendly personal computers and workstations. This is not only due to hardware developments. Frequently occurring functions in applications such as user interfaces and databases have developed into powerful reusable products with standardized interfaces. The desk top metaphor, although first critically received by many computer scientists, can be considered a true innovation.

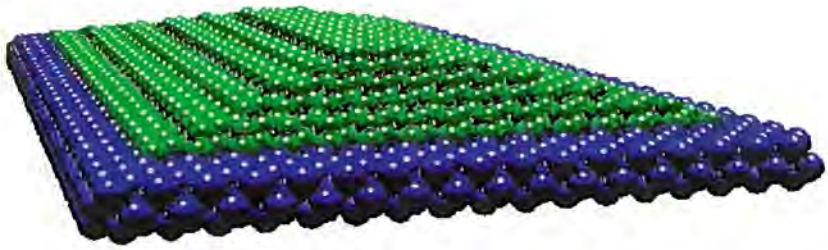


Figure 3. Upper layers of a protracted 'hut cluster', containing about ten thousand atoms created on a Si(100) 1x2 surface by molecular beam epitaxy. Computer simulations of complex systems like the dynamics of such clusters require considerable (parallel) computing power and become more and more an essential part of scientific research. (Courtesy Delft University of Technology, department of Applied Physics/Physics Informatics.)

4. The experimental side

The third view on computer science is experimental. In general, any software system is based on a set of requirements. Some of these requirements are functional, some are non-functional (such as performance). Requirements may be explicit or implicit, quantifiable or not quantifiable. More important, the functionality space is not one-dimensional. Many non-comparable aspects need to be taken into account before the question whether an application serves its purposes can be properly answered.

As a consequence, many software systems are so complex that the only way to validate new concepts is to set up experiments. This is normal practice in any experimental science and may take the larger share of a project's funding. Surprisingly, this is hardly ever done in computer science [2]. There is a lack of experimental evidence in most computer science projects, mainly because there is no money left (or asked for) for validation. Here the binding of computer science to mathematics works out negatively. The main research method in mathematics is analytical and there is no real tradition in performing experiments as part of the research method. As a consequence, most computer science results only consist of claims, without

Figure 4. Recent developments such as those around Internet corroborate the ongoing dramatic influence of computing on communication.

ever proving them to come true in an experimental setting.

Another problem is that building software systems is often a tremendous task, which usually does not contribute to academic research records. Even worse, time spent on writing programs cannot be spent on writing papers. With the current emphasis in academia on the quantity of publications, this indeed will remain a problem for some time.

5. Impact on other sciences

Apart from internal developments, computer science also has introduced a new research method in traditional sciences. For instance, in physics (computer) simulation has become an important third research method, complementing theory and laboratory experiment (see figure 3). In general, the field of modelling and simulation has been given a large impulse through the availability of powerful and relatively cheap computers.

Besides having introduced a new research method, computer science has also extended the corpus of other sciences. An example is management science, where information technology is considered a new production factor along with human resources and capital.

6. What next? Computation and communication

Meanwhile research in computer science itself is very much driven by the astonishing development of computer hardware. It is not that the basic principles of digital computation have changed so much, in fact nothing really fundamental has changed since the day of Von Neumann's conception of the principle of digital computers, it is the mass production and minia-

turization of basic devices such as memories and processors which is the main driving force in today's computer science research.

The above developments are also bringing together the field of computation and communication (see also figure 4). Research and developments in both areas have long been quite separate, even each with their own jargon and terminology. Surely, digital computation did enter the communication field years ago, but mainly for its internal operation (i.e. in digital branch exchanges). But the merge of communication with computation opens completely new fields of application. For the first time computers will be used to create new economic activities rather than just automating the existing ones.

The impact on research will be large. Many research questions need to be addressed. If we can link computers together without (technical) problems, irrespective where they are placed, questions arise whether we can manage such complex systems. System configurations will become much more dynamic and will have to be maintained while in operation. The question of interoperability of systems and languages will have to be addressed again. The strange thing is that within the sequential computer we have not been able to realize proper solutions for this problem. However, distributed systems can simply not be realized without having solutions for the interoperability problem.

In trying to come up with answers to these questions we are faced with the problem that we really do not know where we are heading with this technology. And we cannot find out without really building and experimenting with systems and applications. In short, all faces of Computer Science are needed, in mutual cooperation, to find the appropriate answers to the challenges imposed upon us.

References

1. A. Ralston, E.D. Reilly (eds.). (1993). *Encyclopedia of Computer Science*, 3rd ed., IEEE Press. Van Nostrand Rheinhold.
2. R.L. Glass (1994). The software research crisis. *IEEE Software*, November issue.
3. R. Kurki-Suonio (1994). Real Time: further misconceptions (or half-thruths). *IEEE Computer*, June issue.
4. N. Carriero, D. Gelernter (1992). Coordination languages and their significance. *Communications of the ACM, 35(2)*.
5. (1994). *European Information Technology Observatory 94*. EITO, Frankfurt.

National Mathematics Programme

Polynomial Splines in Two Variables

C.R. Traas

1. Introduction

The SMC research project 'Numerical and Fundamental Aspects of Polynomial Splines in Two Variables' was focussed on the special type of functions referred to as *splines*. These functions do have profitable properties with respect to operations like interpolation, approximation and geometric modelling, which make them outstandingly suitable for applications in various fields of industrial design and numerical mathematics. For example, splines are popular tools for the description of curves and surfaces or, more general, *shapes*. In industrial design they are applied to visually represent all kinds of industrial products on the computer screen (cars, aeroplanes, ships, bottles, shoe-soles, tableware, etc.). Also the shapes of natural configurations like landscapes or earth layers can be adequately described. Not only such geometrical objects, but also functional dependencies obtained from, e.g., measurements can be easily represented: radar reflection patterns, *thermodynamic functions*, tomographic data, etc. In addition, spline functions are often used in numerical mathematics as basis functions in Rayleigh-Ritz-Galerkin processes for solution of boundary value problems for ordinary and partial differential equations.

Historically, splines were motivated as tools for interpolation due to poor behaviour of polynomials in this respect. First steps on a higher level were done by I.J. Schoenberg in the 1940's. With the advent of the computer in the 1950's/60's, the development accelerated dramatically and resulted in a

Figure 1. Originally splines were long strips made of very flexible strong material, used as an engineer's tool at the drawing-table. Nowadays the computer has almost completely taken over this design function.

fairly complete theory and practice for splines in one variable. In particular the various recursive algorithms as initiated by, e.g., C. de Boor were of great importance. Much effort also was put in the research concerning splines in two and more variables. However, notwithstanding a lot of progress, many problems are still open. For example, the problem of shape-preserving approximation is still far away from a general solution. In the current research project a number of these problems were studied in depth.

A spline function in s variables is a piecewise analytic function on its domain of definition Ω which is part of the s-dimensional space $\mathbb{R}^s$. If Ω is bounded, then the subdomains on which the spline is analytic form a finite partition of Ω. In the one-dimensional situation the points where the spline is not analytic are called 'knots'. In the current project mathematical techniques were investigated to construct spaces of spline functions having favourable properties with respect to the desired applications. Also the *computability* of such splines has been dealt with extensively. In the project the analytical parts of the splines were restricted to be of polynomial nature. This class of splines is called the class of *polynomial splines*.

Problems arise when the spline is required to be several times differentiable while the domain $\Omega \subset \mathbb{R}^s$ is arbitrarily shaped and $s > 1$. Other types of problems arise when *closed* bodies in $\mathbb{R}^3$ are required to be described using a high degree of (geometric) continuity. In this latter case one uses *parametric* spline surfaces, and the problem is connected to the fact that the surface of a closed body can not be mapped in a continuous one-to-one way into $\mathbb{R}^2$.

Finally, the problem of the *shape-preserving* description of a surface has been addressed in the project. These latter investigations were of mainly theoretical nature.

2. Polynomial splines

2.1. B-splines in one dimension

One of the basic contributions to the theory of polynomial splines in one dimension is the discovery of basis functions with compact support, the so-called B-splines, by H.B. Curry and Schoenberg in 1966 [1]. These are piecewise polynomial functions with a support of $n + 1$ consecutive subintervals, where n is the degree of the polynomial parts, and with $n - 1$ times continuously differentiable connections at the knots. A support of length $n + 1$ subintervals is the smallest possible support for non-trivial splines of degree n and of class C^{n-1}. The number $n + 1$ is called the *order* of the B-spline. The B-splines form a basis in the space of polynomial splines defined over a given partition of the considered domain $\Omega \subset \mathbb{R}$. Their importance is found in the fact of their very simple computability, which is due to the existence of a recursion relation (De Boor and M.G. Cox, 1972). This relation admits a numerically stable way of building up the higher-order B-splines, starting with B-splines of order 1. Also for differentiation and integration simple rules exist. The derivative of a B-spline can be computed as a weighted difference of two B-splines of one order lower (De Boor, 1972). An expression for the integral of a given spline was found by De Boor, T. Lyche and L.L. Schumaker (1976). With these rules the basics for practical computing with polynomial splines in one dimension are available. Further improvements could be attained by giving special attention to the specific properties of splines and their algorithms. For example, it is possible to let coincide, purposedly, several consecutive knots, which can be interpreted as admitting subintervals of length zero. At such a multiple knot the spline will have a lower order of continuity compared with the continuity order at single knots. The above mentioned recursion, however, can still be applied without any special measure. The computability is thus not affected by introducing multiple knots. Furthermore the B-splines can be normalized such that they form a partition of unity at every point in the domain. A consequence is a close relationship in shape between a spline

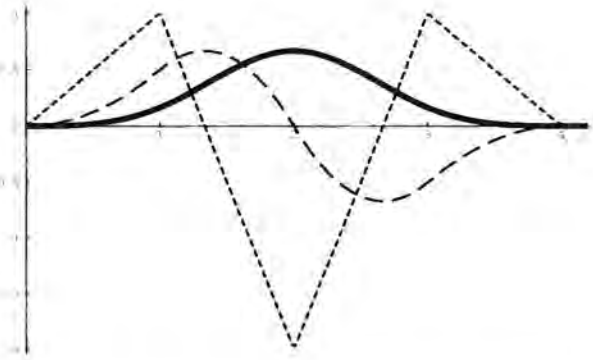


Figure 2. A cubic B-spline (order=4), and its derivatives.

$s(x)$, written as a linear combination of B-splines, and the so-called *control polygon*; this is the polygon which connects the consecutive *control points* by straight line segments. The control points are defined as the points in the x, y-plane which represent the coefficients in the B-spline expansion of $s(x)$: the y-coordinates are the values of the coefficients themselves, and the x-coordinates are the values of the coefficients in the B-spline expansion of the function x over the same knot partition. The similarity in shape allows predictable change of shape of $s(x)$ by changing coefficient values. This is of great importance for shape design purposes. For these latter purposes is also of great importance the possibility to insert additional knots at pre-selected positions, for which a number of simple and elegant algorithms exist.

In figure 2 the normalized cubic B-spline over the set of knots $\{0, 1, 2, 3, 4\}$ is shown (solid line), together with its first derivative (dashed line) and second derivative (dotted line). In figure 3 a cubic spline is depicted over the same set of knots, together with its control polygon. Due to multiplicity of the boundary-knots, the first and last control points coincide with the begin point and end point, respectively, of the curve.

From a theoretical point of view the relation that exists between divided differences of polynomial half-space functions (truncated power functions) and B-splines is of great importance. Properties of B-splines can be derived from this relation in an elegant way. This opens perspectives with respect to the research to B-splines in more than one dimension if the notion of 'divided differences' can be extended to more dimensions in a suitable way. In the present project this item was an important subject of research.

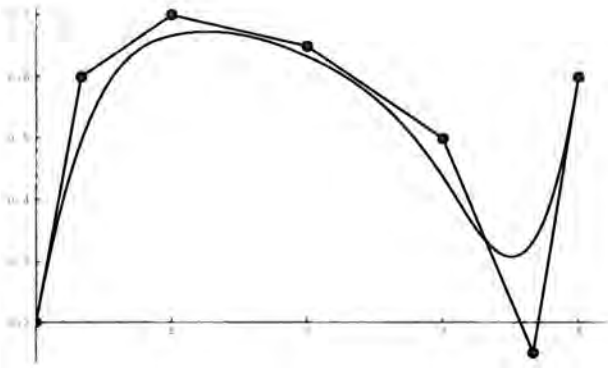


Figure 3. A cubic spline and its control polygon.

2.2. B-splines in more dimensions

B-splines in more dimensions have been topic of research already since several years. An obvious extension to the higher dimensional situation is obtained by constructing *tensor products* of univariate splines. This type of extension is of practical advantage in the sense that it allows the use of repeated univariate algorithms. An obvious drawback, however, is the very limited flexibility with respect to the shape of the domain and the density distribution of the set of knots. A more general extension arises from the notion of *polyhedral spline*. The definition of B-splines, based upon this notion, is strongly geometrical: B-splines in s dimensions are defined as functions, the values of which are proportional to the volumes of corresponding intersections through polyhedra in higher-dimensional spaces. The first actual construction of a B-spline on the basis of this geometrical principle and with a *simplex* chosen for the polyhedron, was performed by De Boor (1976) [2]. Later on other types of polyhedra were used, resulting in the construction of, e.g., *box splines* and *cone splines*. The tensor product splines, mentioned above, could be interpreted as special cases of box splines. Recurrence relations were found soon (L.A. Micchelli, 1980), guaranteeing the relatively simple computability of the splines.

The possibilities for practical use of simplex B-splines were for the first time extensively investigated by R.H.J. Gmelig Meyling (1986). These B-splines appeared to be suitable for high quality approximations of functions in two variables over arbitrary finite domains. However, the computing effort appeared to be high. A further improvement of computing efficiency is needed in order to render these splines really useful for practice. In figure 4 a bivariate quadratic simplex B-spline of class C^1 is shown. Its support

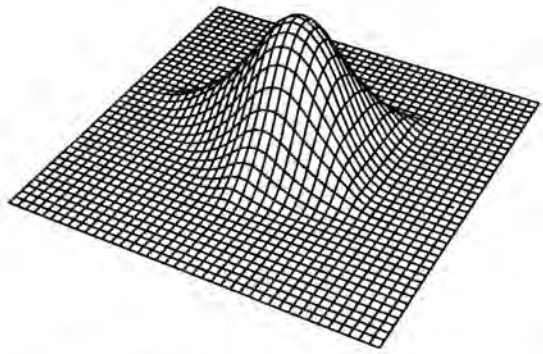


Figure 4. A bivariate quadratic simplex B-spline.

is the convex hull of 5 knots in the plane.

Another extension of splines to more than one dimension, which is not based on polyhedra in higher-dimensional spaces, uses a *triangulation* of the domain and the definition of *Bernstein polynomials* on each of the elements of this triangulation. Using Bernstein polynomials allows in a relatively simple way the construction of a surface of class C^1, or even class C^2, by imposing side conditions on the control points. Pioneering work has been done by Schumaker (1979 and later), in particular with respect to the dimensions of such spline spaces [3]. Also for these splines the practical utility was investigated extensively by Gmelig Meyling (1986).

3. The research in the project

The aim was to consider polynomial splines in two variables. A major part of the fundamental research was devoted to bivariate simplex splines. A first step was the generalization of the notion of univariate divided differences to the higher-dimensional situation. This generalization is based upon a pointwise evaluation of a multivariate function. Next, the simplex spline is expressed as the multivariate divided difference of a generalized polynomial half-space function. The properties of the multivariate simplex splines could be derived from the properties of the multivariate divided differences. Also new proofs were formulated for a number of already known results.

Much attention was paid to the computability of the simplex spline. Using as starting point a publication of E.T. Cohen among others (1987), in which an alternative recurrence relation was presented for evaluation of multivariate simplex splines, new recurrence relations were found for directional derivatives and for inner products of simplex splines. The numerical prop-

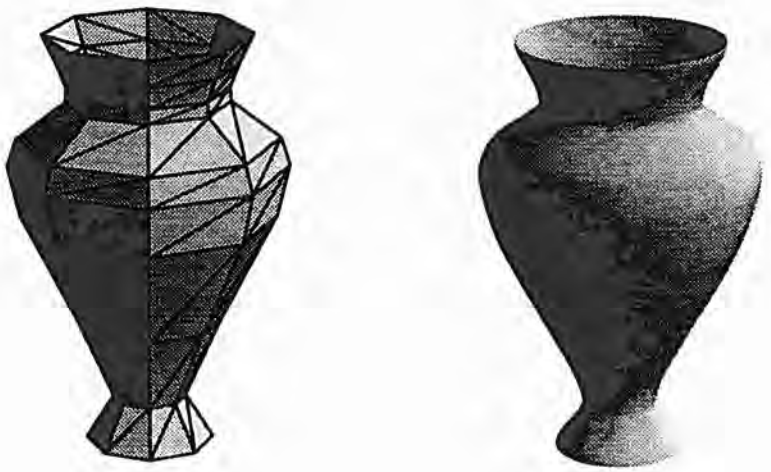


Figure 5. A triangulation (left) and a quintic interpolant.

erties of these new algorithms are investigated (stability, computing effort) and compared with the older algorithms. It appears that progress has been made in particular with respect to the complexity of the algorithms.

An alternative for the computation of simplex splines is based on the concept of *subdivision*. In the case of box splines this is an accepted and practically very useful technique, due to its efficiency and simplicity. For the case of simplex splines little was known about this topic. For this reason the notion of *discrete simplex spline* is introduced and some properties are derived. Discrete B-splines arise when continuous B-splines, defined with respect to a given net of knots, are expressed as linear combinations of continuous B-splines which are defined with respect to another net of knots in the same domain. The latter net usually is a refinement of the first net. It will then be obvious that discrete B-splines take a central position in subdivision processes. The investigations have led to the formulation of an algorithm for subdivision of simplex splines.

Another topic of research was the smooth interpolation of scattered data in three-dimensional space, using spline surfaces. A suitable method was designed using *degenerated* triangular Bézier-Bernstein patches. This degeneration has to be understood as a multiplicity of some of the control

points. The necessity of the use of degenerated patches is a direct consequence of the imposed requirements: (1) the method should be *local*, (2) the patches should be *polynomial*, and (3) the geometric continuity should be of *order 1* at least. Locality means that only local information is used to construct the accessory local part of the surface. Results were obtained for the polynomial degrees 4 and 5. It appeared that the method is suitable for the description of closed bodies as well. In figure 5 an example is given of an object which is described by using degenerated quintic polynomial patches and which is of geometric continuity class C^1. The data set coincides with the vertices of the triangulation.

Finally, attention was given to the smooth approximation and interpolation of convex functions, preserving the convexity. These investigations were mainly of theoretical nature. One result that was obtained is a proof that, whenever a finite-dimensional approximation space is used, the use of *local* operators for interpolation in general will not result in preservation of convexity. Thus an interpolation method which is such that preservation of convexity is guaranteed must be *global*.

The major part of the above research was done by M. Neamtu in the framework of his Ph.D. thesis.

References

1. H.B. Curry, I.J. Schoenberg (1966). On Pólya frequency functions IV: The fundamental spline functions and their limits. *J. d'Analyse Math. 17*, 71-107.
2. C. de Boor (1976). Splines as linear combinations of B-splines. G.G. Lorentz et al. (eds.). *Approximation Theory*, 1-47.
3. L.L.Schumaker (1979). On the dimension of spaces of piecewise polynomials in two variables. W. Schempp et al. (eds.), *Multivariate Approximation Theory*, 396-412.
4. M. Neamtu (1991). *A contribution to the Theory and Practice of Multivariant Splines*, Ph.D. Thesis. Twente University.

Ergodic Theory

M.S. Keane

1. Basic ideas

Ergodic theory is a mathematical endeavour which arose from the study of statistical mechanics by physicists in the latter half of the nineteenth century, as an ongoing attempt to derive the macroscopic, statistical laws of thermodynamics from deterministic microscopic behaviour. The basic concept which is studied in this mathematical discipline is that of the *measure preserving transformation*. Thus my first task is to create an image inside of your head, reader, of what we think of when we hear this collection of words. The most important word, *transformation*, indicates that we are dealing with a change, or movement, of a collection of basic (i.e. indistinguishable except for their names) objects, and the other two words, *measure preserving*, are intended to show that the sizes of subcollections of these objects do not change after the movement, or transformation, is applied.

1.1. A simple example

Here is a simple example. Consider three indistinguishable objects, placed in positions which we simply denote by a, b, and c respectively. Each of the three objects has the same *size*, where it is perhaps best to think of the size of an object as its weight: we generally call this non-negative number the *measure* of the object. The positions a, b, and c are usually called *points*. Now imagine the following movements taking place simultaneously:

the object at point a moves to point b, the object at point b moves to point c, and the object at point c moves to point a. Thus we have defined a measure preserving transformation, since after the movement each position is occupied with an object of the same size as before. The objects, of course, can now disappear from our discussion, since they are only distinguished by their positions and we can think of the measure of an object as a number attached to, or more generally a mass distribution over, the collection of positions: this is a typical mathematical ploy.

1.2. The measure preserving transformation

After this simple example, we jump to the attempt at creating a general picture in your head. A *measure preserving transformation* is defined as a collection of points (i.e. a *set* or a *space*) together with a mass distribution over this space of points, and a transformation assigning to each point of the space another (perhaps in some instances the same) point of the space, such that after simultaneous application of the transformation to each point of the space, the same mass distribution is observed.

Here we perhaps need to remark that our initial example was very simple, in that we were dealing with a finite set of points. The most interesting and natural situations deal with much larger sets of points, such as the interval of real numbers between 0 and 1, or more generally spaces whose 'points' are themselves collections of other objects, e.g. paths of particles or positions of sets of points. In these situations, it is more difficult to define the concept of mass distribution, and there is an entire branch of mathematics developed around the beginning of the twentieth century, called *measure theory*, which lays down the rules for mass distributions and their behaviour under transformations. A thorough knowledge of measure theory is indispensable for research in ergodic theory, although on an intuitive level the concept of mass distribution and mass transportation seems to be easily accessible to a general audience.

1.3. A more interesting example

Let me now try to fill out the abstract picture given above with a more interesting example, which was one of the motivations for the study of ergodic theory at its beginning in the nineteenth century. Imagine a box filled with a large number N of gas molecules (for example, air, or more simply, hydrogen). At any fixed time, we can visualize the situation in the box by writing down the exact position and velocity of each of the molecules in a (very long) vector $\vec{x}$ of $6N$ real numbers. (I have written 'visualize' because of the practical impossibility of carrying out such a description: the number N will be much too large in any reasonable situation.) Now imagine the space X of all possible vectors $\vec{x}$ such that their *energy*, which is simply a number we can calculate from the entries of the given vector $\vec{x}$ by a simple

formula which will not concern us further here, is a fixed number E. The mass distribution we want to consider over X is the natural uniform distribution with total mass one, and the movement is given by starting at time zero in the configuration given by $\vec{x}$ and then calculating the positions and velocities of each of the molecules at time one, putting them all together in another long vector which we denote by $T(\vec{x})$ and call the transformed point. The calculations can be done according to different rules, but let us suppose here that we are interested in the rules given by classical mechanics. The measure preserving transformation thus described is commonly known as 'gas in a box', and the preservation of mass was first proved by Liouville in the middle of the nineteenth century.

2. Questions of interest in ergodic theory

Our next task is to describe some of the basic questions of interest in the field of ergodic theory. From the second example it should be clear that one of the goals is to get away from very detailed, local investigations of the behaviour of individual molecules or points, as in this example it would be impossible to say very much. The simplest way to formulate this restriction is to realize that we wish to deal with successive movements, and in particular to try to describe the long-term behaviour after many many iterations of the measure preserving transformation. In the first example, things are quite clear; after two movements, a is at c, b at a, and c at b, and after three movements everyone is back to his starting spot and things repeat as before. In other words, this is a periodic transformation with period 3. The second example presents more difficulty, but just recently it has been shown by the Hungarian School (for identical molecules of a fixed size and so-called elastic collisions with each other and with the sides of the box) that except for a set of starting points having probability zero, the positions and velocities will come arbitrarily close to *any* given set of positions and velocities again and again as the movement is iterated, after a sufficient number of movements. Thus, with probability one, all of the molecules will eventually collect in the right half of the box (but not stay there), if we wait long enough! This 'inevitable suffocation', although mathematically sound, is also very interesting because it contradicts the second law of thermodynamics, although it has been deduced from the first principles of classical mechanics, the only acceptable physical principles on a microscopic level for a wide class of gas models and densities.

After the above detour into the world of physical interpretation, we now return to mathematics, with a discussion of some mathematical problems and a few results obtained in the past years in The Netherlands and elsewhere connected to ergodic theory. Below we treat three areas of interest: percolation, one-dependent processes, and interval dynamics. We shall try to exhibit the corresponding measure preserving transformation, but it

will not be possible to arrive at a detailed understanding of the underlying connections and proofs. Most of the work discussed has been carried out at Delft University of Technology, and substantially supported by the NWO/SMC-grant 'Coding Problems in Ergodic Theory', as well as other local and national funding.

3. PERCOLATION

3.1. Mathematical models of percolation

In mathematical models of percolation, the underlying measure preserving transformation is not temporal, but spatial. The simplest model runs as follows. Imagine a regular grid of interconnected pipes in three-dimensional space, and suppose that a certain percentage of the pipes are open, allowing liquid to flow through them, while the remaining pipes are blocked. We assume that this configuration has been obtained by some random mechanism, for instance an independent coin toss with the same coin for each pipe. The space X in this situation is the collection of all (infinite) prescriptions of which pipes are open and which are blocked, while the movement is a spatial movement of translation in a direction parallel to some of the pipes, by the length of one of the pipes. There are in this example actually six different directions, so we have six measure preserving transformations, coming in pairs which are inverses of each other. The mass distribution is described by the random pipe blocking mechanism. The basic idea of percolation theory, developed by physicists earlier but put on a sound mathematical basis in the 1950's, is that if only a small percentage of the pipes are blocked, then there will be paths stretching to infinity along which liquid can flow, but if the blocking percentage is large, then all of the liquid is localized and cannot escape from a finite region. Thus there should be a critical blocking percentage, below which these infinite open paths appear and above which there are no infinite paths. (See also figure 1.) The basic applications of
92 these ideas are in the areas of oil exploration and spread of disease, and it is interesting to be able to prove that critical blocking percentages (generally called critical probabilities) exist, calculate them, and more generally describe the nature of the random picture of a realization of the process. Both geometric and measure theoretic issues are important here.

3.2. Recent results

In the short space available, it would be impossible to detail the connection between ergodic theory and percolation, and we must be content with the statement that this point of view has proven very fruitful in understanding and solving many of the open problems, and providing simple proofs of theorems previously believed to be very complicated in nature. I list shortly some of our results—the reader should be aware that the selection is made

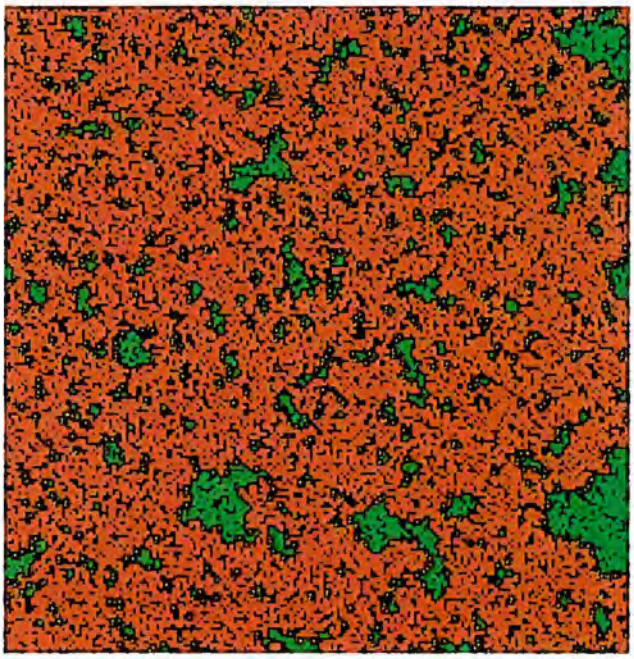


Figure 1. A computer realization of site percolation on the square lattice. Red is percolating.

to indicate the work carried out in the 1980's in The Netherlands, and is not a representative sample of the many interesting ideas in the field of percolation.

- The fundamental Van den Berg-Kesten-inequality, permitting calculation of several critical probabilities, and also of basic theoretical importance ([1]).
- The uniqueness of infinite clusters of open pipes and of densities of clusters in a large range of physically feasible percolation models ([2],[3]).
- Continuity results for percolation probability functions ([1]).
- Exact calculations for critical probabilities and percolation probability functions in circle percolation models ([5]).
- Connectivity and uniqueness in Mandelbrot percolation ([5]).

Most of these results do not directly use established methods of ergodic theory; instead, they raise substantial new questions concerning measure preserving transformations.

4. One-dependent processes

In ([6]), a long-standing conjecture concerning the existence of one-dependent processes which are not two-block factors of independent processes was settled, giving rise to a substantial collection of new one-dependent processes. Let me try to explain the idea behind one-dependent processes. First of all, it is easiest to think of a process as a doubly infinite sequence of two symbols, say 0 and 1, chosen in some random manner. For instance, suppose that for each element of the sequence we flip a coin which is labelled 0 on one side and 1 on the other (the coin need not be fair, but it should be the same coin for every element of the sequence). This gives rise to an independent (stationary) process. A generalization of independent processes, widely studied, is that of Markov processes, in which there are two coins, and the one flipped depends on the outcome of the flip in the preceding element of the sequence. Markov processes have the property that the future flips are independent of the past flips if one supposes the value of the present flip to be known. The *definition* of a one-dependent process is one in which the future is independent of the past, when no knowledge of the present is assumed. That is, the observer sees an event in the past, goes to sleep at the present and misses an observation, and then observes independence in the future with respect to what he saw in the past. It is easy to see that if we take any independent process (with perhaps more than two symbols, even more than a finite or countable number of states), and make another process by a function depending only on two successive observations with values 0 and 1, then this process is one-dependent. The conjecture was that every two-state one-dependent process arises in this manner, and ([6]) contains a large number of new one-dependent processes, which cannot arise
94 in this manner. It is not easy to see how ergodic theory can be of help in this problem, and the underlying transformation is difficult to describe, and not even measure preserving. The methods indicate a connection to and a generalization of quantum probabilistic reasoning, which is not yet well understood and will certainly be the subject of further investigation.

5. Interval dynamics

One of the most interesting problems of ergodic theory is to establish existence of and to calculate the values of invariant measures for a given transformation of a space. In particular, one-dimensional transformations (maps of the unit interval to itself) received a large amount of attention in recent years. Thesis [4] treats existence of invariant measures for such transformations, and provided cornerstones for a number of subsequent results.

6. Classification

Ergodic theory provides an explanation for the apparent randomness observed in physically deterministic models. One can attempt to determine the different types of randomness by *classifying* measure preserving transformations. One of the most fruitful approaches to the classification problem is provided by the theory of finitary codes, developed by M. Smorodinsky and myself in the 1970's. At present, a thorough study of such classification of different types of pure randomness, both in classical and quantum descriptions, is being prepared.

7. Invariant measures

As we have mentioned above, the problem of finding and making explicit invariant measures for a given transformation or transformations is central to ergodic theory. I cannot resist closing this essay with an interesting open problem, due to H. Furstenberg. Let S and T be the transformations of the unit interval defined by $Sx = 2x \bmod 1$ and $Tx = 3x \bmod 1$. Then the normalized Lebesgue measure (uniform distribution on the unit interval) is invariant under both S and T. Does there exist another continuous probability measure on the unit interval with this property?

8. Conclusion

In this short essay we have attempted to briefly describe the basic idea underlying the mathematical discipline of ergodic theory, to give a short description of its physical origins, and to describe summarily some aspects of work at Delft University of Technology using ergodic theory to answer fundamental questions inside the discipline and to contribute to problems in related fields. The essential reason for the wide range of application is the fundamental nature of the notion of a measure preserving transformation, together with its surprising complexity.

References

1. J. van den Berg (1985). *Some Contributions to Percolation Theory and Related Fields*, Ph.D. Thesis, Delft University of Technology.
2. R.M. Burton, M.S. Keane (1989). Density and uniqueness in percolation. *Communications in Mathematical Physics*.
3. A. Gandolfi (1989). *Clustering and Uniqueness in Mathematical Models of Percolation Phenomena*, Ph.D. Thesis.
4. M. Martens (1990). *Interval Dynamics*, Ph.D. Thesis, Delft University of Technology.
5. R. Meester (1990). *Two Models of Dependent Percolation*, Ph.D. Thesis. Delft University of Technology.

6. V. DE VALK (1988). One-Dependent Processes. Ph.D. Thesis, Delft University of Technology.

Markov Decision Chains

A. Hordijk

1. Introduction

Many real-life phenomena have a stochastic dynamic behaviour. Mathematical models for analyzing these phenomena are stochastic processes. For example, in order to study the queue-length at a counter, the mathematical model supposes an arrival process of customers and a distribution of service times. The most simple process already studied early this century by A.K. Erlang, the pioneer in queueing theory (see figure 1), assumes that the probability of an arrival in an interval is linear in the length of this interval with a rest term that is of smaller order than the length of the interval. A similar assumption is made for the service process. This model with Poisson arrivals and exponential service times is denoted by $M/M/1$. Erlang used this mathematical model to compute the long run blocking probability of a telephone-exchange. His goal was to study the quality of service provided by the Danish telephone company he was working for.

In modern telecommunication technology high-speed networks are designed to carry different types of traffic, like audio, video, and data.

One of the challenging problems is to derive the optimal admission control. Given a certain load in the network, should a new arrival, which generates traffic of a certain type, be accepted to the network or should it be blocked? The mathematical problem now becomes: what is the optimal control of the underlying stochastic process?

Figure 1. A.K. Erlang, the pioneer in queueing theory.

2. Markov decision chains

There are several variants of this mathematical model. Let us describe in more detail the discrete-time Markov decision chain (MDC) with a discrete state space. In this model at discrete-time points, which may be stochastic, a decision or control has to be taken. In the admission control model these time points are the arrival times (epochs) of customers. The state of the controlled stochastic process is an element of a subset of the points with integer coordinates in a space of finite dimension. In the admission control model this is the number of customers of the various types at the various nodes in the network.

Each transition from a state at a decision epoch to the state at the next decision epoch has a certain probability. These transition probabilities depend on the chosen control. The control also influences the stochastic rewards and costs until the next decision epoch. For example, the acceptance or rejection of an arrival induces different costs and/or rewards.

The controlled stochastic process may be considered over a finite or an infinite time horizon. In the earlier case the total expected cost is relevant, in the latter, infinite case the total (expected) discounted cost is often considered: when discounted, the cost and/or reward at time instant t is multiplied by α^t with α the discount factor, thus yielding a finite total expected cost. For discount factors close to one, the Laurent expansion of the total discounted cost in the variable $(1-\alpha)$ is important:

$$\frac{g}{1-\alpha} + u_0 + (1-\alpha)u_1 + \dots$$

In this expansion, g in the first term is the average cost per time unit, and the second term u_0 gives the bias which is the limit of the difference of the total cost over a finite time horizon t minus t times the average cost, as t tends to infinity. All higher order terms have similar interpretations. The Laurent expansion and all its terms depend on the chosen control or policy. One considers several optimality criteria in the nondiscounted case. Average optimality means optimizing the first term of the Laurent expansion. Bias optimality corresponds to lexicographic optimization of the first two terms. (In comparing two policies, this means that if the average cost of policy 1 is lower than that of policy 2, or if, the average costs being equal, the

bias of policy 1 is lower than that of policy 2, then policy 1 is preferred to policy 2.) And in more sensitive optimality criteria more terms of the Laurent expansion are taken into account. The most sensitive criterion, Blackwell optimality, is lexicographic optimization over all terms of the Laurent expansion.

The history of Markov decision chains goes back to the fifties. The first papers were on optimal inventory control. The pioneer R. Bellman wrote his book on Dynamic programming (1957) including a chapter on MDCs. This book also has a chapter on Markov games, a closely connected mathematical model. In Markov games, which were introduced by L. Shapley (1953), there are two or more controllers, called players. The players have different objective functions and mostly play against each other. R.A. Howard, in his book Dynamic Programming and Markov processes (1960), focusses on algorithms and applications.

G. de Leve introduced MDCs in The Netherlands with his Ph.D. thesis: Generalized Markovian decision processes (1964). This started a school of researchers in this field, first at CWI, and later also at universities.

In the late sixties a rather complete theory was available for MDCs with a finite number of states. This theory contains theorems on the existence of optimal policies for the discounted, average, bias and more sensitive optimality criteria. Moreover, optimality equations and methods to solve them were obtained. For a denumerable state space only isolated results were available.

Although a dozen of papers on denumerable MDCs are still appearing each year, we can safely say that a rather complete theory for the denumerable case has been established now, almost twenty years later.

3. Denumerable Markov decision chains

One can claim that all problems in practice have a finite state space, and one can question the importance of a theory for nonfinite models. However, in many applications the size of the state space is large but unknown and then often the denumerable state space is the natural model. Also the simple structure of the optimal control is often lost, when the denumerable state space is truncated to a finite one. For example, in the admission control problem the optimal control in the case of linear holding costs is of control-limit type, i.e., a customer of specific type is accepted as long as the total number of customers of that type is below a certain number, the control-limit. For computing the optimal control and even more for implementing it in practice, the simple structure is crucial. By truncation of the state space this simple structure is lost.

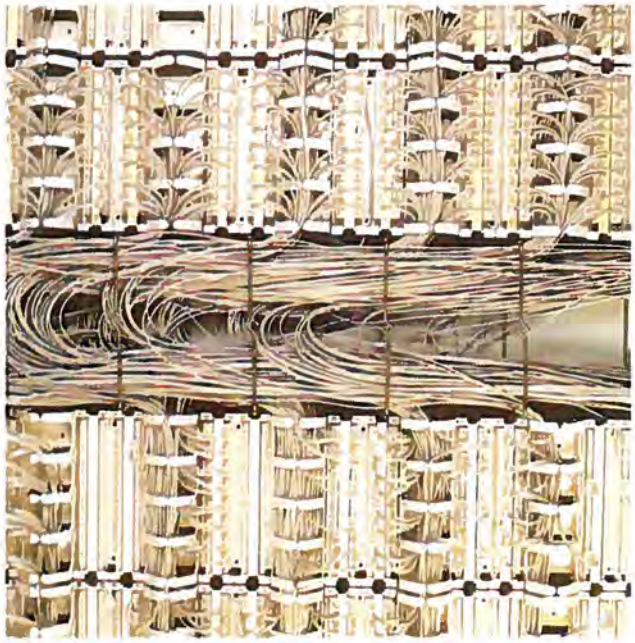


Figure 2. The efficient operation of modern telephone exchanges poses several challenging research problems. (Photo PTT Telecom.)

3.1. Ergodic theory for Markov decision chains

Several of the main steps in developing the theory for denumerable MDC were part of the NWO-SMC projects on Markov decision chains. Let me mention some major results. Whereas the Laurent expansion of the total discounted cost in the finite model always exists, strong recurrence conditions, which guarantee that the stochastic process will return sufficiently 'fast' to a compact set, were necessary for a denumerable number of states. For a satisfactory theory the recurrence conditions should not only be satisfied in the finite state model, they should also be fulfilled in denumerable state applications like the admission control of a telecommunication network.

In classical Markov chain theory there is an extended ergodic theory. In this theory the limiting time average is analyzed, i.e., the behaviour of

$$\frac{1}{T}\sum_{t=1}^{T} X_t$$

is studied as T tends to infinity. The limit, provided it exists, is the long run average reward per time unit, in case X_t is the reward at time t. In

applications discounting is often not appropriate and then the most often used criterion is this long run average reward.

In Markov decision theory not one Markov chain is considered, but a compact set of Markov chains, indexed by the stationary policies, is relevant. For developing a theory for denumerable MDC's it was important to generalize the ergodic theory for one Markov chain to a compact collection.

W. Doeblin was the pioneer in the ergodic theory for Markov chains. His condition for ergodicity, later on called the Doeblin condition, required that the expected recurrence time to a finite set is uniformly bounded in all starting states. Clearly, this condition is too restrictive for almost all queueing models. For example in the simple model with one server, Poisson arrivals and exponential service times (the $M/M/1$ queue), the required number of transitions to the empty state is at least as large as the starting number of customers, so the expectation can never be uniformly bounded for all starting positions. In Markov chain theory Doeblin's work was generalized by T.E. Harris and his recurrence condition is appropriate for queueing models. In Markov decision chains there is the natural requirement that not only the chains are recurrent to a compact set, but that also the expected total cost until this recurrence time is finite. This inspired a condition, which we later on called μ-recurrence. In this assumption the vector μ is a bounding vector of the vector of immediate costs and/or rewards. With this bounding vector weighted supremum norms can be introduced. It is the appropriate extension of the supremum norm, i.e., the μ-norm of vector x is

$$\|x\|_\mu = \sup_i \frac{|x_i|}{\mu_i}, \quad i \in E,$$

where E denotes the state space.

With this vector norm the corresponding norm on the space of matrices is given by

$$\|A\|_\mu = \sup_i \frac{\sum_j |A_{ij}|\mu_j}{\mu_i}.$$

For a given Markov chain let P be the matrix of transition probabilities, and for taboo-set B let ${}_BP$ be the matrix obtained from P by replacing P_{ij} by zero if $j \in B$.

Now the μ-recurrence condition is

$$\exists \text{ finite } B \text{ such that } \|{}_BP\|_\mu < 1.$$

It generalizes Doeblin's condition, since for the bounding vector $\mu = e$ with $e_i = 1\ \forall i$, e-recurrence is equivalent to Doeblin's condition.

The strong ergodic theorem for Markov chains can be stated as (we assume for the ease of presentation that the Markov chain is aperiodic):

$$\|P^t - \Pi\|_\mu \xrightarrow[t\to\infty]{} 0,$$

with Π the matrix of stationary probabilities.

The research in the SMC-NWO-project resulted in a theory for denumerable MDC that uses as basic assumption μ-uniform recurrence for all stationary policies, i.e., $\exists$ finite B such that

$$\sup_f \| {}_B P(f)\|_\mu < 1$$

where $P(f)$ is the matrix of transition probabilities under the stationary policy f. Key results are:

- The continuity of the Laurent expansion as function of the policy.
- The extension of the strong ergodic theorem for Markov chains to μ-norms and Markov decision chains.

The extension to μ-norms is also an original contribution to Markov chain theory. It inspired important research by S.P. Meyn and R.L. Tweedie for Markov chains with a general state space. The generalization of the strong ergodic theorem for Markov decision chains with a general state space is currently in progress.

3.2. Markov decision chains with partial information

With the many applications in telecommunication, models with decentralized control become important. Consider a communication network. In a certain node a controller has to route customers or packets to one of the neighbouring nodes. His control depends of course on the destination of the packet, it may also depend on the number of jobs on the outgoing links of the node. If we model this as a MDC then the control depends on partial information, the controller may not use all information in the complete state description of the network, because his control may not depend on the numbers of customers in links not adjacent to his node.

Within the SMC-NWO project in recent years MDCs with partial information have been investigated. An algorithm has been constructed for computing a memoryless policy that uses partial information and is close to optimal or optimal in that class of policies. The usual approach to solve a MDC with partial information (or partial observation) is to convert this problem to a MDC with a continuous state space via a posteriori probabilities. The drawback of this approach is that the resulting MDC is unsolvable and also that the optimal policy at time t depends on all realized states and actions from time 1 up to time t, so it is far too complicated to implement it in practice. The new approach provides surprisingly good and implementable policies in the various models studied until now.

Applying MDCs in practical problems remains an art, for each problem a special problem oriented method has to be constructed. The main reason is

m	N
1	3
2	11
3	49
4	261
5	1631
6	11743
7	95901
8	876809
9	8877691
10	98641011

Figure 3. The number of states (N), as a function of the number of customers (m).

the curse of dimensionality in real-life applications, since in most cases the number of states increases exponentially fast with the size of the problem. Figure 3 shows the number of states of a recently analyzed closed queueing network with customer routing and only two nodes.

Usually the situation is more favourable, and often we can handle networks with four nodes. However, clearly a lot of research is still waiting in order to overcome this dimensionality problem.

References

1. R. Dekker, A. Hordijk (1992). Recurrence conditions for average and Blackwell optimality in denumerable state Markov decision chains. *Math. Oper. Res. 17*, 271-289.
2. R. Dekker, A. Hordijk, F.M. Spieksma (1994). On the relation between recurrence and ergodicity properties in denumerable Markov decision chains. *Math. Oper. Res. 19*, 539-559.
3. A. Hordijk, F.M. Spieksma (1992). On ergodicity and recurrence properties of a Markov chain with an application to an open Jackson network. *Adv. Appl. Prob. 24*, 343-376.
4. A. Hordijk, J.A. Loeve (1994). Undiscounted Markov decision chains with partial information: an algorithm for computing a locally optimal periodic policy. *ZOR-Math. Meth. Oper. Res. 40*, 163-181.
5. S.P. Meyn, R.L. Tweedie (1993). *Markov Chains and Stochastic Stability*. Springer-Verlag, London.
6. H.C. Tijms (1988). *Stochastic Modelling and Analysis: a Computational Approach*. Wiley, Chichester.

Infinite-Dimensional Linear Systems Theory

R.F. Curtain

1. Introduction

Systems theory is the study of mathematical properties of dynamical systems under the influence of external inputs of two distinct types; one which cannot be influenced (the disturbance input) and one which is to be chosen so as to influence the system in some desired manner (the control input). Usually, the performance objective is expressed in terms of the property of an output (the to-be-controlled output) and additional information is available in terms of another output (the observation). In figure 1, we illustrate schematically the general situation of a dynamical system Σ which we aim to influence by another dynamical system Σ_c called the controller. The dynamical system Σ denotes the physical system we wish to influence and the dynamical system Σ_c denotes the controller we seek to design to achieve our objectives. Notice that considered as a dynamical system, Σ_c has as its input the observation, and, as its output, the input to Σ. A typical control problem is to assume that one has a mathematical description of the system Σ, together with how the inputs and outputs interact with it, and to ask how to design a controller Σ_c (in terms of a mathematical description) so as to achieve certain desired performance objectives under the influence of the inputs. It is important to note that everything depends on time, and in mathematical systems theory, both the system Σ and the controller Σ_c are described in terms of time-dependent equations. In most applications, the system is modelled either by a system of coupled ordinary differential

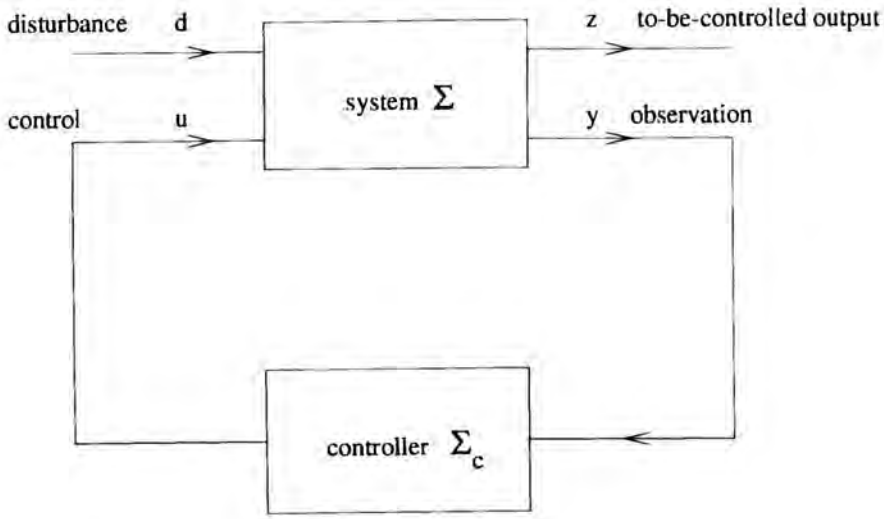


Figure 1. System and controller in closed-loop.

equations (or difference equations) whose solutions depend on the inputs. The outputs are modelled as combinations of the solutions of the ordinary differential equations (or difference equations). Typical problems posed are to design a controller Σ_c as in figure 1, so that the resulting closed-loop system has some desired behaviour; for example, so that the to-be-controlled output z tracks a given trajectory in the presence of unknown disturbances
106 d. The controller produces the input u to the system on the basis of the observation y. An application of such a result would be to devise a strategy to keep a satellite in a prescribed orbit in the presence of atmospheric disturbances.

There are many other problems which have been posed and solved and re-solved in the literature. Moreover, one can pose the same problem for a different class of mathematical descriptions of the system. Not all systems can be adequately described by linear differential or difference equations: many exhibit nonlinear, hysteresis or distributed properties. Below we discuss two particular types of control problems for a particular class of systems, infinite-dimensional linear systems.

2. Infinite-dimensional linear systems

Concrete examples are systems described by linear partial differential equations (PDE's) or by delay equations, the solutions of which depend linearly on the inputs. These arise, for example, in modelling the feedback control of large flexible space structures, chemical processes with delayed control action and noise suppression of engines in modern aircraft. The outputs will be linear functionals of the solutions to the partial differential or delay equation. This explains the modifier 'linear' and the 'infinite-dimensional' arises from the fact that the mathematical description is with respect to a Hilbert space, an infinite-dimensional vector space in which the vector represents the state of the system. Often, the term distributed parameter systems is used instead, especially in the engineering literature. In the early sixties there appeared several papers on system theoretic properties, such as controllability and observability, and on some time-optimal control problems. However, two seminal books which had a special impact on the field are [1] by A.G. Butkovskiy and [2] by J.L. Lions. Both considered versions of the classic optimal control problems for systems described by linear partial differential equations, and obtained nice generalizations of the known solutions for ordinary differential equations. In fact, this linear optimal control problem dominated the literature for two decades. One reason for this is that introducing control on the boundary or delayed control action results in a complicated mathematical description that presented an interesting challenge to PDE experts. The motivation of the two research projects described below was to get away from this overstudied problem and to investigate different system theoretic problems which had already had considerable success in the finite-dimensional literature, that is, for systems described by ordinary differential equations. The first project was on *Geometric theory* and resulted in the publication [3]; the second was on H_∞*-Optimal control theory* and resulted in the publication [4]. It is interesting to compare these topics in one article, because, while both exist as elegant, complete mathematical theories for finite-dimensional linear systems, only one generalizes to a useful theory for infinite-dimensional systems. Fortunately, both are interesting mathematical structures and both have increased our understanding of the now established field of *Infinite-dimensional linear systems theory*.

3. Geometric theory

A classic problem in this area is the following disturbance decoupling problem: for the system Σ in figure 1 construct a controller Σ_c such that the input u (depending on the observation y) produces an output z which is independent of the disturbance input d.

If we can achieve such a disturbance decoupling, it clearly has useful applications, for example, in the process industry. Think of Σ as a model of

a distillation column, u as the flow-rate of the liquid stream input, d as the fluctuations in the composition of the feedstream, y as the observed difference in the composition of the products and z as the difference in the composition of the main product. If we could construct a controller Σ_c to produce a time-dependent input u so that z becomes independent of the fluctuations in the composition of the feedstream, this would be an extremely useful device. Clearly, it is too much to expect that one can always achieve disturbance decoupling. So the mathematical problem is to investigate under which conditions this is achievable. For linear finite-dimensional systems there is a very elegant necessary and sufficient algebraic-geometric condition for disturbance decoupling in terms of the system operators (A, B, E, C, D) and a certain (A, B)-invariant subspace, $\mathcal{V}(A, B)$.

Given $n \times n$ and $n \times m$ matrices A and B, we say that a subspace $\mathcal{V}$ of $\mathbb{R}^n$ is (A, B) invariant if $A\mathcal{V}(A, B) \subset \mathcal{V}(A, B) + ImB$, where $ImB = \{x \in \mathbb{R}^n : x = Bu, u \in \mathbb{R}^n\}$.

Moreover, (A, B)-invariance can be readily tested and a controller achieving the decoupling can be constructed. It is also possible to design controllers with additional properties, such as the stability of the closed-loop system. An additional pleasing aspect of this geometric theory for finite-dimensional systems is that the mathematical and system theoretic concepts also turn up in completely different control problems, i.e., they have an intrinsic system theoretic significance. While the early work on geometric theory was done in Canada and Italy, some interesting later developments had taken place in The Netherlands, and so it was natural to ask whether any of these problems also have solutions for infinite-dimensional systems. At the time, there was little on this in the literature, but it was clear that all the algebraic properties would carry over. However, there had been some counter-examples in the literature which indicated that there would be problems with the topological aspects. This proved indeed to be the case: the basic catch was that while the (A, B)-invariant subspace $\mathcal{V}$ always exists, it is not always closed, and the disturbance decoupling problem is only solvable if $\mathcal{V}$ is closed (in that case the whole finite-dimensional theory can be generalized). In spite of the lack of a nice generalization, [3] comprises a detailed analysis of the problem, an explanation of the lack of a solution and several examples which give insight into the complex situation for some typical PDE and delay equations. To this day this is the most complete account of geometric control in infinite dimensions. It seems likely that one could obtain a more elegant theory by relaxing the requirement that the output z is completely decoupled from the disturbance to the requirement that it be *almost decoupled*. This, however, remains an interesting conjecture, and a topic for future research.

Figure 2. H_∞-control theory is currently applied to reduce noise in aircraft engines. (Photo Capital Press.)

4. H_∞-CONTROL THEORY

H_∞-optimal control problems were formulated for finite-dimensional systems in the 1980's and such problems continue to attract considerable interest to this day. The term H_∞ arose because the mathematical problem involved is the minimization of the norm of a certain operator in the Hardy space H_∞. A recent application of this theory is to devise an active control mechanism to reduce the noise level created by a new class of powerful engines for aircraft (see [7] and also figure 2). An idealized version of this is the so-called *disturbance attenuation problem*: to design a feedback controller Σ_c for a system Σ as in figure 1 so that the closed-loop system is stable and the influence of the disturbance d on the output z is minimized. This can be seen as a weakened version of the disturbance decoupling problem. Indeed, this problem has a solution under fairly mild conditions and the contribution of the research in [4] was to extend the existing theory for finite-dimensional systems (established in the United States and the United Kingdom) to a class of infinite-dimensional linear systems. Unlike the disturbance decoupling problem, the disturbance attenuation problem has a complete generalization to infinite dimensions. Since the solution involves algebraic operator Riccati equations similar to those occurring in the quadratic control problem in [1], this was not altogether surprizing. What was surprizing, was the time it took before the infinite-dimensional case was solved. The class of H_∞-control problems covered by this theory is in fact wider than the disturbance attenuation problem mentioned above.

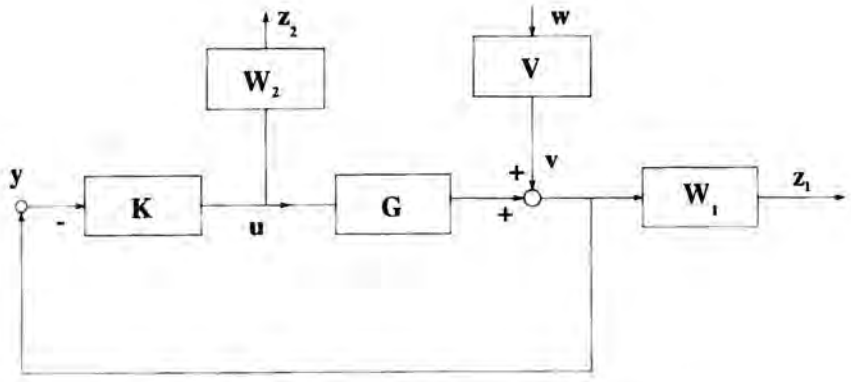


Figure 3. Weighted-mixed-sensitivity design.

Using clever algebraic manipulations, other problems can be formulated just as in figure 1, except that the system Σ is now an abstract system containing the physical system and various weighting transfer matrices, depending on the particular control problem considered. Examples of problems which can be formulated in this abstract way include designing robust controllers, (i.e., controllers which stabilize a whole family of systems) and controllers which also allow for performance objectives such as reduced sensitivity to disturbances in a given frequency band. One such popular controller design is called the *weighted-mixed-sensitivity* design (see [6]). The aim is to design a controller such that in the configuration of figure 3 influence of the disturbance signal w on the output

$$z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

is minimized. The weighting transfer matrices W_1, W_2 and V are designed to enhance the performance with respect to robustness and other characteristics of the output. This can be reformulated as a standard problem as in figure 1 with as new extended system Σ_e with transfer matrix

$$\left(\begin{array}{c|c} W_1 V & W_1 G \\ 0 & W_2 \\ \hline -V & G \end{array}\right)$$

G is the transfer matrix of the original system Σ. This is the compelling feature of the H_∞-formulation: it covers many problems simultaneously. After the publication of [4], or rather at the time of preprints of earlier results, the H_∞-control problem attracted considerable interest in the United

States, France and Rumania, the aim being to generalize the theory to an even wider class of systems. This achieved some success and research in this area is continuing to flourish. The downside is that the 'solutions' are theoretical in nature; the controllers are infinite-dimensional and not readily implementable. More desirable would be algorithms for designing rational controllers for infinite-dimensional systems which have desirable robustness and other properties. To do this one needs approximation results and although some progress in this direction has been made in [5] and [6], much more research remains to be done. Finally, we remark that the solution in [4] is a state-space approach in which the solution is given in terms of operators A, B, C, D. There exist alternative frequency-domain approaches in terms of a transfer matrix description as well. The appeal of the state-space approach is that it reveals the connection with the classical quadratic optimal control problems in [2].

References

1. A.G. Butkovskiy (1969). *Theory of Optimal Control of Distributed Parameter Systems*, American Elsevier. (Translation of the original Russian version published in 1965.)
2. J.L. Lions (1971). *Optimal Control of Systems Described by Partial Differential Equations*, Springer Verlag, Berlin. (Translation of the original French version published in 1968.)
3. H.J. Zwart (1989). *Geometric Theory for Infinite Dimensional Systems*, Lecture Notes in Control and Information Sciences, Vol 115, Springer Verlag, Berlin.
4. B.A.M. van Keulen (1993). *H_∞-Control for Distributed Parameter Systems: A State-Space Approach*, Birkhäuser, Boston.
5. J. Bontsema (1989). *Dynamic Stabilization of Large Flexible Space Structures*, Ph.D. Thesis, Groningen University, The Netherlands.
6. R.F. Curtain, Y. Zhou (1994). *A Weighted-Mixed-Sensitivity H_∞-Control Design for Irrational Transfer Matrices*, University of Groningen, The Netherlands, Report W-9416.
7. H.T. Banks, M.A. Demetriou, R.C. Smith. Robustness studies for H_∞-feedback control in a structural acoustic model with periodic excitation. To appear in the *Journal of Robust and Nonlinear Control*, R.F. Curtain, A. Stoorvogel (eds.). A Special Issue on Infinite-Dimensional Systems.

Coding Theory

J.H. van Lint

1. Introduction

Coding theory, or more specifically, the theory of error-correcting codes is younger than the Foundation Mathematical Centre. We go back to the late 1940's. In those days computers were able to recognize bit errors (due to some technical failure) and if this happened, the process would be terminated. The idea is simple. Sequences of binary symbols (0 and 1) of a fixed length n are processed. Such sequences will be referred to as *words*. The only words that were allowed to be used were those with an even number of 1's. Clearly an error (which was luckily improbable) in a single bit would cause a violation of the parity rule and the computer stopped. In sufficiently long programs this would eventually happen. As a consequence R.W. Hamming of Bell Laboratories (USA) quite often found his computer not at work when he returned to the laboratories in the morning: an error had been *detected*. His irritation over the fact that an error could be detected but not corrected, led to his construction of the *Hamming code*, a so-called single-error-correcting code. The idea of the code can be easily understood from the following simple and elegant description, due to R.J. Mc Eliece. We assume that information is presented as a long string of 0's and 1's. These are to be communicated from a 'sender' to a 'receiver' over a medium that we shall call 'the channel'. This channel has the unpleasant property that, with a (luckily small) probability p, sometimes a 0 is changed into a 1 or a 1 into a 0 on the way to the receiver. We wish to significantly

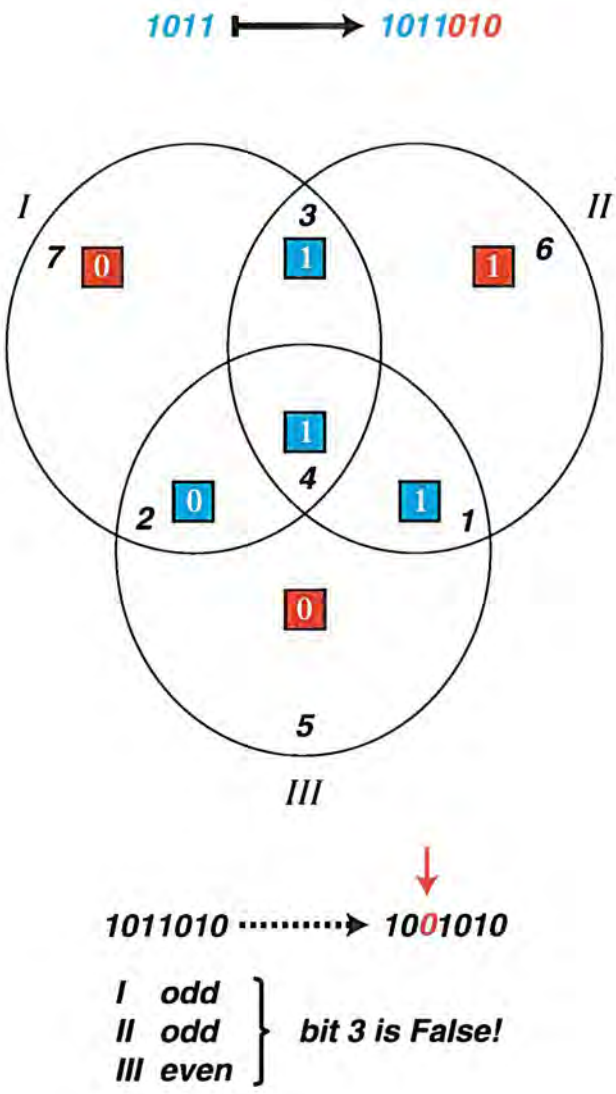


Figure 1. The Hamming code.

increase the probability of correct interpretation of the received sequence of 0's en 1's. For this we are willing to pay a toll (loss of time or energy or space, all depending on the practical application of this model). Here is what Hamming did. The sequence of 0's and 1's is divided into fourtuples, each of which is mapped into a seventuple. The seventuples are transmitted. The mapping is described in figure 1.

1.1. An example

Let (a_1, a_2, a_3, a_4) be a fourtuple. These bits are put into the positions 1,2,3,4, in the figure. From the positions 5,6,7 we find three so-called *redundant* bits by the following parity rule: each of the circles I,II,III should contain an even number of 1's. The reader will easily see that if one bit is received incorrectly, the receiver can see from figure 1 which circles violate the parity rule. This clearly identifies the position of the erroneous bit and *correction* can take place! To understand why this became an extremely exciting area in mathematics, we must analyse the '*code*' described above. This will allow us to quote the theorem that started coding theory. It is known as Shannon's Theorem, put forward in his monumental paper in 1948 (see [1]).

In the example above, each received sequence consists of four bits of *information* and three redundant bits. We say that the *information rate* R equals $4/7$. As an exercise, the reader can check that if the probability p that a bit is transmitted erroneously by the channel is say 0.001, then the probability that a fourtuple is misinterpreted by the receiver is roughly 2.10^{-5}. Clearly not using coding would imply an error probability of 4.10^{-3}. This is an impressive improvement in error probability and the toll is a decrease of an information rate from 1 to $4/7$. It may be of interest to the reader to know that, on the most impressive application of coding theory, to wit the *compact disc*, the information rate is $3/4$, i.e., one fourth of the disc does not contain music but redundancy added by coding theorists, responsible for the superb quality of the music!

The channel that we described above is known as the *binary symmetric channel*. The model assumes that a bit-error is a random event with a given probability p. For such a channel, we define the *capacity* C as $C = 1 + p \log p + (1 - p) \log(1 - p)$. (Logarithms to base 2.) In our example, we have $C \approx 0.99$.

Shannon's theorem states that for a binary symmetric channel with capacity C and for any $\varepsilon > 0$ and any $R < C$, there exists a code with rate at least R and error probability (after decoding) less than ε. This sounds unbelievable. What one should realize is that the codes of this theorem are *extremely long*.

1.2. Applications

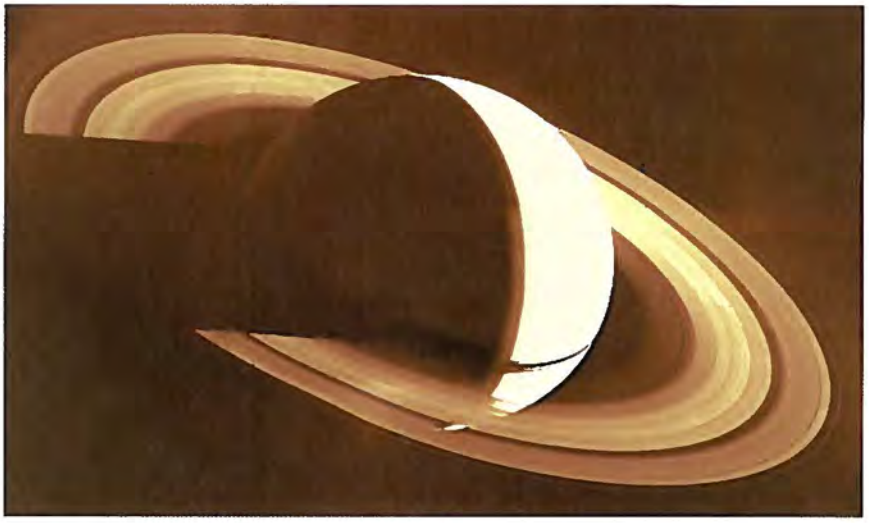
Where has coding theory gone? We mention some of the important areas of application (see also figure 2). As mentioned in the introduction, computing is an important area of application. Since most modern communication is digital, error-correction plays a role there. Spectacular applications were the photographs taken by several satellite missions (Mars, Saturn, Jupiter). Without coding theory there would not have been pictures at all. In recent times the CD is the most notable application. In these examples, the channels are quite different. For telephone, it is light in optical fibre, for satellites radio communication (where the source of errors is thermal noise in the amplifier at the receiver), for CD the errors are caused by dust, air bubbles in the disc, scratches, finger prints, etc. Practical requirements can differ considerably: processing the signals from Mariner Mars took one day; the CD-player has a delay (for decoding) of a fraction af a second. Obviously, a lot of energy has gone into finding good (i.e., fast) decoding algorithms.

1.3. Parameters

To understand some of the mathematical developments, we need some parameters. We use n for the length of the code. The alphabet is not necessarily $\{0,1\}$. We use q for the size of the alphabet. In algebraic coding theory, q is a prime power and the alphabet is a finite field. (For CD we have $q = 2^8$.) Up to now, we have not mentioned the most important parameter of a code, to wit its *distance*. The distance of two words is the number of places where they differ (e.g., 101011 and 100010 have distance 2, since they differ in positions three and six). The *minimum distance* d of a code C is the minimum value of the distance between two distinct codewords. Obviously for a code with minimum distance $d = 2e + 1$, it is theoretically possible to correct up to e errors. The most important problem in combinatorial Coding Theory is to establish bounds for the number $A_q(n,d)$, the *maximal* number of codewords in a code C of length n over an alphabet of size q, and with minimum distance d. For example $A_2(7,3) = 16$ and the Hamming code described above realizes this bound. Clearly, for practical applications one has tried to find relatively good codes of moderate length, i.e., codes with given n, q, and d, for which $|C|$ is close to $A_q(n,d)$. Quite often, mathematical results are not appealing to engineers because the construction of the code gives no hint as to how the receiver can decode (quickly).

1.4. Covering codes

We now turn to so-called *covering codes*. This area of research is very important for error-correcting codes themselves but is also, in some sense, complementary. Some of the applications concern *data-compression* (important for high definition television). Here the problem is that we have



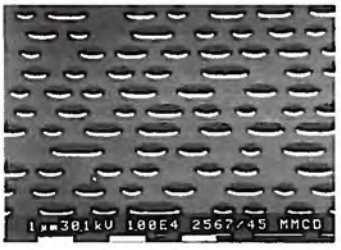


Figure 2. Coding theory is applied in several areas. Spectacular examples include: Voyager pictures of Saturn (above) and CD-players (below, high density pit structure shown on the right; courtesy Philips NV Eindhoven).

too much information to transmit and that we are willing to accept small deviations in order to save time or space. We illustrate this using our earlier example of the Hamming code. We first introduce some geometric terminology. If $\mathbf{c}$ is a codeword in a code C, then the *ball* of radius r around $\mathbf{c}$ consists of all words $\mathbf{x}$, such that $d(\mathbf{x}, \mathbf{c}) \leq r$. Note that if C has minimum distance $d = 2e + 1$, then the balls of radius e around the codewords are disjoint. There can be many words $\mathbf{x}$ that are not in any of these balls. In practice of error-correction this is important. A received word that is within a ball of radius e around a codeword $\mathbf{c}$ is decoded as $\mathbf{c}$. If $\mathbf{x}$ is received and $\mathbf{x}$ is not in any of these balls, the receiver knows that too many errors have occurred for him to handle the situation.

Back to data compression. We have to send information consisting of seven bits. If the receiver makes one error, this is close enough for him to interpret our meaning. (In TV this is achieved by the redundancy of the overall picture.) For each seventuple we consider the ball in the Hamming code to which it belongs, determine the centre and transmit only the bits a_1, a_2, a_3, a_4 corresponding to that centre. The receiver is never more than one bit off (for the seventuple) and we have only 4/7 of the amount of information to be transmitted, a considerable gain. This leads to the definition of the *covering radius* of a code C (a subset of all words of length n over an alphabet of size q). The covering radius is the smallest number ∂ such that the balls of radius ∂ around the codewords of C cover all words of length n. For the Hamming code we have $\partial = 1$. This code is called *perfect* because the balls of radius 1 around codewords are not only disjoint but they also cover the space. In practice, perfect codes are far from perfect: the occurrence of too many errors should in general not escape the receiver. There has been far less research on covering codes than on error-correcting codes.

1.5. Coding theory and algebraic geometry

For a long time, coding theory was an area for electrical engineers. Many of their interesting achievements later turned out to be equivalent to mathematical methods and results that had been known for a long time. It was not until the 1970's that mathematicians became interested in coding theory (influenced partly by activities in The Netherlands). Especially the relations between coding theory and design theory (an area with its origin in statistics and quality control) led to a surge of interest. We will go into this below. Algebraic methods became increasingly important (especially through the work of Ph. Delsarte). A peculiar connection to simple groups pulled in another area, so that by the 1980's coding theory was respectable for group theorists, algebraists and of course combinatorialists. And then algebraic geometry appeared on the scene of coding theory, 'bien étonné de

se trouver ensemble'! It is extremely difficult to explain the connection but let's try.

Everyone is familiar with the fact that the rational numbers are a subfield of the reals. Everyone is familiar with the curve S known as the unit circle: $\{S = (x, y) : x^2 + y^2 = 1\}$. The point $(3/5, 4/5)$ on S has rational coordinates but in general the coordinates on S will be irrational. We are interested in codes over an alphabet F_q (the finite field with q elements). This field is a subfield of its algebraic closure F (which is an infinite field). In algebraic geometry one studies curves (defined using coordinates in F), given by an algebraic equation (like the circle S above). One of the important problems is to determine the 'rational' points on S. Here, rational means that the coordinates are in the subfield F_q.

We now describe the link to coding theory. Let S be an algebraic curve over F with n rational points $P_1, P_2, \ldots, P_n$. Consider a suitably chosen set $\mathcal{F}$ of rational functions defined on S. The code C is defined as the collection of words $(f(P_1), f(P_2), \ldots, f(P_n))$ of length n obtained by letting f run through $\mathcal{F}$. If we know enough about the curve S, it is possible to make (interesting) assertions about the code C, i.e., about its minimum distance. Using some deep results from algebraic geometry, M.A. Tsfasman, S.G. Vlădut, and Th. Zink in 1982 proved the existence of codes that are far better than anything that was believed possible until then. Clearly a sensational development. This result has led to quite a lot of research in which The Netherlands has made significant contributions. Three directions can be mentioned. First, studying special classes of curves to see if reasonably good codes can be found. Second, finding decoding algorithms, preferably good enough to get engineers interested. (Notice that the problem has reversed.) Recently, there has been progress in translating the results from algebraic geometry into terminology that avoids the deep mathematics but produces nearly the same results [2]. This is an exciting area that has just been opened.

2. Coding theory at CWI

Research on coding theory at CWI started in 1972. This marked the beginning of the strong collaboration with the Discrete Mathematics group at the Eindhoven University of Technology (TUE, which has lasted). Important contributions from the early years are several results on bounds on codes due to M.R. Best and A.E. Brouwer. In 1974 CWI organized the Advanced Study Institute on Combinatorics at Nijenrode Castle. It is still considered one of the major events in this area of the past 25 years! At the meeting Ph. Delsarte (invited speaker) presented his theory of the association schemes of coding theory. It has had a very strong influence on the research at both CWI and TUE. One of the open problems mentioned in the lecture of J.H. van Lint on perfect codes was solved by M.R. Best in his Ph.D. thesis

(1982). In 1975 a course on coding theory was held at the Mathematical Centre (CWI's name before 1983). This led to MC Syllabus 31 which was the basis for a book on Coding Theory [3].

3. Ph.D. projects (NWO/SMC) performed at TUE

One part of the Ph.D. thesis of H.J. Tiersma (1989) was concerned with codes from algebraic geometry, namely those based on Hermitean curves. The main part of the thesis concerns constructions and bounds for codes for channels that differ from the binary symmetric channel described above. We do not go into details but the idea is that more than one user uses the channel and information goes both ways, or it is added, etc.

G.J.M. van Wee (Ph.D. thesis 1991) also dedicated some of his time to algebraic geometry codes. The question that is answered is 'Which linear codes are algebraic-geometric?' (joint work with R. Pellikaan and B.Z. Shen). It turns out that in the class of codes known as Hamming codes only those with at most two redundant symbols can be constructed with an algebraic curve, with the exception of our example of figure 1.

The main part of the thesis concerns covering codes. One result deserves special mention. It is an elementary but ingenious counting argument that produces a lower bound on the number of words in a code of given length and covering radius. It is now known as the Van Wee bound. For this Ph.D. thesis Dr. van Wee was awarded the prestigious 'Dissertationspreis' of the Gesellschaft für Mathematik, Ökonomie und Operations Research in 1992.

In the meantime the algebraic-geometry code research group at the TUE, supervised by R. Pellikaan, had obtained international recognition. This led to several visits from researchers from abroad. An important further step in this process was the Ph.D. thesis by I.M. Duursma (1993) on Decoding Codes from Algebraic Curves. Mathematically speaking, the problem is solved but for practical use it is essential that far more efficient methods are found. This research (often jointly done with visitors) has led to sufficient insight in these codes to be able to describe them in a more elementary way, thus opening the door to practical use. The decoding methods also led to new decoding techniques for cyclic codes. This extremely successful SMC-project led to 12 publications and 25 lectures abroad by Dr. Duursma!

In 1994 two SMC-projects resulted in a Ph.D. thesis. Feng-Wen Sun constructed decoding techniques and a modulation scheme for band-limited communications. The channels concerned differ considerably from those above. Either the signal is continuous and noise is Gaussian noise and not discrete, or the errors are not discrete but weighted in some way. Despite this different approach to signalling there is strong interplay with traditional coding theory in the thesis.

R. Struik extended the work of Van Wee on covering codes. In fact, he gave an improvement of the Van Wee bound. This led to several new

records. Furthermore the thesis analyses the codes with covering radius 2 or 3 and presents several new constructions for covering codes.

Both for the area of covering codes and for codes from algebraic geometry it is clear that the results mentioned above have made it possible to formulate many new interesting projects. It is therefore quite desirable that this sequence of SMC-projects is continued in the future.

References

1. C.E. Shannon (1948). A mathematical theory of communication. *Bell Syst. Tech. J. 27*, 379-423 and 623-656.
2. T. Hoholdt, J.H. van Lint, R. Pellikaan (1996). Algebraic geometric codes. R.A. Brualdi, W.C. Huffman, V. Pless (eds.). *Chapter of Handbook of Coding Theory*, Elsevier Science Publishers, Amsterdam, to appear.
3. J.H. van Lint (1982). *Introduction to Coding Theory*, Springer-Verlag.

Analysis on Lie Groups

G. van Dijk

1. Introduction

The project combines two fields: analysis and the theory of Lie groups, and thus leads to a challenging enterprise. It is also a hard enterprise since ample experience in both fields is required to be successful in research. The project comprises the following closely related topics:

- Harmonic analysis and representation theory
- Special functions related to root systems
- Special chapters in functional analysis and applications to Gelfand pairs.

Below we shall briefly describe the historical development of the topics, explain its basic techniques, and discuss recent developments. We also outline the work carried out in this project in The Netherlands, which was mainly sponsored by SMC.

2. Harmonic analysis and representation theory

A main theme is the analysis of functions on a Lie group G or, more generally, on a space M on which G acts homogeneously. A good example is the Lorentz group acting on the forward light cone. In particular one is interested in the decomposition of the space $L^2(M)$ of square integrable functions on M as a sum or integral of irreducible subspaces: the so-called Plancherel decomposition. In the classical cases, where M is the real line or the circle, with its group of translations, this amounts to Fourier analysis.

If M is the sphere with its group of rotations acting on it, one obtains the decomposition in spherical harmonics.

It is a general phenomenon that G-invariant operators such as invariant differential operators leave the decomposition of $L^2(M)$ into irreducible components invariant and act by scalar multiplication on each of the components. In order to gain insight in the Plancherel decomposition of $L^2(M)$ into irreducibles, one studies

1. the fine structure of representations of G related to M, and
2. eigenfunctions and eigendistributions related to such representations.

The Plancherel decomposition has been completely determined for any real semisimple Lie group G and the associated Riemannian symmetric space, by the work of Harish-Chandra. It has not yet been determined for the interesting class of pseudo-Riemannian symmetric spaces, leaving many challenging problems to be solved. A well-known example of a pseudo-Riemannian space is the hyperboloid of one sheet in $\mathbf{R}^n$ (see figure 1), while the two sheeted hyperboloid is a standard example of a Riemannian symmetric space. In recent years important progress has been made:

(a) For pseudo-Riemannian symmetric spaces of rank one the decomposition has been obtained explicitly by the work of V.F. Molchanov, J. Faraut, and G. van Dijk and his Ph.D. students; for a rank two space an explicit decomposition was obtained by N. Bopp. S. Sano and P. Harinck were successful in the group-like case $G_{\mathbf{C}}/G_{\mathbf{R}}$.

(b) For spaces of general rank the classification of the discrete series has been achieved in the work of M. Flensted-Jensen and Oshima-Matsuki. Later G. Olafsson and B. Orsted made a deep study of the spaces which admit a so-called holomorphic discrete series.

(c) In a very interesting, not yet completely published paper, E.P. van den Ban and H. Schlichtkrull have determined a 'partial' Plancherel theorem for the most continuous part of the spectrum.

2.1. The role of differential equations

The study of eigenfunctions and eigendistributions, which is such an important tool in harmonic analysis as described above, has its own independent interest. It provides examples of systems of partial differential equations for which one can obtain much more detailed information about the solutions than in general. Such examples include asymptotic and convergent expansions, integral formulae, meromorphic extensions and analysis of the singularities are among the phenomena which can be understood by combining general principles of the theory of differential equations (such as ellipticity, holonomicity and regular singularities) with the additional information provided by the group actions. We refer to the work of Van

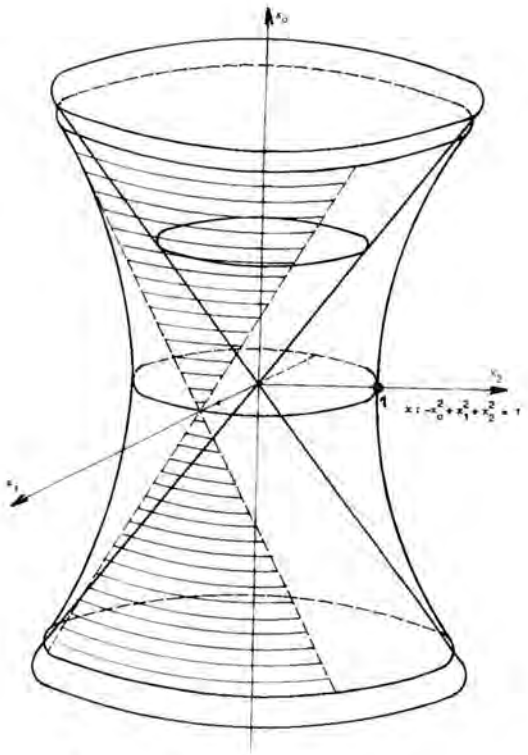


Figure 1. A pseudo-Riemannian space.

den Ban-Schlichtkrull on boundary values and Helgason's conjecture, and to the work of Kolk-Varadarajan on Lorentz invariant distributions on the light cone. Also the work on rank one spaces, mentioned above, provides a good example of the strength of the role of differential equations in harmonic analysis.

3. Special functions related to root systems

The theory of special functions is very closely related to the representation theory of Lie groups: most special functions appear as matrix coefficients of representations of special Lie groups. This connection was already known in the previous century, when the various special functions were introduced, for instance by C.F. Gauss. However, it was H. Weyl who emphasized this connection more clearly in his basic work on representations of semisimple Lie groups. It turns out to be fruitful in two ways: on the one hand, results from special function theory, analytically derived without knowledge of group theory, are needed to answer some of the questions of harmonic

analysis on Lie groups; on the other hand, the group theoretic interpretation of the special functions suggests results, which would probably never have been discovered without this interpretation.

Fairly recent is the development of a special function theory in several variables, as suggested by the theory of special functions on symmetric spaces of higher rank. This research was initiated by T.H. Koornwinder in his Ph.D. thesis, by working out some rank two examples. It is remarkable that this theory admits a perfect generalization of the one variable case. The differential equation, the Rodrigues formula, the beta-function integral of Euler, all have a perfect analogue, which can be calculated explicitly. In particular the multivariable beta-function integral had attracted some attention before, by the work of A. Selberg, F. Dyson, M.L. Mehta and by the Macdonald conjectures. These conjectures were solved because of the work of G.J. Heckman and E.M. Opdam, who used the multivariable hypergeometric function theory associated with root systems. This is a general feature: the role of the Lie group is taken by the root system of its Lie algebra. So the group structure is lost but some connection remains, namely by means of the root system. This leads naturally to a much more algebraic and geometric theory of special functions. In figure 2 we give, only as an illustration of the complexity of root systems, a partial table of its Satake diagrams, which contains detailed information for the theory.

Many questions remain open, of which the most natural ones are: is there a good spectral theory (suggested by Harish-Chandra's Plancherel formula for symmetric spaces), and what about q-analogues (as started by Macdonald)? The second question is at the moment a quickly developing subject, because of the connection with *quantum groups*. We refer to work by M. Noumi, Koornwinder and H.T. Koelink. As to the first question, recently a new tool was developed in the study of special functions associated with root systems by C.F. Dunkl, by the introduction of his so-called Dunkl operators. The original philosophy to study equivalents of the radial parts of Laplace operators (by regarding the root multiplicities as parameters), has the disavantage that in general more than one Laplace operator exists whose form is unknown. Dunkl's operators have explicit expressions, commute and are related to Laplace operators of Cartan motion groups. Later I. Cherednik has adapted these operators for the group case as well. This new impulse has provided answers to question one. We refer to work of Opdam, Heckman and M.F.E. De Jeu. Dunkl's operators have nowadays a worldwide interest.

4. Special chapters in functional analysis and applications to Gelfand pairs

The decomposition of Hilbert spaces as integrals of irreducible component spaces involves aspects of direct integral theory, that can be clarified by

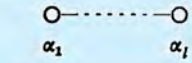
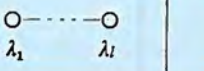
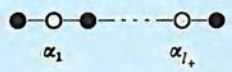
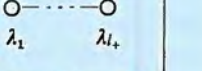
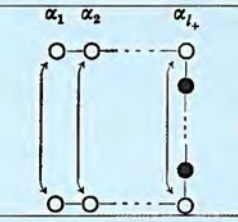
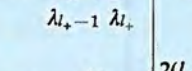
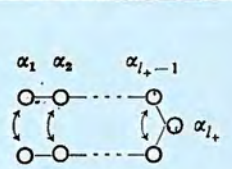
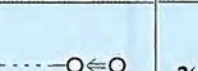
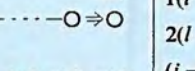
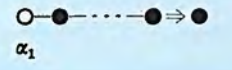
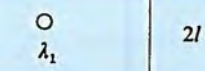
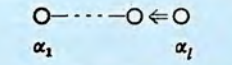
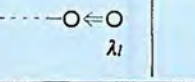
Type	$\Phi \longleftrightarrow \Psi$	$\Sigma \longleftrightarrow \Upsilon$	$m(\lambda_i)$	$m(2\lambda_i)$
AI			1	0
AII		 $(l = 2l_+ + 1, l_+ \geqslant 1)$	4	0
AIII		 $\left(2 \leqslant l_+ \leqslant \frac{l}{2}\right)$	$2(i < l_+)$ $2(l - 2l_+ + 1)$ $(i = l_+)$	0 1
		 $(l = 2l_+ - 1, l_+ \geqslant 2)$	$2(i \leqslant l_+ - 1)$ $1(i = l_+)$	0 0
AIV			$2(l - 1)$	1
BI		 $(l \geqslant 2, 2 \leqslant l_+ \leqslant l)$	$1(i < l_+)$ $2(l - l_+) + 1$ $(i = l_+)$	0 0
BII			$2l - 1$	0
CI			1	0

Figure 2. Satake diagrams of root systems [5].

establishing the link with the integral representation theory of G. Choquet and certain generalizations of it, due to E.G.F. Thomas. The pair (G, H) is said to be a Gelfand pair if the cone of H-invariant distributions of positive type on G is simplicial. These and other characterizations are relevant for the concrete determination of Gelfand pairs. Gelfand pairs are named after I.M. Gelfand and have initially only been studied in the case of compact H. A classical example of a Gelfand pair is a pair (G, H) such that G/H is a Riemannian symmetric space, e.g., $G = \mathrm{SL}(n, \mathbf{R})$, $H = \mathrm{SO}(n, \mathbf{R})$. These pairs are well studied and lead to a very beautiful theory, which is mainly due to Harish-Chandra and S. Helgason. A more general situation, involving finite-dimensional representations of H has been studied by H. van der Ven. In the general situation where H is noncompact, a nice result has been obtained by Van Dijk, studying rank one pseudo-Riemannian pairs. These pairs are Gelfand pairs, with the exception of the pair $(\mathrm{SO}_0(1, n), \mathrm{SO}_0(1, n-1))$. The Plancherel formula for the space associated with this pair has multiplicity two in the continuous spectrum.

5. Concluding remarks

One of the most interesting new lines of research of the last ten years in the field on Lie groups is without any doubt the second item: special functions related to root systems. Several new topics of research come to The Netherlands from other countries, mostly from the United States. However, this topic is to a great extent really Dutch, with pioneering work by Koornwinder, Heckman and Opdam. This does not happen very often. It is something to be proud of.

References

1. E.P. van den Ban, H. Schlichtkrull (1993). Multiplicities in the Plancherel decomposition for a semisimple symmetric space. *Contemp. Math. 145*, 163-180.
2. G. van Dijk (1994). Group representations on spaces of distributions. *Russian J. Math. Physics 2(1)*, 57-68.
3. G.J. Heckman, E.M. Opdam (1987). Root systems and hypergeometric functions I. *Compositio Math. 64*, 329-352.
4. T.H. Koornwinder (1993). Askey-Wilson polynomials as zonal spherical functions on the SU(2) quantum group. *SIAM J. Math. Anal. 24*, 795-813.
5. G. Warner (1972). Harmonic analysis on semisimple Lie groups I. Springer-Verlag, p. 30.

Functional Analysis and Optimization Problems in Hydrodynamic Propulsion

J.A. Sparenberg, P. Sijtsma, H.P. Urbach

1. Introduction

Hydrodynamic propulsion is of interest in the biological sciences for the study of swimming creatures, but it is also important in technics, already since ships came into use. We direct here our attention to the propulsion of ships as it is studied at (technical) universities, at ship research institutes and sometimes at shipyards and screw factories. The research described below was inspired by the desire in the shipbuilding industry in the late 1970's to diminish propulsion costs in view of rising energy prices. In general the theoretical research is directed to the solution of practical problems. Analysis in the form of 'classical' applied mathematics in combination with extensive computer programs is employed for the application of lifting surface theories to propellers. By a propeller we mean not only the well-known screw propeller, but also periodically moving wings which cause a thrust. An example of the latter is the Voith-Schneider propeller.

Most propellers are placed at the stern of a ship for the following reason. By its slight viscosity the water flowing along the hull of the ship is dragged with it and obtains kinetic energy with respect to the water at large distance. It can be shown that by placing the propeller at the stern of the ship, part of this kinetic energy can be regained by which the efficiency of the propeller increases.

Figure 1. Four-bladed screw propeller with endplates. (Photo: Groningen Propeller Technology B.V., The Netherlands.)

We now mention two difficulties for the calculation of the performance of the propeller which are caused by its above mentioned efficiency increasing position at the stern. First, at the stern of a ship the flow of the water is 'untidy'. The water dragged with the hull becomes turbulent and because
130 it has to follow the shape of the hull, it has to converge at the stern. Besides this the wave pattern at the free surface above the propeller causes a velocity field which varies with depth. Second, the stern forms partly a rigid boundary of the flow domain and hampers the water to be set into motion by the propeller. The same holds for the rudder and also the free surface acts as part of the boundary of the region in which the propeller operates.

Besides the foregoing ones, another type of difficulty can occur, perhaps more specifically with respect to the screw propeller, namely when the propeller is heavily loaded. Then the interaction between vortices (which causes the nonlinear roll-up of the shed vorticity) becomes important and also time-dependent cavitation can be present at the blades.

These are not ideal circumstances for the application of elegant mathematics in order to describe the performance of the propeller. For that

sake we have to make simplifications. However, in that case we have to be careful with the application of the results to reality. A rather accessible situation occurs when we suppose that the propeller acts in an incompressible, inviscid, unbounded and otherwise undisturbed fluid and translates with constant velocity and delivers a prescribed thrust. Often it is also assumed that this thrust is sufficiently small so that a (semi-)linearized theory can be used in which squares of velocities induced by the shed vorticity can be neglected with respect to these velocities themselves.

Hence, one of the simplifications is the neglect of viscosity of the fluid. However, loss of efficiency of the propeller caused by viscosity can be very important, especially with respect to optimization problems. Luckily we can introduce in an inviscid optimization theory experimentally or theoretically obtained results of the viscous resistance of plates, by which the viscosity can often be incorporated satisfactorily.

In technically useful optimization calculations it is, for instance with respect to the screw propeller, not always necessary to use functional analytic methods, because by experience built up in the course of time it is known that optimum screw propellers do exist under certain simple constraints. However, there are propeller types for which we are not sure that an optimum propeller does exist within a 'set' of admitted propellers. This can happen rather easily with propellers consisting of periodically moving thrust producing wings, of which we shall discuss two examples. Both examples are two-dimensional, because besides the mentioned simplifications it is also assumed that the wings are infinitely long. First, the small amplitude motion of a thrust producing flat profile and, second, the large amplitude motion of a lifting line (i.e., a line on which the forces are assumed to be concentrated) for which there is an unequality constraint on its lateral force action. It is clear that both models are highly idealized versions of a practically possible propeller. Nevertheless it is very elucidating to understand their working in the simplified case.

2. Optimization of small amplitude motions of a flat profile

We shall discuss small amplitude motions of a rigid profile through a previously undisturbed fluid [1]. With respect to the Cartesian coordinate system (x, y) shown in figure 2 the motion is given by

$$y = h(x,t) = \frac{\ell}{2}a(t) + \alpha(t)(x - Ut), \quad -\ell \leq x \leq \ell, \tag{2.1}$$

where 2ℓ is the length of the profile, $\frac{\ell}{2}a(t)$ and $\alpha(t)$ are the so-called heaving and pitching parts of the motion, and U is the constant velocity of the profile in the positive x-direction.

Let $T(h)$ be the mean thrust generated by the periodic motion h with period τ_0 and let $E(h)$ be the mean increase of kinetic energy of the fluid

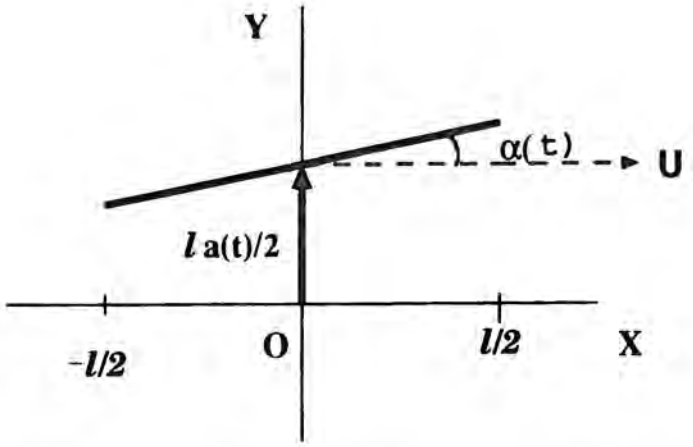


Figure 2. Flat profile of length 2ℓ moving through a previously undisturbed fluid at velocity U.

during one period. The efficiency of the motion h is useful work divided by total work, hence

$$\eta(h) = \frac{UT(h)}{UT(h) + E(h)}. \tag{2.2}$$

The aim of the optimization is to minimize the lost energy E subject to the constraint that a prescribed mean thrust $\overline{T}$ is generated and furthermore subject to some additional constraints, e.g., on the amplitude of the motion. The thrust is obtained by summing the integrated x-component of the pressure jump across the profile and the suction force at the leading edge. The suction force always acts as a positive thrust. The relative contribution of the suction force to the total thrust will be constrained. The reason is that in certain cases it can be shown that without this constraint optimum motions do not exist. The constraint on the suction is also useful from the mechanical point of view, because large suction forces cause the separation of flow from the profile.

Furthermore, from the engineering point of view it is desirable to constrain the amplitude of a point of the profile. Hence the optimization problem that we shall consider is for given $\overline{T} > 0$, $r > 0$ and $C_\searrow > 0$,

$$\begin{aligned} &\underset{h \in \mathcal{H}}{\text{minimize}}\ E(h), \ \text{subject to}\ T(h) = \overline{T},\ T^s(h) \le r\overline{T}, \\ &\max_t |h(x_p, t)| \le C_\searrow, \end{aligned} \tag{2.3}$$

where $\mathcal{H}$ is the function space in which the optimum motion is sought, $T^s(h)$ is the mean suction force, and x_p is the x-coordinate of the point whose amplitude is constrained.

We assume that the motions have small amplitudes so that a linearized theory can be applied. In this theory all flow quantities can be explicitly expressed in terms of a and α, more precisely in terms of their Fourier coefficients. By writing

$$\begin{aligned} a(t) &= \sum_{n=-\infty}^{\infty} \tilde{a}(n) \exp(2\pi i n t/\tau_0), \\ \alpha(t) &= \sum_{n=-\infty}^{\infty} \tilde{\alpha}(n) \exp(2\pi i n t/\tau_0), \end{aligned} \tag{2.4}$$

we find for example

$$E(h) = \sum_{n=-\infty}^{\infty} (\tilde{a}(n), \tilde{\alpha}(n))\, \mathcal{E}(n\sigma_0) \begin{pmatrix} \tilde{a}(n) \\ \tilde{\alpha}(n) \end{pmatrix}^*, \tag{2.5}$$

where * denotes complex conjugation, $\sigma_0 = 2\pi l/(\tau_0 U)$ and $\mathcal{E}(\sigma)$ is for all $\sigma \neq 0$ a nonnegative selfadjoint $(2,2)$–matrix, and therefore E is a convex quadratic functional. Analogous expressions hold for $T(h)$ and $T^s(h)$ with selfadjoint matrices $\mathcal{T}(\sigma)$ and $\mathcal{T}^s(\sigma)$ instead of $\mathcal{E}(\sigma)$. $\mathcal{T}^s(\sigma)$ has one positive and one vanishing eigenvalue, whereas $\mathcal{T}(\sigma)$ has one positive and one negative eigenvalue, in agreement with the fact that the suction is always nonnegative, whereas the total thrust can be negative as well as positive. Hence T^s is convex, whereas T is not.

In order to prevent unessential constraints on the smoothness of the motions, one should choose for $\mathcal{H}$ the largest function space for which all functionals occurring in the optimization problem are well-defined and norm-continuous. Because the nonzero eigenvalues of the matrices $\mathcal{E}(\sigma)$ and $\mathcal{T}^s(\sigma)$ are $\sim \sigma^2$ for $\sigma \to \pm\infty$, this means that we require a, and α to be in the Sobolev space $H^1_{\tau_0}$ defined by

$$\begin{aligned} H^1_{\tau_0} = \{ f \in L^2_{loc}(\mathbf{R});\ \ & f' \in L^2_{loc}(\mathbf{R}), \quad \text{and} \\ & f(t+\tau_0) = f(t) \text{ for all } t \}. \end{aligned} \tag{2.6}$$

With the standard scalar product $H^1_{\tau_0}$ is a Hilbert space. It is well known that the functional $h \mapsto \max |h(x_p, t)|$ is continuous with respect to the weak topology of this space.

In studying the existence of optimum motions it is natural to attempt to apply the general theorem which says that a lower semi-continuous (l.s.c.) functional attains its infimum on a compact set. Now the constraint set in

(2.3) is bounded and norm-closed, but not norm-compact. One would like to choose a smaller topology on the space $H^1_{\tau_0}$, such that the set is compact and such that E is l.s.c. Because E is convex and norm-continuous, it is l.s.c. with respect to the weak topology on $H^1_{\tau_0}$, and this weak topology is in practice the smallest topology for which E has this property. Nevertheless, the set in (2.3) is not compact in this topology either. As is often the case in infinite-dimensional optimization problems of hydrodynamic propulsion, the trouble is caused by the equality constraint on the mean thrust.

In spite of the fact that the set in (2.3) is not compact in the weak topology, it is possible to prove the existence of an optimum motion. The idea of the proof is based on the important observation that the difference $G(h)$ between the useful work and the lost energy:

$$G(h) = UT(h) - E(h) \tag{2.7}$$

can be shown to be weakly continuous. Problem (2.3) is equivalent to

$$\begin{aligned} &\text{maximize } G(h), \text{ subject to } T(h) = \overline{T},\ T^s(h) \leq r\overline{T}. \\ &a, \alpha \in H^1_{\tau_0} \\ &\max_t |h(x_p, t)| \leq C_\infty. \end{aligned} \tag{2.8}$$

In addition to (2.8) we introduce the optimization problem obtained by replacing the equality constraint on the generated thrust by an inequality:

$$\begin{aligned} &\text{maximize } G(h), \text{ subject to } T(h) \leq \overline{T},\ T^s(h) \leq r\overline{T}. \\ &a, \alpha \in H^1_{\tau_0} \\ &\max_t |h(x_p, t)| \leq C_\infty. \end{aligned} \tag{2.9}$$

This problem is not necessarily equivalent to (2.8), because it could have a solution with $T(h) < \overline{T}$. It follows from $UT(h) = G(h) + E(h)$ and from the mentioned properties of G and E that T is l.s.c. with respect to the weak topology. Because T^s is convex and norm-continuous, it has this property also. Therefore, the set in (2.9) is weakly closed and since it can be shown to be bounded also, it is weakly compact. We conclude therefore that problem (2.9) has at least one solution. Furthermore, one can prove that at least one solution satisfies $T(h) = \overline{T}$ and therefore is also a solution of problem (2.8). Because problems (2.3) and (2.8) are equivalent, we conclude that optimization problem (2.3) has indeed at least one solution.

When the required mean thrust is smaller than a treshold value $\overline{T}_1$, the solution is a pure harmonic with lowest frequency $1/\tau_0$. When the required mean thrust is larger than $\overline{T}_1$, the constraint on the amplitude of the motion becomes active and both the heaving and the pitching components consist of infinitely many nonvanishing harmonics. The amplitude constraint is active for two intervals of time per period during which the point x_p is at rest at

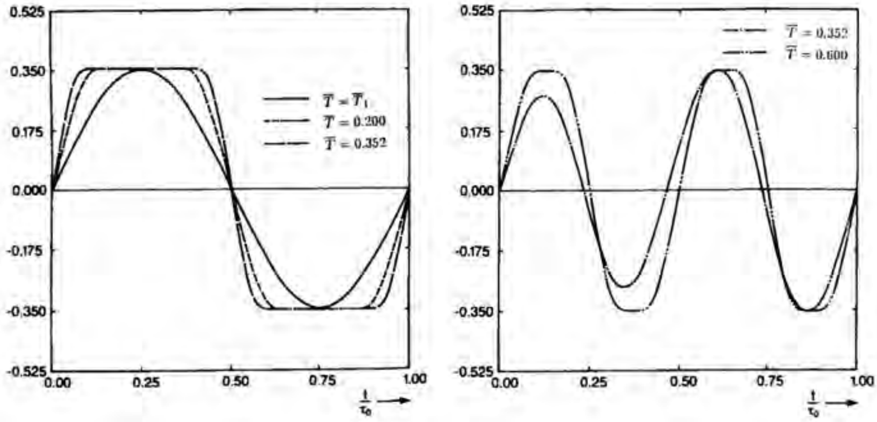


Figure 3. The motion of the quarter-chord point as function of time for optimum motions corresponding to $\sigma_0 = \pi/3$, $x_p = 0.5l$, $r = 0.4$, $C_\infty = 0.35l$ and for four values of the required mean thrust: $\overline{T} = \overline{T}_1 = 0.066$, $\overline{T} = 0.200$, $\overline{T} = 0.352$ and $\overline{T} = 0.600$. All optimum motions shown in the left figure have dominant lowest harmonic, whereas those shown at the right have dominant second harmonic.

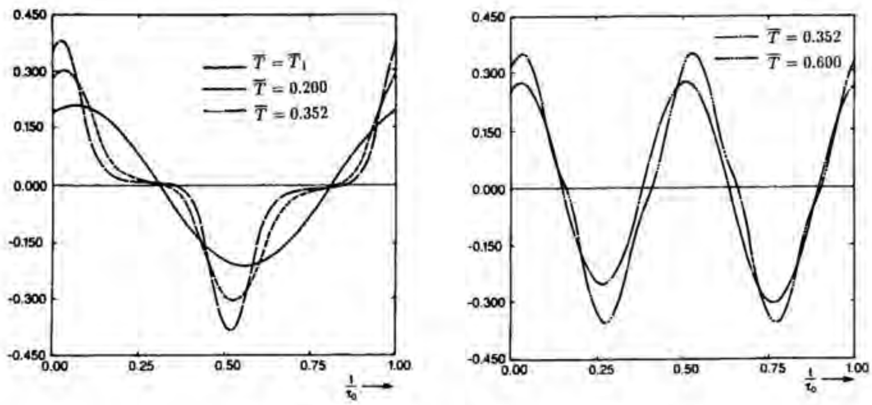


Figure 4. The pitching $\alpha(t)$ of the optimum motions of figure 3.

the maximum stroke while the profile slowly pitches around it. For $r = 0.4$, $x_p = 0.5l$ (quarter-chord point), $C'_\infty = 0.3525l$ and for several values of the thrust, $h(x_p, t)$ and the pitching $\alpha(t)$ of the optimum motion are shown in figures 3 and 4. For $\overline{T}_1 \leq \overline{T} < 0.3525$ the lowest harmonic is dominant. For $\overline{T} = 0.3525$, a second optimum exists having dominant second harmonic, and for larger required thrust the second harmonic remains dominant until the third takes over at a certain higher treshold value. All computed optimum motions generate the maximum allowed mean suction force $r\overline{T}$, even when r is much larger than 1.

Several properties of the optimum motions can be derived from a Lagrange multiplier rule that is obtained by the application of a Kuhn-Tucker type of theorem (which provides a method to solve certain minimization problems with inequality constraints). However, the classical Kuhn-Tucker Theorem cannot be used because the functional $h \mapsto \max|h(x_p, t)|$ is not Gateaux differentiable. When this type of constraint occurs, the theory of generalized differentials has to be applied.

3. Optimum large amplitude sculling propulsion with an inequality constraint on the side force

Next, we consider the large amplitude motion of a lifting line [2], which can represent a one-wing sculling propeller, mounted vertically at the stern of a ship (see figure 5). The sculling wing W moves sideways back and forth, while its angle is adjusted such that a thrust is created. From the hydrodynamical point of view, a large lateral amplitude is profitable, since this can result in a high efficiency.

The lifting line or concentrated bound vortex $\Gamma(t)$, which represents this sculling wing, moves through the water along a line G (see figure 6). The strength of the vortex varies with time, corresponding with the blade angle variation of the wing. Since the bound vorticity $\Gamma(t)$ varies with time, free vorticity is shed into the water. In other words, the fluid behind the lifting line is put into motion and its kinetic energy increases with time. It is clear that the lost kinetic energy should be kept as small as possible.

By the motion of the lifting line a 'lift' force is evoked, acting perpendicularly to the local direction of motion, hence normal to G. By Joukowski's law, the magnitude of this force is proportional to the product of the vortex strength and the velocity of the lifting line. The lift force can be decomposed into two components: a thrust component T in the direction of motion of the ship and a side force component S perpendicular to it. The mean value of the thrust should be equal to the ship's hull drag at a desired speed. The most efficient propulsor is the one that produces the least kinetic energy under the constraint of a prescribed mean thrust.

The fluctuating side force is an evident drawback of a one-wing sculling propeller. It can have a disturbing influence on the course of the ship.

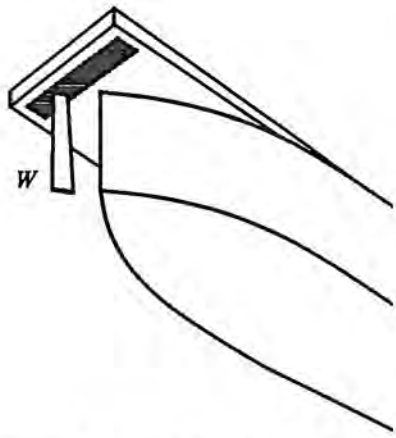


Figure 5. Stern of a ship, equipped with a one-wing sculling propeller W.

Therefore, it is studied what the effect is on the optimum motion if a constraint is put on this lateral force. To be more precise, a maximum is put on its absolute value. The optimization problem we consider is to find an optimum time-dependent bound vorticity of the lifting line for which the kinetic energy generated per period of time is minimal, under the constraints of a thrust with a prescribed mean value and a side force with a maximum value. Contrary to the previous section, the path G is chosen in advance.

The problem that we are confronted with consists of two parts. First, it has to be proved that an optimum bound vorticity *exists*. Second, the optimum bound vorticity has to be *constructed*. In the following, an outline is given of the procedure that is followed to solve these problems and which role is played by functional analysis. Since the mathematical implications are rather complex, a two-dimensional model is used. This means that the lifting line is assumed to be infinitely long, having a constant strength in spanwise direction. Furthermore, a linearized theory of an inviscid fluid is adopted, implying that the shed free vorticity keeps its strength and remains on the place where it is formed, that is on G.

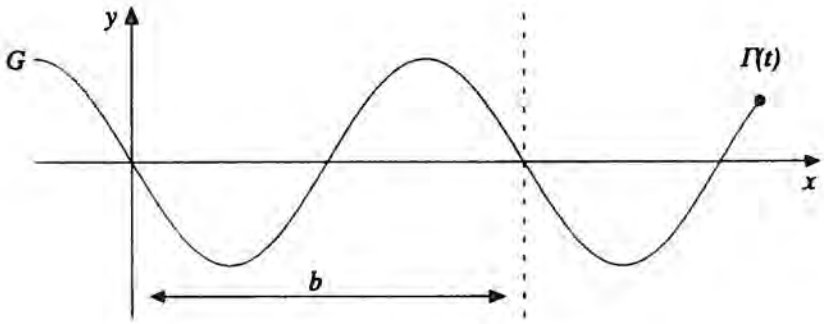


Figure 6. The motion of the lifting line $G(t)$.

The first step, before we can prove the existence of an optimum motion, is to state the optimization problem unambiguously in a proper mathematical sense. For that purpose, let us consider a Cartesian reference frame x, y, in which the positive x-direction is defined by the direction of motion of the ship (which is assumed to move in a straight line).

Now let us say that we find ourselves at a certain position x, y. The ship passed by a very long time ago and disappeared behind the horizon ($x = \infty$), so we do not experience the unsteady motion of the lifting line. On the other hand, the ship started its motion somewhere behind the opposite horizon ($x = -\infty$), so starting effects can be neglected as well. What remains is a steady flow induced by the free vorticity on G. During one time period of the wing motion, an amount of kinetic energy is added to the fluid which is equal to the kinetic energy in the strip:

$$\Omega = \{(x, y) \in \mathbb{R}^2 : 0 \leq x \leq b\}, \tag{3.1}$$

where b is the distance covered by the ship during one period.

Returning to our steady state model, the only vorticity that is present in the fluid is on G. So, outside G a velocity potential ϕ exists. The kinetic energy, produced per time period can then be expressed as:

$$E(\phi) \int_{\Omega} \int \| \nabla \phi \|^2 dx dy. \tag{3.2}$$

The potential ϕ is not continuous over G. The free vortices on G induce a jump $[\phi]$, which is such that the derivative of $[\phi]$ along G is equal to the strength of the free vorticity. We can choose the potential ϕ such that the jump $[\phi]$ at a certain location on G is equal to the vortex strength $\Gamma(t)$ of the lifting line at the time it passed by.

Since the location of G is prescribed, it is allowed to treat $[\phi]$ as a function of x only. The constraint of the prescribed mean thrust can be expressed as a weighted integral of $[\phi]$, symbolically written as:

$$T([\phi]) = T. \tag{3.3}$$

The inequality constraint on the side force can be expressed, directly in terms of $[\phi]$, as:

$$-\bar{S} \leq [\phi](x) \leq \bar{S}, \text{ for every } x. \tag{3.4}$$

Summarized, our aim is to find a potential ϕ, satisfying (3.3) and (3.4), for which the lost energy $E(\phi)$, (3.2), is as small as possible.

To define the optimization problem well in the mathematical sense, we have to identify a convenient function space for ϕ, in which the optimization can be carried out. This space has to be such that (3.2), (3.3) and (3.4) are well-defined. An appropriate candidate for this is the Sobolev space $H^1(\Omega)$.

consisting of square-summable functions on Ω, of which the derivatives are square-summable too. In this space, (3.2) is automatically well-defined.

At the upper and lower side of G, boundary values of $\phi \in H^1(\Omega)$ exist in the sense of the so-called 'trace'. These boundary values are elements of the Sobolev space $H^{1/2}(0,b)$, which is defined using derivatives of non-integer order. Since $H^{1/2}(0,b)$ is contained in the space of square-summable functions $L_2(0,b) = H^0(0,b)$, it is clear that (3.3) is also well-defined. The side force constraint (3.4) has to be slightly weakened as:

$$-\bar{S} \leq [\phi](x) \leq \bar{S}, \text{ for } \textit{almost} \text{ every } x. \tag{3.5}$$

This is because Sobolev spaces, in fact, consist of *equivalence classes* of functions.

So, we can formulate the problem as the minimization of $E(\phi)$, (3.2), on the set

$$P = \{\phi \in H^1(\Omega);\ \phi \text{ satisfies (3.2) and (3.4)}\}. \tag{3.6}$$

To prove the existence of a solution we use the well-known fact that a closed, convex subset of a Hilbert space possesses a unique element which minimizes the norm. Indeed, P is a closed, convex subset of the Hilbert space $H^1(\Omega)$. Here we benefit from the fact that the path G is fixed, in contrast with the previous section, where the counterpart of P is *not* convex.

The energy $E(\phi)$ is in general not equivalent with the H^1-norm. However, if we equip $H^1(\Omega)$ with some evident symmetry and periodicity properties, then $E(\phi)$ can be proved to be equivalent with the usual norm on $H^1(\Omega)$. By this strategy, the existence of an optimum motion of the lifting line is proved. Moreover, it follows that there is *only one* optimum motion.

It is noted that, up to now, we have completely ignored the incompressibility of the fluid. It turns out, however, that this omission is not essential. By disturbing the optimum potential ϕ_0 with test functions which are continuous over G, it is seen that $\Delta\phi_0 = 0$ (Δ is the Laplace operator), which means that the velocity field of the optimum potential is free of divergence. Moreover, it follows that the normal velocity is continuous across G, as it should be. So, although we admitted divergence, the solution of the optimization problem is *free* of divergence.

One of the constraints, namely (3.5), is defined on an infinite set. Consequently, one of the Lagrange multipliers is not a scalar, but a function. Then, it is not evident that (a generalized version of) the Kuhn-Tucker theorem is applicable. The classical generalized Kuhn-Tucker theorem requires the set P, in which the optimization is carried out (cf. (3.6)), to be a so-called 'positive cone' (a semi-infinite set). Furthermore, for P the 'regularity condition' should hold, which means that this cone must have a non-empty interior. It is indeed possible to reformulate the optimization problem, such

that P is a positive cone. However, the set of positive functions of $H^{1/2}(\mathbb{R})$, the space in which (3.5) is defined, has *no* interior points.

To overcome this difficulty, an other version of the Kuhn-Tucker theorem, without regularity condition, would be useful. However, in the open literature some variants were found. However, in these theorems the regularity condition was replaced by other, complicated conditions, of which the validity could not be proved in our situation. Therefore, a new Kuhn-Tucker theorem was developed [3] with less complicated requirements, which appeared to hold in our optimization problem.

Herewith, the optimization problem is solved. In other words, in our simple, two-dimensional model, a unique optimum motion *exists* of a sculling wing (represented by a lifting line) with prescribed mean thrust and bounded side force. Moreover, we are able to *construct* this optimum motion.

References

1. H.P. Urbach (1986). *A Functional Analytic Approach to Some Problems of Hydrodynamic Propulsion*. Ph.D. thesis, Groningen University, The Netherlands.
2. P. Slitsma (1992). *On the Hydrodynamics of Optimum Sculling Propulsion of Ships and on the Linearized Lifting Surface Theory*. Ph. D. thesis, Groningen University, Netherlands.
3. R. van der Meer (1988). *Gegeneraliseerde Stelling van Kuhn-Tucker*. Master thesis (in Dutch), supervisor E.G.F. Thomas, Groningen University, The Netherlands.

Singularity Theory

T. de Jong, J.H.M. Steenbrink

1. Introduction

The existence of a 'Dutch Singularity School' was first noticed by N.H. Kuiper at the *Colloque sur la monodromie* in Metz in February 1974. E.J.N. Looijenga, D. Siersma and J.H.M. Steenbrink, who were to defend their Ph.D. theses in Amsterdam that year, were present at that meeting and their work was a topic of the discussion. Five years later, this triple started the ZWO-project *Singularity Theory*, after J. Seidel invited them to set up a common activity of larger scale than usual at ZWO (predecessor of the present National Research Council NWO). Under this flag, W.A.M. Janssen, G.R. Pellikaan, D. van Straten and T. de Jong completed their Ph.D. theses and several others, such as J. Stevens and H.J.M. Sterk, were strongly influenced by its activities.

In this article we will focus on one characteristic aspect of the project: the contributions of Pellikaan, Van Straten and De Jong to the deformation and classification theory of singularities.

2. Singularities

2.1. Introduction

In the context of this article, the subject of singularity theory is the local study of complex analytic sets. Let U be an open subset of $\mathbb{C}^n$ and let $f_1, \ldots, f_k$ be holomorphic functions on U. Then the set of common zeroes

of $f_1, \ldots, f_k$ is called an analytic subset of U, and every analytic set is a union of such subsets. The interest in such sets arose when people realized that analytic sets can be quite rich from the topological point of view. Consider one holomorphic function f in complex variables z_1, z_2. Suppose that $f(0,0) = 0$. If one of the partial derivatives of f at $(0,0)$ is non-zero, then there exists a holomorphic function g such that (f, g) is a holomorphic coordinate system at $(0,0)$ and, hence, the analytic set $\{f = 0\}$ is similar to a linear subspace at $(0,0)$. However, if $df(0,0) = 0$, then the situation is quite different. The richness of the local topological structure can be seen by intersecting the set $\{f = 0\}$ with a small sphere centered at $(0,0)$; the result is an algebraic knot or link inside the three-dimensional sphere. In higher dimension one can construct exotic spheres in this way.

A *germ of an analytic set* at 0 in $\mathbb{C}^n$ is an equivalence class of zero-sets $\mathcal{V}(f_1, \ldots, f_k)$ defined in some open neighbourhood U of $0 \in \mathbb{C}^n$. It is therefore represented as

$$\{(a_1, \ldots, a_n) \in U : f_1(a_1, \ldots, a_n) = \ldots = f_k(a_1, \ldots, a_n) = 0\}.$$

Two such sets are called equivalent if their intersections with a sufficiently small neighbourhood of 0 agree.

A more subtle notion is that of *germ of an analytic space*, also called *singularity*. Here one does not consider the zero set alone, but also the functions which define this set. For example, the space defined by the equation $x^2 = 0$ in $\mathbb{C}$ (a 'thick' point of multiplicity two) is considered to be different from a regular point (defined by $x = 0$). Equivalently, a germ of an analytic space can be defined by its 'ring of holomorphic functions', which is a quotient of the convergent power series ring $\mathbb{C}\{x_1, \ldots, x_n\}$. An important example is the singular locus of a hypersurface singularity $f(x) = 0$, which is defined as an analytic space germ by the equations $f(x) = \partial_1 f(x) = \ldots = \partial_n f(x) = 0$. Any analytic space has an underlying analytic set. Conversely, for any germ of an analytic set there is an associated analytic space, defined by the ideal of all functions vanishing on this analytic set. Such analytic spaces are called *reduced*.

2.2. Some terminology

The simplest case is where all the defining equations can be taken to be linear and vanishing in the origin. In this case the singularity we get we call *smooth* (its singular locus is empty). Of course, in singularity theory this case is hardly interesting. The simplest singularity which is not smooth is the A_1 singularity, given by a non-degenerate quadratic equation, for example $x_1^2 + \ldots + x_n^2 = 0$. The case $n = 1$ we considered above (the fat point). Below we give real pictures for the A_1 curve singularity $xy = 0$ (see figure 1), and the A_1 surface singularity given by $xy - z^2 = 0$ (see figure 3).

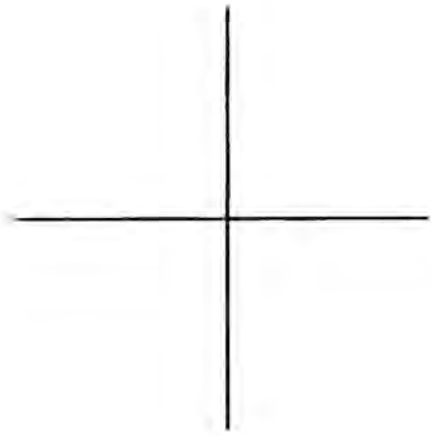


Figure 1.

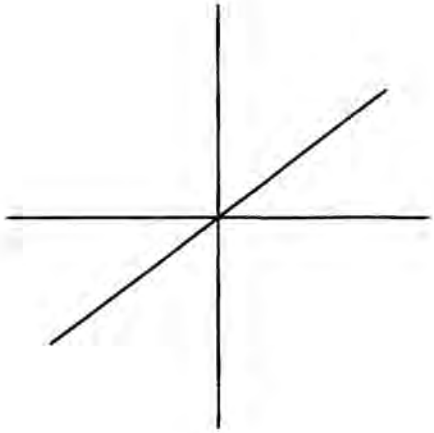


Figure 2.

The above singularities are examples of *hypersurface* singularities, singularities which can be given by one equation. Hypersurface singularities in turn are examples of *complete intersection* singularities: here the number of functions needed to describe the singularity is equal to the *codimension* of the singularity. One of the simplest examples of a singularity which is not a complete intersection is the union of the coordinate-axes in three-space (see figure 2). Here one needs *three* equations: $xy = xz = yz = 0$, whereas the codimension is two.

Even if one is just interested in smooth spaces, it might be interesting,

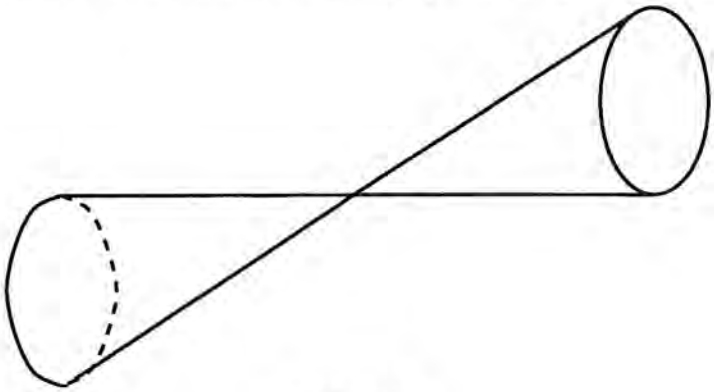


Figure 3.

handy and even 'necessary' to study spaces with singularities. For example, classically smooth curves (Riemann surfaces) were studied by taking a model in the projective plane, which always exists by the theorem of the primitive element. The big advantage of studying plane models is of course that they can be given by just *one* equation. The price one has to pay is that the plane model in general must have singularities. This can be seen for instance by the genus formula. For a *smooth* plane curve with genus g and degree d one has the relation

$$g = \frac{(d-1)(d-2)}{2}$$

from which it follows that a curve of genus two does not have a smooth plane model. Similarly, surfaces were studied by taking a model in $\mathbb{P}^3$. Here one even has to allow non-isolated singularities, i.e., the set of points where the surface is not smooth is a curve itself.

Singularities also occur naturally in the study of so-called *minimal models* of smooth algebraic varieties. Minimal models are known to exist for curves and surfaces for a long time. It was discovered by Mori and Reid, that for a good notion of minimal models for higher dimensional varieties, one has to allow *singularities* on the minimal model. Another interesting motivation is the study of exotic spheres. Exotic spheres are differential manifolds homeomorphic but not diffeomorphic to the standard sphere. Interesting examples of these appear as links of singularities. (The link of an analytic set is the intersection of a suitable representative of this analytic set with a small sphere.) For example, the link of the singularity defined by the equation

$$x_1^5 + x_2^3 + x_3^2 + x_4^2 + x_5^2 = 0$$

is an exotic sphere of dimension seven.

3. Deformations of singularities

One way to study singularities started off with the book of J. Milnor [2]. He considered hypersurface singularities, defined by a holomorphic function f. Take the ball B_ϵ with center 0 and radius ϵ in $\mathbb{C}^n$ and a disc D_η with center 0 and radius η in $\mathbb{C}$, such that $0 < \eta << \epsilon << 1$. One of the main results of Milnor is that the map

$$f : B_\epsilon \cap f^{-1}(D_\eta \backslash \{0\}) \to D_\eta \backslash \{0\}$$

is a C^∞ fibration. The 'general fibre' $f = t$ for $t \in D_\eta \backslash \{0\}$ is called the Milnor fibre. In case we have an *isolated* singularity, the Milnor fibre is homotopy equivalent to a finite wedge of spheres of dimension $n-1$: the number of those spheres is called the Milnor number of the singularity.

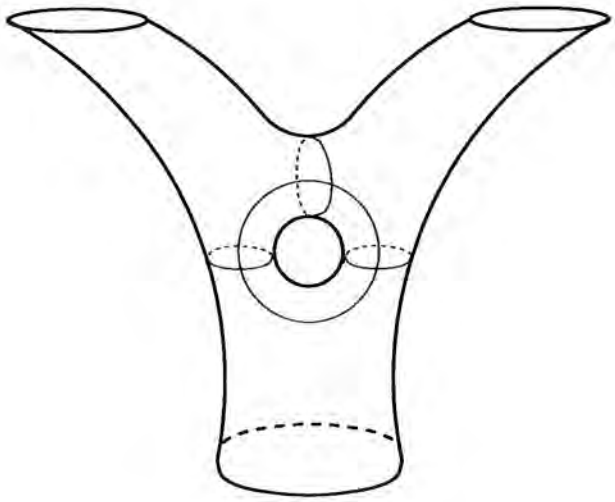


Figure 4.

There is a simple formula for computing the Milnor number μ, as the $\mathbb{C}$-dimension of the algebra $\mathbb{C}\{x_1, \ldots, x_n\}/(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$.

Let us consider the example of the D_4-singularity $x^3 - y^3 = 0$. One can take here even $\epsilon = \infty$ and $\eta = 2$, so the Milnor fibre can be given by $x^3 - y^3 = 1$. Therefore, the Milnor fibre is an elliptic curve with the three points at infinity removed. Topologically the Milnor fibre is as in figure 4.

We see that one can retract the Milnor fibre on the four drawn circles. If one now makes a graph with vertices corresponding to these circles and edges corresponding to their points of intersection, one obtains the Dynkin diagram of type D_4!

One can ask in general whether for a given isolated singularity X there exists a flat one-parameter family $X_T \to T$ (T is a small disc in $\mathbb{C}$) such that for $t = 0$ one has the original singularity X and for $t \neq 0$ the fibre is *smooth*. Here 'flat' is a technical notion (t, a parameter on T, is to be a non-zero divisor on the space X_T; this insures that much information of the zero-fibre can be read off the general fibre. It implies for instance that each component of X_T maps surjectively to the parameter space T, explaining the word 'flat'). For complete intersection singularities, every small perturbation of the functions defining the singularity gives a flat one-

parameter deformation, but for non-complete intersections the situation is much more complicated. For instance, the space defined by the equations

$$xy - t = xz - t = yz - t = 0$$

is *not* a flat one-parameter deformation of the coordinate axes in $\mathbb{C}^3$. In general it is not clear that one can give non-trivial (i.e., not isomorphic to a product) deformations of a singularity at all! Indeed there exist examples of singularities which are rigid, i.e., admit only trivial deformations. But there exist also examples of singularities, the easiest one being the cone over a rational curve in $\mathbb{P}^1$, due to H. Pinkham which admit two one-parameter deformations for which the general fibres are smooth, but not homeomorphic!

4. Classification of singularities

One can try to classify singularities up to holomorphic coordinate changes. This goal is too ambitious in general, but a beginning of the classification of hypersurface singularities was made by V.I. Arnol'd, R. Thom, Mather and Siersma in the early 1970's. The list starts with the simple singularities:

$$A_k : x_1^{k+1} + x_2^2 + \ldots + x_n^2 = 0$$

$$D_k : x_1^2 x_2 + x_2^{k-1} + x_3^2 + \ldots + x_n^2 = 0; \qquad k \geq 4$$

$$E_6 : x_1^4 + x_2^3 + x_3^2 + \ldots + x_n^2 = 0$$

$$E_7 : x_1 x_2^3 + x_1^3 + x_3^2 + \ldots + x_n^2 = 0$$

$$E_8 : x_1^5 + x_2^3 + x_3^2 + \ldots + x_n^2 = 0.$$

Here 'simple' is a technical term, meaning more or less that the singularity can only deform in a finite number of isomorphism classes of other singularities. The labels A, D, E come from the Dynkin diagrams of simple Lie groups. In fact, there exist at least 15 different characterisations of these simple singularities.

A peculiar aspect of Arnold's classification is that singularities tend to appear in series. We quote Arnol'd [1]: 'Although the series undoubtedly exist, it is not at all clear what a series of singularities is'. And: 'It is only clear that the series are associated with singularities of infinite multiplicity' (non-isolated singularities). Indeed, in the example of A_k and D_k singularities one can formally put $k = \infty$ to get the non-isolated singularities of figure 5.

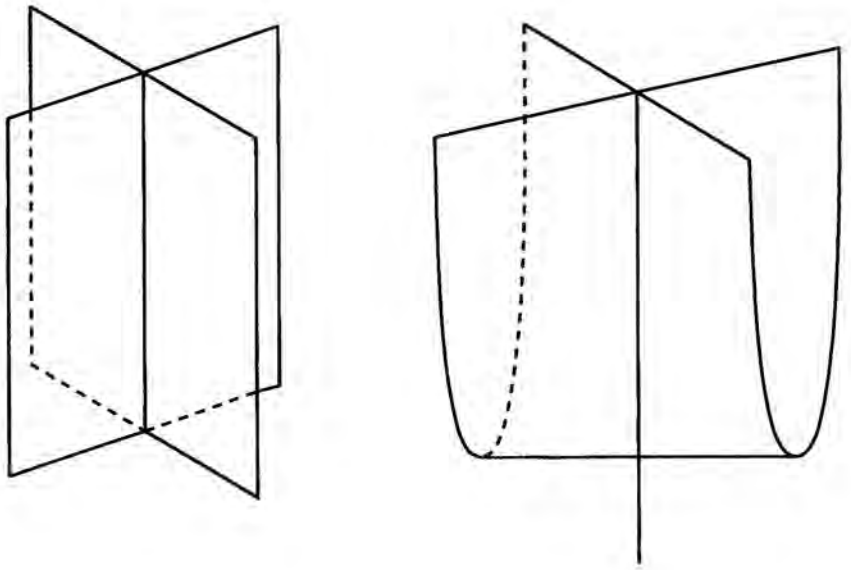


Figure 5.

5. The project Singularity Theory

5.1. Hypersurface singularities

In this section we discuss some of the results obtained in the SMC-project Singularity Theory. Inspired by the remarks of Arnol'd, Siersma and later his student Pellikaan started to study the simplest types of non-isolated singularities: hypersurface singularities with one-dimensional singular locus and transverse type A_1. Transverse type A_1 means that if one takes a transverse slice at the general point of the singular locus the intersection is an isolated A_1 singularity. For example, in figure 6 the first singularity has transverse type A_1 whereas the second has not.

One of the goals of Siersma and Pellikaan was to understand the topology of the Milnor fibre of such singularities. In the case of an isolated singularity, this can be done by deforming the defining function to a Morse function, i.e., a function with only singularities of type A_1. The number of Morse points appearing is just the Milnor number. This 'morsification method' was generalized to the case that the reduced singular locus is a complete intersection, and led to so-called admissible deformations. Loosely speaking,

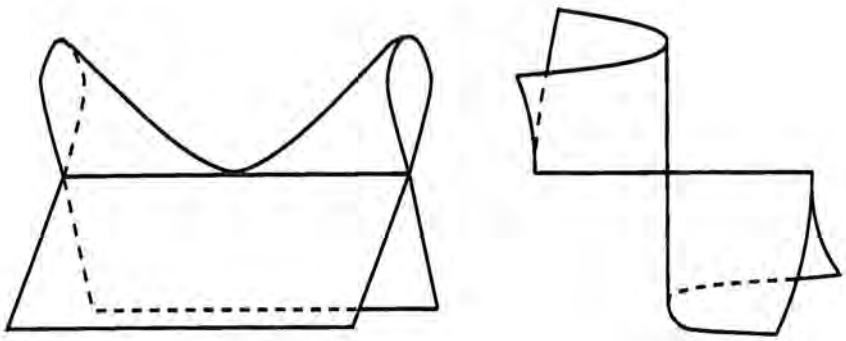


Figure 6.

these are deformations of the pair (f, Σ) where Σ is the singular locus of f.

An example of a deformation which is not admissible is the deformation of A_∞ to A_k, given by the equation $yz - tx^{k+1} = 0$. For the special fibre the singular locus is a line, but for the general fibre the singular locus is just one point. Therefore this does not induce a flat deformation of the singular locus, and the deformation is not admissible. The deformation suggested by figure 7 however, is admissible.

Using these admissible deformations Siersma and Pellikaan proved a theorem on the homotopy type of the Milnor fibre for hypersurface singularities whose singular locus is a complete intersection with transverse type A_1. Except for some special cases, the homotopy type turns out to be a wedge of spheres (as in the isolated singularity case), and a formula for the number

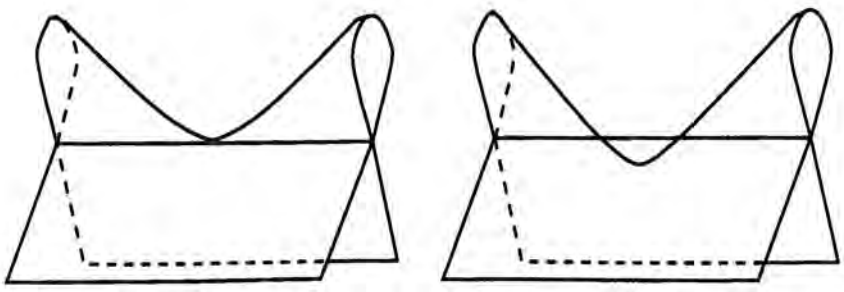


Figure 7.

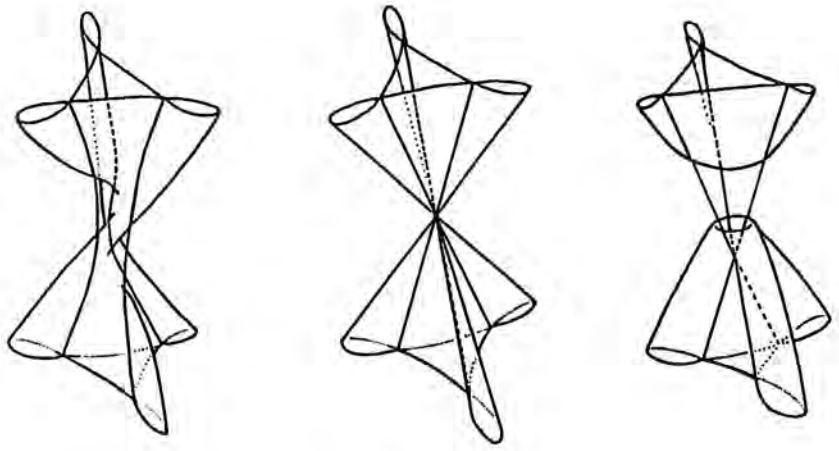


Figure 8.

of those was given by De Jong. Van Straten proved similar formulas using differential forms. Later on De Jong extended these results to certain cases of hypersurface singularities whose singular locus is a line and whose transverse type is a simple isolated singularity.

5.2. Weakly normal surface singularities

In his Ph.D. thesis, Van Straten studied weakly normal surfaces. Important examples of these are surfaces which are obtained as generic projections of smooth (or even normal) surfaces in $\mathbb{C}^3$. Such a surface has a singular locus which is a curve with isolated singular points itself. The structure of the surface near these special points is investigated by 'improving' them, i.e., replacing them by certain curve configurations, analogous to the process of resolution of isolated singularities by blowing-up. Van Straten generalized many results from the theory of normal surface singularities to this class of non-isolated surface singularities, and complemented in this way the knowledge obtained by Pellikaan. Also this approach led to a rich treasure of examples, obtained from a rough classification of weakly normal surface singularities by the structure of their improvements.

5.3. Admissible deformations

Pellikaan gave the following very interesting example of an admissible deformation: consider the hypersurface singularity given by the equation $(xy)^2 + (yz)^2 + (zx)^2 = 0$: the cone over a quartic curve in the complex projective plane with three A_1 singularities (see figure 8). This hypersurface

singularity is a projection of the cone over the rational normal curve of degree 4. Pellikaan wrote down explicitely two essentially different admissible deformations, as in figure 8. It turned out that they correspond exactly to the two different deformations of the cone over the rational normal curve of degree 4 discovered by Pinkham! Note that the surface in Pellikaan's example is a generic projection of the cone over the rational normal curve of degree 4.

Inspired by Pellikaan's example, De Jong and Van Straten started to develop the following program: given a normal surface singularity, project it to $\mathbb{C}^3$ to obtain a weakly normal surface, given by an equation $f(x, y, z) = 0$ and with singular locus Σ. Try to determine which deformations of the projected surface are obtained as projections of deformations. Surprisingly, the deformations they found were precisely the *admissible deformations* of (f, Σ), which were introduced by Siersma and Pellikaan. The advantage of course is that one needs just one equation to describe the projection, the disadvantage being having to allow non-isolated singularities. This *projection method* has been very fruitful: the base space of a semi-universal deformation of rational quadruple points could be determined, in spite of the fact that equations for these singularities have never been written down. Using the projection method one also sees that in series of singularities (which we still do not know what they are) deformation theory behaves well, i.e., for two members of a series, it is easy to compare the deformations of one with the other. Further applications of this projection method are still being discovered.

References

1. V.I. Arnol'd (1981). *Singularity Theory. Selected Papers.* Lecture Note Ser. 53, Cambridge Univ. Press, Cambridge.
2. J. Milnor (1968). *Singular Points of Complex Hypersurfaces.* Annals of Math. Studies 61, Princeton Univ. Press.

The Moduli Project, 1981-1988

G.B.M. van der Geer, F. Oort, C.A.M. Peters

1. Some early history

1.1. Introduction

The term 'moduli' was introduced by B. Riemann in 1857:

> '... *es hängt also eine Klasse von Systemen gleichverzweigter* $2p+1$*-fach zusammenhangender Funktionen und die zu ihr gehörende Klassen algebraischer Gleichungen von* $3p-3$ *stetig veränderlichen Grössen ab, welche die Moduln dieser Klasse genannt werden sollen.*'
> B. Riemann—*Theorie der Abel'schen Funktionen*, Journ. reine angew. Math. (Crelle), 54 (1857), pp. 115-155. (see p. 134).

In mathematics the word 'moduli' has various meanings. In our context, however, it only occurs in plural and refers to the essential parameters on which certain algebraic structures depend (usually in a continuous way). Used in this sense the term stems from Riemann, who introduced it in his study of Riemann surfaces. He described these as coverings of the (Riemann) sphere and proved that the number of essential parameters equals $3g-3$ for a Riemann surface of genus $g \geq 2$ (equivalently, for an algebraic curve of that genus). The genus g is a discrete invariant which only assumes

non-negative integral values, whereas the (complex) $3g-3$ parameters vary continuously. The word 'moduli' indicates the (number of) parameters on which a geometric structure like a Riemann surface depends. Frequently these 'moduli' themselves satisfy algebraic equations and, hence, can be identified with the points of an algebraic variety, called the 'moduli space'.

Elliptic curves form an example. An elliptic curve E is characterized—up to isomorphism over an algebraically closed field—by the invariant $j(E)$ and hence corresponds with a point on the affine line $\mathbb{A}^1$. The set of (isomorphism classes of) elliptic curves corresponds with the affine line. Generalization of this to elliptic curves endowed with an additional structure, such as a point of a certain order on the curve, leads to 'modular curves'. The geometry and number theory of modular curves were extensively studied in the beginning of this century by F. Klein and others.

The notion of elliptic curve can be generalized by considering algebraic curves of higher genus ($g > 1$) and by considering group varieties of higher dimension ($g > 1$). This in turn leads to generalizations of the modular curves mentioned above: the moduli space $\mathcal{M}_g$ of algebraic curves of genus g and the moduli space $\mathcal{A}_g$ of abelian varieties of dimension g (with a polarization). Both generalizations and both types of moduli space are in the center of present-day mathematics. However, the study of these spaces was (and is) far from easy and the theory was developed only with great difficulty. Here O. Teichmüller's work on the moduli of algebraic curves was important.

1.2. History in a nutshell

A full historic overview of the period 1860-1960 would take far more space than is available here. Apart from some intermediate results, in 1960—remarkably enough—still no satisfactory algebraic-geometric theory of moduli spaces was developed. The reason is that it is not only important to know that the parameters or moduli satisfy certain equations, but as much that these equations are universal. However, tackling that problem—and even formulating it properly—required the revolutionary conceptual apparatus, called the theory of *schemes*, introduced by A. Grothendieck in the 1960's in algebraic geometry.

His fundamental work, building on results by A. Weil, O. Zariski, J.-P. Serre and many others, allows a unified treatment of complex geometry and number theory: in short, algebraic geometry in all its aspects. Grothendieck introduces the concept of 'representable functor' and shows that the fundamental problem of moduli is to determine whether certain functors are representable. From this viewpoint all properties of moduli spaces acquire a 'modular' interpretation, which enables in principle a far better understanding of this class of varieties than of other ones.

Characteristic for this transition period is J.-I. Igusa's work [2] on the

moduli (over $\mathbb{Z}$) of genus 2 curves and the way it was received in the 'French' world of mathematics. P. Samuel starts his Séminaire Bourbaki lecture as follows:

> '*Signalons aussitôt que le travail d'Igusa ne résoud pas, pour les courbes de genre 2, le 'problème des modules' tel qu'il a été posé par Grothendieck à diverses reprises dans ce Séminaire.*' P. Samuel—*Invariants arithmétiques des courbes de genre 2.*' Sém. Bourbaki 14 (1961/62), Exp. 226, Décembre 1961.

D. Mumford takes up Grothendieck's challenge in trying to construct the moduli spaces of curves and abelian varieties (in the sense of Grothendieck) [1]. For many years he is the innovator and great stimulator for the development of the basics of moduli and succeeds in drawing many researchers into this field. It is fair to say that Mumford learned us, among other things, to handle Grothendieck's new, formidable conceptual apparatus.

Following this fundamental work, including the compactification of $\mathcal{M}_g$ jointly with P. Deligne, harvesting started with the theorem of Harris and Mumford (1982), stating that for g sufficiently large the moduli spaces $\mathcal{M}_g$ are 'of general type'. It was the first major success. The concept of moduli spaces has spread ever since over large parts of present-day mathematics and has fully proved its value.

Evidence for this is abundant. E. Witten showed in the mid-1980's that the moduli spaces of curves are of fundamental importance in theoretical physics. Here the Riemann surface appears as a 'dressed-up' version of the Feynman diagram in physics. Interestingly, physical intuition has led to the amazing 'Witten conjectures' concerning the cohomology of the moduli spaces of curves (proved by Kontsevich a few years afterwards). Moduli spaces are also central in the recent proof of Fermat's Last Theorem by A. Wiles.

Present research also extends to other moduli spaces than those of curves and abelian varieties. In particular we are witnessing an explosive growth of research on moduli of vector bundles.

Looking back we must admit that at that time none of us foresaw such a spectacular development and success for the theory of moduli.

1.3. Intercity Seminar

This seminar has its origins in a seminar started around 1958 under the name 'schovenclub'. This owed its existence to the inspiring personality of N.H. Kuiper who felt that Dutch mathematicians should become acquainted with the concept of 'sheaf'. Thus a platform for cooperation was established

that has shown a great vitality up to the present day. The seminar's permanent aim was the discussion of new developments, usually in geometry and algebra, and later including number theory. During the period 1958-1981 there was no fixed structure and subjects varied. For example, starting in the late 1970's E.J.N. Looijenga, C.A.M. Peters and J.H.M. Steenbrink initiated joint activities in the area of complex geometry and singularities, and at the instigation of F. Oort the arithmetic aspect of geometry was also emphasized. Often new and unpublished results were presented, which enabled young researchers to familiarize with new aspects, sometimes long before publication in international journals. This included difficult and deep results, which were not easy to master.

In 1980-1981 the Intercity Seminar focused on modular curves. This proved to be a nursery for new talent and a source of intensive cooperation. Around the same time the Dutch research structure in mathematics was enhanced by the creation by NWO of 'Landelijke Werkgemeenschappen', managed by SMC. Within this framework two projects: 'Moduli' and 'Singularities', were initiated. These projects have guided for many years jointly and alternately—the activities of what was called the 'Intercity Seminar'.

2. The Moduli Project

The project was jointly proposed by G.B.M. van der Geer, F. Oort and C.A.M. Peters; H.W. Lenstra Jr. and J.P. Murre acted as advisors. Research results included three Ph.D. theses by L.N.M. van Geemen (1985; cum laude), C.F. Faber (1988) and J. Top (1989). The project's applicants, all experts in algebraic geometry, differed in education and interests, which turned out to be quite an advantage. Researchers in the project included, apart from the three project leaders, several graduate students and senior researchers. Collaboration formed one of the most fascinating aspects for each of us. An ongoing avalanche of new results in the field—a pleasant surprise—was gratefully exploited and frequently led to the set-up of new research.

Over the years the project gave us the opportunity to invite several mathematicians for short visits as well as for longer stays. These stimulating visits have led to a broad spectrum of research, useful developments and interesting publications.

Characteristic for the Moduli Project was that it naturally emerged from an existing and well-functioning collaboration between the members of a small, enthousiastic group of mathematicians. The initiators' foremost concern was to stimulate the field of algebraic geometry with NWO-support. Inviting eminent mathematicians from abroad was considered as important as appointing promising graduate students, certainly in a time of ever shrinking academic budgets (when, indeed, will that stop?).

In the sections below we briefly describe the work of the three Ph.D. theses completed in the framework of the Moduli Project. This work was, in the focus of international developments at the time, of high quality, and in all cases proved to be a stepping-stone to further research, presently in full swing.

3. THE SCHOTTKY PROBLEM

The period mapping (Torelli mapping)

$$j : \mathcal{M}_g \to \mathcal{A}_{g,1}$$

assigns to an (isomorphism class of an) algebraic curve C its principally polarized Jacobian $(\mathrm{Jac}(C), \Theta_C)$. This mapping is injective at geometric points, as Torelli proved in 1914 (over the complex numbers). The closure of the image of this mapping

$$(j(\mathcal{M}_g))^c =: \mathcal{J}_g \subset \mathcal{A}_{g,1}$$

is usually called the Torelli locus or Jacobi locus. For $g \geq 4$ this yields a lower-dimensional subvariety in $\mathcal{A}_{g,1}$. Riemann had asked for a characterization of this subvariety of the 'periods'. F. Schottky, in 1888 for $g = 4$, and F. Schottky and H. Jung in 1909 for general g, indicated relations which were expected to characterize the Jacobi locus. When Van Geemen [3] started to work on this classical problem, only partial results were known. His main result states that *the Jacobi locus is a component of the Schottky locus for all g*. His proof contains an induction on g, the most important idea being to intersect the Schottky locus with the boundary of the moduli space, in a blown-up compactification of the moduli space $\mathcal{A}_g$. Modestly, Van Geemen notices that this idea was already present implicitly in work by Schottky and by F. Frobenius in 1888 and 1889, but he deserves credit for being the first to understand the argumentation and to carry out the proof. His work links to very different methods developed for the same problem by G.E. Welters, E. Arbarello and C. De Concini, as well as by T. Shiota in connection with the Novikov conjecture (stating that the Jacobi locus is described by solutions of the Kadomtsev-Petviashvili differential equations).

These elegant and fine results on a classical problem are indicative for Van Geemen's insight in geometry which he combined with an extensive arsenal of techniques in algebraic geometry. Clearly he vastly profited from his participation in the Moduli Project and – vice versa – the project from his insight and dedication.

4. CHOW RINGS OF MODULI SPACES OF CURVES

Studying varieties frequently involves the use of a cohomology theory. However, the ring of cycles modulo rational equivalence on that variety provides a finer invariant.

Mumford was one of the first to apply this method to moduli spaces of curves [6]. Since the moduli space is usually singular, $\mathbb{Q}$-coefficients have to be used. Moreover, since a regular covering was lacking at that time (it was discovered only much later), even the definition of a Chow ring is not obvious in this case. Mumford founded this theory and he computed the Chow ring for the moduli space of curves of genus 2.

In his Ph.D. thesis [4], Faber studies the Chow rings of a variety of moduli spaces of curves. His main result is the complete computation of the Chow ring $\mathcal{A}^*(\overline{\mathcal{M}_3})$ of a compactification of the moduli space of curves of genus 3. This space has a natural interpretation as the union of subspaces, like the boundary $\overline{\mathcal{M}_3} - \mathcal{M}_3$, the hyperelliptic locus $\mathcal{H}_3$, and the moduli space $(\mathbb{P}^{14} - \Delta)/PGL(3)$ of plane, non-singular curves of degree 4. In this way generators of the various Chow groups of $\overline{\mathcal{M}_3}$ can be given. Now the difficult part starts: which are the relations between the obvious generators? Faber invents a fascinating method and applies it with virtuosity. In order to find relations between classes of cycles of codimension k on this 6-dimensional variety $\overline{\mathcal{M}_3}$, 'test objects' in codimension $6 - k$ are selected, and all intersection products between cycles and test objects are calculated, until from that the structure of the Chow ring follows. When shortly afterwards E. Witten proposed his conjectures concerning the intersection numbers, these could be successfully tested with Faber's results for $g = 3$.

5. L-SERIES IN GEOMETRY

Whereas the Ph.D. theses by Van Geemen and Faber treated typically geometric problems, Top's work [5] is closer to number theory, although geometry is frequently drawn upon as motivation or as a tool. L-series are central here. These show up, for example, in connection with cycles on an abelian threefold. Ceresa proved that the cycle $C - C^-$ for a generic curve of genus 3 yields a non-torsion class. Top links this with a conjecture by S. Bloch concerning zero's of an L-series; his work provides evidence for this conjecture.

Top also considers L-series occurring in the theory of Siegel modular forms. In this difficult field understanding of the GL_2 case is dawning, but other cases still seem far beyond our grasp. Top works out examples, illustrating a conjecture by H. Yoshida about Siegel modular forms of weight 2 belonging to $Sp(\mathbb{Z})$ – an area now in full swing.

6. WHAT HAPPENED AFTERWARDS

These developments, from the 'schovenclub' through the 'intercity seminar' to Moduli and Singularities , have generated a group of Dutch mathematicians well-versed in algebraic geometry. Abroad our young researchers, in collaboration and at meetings, now easily match world level, as is evidenced by their addresses at important conferences and their usually easy acqui-

sition of good positions (abroad!) in these difficult times. Our research is linked, in depth and in diversity, with the work of the best people in the field, and is published in the leading journals. The breeding ground, to which the Moduli Project belonged, proved to be very valuable.

Collaboration did not stop when the Moduli Project terminated. Arithmetic aspects were elaborated in a project 'Arithmetic Algebraic Geometry', and geometric aspects of moduli theory are addressed in a project studying moduli of curves and of Riemann surfaces. Thus collaboration started in the Moduli Project has borne fruit and scientific methods developed in the project are used and expanded.

We expect moduli spaces to gain significance and to be a focal point in the research of the coming decades. As in all good mathematics, closer study generates more questions than solutions. Assessing the project's impact after ten of twenty years may yield an even more positive picture than at present.

References

1. D. Mumford (1965). Geometric invariant theory. *Ergebn. Math. 34*. Springer-Verlag. (Second printing: D. Mumford, J. Fogarthy. Springer-Verlag, 1982.)
2. J.-I. Igusa (1960). *Ann. Math. 72*, 621-649.
3. L.N.M. van Geemen (1985). *The Schottky Problem and Moduli Spaces of Kummer Varieties*, Ph.D. Thesis, Utrecht University. (See: Siegel modular forms vanishing on the moduli space of curves. *Invent. Math. 78*, (1984), 329-349.)
4. C.F. Faber (1988). *Chow Rings of Moduli Spaces of Curves*, Ph.D. Thesis, University of Amsterdam. (See: *Ann. Math. 132*, (1990), 331-419 and 421-449.)
5. J. Top (1989). *Hecke L-series Related with Algebraic Cycles or with Siegel Modular Forms*, Ph.D. Thesis, Utrecht University. (See: R. Salvati-Manni, J. Top. Cusp forms of weight 2 for the group Γ_2 (4,8). *Amer. J. Math. 115*, (1993), 455-486.)
6. D. Mumford (1983). Towards an enumerative geometry of the moduli space of curves. Arithmetic and geometry (dedicated to I. Shafarevich), Vol.II: Geometry. *Progr. Math. 36*, Birkhäuser.

Intuitionistic Logic and Topos Theory

I. Moerdijk

1. Brouwer

Any scientific activity is governed by logic. The word 'logic' here usually refers to 'sound' reasoning, and the forms of reasoning which are sound may differ somewhat from one discipline to another. For example, the logic of ethics, dealing with statements about what actions should or should not be performed, and the logic governing reasoning involving probabilities, both differ from the one pure descriptive, factual reasoning. The latter logic is usually thought of as the logic which applies to mathematics, which, after all, is the prime example of a scientific discipline where statements are clear and unambiguous, hence either true or false.

This, however, tacitly assumes agreement on the nature of mathematical knowledge, on what it means to 'know' that a mathematical statement is true. It was convincingly shown by the Dutch mathematician L.E.J. Brouwer (1881-1966, see figure 1) that there is no unique unambiguous approach to mathematical truth. Brouwer developed a *constructive* foundation of mathematics, in which a mathematical statement is only viewed as established when constructions are given for all the objects asserted to exist by the statement. To see how this affects the logic, note that from this point of view, 'classical' rules of logic such as *Tertium non datur* ('p or not p') and proof by contradiction ('if the assumption that not p leads to a contradiction, then p') are no longer valid. This becomes particularly clear for existential statements. For Brouwer, a statement of the form 'there

Figure 1. L.E.J. Brouwer (Photo: Brouwer archive).

exists an object x with property p' is established only when one can describe an explicit construction of such an object x; thus, it doesn't suffice to prove that the assumption that no object x can have property p leads to a contradiction.

Brouwer's ideas were later formalized and made into a clear logical system of axioms, called 'intuitionistic logic', first and notably by A. Heyting. This system, and variants of it, led to several interesting early developments, such as Gödel's embedding of intuitionistic logic into modal logic (around 1933), Kleene's algorithmic interpretation of intuitionistic logic (where the 'constructions' of Brouwer are interpreted as algorithms for numerical functions, around 1945), and Kripke's completeness proof (1965) using what are now called Kripke models.

It is often thought that intuitionistic logic, and the mathematics based on it, are properly contained in ordinary, 'classical' logic and mathematics. Now it is indeed true that intuitionistic logic per se is weaker than classical logic, but this also implies that there is room for new concepts and theorems, perhaps inconsistent with ordinary logic. For example, based on a suitable

analysis of the notion of a real number, Brouwer argued that all functions from the reals to itself are continuous. This result was later made more rigorous by G. Kreisel and A.S. Troelstra, who extended Heyting's axioms to a system for analysis, containing so-called 'choice sequences'. In this system, Brouwer's continuity result is formally derivable.

Thus, intuitionistic logic turned out to be very rich in mathematical content, by giving rise to new methods and models of formal logic, and in the ways it deviates from and possibly extends classical mathematics. Nevertheless, it seems fair to say that it was not part of the mainstream activities in mathematics in general, and within mathematical logic in particular. This situation changed drastically in the past two decades, for two reasons. First, with the increasing interaction between logic and computer science, intuitionistic logic turned out to play a central role, in semantics of programs as well as in proof theory (program extraction from formal derivations). Secondly, the relation to sheaves and topoi, discovered in the early seventies, gave an enormous impulse to intuitionistic logic, and started a whole new and broader line of development, now generally refered to as categorical logic. It is this second reason that I wish to explore here.

Figure 2. A. Grothendieck (Courtesy Birkhaüser, Inc., Boston).

2. Grothendieck

The work of the French mathematician A. Grothendieck (see figure 2) in the sixties and seventies formed a revolution in algebraic geometry, and later led to the solution by P. Deligne of the famous Weil conjectures. Central among the many new concepts and methods introduced by Grothendieck was his generalization of the notion of 'space'. The basic idea was that for the construction of many invariants of a space, it suffices to know the system of all

'sheaves' which can be defined over the space. A sheaf is something like a continuously varying function on the space whose values are sets—or more often, sets with some algebraic structure, such as abelian groups. Grothendieck then observed that such sheaves could be defined not only for spaces, but for much more general structures. This gave rise to the notion of a 'site'. A site is a category, to be thought of as a system of 'neighbourhoods', equipped with an a priori given notion of when a family of such neighbourhoods covers another neighbourhood. For any such site one can define the system of all sheaves on the site. Such a system defined from a site is called a *topos*. According to Grothendieck topoi—and not just spaces—are the central geometric objects to be studied.

The relation to logic and set theory arose from the work of F.W. Lawvere and M. Tierney, concerned with simplifications of Grothendieck's axioms for sheaves and sites. Lawvere and Tierney discovered that many of the properties of topoi could be derived from a very simple set of axioms. These axioms describe elementary properties of sheaves: for example, that for any two sheaves S and T one can form the product sheaf $S \times T$, the 'function sheaf' T^S of all maps from S to T, and the 'powersheaf' PS of all subsheaves of S. They also discovered that any topos can be viewed as a 'universe of sets'. This means that one can interpret the axioms of set theory in a topos, and view this topos as a world in which one can do mathematics. Such a topos world is exactly like the ordinary world of sets in which mathematicians work, except that the logical rules of Tertium non datur and proof by contradiction do not hold. In fact, it is a world with a logic which differs from ordinary, classical logic: a world with precisely the intuitionistic logic of Brouwer and Heyting!

3. Topos models for intuitionistic logic

The discovery of this striking coincidence between geometric structures and intuitionistic logic immediately led to the construction of a great variety of natural mathematical models of specific intuitionistic theories. For example, the principle mentioned above of continuity of all real functions, at first thought of as a rare phenomenon, turned out to be true in many of Grothendieck's most general topoi (the so-called topological gros topoi). Using these topoi, one discovers natural models of the Kreisel-Troelstra theory of choice sequences (G.F. van der Hoeven and I. Moerdijk, 1984).

The use of topoi as models for logic and set theory also led to new explanations of classical independence results for Zermelo-Fraenkel set theory. Thus, Tierney showed (1972) how to interpret Cohen's famous proof of the independence of the Continuum Hypothesis as a topos theoretic construction, and later (1980) P. Freyd gave a strikingly simple proof, based on topos theory, of the independence of the axiom of choice. Around the same time, J.M.E. Hyland showed that Kleene's algorithmic interpretation of intuition-

istic logic can also be viewed as the construction of a topos, thus providing an extension of Kleene's interpretation to higher order logic. Hyland's topos has various properties which make it strikingly different from ordinary set-theoretic universes. For example, it contains a large class of countable sets which has the remarkable property that the product of all its members is again a member of the class, while avoiding the paradoxes of set theory that such phenomena usually give rise to. This property has applications to the semantics of certain strong functional programming languages (versions of the so-called polymorphic lambda calculus).

The Continuum Hypothesis, formulated by G. Cantor in 1878, states that every infinite subset of the continuum $\mathbf{R}$ (i.e., the set of all real numbers) is either equivalent to the set of natural numbers or to $\mathbf{R}$ itself. D. Hilbert posed, in his celebrated list of problems presented at the International Congress of Mathematicians in 1900 in Paris, as Problem nr.1 that of proving this hypothesis. The independence of the Continuum Hypothesis, proven in 1963 by P.J. Cohen, means that it neither can be deduced from, nor contradicts the other axioms of set theory (e.g. the Zermelo-Fraenkel system), assuming these axioms to be non-contradictory. The Axiom of Choice (E. Zermelo, 1904) states that if $\mathbb{S}$ is a system of non-empty sets, then there exists a set A having exactly one element in common with every set S of $\mathbb{S}$. This axiom was put forward in connection with the question, posed by Cantor, whether of two sets there is always a largest. With the Axiom of Choice there is. The axiom met with considerable resistance, because it produced some counter-intuitive results. The independence in the above sense of the Axiom of Choice was also proved by Cohen. These independence results bear some similarity to the famous parallel postulate in Euclidean geometry, the discussion about which led to the discovery of non-Euclidean geometry. However, the independence proofs in logic are very different from those in geometry.

The newly discovered relation between intuitionistic logic and topos theory also led to an effective and well-motivated development of parts of intuitionistic mathematics. Notably, P.T. Johnstone and others developed a version of intuitionistic topology which avoids the use of points, and is now known as locale theory. This theory immediately went far beyond what was known up to then in intuitionistic topology. Furthermore, by applying the theory inside a topos (remember, a topos 'is' an intuitionistic universe in which one can do mathematics), various new presentation theorems for topoi were discovered. For example, Freyd proved that any topos allows a particularly nice embedding into the category of $\mathbb{Q}$-equivariant sheaves on a locale X equipped with the action by the group $\mathbb{Q}$ of rational numbers. A. Joyal proved with Tierney (1984) that any topos can be described as

a more general category of equivariant sheaves on a locale. Later (1990), Joyal and Moerdijk proved that for any topos there exists a locale with the same weak homotopy type.

4. Further developments

The discovery of the relation between topoi and logic stood at the origin of an entirely new subject within logic, now called 'categorical logic'. In categorical logic, one tries to study logical systems in a way independent from their description in a specific language. Instead, a logical system is described by certain closure conditions on categories. The 'free' category possessing these closure conditions then replaces the older description of the logical system by a formal language, while more concrete, mathematical categories possessing these closure conditions correspond to models of the system. Semantics is now simply a functor between categories. (In hindsight, this is somewhat analogous to group theory, where abstract groups come instead of generators and relations, and representations of groups are homomorphisms into 'concrete' groups of automorphisms.) In categorical logic, the notion of a topos replaces that of a logical system for (a weak form of) set theory.

Similarly, logical systems for the typed lambda calculus correspond to cartesian closed categories [3]. First order logic corresponds to Grothendieck's theory of pretopoi (or coherent topoi). Grothendieck's fibered categories also have turned out to be very useful, in particular for describing type theories with so-called dependent types; see the Ph.D.-theses of D. Pavlovic (Utrecht, 1990) and B.P.F. Jacobs (Nijmegen, 1991).

This formulation of logic using categories has turned out to be very flexible and useful, in discovering analogies between logical systems, in finding new models, and, perhaps most importantly, in making methods of central parts of mathematics such as algebra and topology applicable to logic. By way of example, I may mention the work of M. Makkai (1993) and M. Zawadowski (1995) on descent theory. Descent theory in algebraic geometry is concerned with conditions under which one can 'descend' structures defined for a space X along a mapping $X \to Y$ to similar structures defined over Y. Descent theory was also used in the work of Joyal-Tierney and Joyal-Moerdijk mentioned above. The work of Makkai and Zawadowski relates descent theory, via the methods of categorical logic, to classical results in logic concerning definability and interpolation, some of which go back to the Dutch logician E.W. Beth (1909-1964).

Using the correspondence between suitable categories (pretopoi) and first order logic, Makkai and Zawadowski have been able to develop a descent theory for first order logic, and show that this theory leads to definability and interpolations theorems which are considerably stronger than the classical ones just mentioned.

This is one of many examples of an active and highly interesting interaction between logic and geometry, which has led and will lead to many new concepts and results in mathematical logic, and from which a fruitful feedback to geometry in general and topos theory in particular is emerging.

REFERENCES

1. A. GROTHENDIECK ET AL. (1972). *Théorie des Topos et Cohomologie Etale des Schémas* ('SGA4'), Springer Lecture Notes in Mathematics *269* and *270*.
2. A. JOYAL, I. MOERDIJK (1995). *Algebraic Set Theory*, London Math. Soc. Lecture Notes Series, Cambridge University Press.
3. J. LAMBEK, P.J. SCOTT (1986). *Introduction to Higher Order Categorical Logic*, Cambridge University Press, Cambridge.
4. S. MAC LANE, I. MOERDIJK (1992). *Sheaves in Geometry and Logic — A first introduction to topos theory*, Springer-Verlag.

Statistics and Medieval Astronomical Tables

B. van Dalen

1. Introduction

Around the year 150 A.D. the Greek scientist Claudius Ptolemy, who lived and worked in Alexandria in present-day Egypt, compiled his main astronomical work, the *Almagest*. In this work he not only collected the most important achievements of his predecessors, in particular Hipparchus (Rhodes, second century B.C.), but he also developed the first accurate geometrical models for the motion of the moon and the five visible planets as seen from the earth. Ptolemy computed large sets of tables, mostly of complicated functions based on plane or spherical trigonometry, by means of which the geocentric positions of the sun, moon and planets could be calculated at the cost of only a small number of additions and multiplications.

Ptolemy's astronomical work was highly influential till the end of the Middle Ages. From the ninth till the sixteenth century Muslim astronomers from Afghanistan till Spain and from Yemen till Constantinople (see also figure 1) used the *Almagest* as a prototype for their own astronomical handbooks with tables, which were mainly written in Arabic or Persian. Only incidentally they made modifications to the Ptolemaic planetary models, but in many cases they increased the accuracy of the calculations underlying the tables, in particular by determining more accurate values for the basic trigonometric functions. Furthermore, they made new observations of the motions of the sun, moon and planets, on the basis of which they improved upon the values of the parameters underlying the models. In some

Figure 1. Astronomers at work in the observatory of Constantinople (16th century).

cases, e.g., the eccentricity of the solar orbit or the obliquity of the ecliptic (the angle between the equator and the plane in which the apparent yearly motion of the sun takes place), these values had actually changed during the centuries. In other cases, e.g. the length of the solar year, the availability of observations over longer time spans allowed a more accurate determination.

Frequently the extant manuscripts of medieval astronomical handbooks are quite different from the original works. Since the manuscripts were copied by hand, many scribal errors occurred, in particular in the numbers in the tables. Often we find additions to the text or the tables by later users and in many cases parts from other works were inserted in place of missing, outdated or unsuitable parts of the original work. Knowledge of the origin of such added materials could give us important information about the development of medieval astronomy and the influence of earlier astronomers on later ones. However, in many cases the added material is not properly attributed.

A useful method for determining from which sources tables in an astronomical handbook derive is the comparison of mathematical properties of the tables, e.g. the underlying function and parameter values, peculiarities of the method of computation such as intermediate rounding, the use of (inverse) interpolation or inaccurate auxiliary tables, etc. This kind of information can sometimes be found in tabular headings or explanatory text, but is often unreliable or simply missing. Not always is a given table based on the indicated parameter values and only incidentally leads a recomputation following the rules in the explanatory text precisely to the tabular values. Sometimes tables are based on two different values for the same astronomical parameter.

From the above it follows that for many tables in medieval astronomical handbooks methods by which the mathematical properties can be determined directly from the tabular values are indispensable. Until recently such methods were only applied *ad hoc*: tables were recomputed for various historically attested parameter values and methods of computation in order to see which value and method led to the best agreement, the use of linear interpolation was recognized from groups of constant tabular differences, etc. Although many important results were found in this way, there remained a large class of tables which defied a mathematical explanation. Only in the last decade has a systematic approach to the analysis of medieval astronomical tables with the use of advanced mathematics and statistics and special computer programs been practiced by a small number of scholars.

2. The SMC research project in Utrecht

In a SMC-supported research project at the Mathematical Institute of Utrecht University an extensive investigation of the application of statistical

methods for the analysis of medieval astronomical tables was made. Firstly, the conditions under which statistics can be applied to such tables, i.e. under which errors in tables can be assumed to be random variables, were explored (see below). Secondly, several estimators were developed for determining unknown parameter values underlying medieval astronomical tables. Since such tables mostly display values for complicated functions based on plane or spherical trigonometry, the estimators are non-trivial. They include:

- **Weighted estimator.** Assume that we have a table with values $T(x)$ for a function f_θ depending on a single parameter θ. Let g be a function of two variables such that $g(x, f_\theta(x)) = \theta$ for every argument x and every value of θ. The value of θ underlying the table can then be estimated from each single tabular value $T(x)$ using $\theta \approx g(x, T(x))$. Bias and variance of these estimators can be approximated using a Taylor expansion of g. Finally, a more accurate estimator is obtained by computing a weighted sum of the separate estimators.
- **Fourier Estimator.** Let g be a 2π-periodic odd function and c an unknown constant. Let f be defined by $f(x) = g(x - c)$ for every x. If the Fourier series of f converges and the Fourier coefficients a_k and b_k for $k > 0$ are given by

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos kx\,dx \qquad \text{and} \qquad b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin kx\,dx$$

  respectively, we have $a_k \cos kc + b_k \sin kc = 0$ for every k. From a table for f we can calculate approximations $\tilde{a}_k$ and $\tilde{b}_k$ to the Fourier coefficients by replacing the integral by a finite sum and the (unknown) functional values by tabular values. Then $\tan c$ can be estimated by the quotients $-\tilde{a}_k/\tilde{b}_k$ (or $\cot c$ by $-\tilde{b}_k/\tilde{a}_k$). These estimators have some interesting properties, e.g. they become degenerate if c lies precisely between two consecutive arguments of the table for f.
- **Least Number of Errors Criterion.** According to this criterion an unknown parameter value underlying an astronomical table is determined in such a way that the number of errors in the table is as small as possible. The criterion can be given a statistical interpretation if we assume a probability distribution for the tabular errors.
- **Least Squares Estimator.** The method of least squares was used for determining multiple unknown parameters from a single table.

The accuracy of the estimators listed above was determined and confidence intervals for the unknown parameters computed. Special user-friendly computer programs were written to deal with the sexagesimal number system (i.e. with base 60) in which values in medieval astronomical tables were usually given. In these programs the estimators described above and many specific methods of analysis can be conveniently applied.

The SMC research project in Utrecht led to a large number of interesting historical results which had not been possible without the use of statistical methods and special computer programs. One of these was the recovery of some of the lost tables of the important tenth-century Muslim astronomer Abū'l-Wafā' in a thirteenth-century astronomical handbook (see below). Furthermore, the method of computation of various tables for complicated functions which had thus far defied explanation, was discovered.

3. CAN STATISTICS BE APPLIED TO MEDIEVAL ASTRONOMICAL TABLES?

In order to apply statistical methods to ancient and medieval astronomical tables, we assume that, in some sense, the tabular values behave as random variables. We will investigate this in the case of three spherical-astronomical tables by Abū'l-Wafā'. This important Muslim mathematician and astronomer lived and worked in tenth-century Baghdad and is known in particular for the advances he made in spherical astronomy and in the calculation of accurate sine values. Abū'l-Wafā''s major astronomical work, the *Almagest*, has a structure similar to Ptolemy's *Almagest*. Most of the explanatory text concerning trigonometry, spherical astronomy and the planetary models is extant in a manuscript in the Bibliothèque Nationale in Paris. The text includes geometrical proofs, directions for the calculation of many useful quantities in spherical astronomy, and a large number of numerical examples. However, the tables belonging to the original work, explicitly indicated to follow the explanatory sections, are not present; a possible reason is that the person who ordered the copying of the manuscript, was only interested in the mathematics, not in the tables.

The Bibliothèque Nationale also possesses a unique manuscript of the astronomical handbook of al-Baghdādī (thirteenth century), which is a mixture of material from earlier works. al-Baghdādī explicitly states that he used some of the lunar tables of Ḥabash al-Ḥāsib (Baghdad, c. 840) and the solar equation table of Abū'l-Wafā'. Furthermore, he copied some inaccurate tables for spherical astronomy from Kūshyār ibn Labbān, a contemporary of Abū'l-Wafā'. al-Baghdādī also included a more accurate set of tables for trigonometry and spherical astronomy. By comparing the mathematical properties of these tables with the explanatory text in Abū'l-Wafā''s *Almagest*, it is possible to show that the tables derive from that work.

We will now investigate the tabular errors in three of Abū'l-Wafā''s tables occurring in the astronomical handbook of al-Baghdādī. In each case $T(x)$ denotes the tabular value for argument x and the tabular error $e(x)$ is defined as the difference between $T(x)$ and the exact functional value $f(x)$: $e(x) = T(x) - f(x)$. A tabular value is called correct if it is equal to $f(x)$ rounded to the number of digits of the table concerned; otherwise, the tabular value is said to contain an error. All tabular values considered below are given in sexagesimal notation, i.e. to base 60. In transcriptions of

sexagesimal numbers a semicolon denotes the sexagesimal point and further sexagesimal digits are separated by commas. For instance, the sexagesimal number 51;57,41,29 denotes $51 + 57 \cdot 60^{-1} + 41 \cdot 60^{-2} + 29 \cdot 60^{-3}$ and will be said to have three sexagesimal fractional digits or to be given to sexagesimal thirds.

Example 1: Abu'l-Wafa''s sine table
Like many ancient and medieval astronomers, Abū'l-Wafā' tabulated the trigonometric functions for a base circle with radius 60. Thus his sine table displayed values for $\mathrm{Sin}\, x \stackrel{\mathrm{def}}{=} 60 \cdot \sin x$. These values were given to sexagesimal thirds (e.g., $T(60) = 51;57,41,29$), which corresponds to an accuracy of approximately seven decimal digits. Abū'l-Wafā''s original table is known to have had values for each quarter of a degree, but the copy which is extant in the handbook of al-Baghdādī only displays values for each integer degree. Most of the ninety tabular values are correct; only nine values contain an error of ± 1 sexagesimal third. Figure 2 displays the tabular errors $e(x) \stackrel{\mathrm{def}}{=} T(x) - \mathrm{Sin}\, x$ in Abū'l-Wafā''s sine table for all arguments $x = 1, 2, 3, \ldots, 90$. The nine tabular errors outside the range from $-0;0,0,0,30$ to $+0;0,0,0,30$ correspond to the nine errors just mentioned. All tabular errors within the range correspond to correct tabular values; they are precisely the errors made in rounding exact sine values to the number of digits of the table. It appears that these rounding errors behave like independent random variables with a uniform distribution on the interval $[-0;0,0,0,30\,,\ +0;0,0,0,30]$. Below this idea will be further pursued.

Example 2: Abu'l-Wafa''s tangent table
Abū'l-Wafā''s original tangent table displayed values for $\mathrm{Tan}\, x \stackrel{\mathrm{def}}{=} 60 \cdot \tan x$ for each quarter of a degree; the values for arguments up to 45° had four sexagesimal fractional digits (e.g., $T(30) = 34;38,27,39,38$), those for arguments larger than 45°, three (e.g., $T(60) = 103;55,22,58$). Again in his version of the table al-Baghdādī left out the tabular values for non-integer arguments.

The tabular errors $e(x) \stackrel{\mathrm{def}}{=} T(x) - \mathrm{Tan}\, x$ in the tangent table do not display a uniform pattern like that in the sine table. Instead they contain a clear tendency towards larger errors for larger arguments. This pattern can be explained as follows:
medieval astronomer usually calculated the tangent from his previously calculated sine values according to

$$\mathrm{Tan}\, x = \frac{60 \cdot \mathrm{Sin}\, x}{\mathrm{Sin}\,(90 - x)}.$$

If he used sine values from a sine table with a fixed number of sexagesimal fractional digits, the relative error in the denominator increased as x approached 90°. Since the error in the numerator remained of the same order,

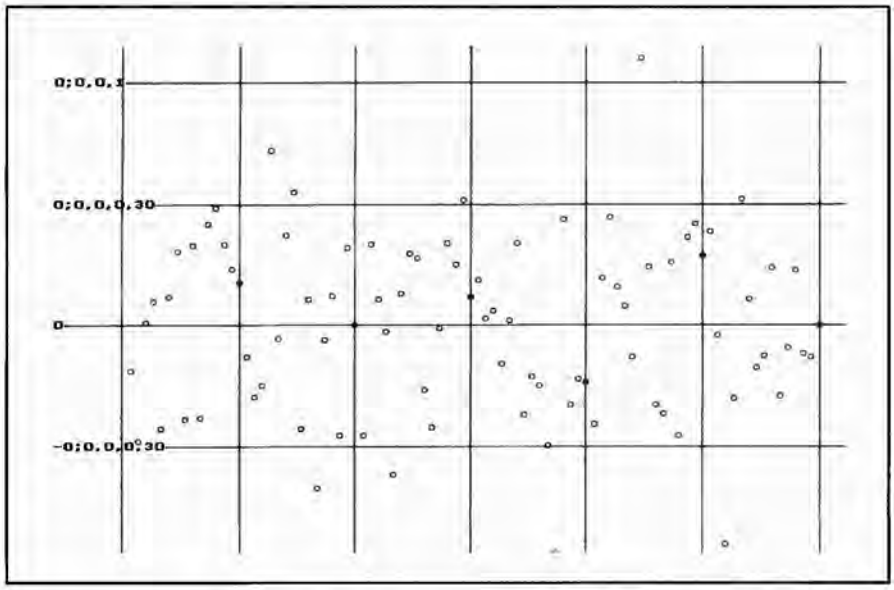


Figure 2. Plot of the tabular errors in Abū'l-Wafā''s sine table.

the absolute error in the quotient increased. This is precisely what happens with Abū'l-Wafā''s tangent table.

Figure 3 shows the differences between the tangent values presented by al-Baghdādī and values recomputed from his sine table as described above. The increasing tendency towards 90° has completely disappeared; instead, both in the first half of the table (although this is difficult to see from figure 3) and in the second half the differences appear to behave as independent random variables. Although the calculation of tangent values from the sine table as described above is a relatively easy and in principle completely deterministic process, we note that the differences shown in figure 3 apparently contain two sources of randomness: firstly, the random error in the sine values; secondly, a more or less random error made in the division of the sine values: apparently the quotient was not calculated to its full accuracy and the last digit was to a certain extent guessed. This led to the 15 differences larger than 0;0,0,0,0,30 for arguments between 0° and 45° and to the 13 differences larger than 0;0,0,0,30 for arguments between 45° and 90° which can be seen in figure 3.

It can be checked that changes in the underlying sine values of only one sexagesimal third lead to clearly different tangent values. Therefore we can be certain that the tangent table in the astronomical handbook of al-Baghdādī was computed from his sine table. The outlying difference for argument 41° is probably due to a scribal or computational error.

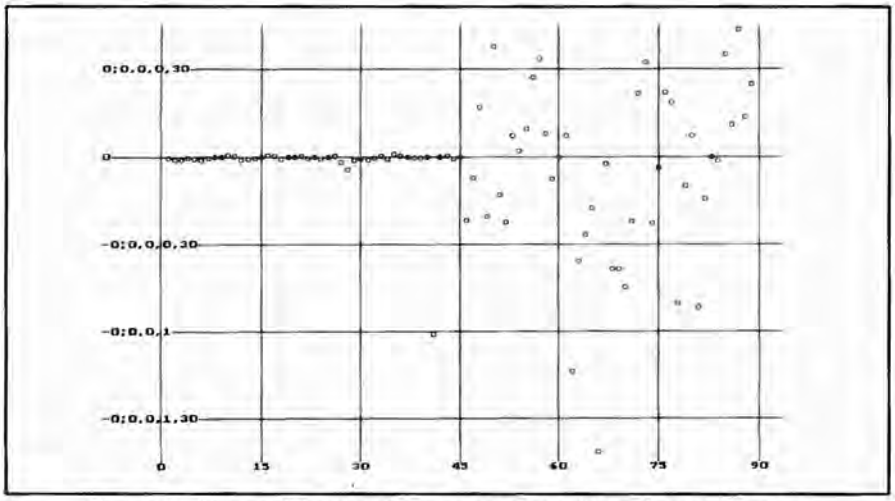


Figure 3. Plot of differences between tangent table and reconstruction.

Example 3: Abū'l-Wafā''s solar declination table
The solar declination is the orthogonal distance on the heavenly sphere between the sun and the equator. It can be calculated by applying the sine law to the spherical triangle ΥSP in figure 4, where E denotes the earth, s the sun, λ the solar longitude (i.e., the length of the arc ΥS along the ecliptic), ε the obliquity of the ecliptic, and δ the declination. We have

$$\delta(\lambda) = \arcsin(\sin\varepsilon \cdot \sin\lambda)$$

for each value of λ. Practically every medieval astronomical handbook contained tables for the solar declination and other spherical astronomical functions connected to the transformation of equatorial coordinates into

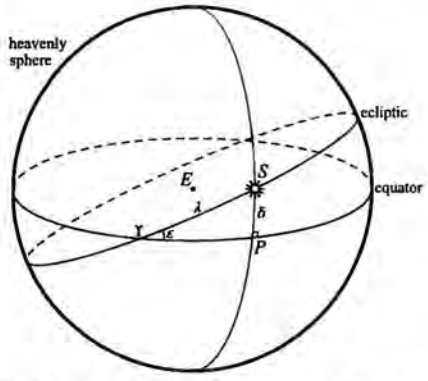


Figure 4. The calculation of the solar declination.

ecliptical ones and vice versa, or to the determination of the time of the day from observations of the sun or a fixed star.

The tabular errors in Abū'l-Wafā''s table for the solar declination, which gives values to sexagesimal thirds for arguments $1, 2, 3, \ldots, 90$, do not display a uniform pattern like the ones above. Instead the errors tend to be negative rather than positive and they are up to twenty times as large as the level of rounding of the table (here we disregard the apparent outlier for argument 79°). An explanation can only be found by following the calculation of the tabular values step by step. For the calculation of the solar declination according to the formula above, Abū'l-Wafā' needed:

1) sine values for arguments $1, 2, 3, \ldots, 90$,
2) a value for $\sin \varepsilon$, and
3) a method to calculate an arcsine.

Naturally he could take the required sine values directly from his table of sines. Furthermore, he could calculate $\sin \varepsilon$ by performing interpolation in his sine table or by making a separate, more accurate, but highly elaborate calculation. Finally, he had to determine the arcsines by performing some type of inverse interpolation in his sine table. In his *Almagest* Abū'l-Wafā' consistently used the value 23;35 for the obliquity of the ecliptic and 24;0,17,38 for $\mathrm{Sin}\,\varepsilon$. This value can be obtained by performing a linear interpolation between the accurate values 23;55,29,48 for Sin 23;30 and 24;9,53,17 for Sin 23;45 (as we have seen above, Abū'l-Wafā''s original sine table displayed values for each quarter of a degree). Abū'l-Wafā' also described how to determine the arc corresponding to a given sine value, namely by means of inverse linear interpolation in his sine table. (Second order interpolation schemes were also known around the year 1000 A.D., but were not widely used for the computation of tables.)

It can be checked that Abū'l-Wafā''s table for the solar declination was indeed computed using the value 24;0,17,38 for $\sin \varepsilon$ and inverse linear interpolation between accurate sine values for each quarter of a degree. If we calculate the differences between Abū'l-Wafā''s declination values and values reconstructed according to this method of computation, we note the following:

- Of the 90 differences seven have an absolute value larger than 4 sexagesimal thirds. Several of these outliers are close to round numbers (5, 10 or 40 thirds), which makes it plausible that they were caused by scribal or computational errors at some stage of the calculation. The absolute value of three more differences is larger than $1\frac{1}{2}$ thirds.

- 53 differences lie between −0;0,0,0,30 and +0;0,0,0,30 and hence correspond to declination values correctly calculated and rounded according to the method of computation described above. These differences

appear to behave like independent uniformly distributed random variables.

- The remaining 27 differences have an absolute value between 0;0.0.0.30 and 0;0.0.1.30 and hence correspond to declination values in whose calculation small errors were made. It appears that the behaviour of these errors can be considered to be random. The following causes of the errors can be conjectured:
    1. For non-integer arguments we have used correct sine values to sexagesimal thirds because Abū'l-Wafā''s complete original table is no longer extant. However, since the values for integer arguments contain nine errors, it is probable that also the values for non-integer arguments contain errors. For example, the three errors in the reconstructed declination values for arguments 81 to 83 disappear if we assume an error of one unit in one of the underlying sine values.
    2. The product $24;0,17,38 \cdot \sin\lambda$ was probably not calculated to its full accuracy. It might have been rounded or some of the smaller terms of the product may simply have been left out.
- The differences are much larger in number and in size if inverse interpolation within smaller or larger intervals of the sine table is used. Apparently, the error pattern in the table is highly characteristic for the intervals used for the interpolation.

Both in the case of the tangent and in the case of the solar declination we have seen that if we take into account the systematic causes of error in the tabular values and leave out the outliers caused by unpredictable scribal or computational errors, we are left with errors that seem to behave like independent random variables with distributions that can be assumed to be equal for practical purposes. We will now discuss the distribution of the tabular errors in some more detail.

4. The probability distribution of tabular errors

We have seen that the errors in medieval astronomical tables consist of the following components:

1. Scribal and computational errors, which are unpredictable but can sometimes be corrected on the basis of the form of the numerals used or by reconstructing the consecutive steps in the calculation of the tabular values.
2. Systematic errors, which are due to the specific method of computation of the tabular values (cf. the tangent and solar declination tables discussed above).

3. Random errors, which can be considered to be independent and identically distributed. For a table with hardly any errors, this component is constituted by the rounding errors (cf. the correct values in the sine table discused above); below we will argue that in many practical cases the rounding errors can be considered to be independent and uniformly distributed. Small non-systematic errors in a table (as we have found in the tangent and solar declination tables) can be assumed to be the result of inaccuracies in the steps of the calculation: rounding of intermediate results, use of auxiliary tables with errors, use of interpolation in auxiliary tables, etc. In this case the tabular errors could be assumed to have a uniform distribution on the interval corresponding to the level of rounding of the tabular values and 'normal tails' outside that interval.

4.1. Distribution of rounding errors

In particular in Abu'l-Wafā''s sine and tangent tables we have seen that the rounding errors seemed to behave like random variables with a uniform distribution. We will now argue that under certain conditions rounding errors in a correct table of an astronomical function have approximately a uniform distribution and can be assumed to be independent. First we note that the values produced by the linear congruential random number generator $x_{k+1} = (ax_k + c) \bmod m$ are rounding errors of a table with equidistant arguments and unit m for an exponential function $f(x) = ba^x + d$, where the constants b and d depend on a, c and the initial value x_0 of the sequence of random numbers. It seems reasonable to expect that under certain conditions, in particular if the tabulated function is not (almost) linear and if the number of sexagesimal fractional digits of the tabular values is sufficiently large, the rounding errors in tables occurring in ancient and medieval astronomical handbooks have approximately a uniform probability distribution and are independent. Thus we can for instance conjecture the following:

Let $T_{k,n}$ be a correct table with k sexagesimal fractional digits for the non-linear function f, such that $T_{k,n}$ has n tabular values for equidistant arguments x_i, $i = 1, 2, 3, \ldots, n$ in a fixed interval. For every argument x_i the tabular error $e_{k,n}(x_i)$ defined by $e_{k,n}(x_i) = T_{k,n}(x_i) - f(x_i)$ is the rounding error that we make by rounding the exact functional value $f(x)$ to the number of sexagesimal digits of the table. I will assume that this rounding is performed in the modern way. Let $F_{k,n}$ be the experimental distribution of the normalized rounding errors of the table $T_{k,n}$, i.e., $F_{k,n}(y)$ is the fraction of rounding errors smaller than $60^{-k}y$ for every y. Note that we have $F_{k,n}(y) = 0$ for every $y \leq -\frac{1}{2}$ and $F_{k,n}(y) = 1$ for every $y \geq +\frac{1}{2}$. Let F_k be the limiting distribution of $F_{k,n}$ for $n \to \infty$.

Conjecture. F_k converges to the uniform distribution on $]-\frac{1}{2}, +\frac{1}{2}]$ as k tends to infinity.

For certain types of functions it may be possible to prove this conjecture by means of the theory of *uniform distribution modulo one*. Let $\{x\}$ denote the fractional part of the real number x. For a given sequence x_i, $i = 1, 2, 3, \ldots$ of real numbers let $A_M(a, b)$ be the number of terms x_i, $1 \leq i \leq M$, for which $\{x_i\} \in [a, b\rangle$. The sequence x_i, $i = 1, 2, 3, \ldots$ is said to be uniformly distributed modulo one if

$$\lim_{M \to \infty} \frac{A_M(a, b)}{M} = b - a \quad \text{for every } 0 \leq a < b \leq 1. \tag{4.1}$$

Using the notation introduced above, our conjecture can now be stated as follows: the sequence $\frac{1}{2} + f(x_i) \cdot 60^k$, $i = 1, 2, 3, \ldots$ is asymptotically uniformly distributed modulo one for $k \to \infty$.

Less general results can be obtained if we introduce randomness explicitly. By means of an unpublished theorem by J.H.B. Kemperman it can be shown that for a randomly chosen argument the distribution of the rounding error converges to the uniform distribution as the number of sexagesimal fractional digits tends to infinity. Since the roles of the argument and an underlying parameter can be interchanged, the same holds for a randomly chosen value of an underlying parameter.

We expect that the rounding errors in a table for a non-linear function will become independent as the number of sexagesimal fractional digits tends to infinity. For instance, we may conjecture that the joint experimental distribution of rounding errors for arguments with fixed distances converges to the joint distribution of independent uniform variables as the number of sexagesimal fractional digits of the tabular values approaches infinity.

References

1. B. van Dalen (1993). *Ancient and Mediaeval Astronomical Tables: mathematical structure and parameter values*. Ph.D. Thesis. Utrecht University.
2. E.S. Kennedy et al. (1983). *Studies in the Islamic Exact Sciences*. Beirut.
3. L. Kuipers, H. Niederreiter (1974). *Uniform Distribution of Sequences*. New York.
4. O. Pedersen (1974). *A Survey of the Almagest*. Odense.
5. G.R. Van Brummelen (1993). *Mathematical Tables in Ptolemy's Almagest*. Ph.D. Thesis. Simon Fraser University. Burnaby B.C.

Mathematical Aspects of Nonlinear Dynamical Systems

H.W. Broer, F. Takens

1. Description of the field of research

Dynamical systems are systems that change as time evolves. In the present, mathematical context this means: mathematical models for such systems. These models consist of two main ingredients. The first is a state 'space': a set whose elements are the possible states of the system. The second ingredient is an evolution law, which describes how the state of the system evolves as a function of time, once an initial state is known. We usually assume that whenever the state of a system is known at a certain time t, the evolution law completely determines the state at all later times. This defines a deterministic system, as opposed to a stochastic system where the evolution is in terms of probability. Also we assume that we cannot influence the dynamics, except by choosing the initial state. Dynamical systems that do admit such interventions are the subject of systems- and control theory.

1.1. Linearity

The notion of linearity for dynamical systems usually refers to some equilibrium state, such that the only states of interest are small perturbations of this equilibrium. As an example think of water in a pond with a completely flat surface as its equilibrium, where the perturbations are small surface waves.

In such a context the system is called linear if a superposition principle holds in the following sense: for any two perturbations compatible with the

law of evolution, the 'sum' is also compatible with this law. (In the example of a water surface this holds to a good approximation: two different waves can cross one another without being visibly disturbed.) In 'real' systems this linearity often holds only in first approximation.

1.2. Nonlinearity

Nonlinear systems mostly are far from equilibrium. Usually it is hard to obtain general information about the set of all possible evolutions of such systems. There are some exceptions: systems which are nonlinear, but still can be completely 'solved'. The most famous example of this is the solar system without interaction between the planets (leading to the description of the motion as given by the Kepler laws). Systems that can be analysed completely in this way are called *integrable*. Usually the dynamical behaviour of such integrable systems is not representative for that of general nonlinear systems.

In this respect integrable systems, as well as linear systems, are exceptional. However, both cases also are important for the study of the general case. Indeed, many aspects of nonlinear dynamical systems can be studied in situations obtained from linear or integrable systems by a small perturbation. An example of this is the solar system *with* interaction between the planets, where the interaction is considered small.

An early example concerning this was the work of H. Poincaré, leading to his monumental paper in the Acta Mathematica (1890) which was a first and very influential contribution to what is now called the geometric theory of dynamical systems.

1.3. Degrees of freedom

Apart from the distinction between linear and nonlinear or integrable and non-integrable, there is another distinction we want to point out. On the one hand there are systems with only a finite number of degrees of freedom, i.e., systems, the state of which is specified by a finite set of numbers. Here think of the planetary system, mechanical systems consisting only of a finite number of rigid bodies and springs, electrical circuits, etc. On the other hand there are systems with infinitely many degrees of freedom, usually systems where the specification of a state needs one or more functions. An example is given by the above water surface where the specification of the height of the surface requires a function.

It turns out that systems with infinitely many degrees of freedom in their state space often exhibit some finite dimensional structure to which, due to the law of evolution, all states converge. In such cases the dynamics essentially is that of a system of a finite number of degrees of freedom. In cases where there is no such reduction new dynamical phenomena can

appear. The emphasis in the programme under description mainly is on systems with finitely many degrees of freedom.

2. Methods of research

Investigating nonlinear systems one tries to solve two types of questions. First, how to obtain information about the dynamical properties of a system, the evolution laws of which are given in terms of explicit equations. Second, what types of dynamic behaviour can be expected in typical (i.e., non-pathological) deterministic systems. Most methods, mentioned below, are used to answer both types of questions.

2.1. Analytic methods

Often it is not possible to determine analytically (i.e., without a computer) the future states of a dynamical system given its initial state. Nevertheless there are many important cases where good approximations exist by systems that one can solve in this respect. These approximating systems usually are either linear or integrable. For many years the positions of the planets have been predicted by such methods. It is remarkable how Poincaré motivated this research: "The final goal of celestial mechanics is to resolve the great problem of determining if Newton's law alone explains all astronomical phenomena. The only means of deciding is to make the most precise observations and then compare them to the calculated results." Indeed, these methods of approximation gave the required accuracy: not many years after he wrote this it was found that Mercury did not obey this 'law' — which was one of the observations leading to the general theory of relativity.

2.2. Geometrical methods: far from linear or integrable

These methods rely on abstract existence results often in the form of fixed point theorems. For example think of the theory of invariant manifolds and persistence for dynamical systems with hyperbolic subsets.

2.3. Analysis and interpretation of physical or numerical examples

A driving force in the development of the theory of dynamical systems was the desire to give a mathematical explanation of concrete examples. In recent times a number of such examples were given in the form of mathematical equations. We here mention the examples of Lorenz (1963), Hénon-Heiles (1964), Hénon (1976), Rössler (1976), etc., which were solved only numerically. See figure 1. These numerical solutions showed thoroughly unexpected patterns, requiring completely new theoretical insights for their explanation. In this context we also mention experiments in mechanical systems with resonance, e.g., Moon and Holmes (1979), the dynamics of which has much in common with that of the above Hénon system.

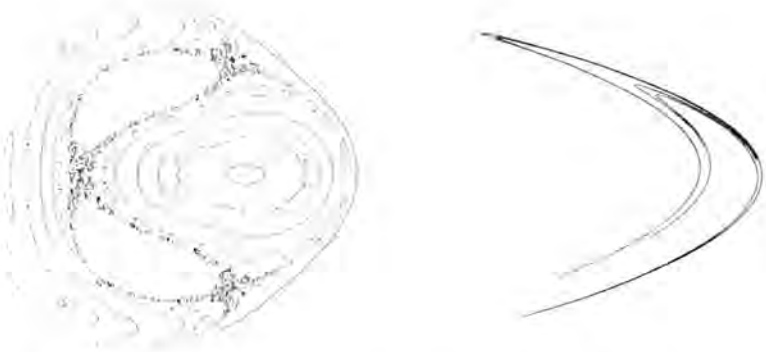


Figure 1. The conservative example of Hénon-Heiles 1964 (left) and the dissipative example of Hénon 1976 (right).

The earlier, but fundamental, work of B. van der Pol at the Philips Natuurkundig Laboratorium (1920 and later) was aimed at the understanding of the dynamics of electrical circuits with nonlinear elements (vacuum tubes).

2.4. Analysis of geometric examples

There have been a number of highly important examples of dynamical systems, not given by explicit equations, but by a geometric description. From this one could prove mathematically the possibility of certain types of dynamic behaviour. In particular, the horseshoe map (Smale 1965) and related

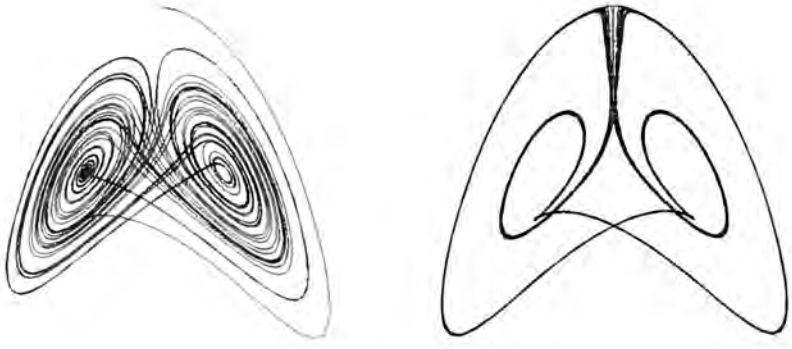


Figure 2. The dissipative example of Lorenz 1963 (left) and a modification, e.g., Palis and Takens 1993 (right).

examples showed that a deterministic system could simulate random behaviour like coin tossing. This horseshoe example isolates the essentials of the homoclinic 'webs' to be discussed below in §3.2.

In this context we consider the existence of so-called chaotic attractors in systems described by differential equations. Interestingly, this existence has not yet been proven mathematically for any system given in terms of explicit equations (without parameters). Nonetheless the geometric examples prove that there must be differential equations, say of polynomial form, which do exhibit this chaotic behaviour. See figure 2.

A similar remark holds for conservative systems used for modelling the dynamics in the world of frictionless mechanics. Here the chaos-question is whether regions of positive measure in the state space are densely filled by single orbits, another fact which is strongly suggested by computer simulations. See figure 1. The mathematical affirmation of this, even for simple systems, is open for at least 30 years and there seems to be no hope in the near future.

These geometric examples are typically related to the second class of questions mentioned before: What kind of dynamic behaviour can one expect in typical deterministic dynamical systems?

3. New concepts

The investigations of nonlinear dynamical systems opened our eyes to new notions, relevant to the description of the different types of dynamic behaviour. We mention the most important ones.

3.1. Chaos

With chaos, or chaotic dynamics, a type of deterministic dynamics is meant which looks like random. This phenomenon is displayed by systems where the evolution, following a typical initial state, is very sensitive to perturbations of this state. In fact, usually such perturbations grow exponentially. The behaviour of these systems can only be predicted over a short period: after this the uncertainty concerning the initial state, possibly combined with the round-off errors in the calculations, make further prediction impossible. A well-known example is the impossibility to predict the weather over a period longer than typically a couple of weeks.

3.2. Fractals

Fractals are self-similar objects. This means that any magnification of a fractal shows the same large-scale structures as present originally. Such structures can show up as attracting sets, i.e., sets to which typical evolutions are attracted in the case of chaos. See figure 3.

This and related occurrences of fractals in dynamical systems have drawn

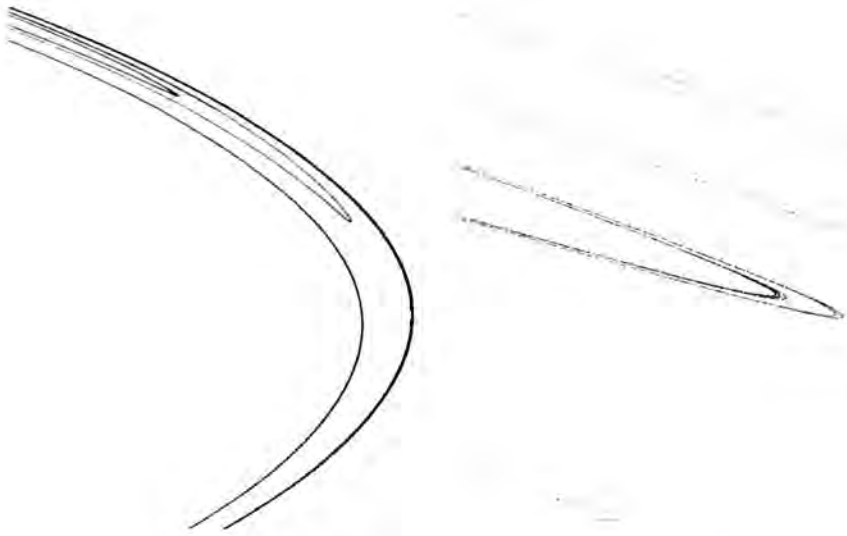


Figure 3. Magnifications in the Hénon attractor.

a lot of attention, not in the least because of the beautiful pictures that can be provided in this way. Especially the group of H. Peitgen (University of Bremen) has been very successful in this respect.

Still it is only fair to say that the relation between chaotic dynamics and fractals is rather complicated. In particular the often heard suggestion that every chaotic attractor also is fractal, is not true. Of course, it is neither true that every fractal is a chaotic attractor ...

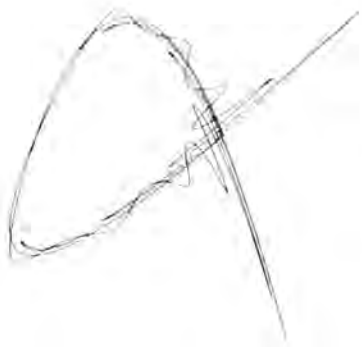


Figure 4. A homoclinic web.

One way in which fractal structures appear in dynamical systems is by the 'web' of stable and unstable separatrices in the presence of a homoclinic intersection. Let us briefly explain this. The stable separatrix of a state p, or point in the state space, consists of all points approaching p as time goes on. Reversing time gives the analogous notion of the unstable separatrix. Intersections of two such separatrices belonging to the same point p is a *homoclinic* intersection. In figure 4 we show such a web for a dynamical system with two degrees of freedom.

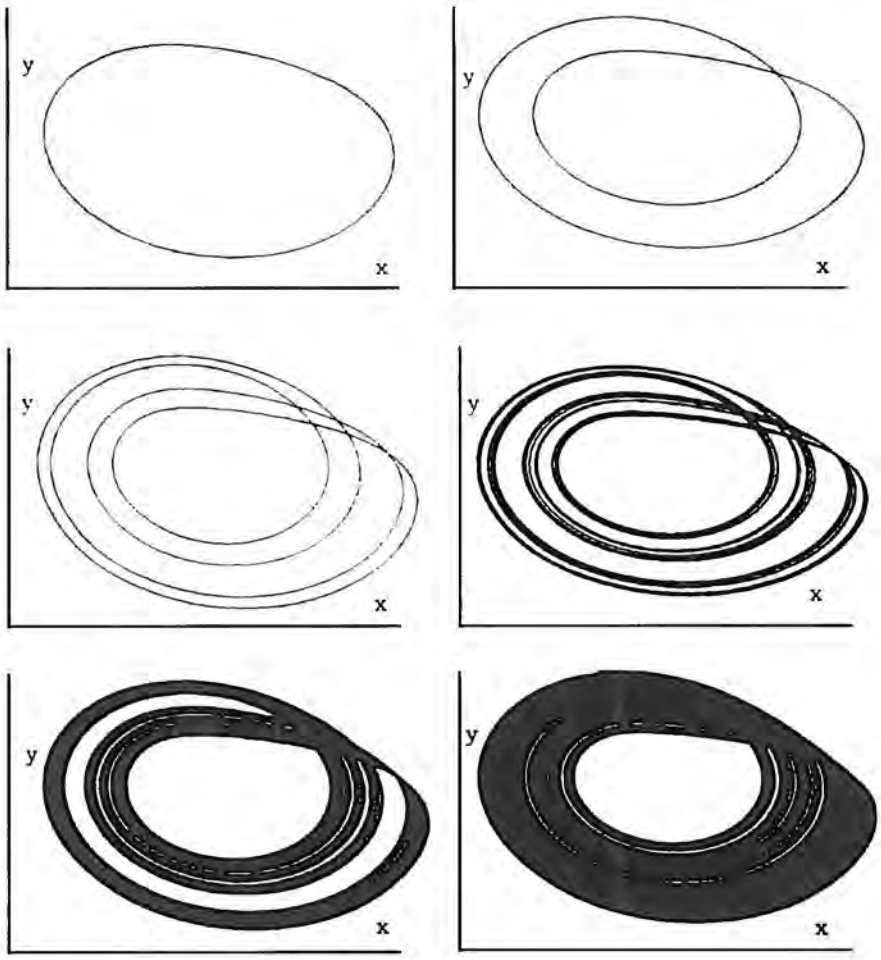


Figure 5. A sequence of bifurcations leading to complicated dynamics.

These webs (or parts of them) are strongly related to chaos in both the dissipative and the conservative context, but only in exceptional cases precise mathematical results on this have been obtained, cf. Benedicks and Carleson [4], and Palis and Takens [19].

3.3. Bifurcations

A bifurcation is a transition between different dynamical regimes. For example think of a dynamical system, the dynamics of which can be changed by tuning one or more dials. Although this may somewhat look like 'input variables' in control theory, the situation is quite different: the dials are to

be fixed when the evolution law is working. It turns out that the occurrence of certain bifurcations implies the presence of other bifurcations. This imposes a complicated hierarchy on the world of bifurcations.

Instances of this include infinite bifurcation sequences leading from stationary to chaotic dynamics. One example is the Feigenbaum sequence, where an infinite repetition of period doublings occurs. This and other examples show a strong persistence: if one perturbs the evolution law a bit, the whole infinite succession of bifurcations remains qualitatively the same. See figure 5.

4. A history

In the above exposition we already mentioned some historical aspects. Here we give a somewhat more systematic description.

4.1. From Newton to the 19th century

Through the Newtonian laws (Principia Mathematica Philosophiæ Naturalis, 1687) it became possible to treat many problems of the dynamics of mechanical systems in a mathematical way. The corresponding analysis was given in terms of explicit solutions (e.g., vibrations) or approximations (such as lunar and planetary motion). These approximations were only known to be reliable, as a description of the motion, over a restricted time interval.

4.2. The stability of the solar system

The stability of the solar system has been considered in many different forms. The main point is that information is asked concerning the dynamics of the (Newtonian) solar system, valid for the whole future, so over an infinite interval of time.

The conservation laws for the energy and the (angular) momentum give some information about the infinite future. But even taking these conservation laws into account, the following still is conceivable. Due to the interaction of planets one of these, say the earth, systematically gains energy (which the others are losing), and finally escapes from the solar system. In this sense the solar system could be unstable. This stability problem was posed by Weierstraß, and became part of the problems in 1885 set for a prize by king Oscar of Sweden. Poincaré won this prize, not by establishing stability, but because of the new insights he revealed, showing the complexity of the problem. This was the content of the paper in the Acta of 1890 mentioned before.

This work of Poincaré can be considered the starting point of what is now called the geometric theory of dynamical systems.

4.3. The theory of nonlinear oscillations

In the beginning of this century, due to the growing electronic technology, there was much interest in nonlinear electronic circuits and their oscillations. This was the subject of the fundamental work of Van der Pol (around 1920), which was later continued by Cartwright, Littlewood and Levinson (around 1950). These developments inspired Smale and his co-workers when they extended the geometric approach of Poincaré (around 1965). This extension made it possible to formulate and partly solve the basic questions behind this, that are nowadays associated to the chaotic dynamics in nonlinear oscillations.

4.4. Theory of bifurcations

The theory of bifurcations was also initiated by Poincaré. Here one investigates how the qualitative properties of a dynamical system can change as a function of one or more parameters. Later contributions are due to Andronov and co-workers around 1940, who started a systematic study of the hierarchy (based on the idea of co-dimension) of bifurcations of mechanical systems, and Hopf (1942) who investigated the transition from stationary to oscillatory behaviour inspired by questions about turbulence in the motion of fluids. Later on, R. Thom made this hierarchy of co-dimensions the basis of his general ideas on morphology and catastrophy theory [23].

Afterwards, when the importance of chaotic dynamics was discovered, one of the main questions in bifurcation theory became how transitions to chaos take place in a persistent (or typical) way. The most well-known scenario's are the transition via quasi-periodic motion by Ruelle and Takens [20] and the transition via period doubling due to Feigenbaum [9]. See figure 5.

4.5. KAM-theory

KAM-theory, around 1960 initiated by Kolmogorov, Arnol'd and Moser, deals with the persistent occurrence of quasi-periodicity in dynamical systems. This is a kind of periodicity with more than one frequency involved. Its first interest was in the conservative systems modelling classical mechanics. The context of KAM-theory again is perturbation theory: it deals with nearly integrable systems such as the solar system, see [2,3].

Concerning the stability of the solar system, this conservative KAM-theory guarantees that positive (Liouville) measure in the state-space is swept out by orderly, quasi-periodic orbits. For the stability problem this means that there is positive probability that the 'actual' evolution of the solar system is quasi-periodic, which certainly would imply stability.

In general, however, it is expected that both the quasi-periodic and the chaotic regime have positive measure. See figure 1. This coexistence of order and chaos makes it hard to infer stability from these qualitative considerations for an explicit initial state and King Oscar's question is still open ...

A related point of interest is that for nearly integrable systems with finitely many degrees of freedom ergodicity does not hold. Ergodicity roughly means that all evolutions in the long run come everywhere in the state space. Since the quasi-periodic orbits yield measure theoretically nontrivial invariant sets, ergodicity does not hold for the systems under consideration. In the case of an infinite number of degrees of freedom often an Ergodic Hypothesis is postulated and it is an unsolved problem to understand the limiting processes involved.

Later on, KAM-theory also became important for 'dissipative' systems depending on external parameters. Here again generally coexistence of quasi-periodic order and chaos holds. As stated earlier, this means that quasi-periodicity can indeed be a transient stage in a sequence of bifurcations from order to chaos. The behaviour of quasi-periodic attractors under variation of parameters was studied by, e.g., Broer, Huitema, Takens and Braaksma [5].

5. Main themes of the programme

We now turn to the NWO-SMC programme 'Mathematical aspects of nonlinear dynamical systems', which was carried out the last couple of years. The research projects involved here easily can be traced back to the history sketched above.

5.1. Resonance phenomena

Let a dynamical system be given, mechanical or otherwise, containing several oscillatory parts that are somehow linked. Then the term 'resonance' refers to an exceptional, though often strong, interaction between two or more of such parts. The simplest interaction involves the equality of frequencies of two oscillatory parts, but also other simple arithmetic relations between frequencies can occur. This is a classical setting for bifurcation theory: small changes of parameters may tune away from resonance, bringing about drastic changes of the dynamics. The programme contains several activities in this area. This research is relevant for technological applications.

Parametric resonance (Broer, Hoveijn (postdoc), Levi (guest from RPI, Troy NY)). A model problem is the following. Consider a pendulum, its point of suspension oscillating vertically. Suppose that we can change the corresponding period at will, so that we may consider it as a parameter. Here a strong form of so-called parametric resonance occurs whenever the period is near the period or near half the period of the pendulum itself. It turns out that for certain values of the system parameters the equilibrium of the pendulum becomes unstable!

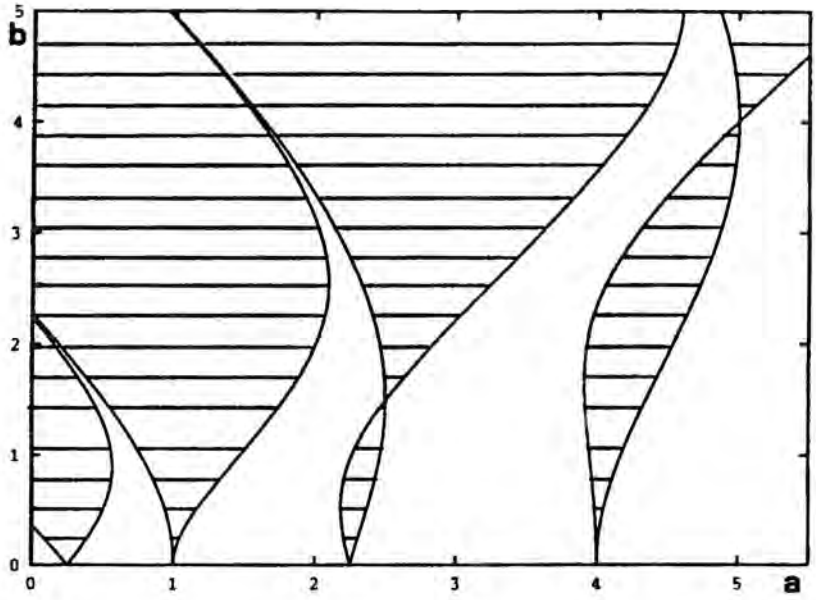


Figure 6. Pockets in the stability diagram of Square Hill's Equation.

One problem is how the dynamical possibilities are organized in parameter space (which here happens to be a plane). Often the regions of instability form tongues, sometimes exhibiting so-called instability-pockets, see figure 6. This is a complicated matter that has been the subject of research since the 1920's. The research of Broer and Levi [6] has contributed to the geometric insight in this.

A variation of this problem occurs when two of such pendulums are coupled by a weak string. If the resonance is such that the sum of the natural frequencies of the pendulums equals the frequency of the forcing, the system is stable only for parameter values in a narrow tongue. The geometric understanding of this phenomenon was enhanced by Hoveijn and Ruijgrok [14].

The fattened Arnol'd family (Broer, Simó & Tatjer (guests from the University of Barcelona), Viana (guest from IMPA, Rio de Janeiro)). V.I. Arnol'd is one of the leading members of the dynamical systems community. For a better understanding of — among other things — resonant dynamics, he has introduced a model system operating on a circle, which by now is well-understood. It seems that extensions of this model to the plane may play a central role in bifurcations to chaos, related to homoclinic points. As more

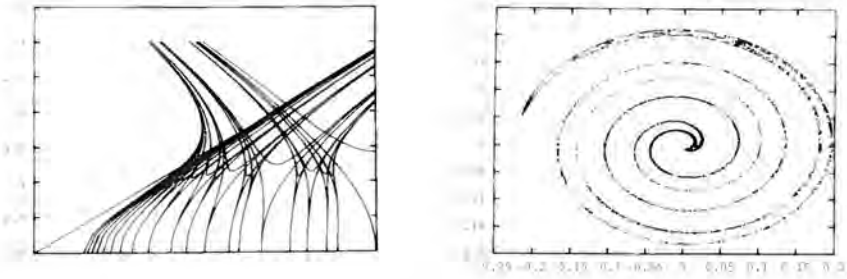


Figure 7. Complexity in the stability diagram (left) and a global Viana-attractor in the fattened Arnol'd family (right).

often, this 'fattened' model itself also turns out to contain a lot of chaos. These phenomena are strongly related to resonance, in particular where the resonance is about to disappear and where several resonance areas start to interact. Viana did theoretical work on this when visiting Groningen. Simó and Tatjer, together with Broer contributed to the understanding of this in a computer assisted way. Among other things, phenomena predicted by Viana's theory were found, see figure 7. Compare [24].

Generic 1:4 resonance (Broer, Krauskopf (Ph.D.-student), Takens). This is a study of all possible dynamical consequences of a loss of stability, associated to a frequency ratio of 1 to 4 (or 3 to 4, which can be reduced to the same problem). A corresponding study was carried out for all other integer ratio's in the 1970's by Arnol'd, Bogdanov, Carr, Khorozov and Takens, but the present case, which is much more complicated than the others, is still not completely solved. There is a conjecture by Arnol'd [3] which, when true, gives a complete solution of this case. In the present project a combination of analytical and numerical methods were used to study and visualize the consequences of the conjecture in terms of the bifurcation structure in a three-dimensional parameter space; see figures 8 and 9. In combination with the study of a certain singularity that acts as an organizing centre, this is convincing evidence in favour of the Arnol'd conjecture; see Krauskopf [15,16].

Resonance in adiabatically forced Hamiltonian systems (Huveneers (Ph.D.-student), Verhulst). Here we have a resonance problem in the context of conservative or frictionless mechanics with two parameters, one detuning the resonance and one related with the average energy. It turns out that here one can successfully transform to the quantummechanical formalism. Although

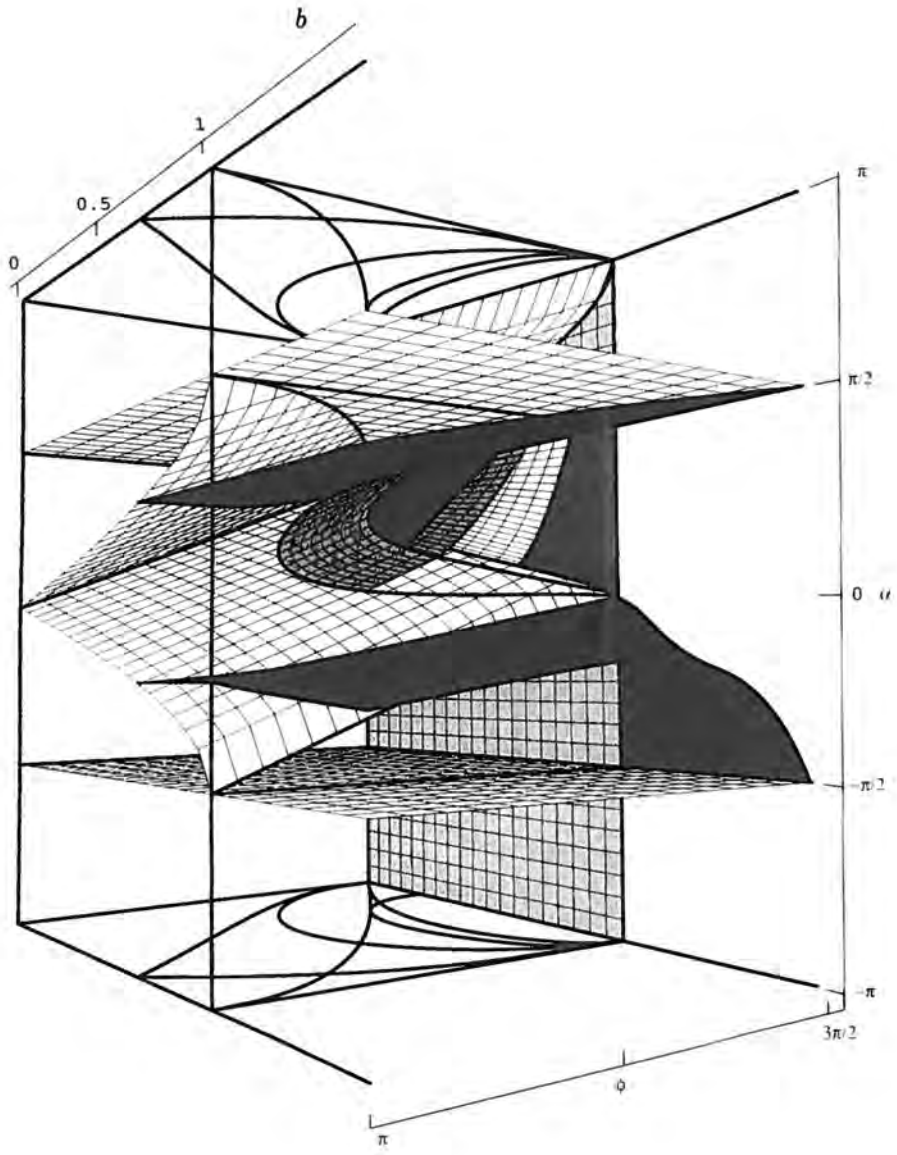


Figure 8. Bifurcation structure in 3D parameter space related to the 1:4 resonance.

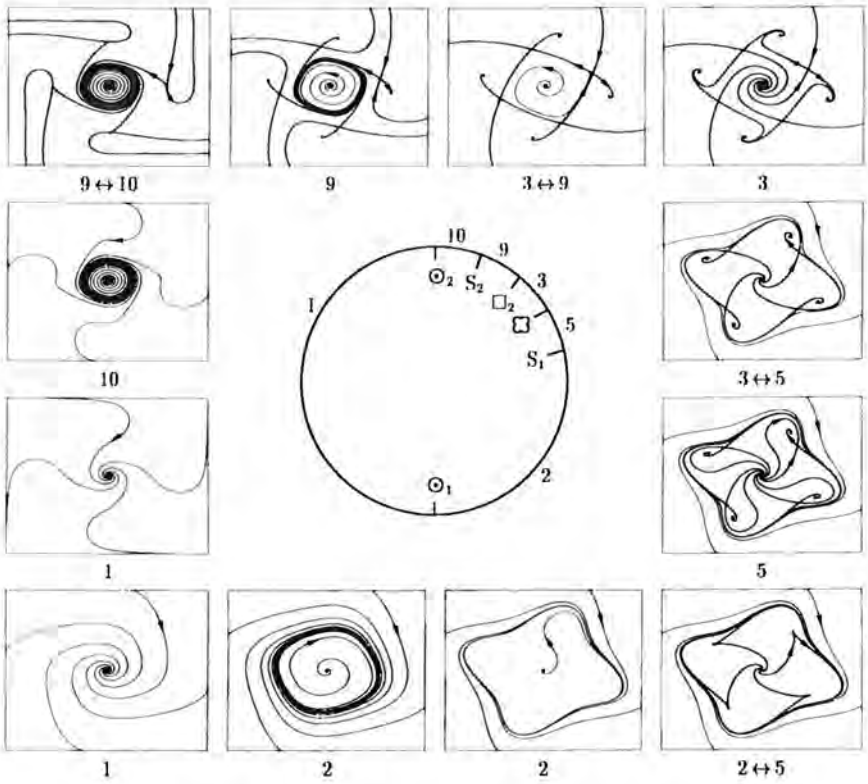


Figure 9. A sequence of phase portraits near 1:4 resonance.

this change of formalism changes the context from nonlinear and finite-dimensional to both linear and infinite-dimensional, the bifurcations can still be interpreted. The bifurcations are analysed in terms of the invariant subspaces of the associated infinite-dimensional Hilbert space.

5.2. Symmetry

Many natural systems exhibit some form of symmetry, which then determines the bifurcations, i.e., the drastic changes in the behaviour one can typically expect. Another reason for studying symmetric systems is that they are often integrable and hence can be used as a first step to study (approximate) more complicated systems. This holds in particular for the symmetric systems arising from truncation of higher order terms, in combination with normal form procedures.

Coupled Josephson junctions (Van Gils, Krupa, Tchistiakov (Ph.D.-student)). This project is concerned with the dynamical properties of a number of identical Josephson junctions, each two of them coupled in the same way. A Josephson junction is a gadget from the theory of superconductivity, modelled by a pendulum with oscillating point of suspension. This present system is invariant under any permutation of the Josephson junctions. The main question, which is related with the applications of these systems, is under which conditions the different junctions operate in a synchronous way. The investigations, which centre around a homoclinic bifurcation, are carried out both by analytical and numerical means.

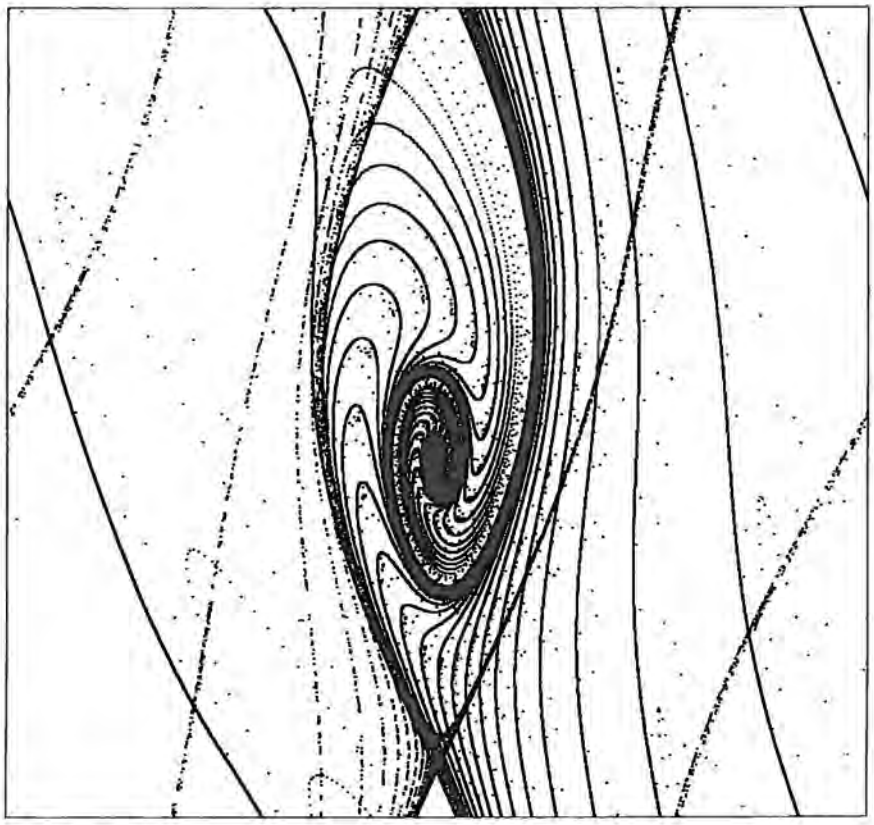


Figure 10. Chaos in the skew Hopf bifurcation.

Skew Hopf bifurcation (Broer, Takens, Wagener (Ph.D.-student)). This project started with the investigation of a transition to chaos in the presence of symmetry, which showed the possibility of mixed spectra for chaotic systems, see Broer and Takens [7]. Mixed spectra were observed in experiments, but no persistent mathematical example was known. The present investigations are concerned with the question what happens to this transition if the symmetry gets (slightly) broken. This needs a generalization of the KAM-theory of quasi-periodic motions. On the other hand, computer simulations indicate that new types of attractors are formed when the symmetry is broken. See figure 10.

Resonance and Symmetry (Hoveijn (postdoc)). Resonant systems can be considered as small perturbations of symmetric systems in the sense that their Taylor series have a formal symmetry, which is inherited by truncations. Therefore, in the setting of resonance it is quite natural to consider symmetric systems.

The presence of a symmetry group makes it possible to lower the dimension of the system by considering the dynamics on the orbit space of the group. For Hamiltonian systems one often can even lower the number of degrees of freedom. This approach raises some interesting problems. In general orbit spaces will have singularities. The first problem is to characterize these just from the symmetry group at hand. In many cases these singularities turn out to be rational, which facilitates this task. The second problem is to determine the global structure of the orbit space. This question is harder, involving real algebraic geometry. Apart from determining the nature of the orbit space there is the question of defining a dynamical system on a phase space with singularities.

For Hamiltonian systems many results have been obtained by Lerman & Sjamaar [17] and Arms et al. [1] where the symmetry group is a Lie group with linear action. More detailed results for particular resonant systems with two degrees of freedom were already found by Churchill et al. [8]. For these systems the singular reduction method is very powerful because here there are no global problems and the singularities are simple. Singular reduction for resonant Hamiltonian systems with more than two degrees of freedom is a subject of further research, for partial results see Hoveijn [13].

In systems of this form one often finds an interplay between various structures, such as a symplectic structure and a symmetry group. Research inspired by the coupled pendulums example led to the characterisation of infinitesimally reversible symplectic matrices.

One-dimensional dynamics (Van Strien, Kozlovsky (Ph.D.-student)). The most complete, and profound, mathematical results concern dynamical systems with only one degree of freedom. Although this is a rather restricted

class it has applications in biology and in fluid dynamics. The dynamics of these one-dimensional systems can be extremely complicated, but still is mathematically well understood. See De Melo and Van Strien [18]. The aim of this project is to try and extend the one-dimensional results to higher dimensions. In the special case of the Hénon map this already turned out to be possible, compare the fundamental work of Benedicks and Carleson [4].

This approach is strongly related to the analysis of the fattened Arnol'd family mentioned above.

5.3. Numerical tools and visualization

There is now software to analyse bifurcations of explicitely given systems (DS tool, AUTO and LOCBIF), but much needs to be done to integrate the possibilities of these programs, and to combine them with the normal form algorithms. The dynamical systems laboratory (DSL) at CWI has been active in this direction and has supported applications of this software, e.g., for the 1:4 resonance mentioned earlier, but also for investigations in population dynamics. The objects visualized are usually of dimension two, or at most three.

Other problems arise when studying higher-dimensional objects, like invariant manifolds. Both the calculation of these objects and their visualization require new methods.

Development of software and applications to population dynamics at DSL (Sanders, Kuznetsov, Levitin (NWO-visitor), Lisser, Kirkilionis, Hantke (Postgraduate)). One of the main projects in this group is the development of a new program CONTENT combining the advantages of the different now existing programs. There is an intensive cooperation with authors of previous programs, like Doedel (author of AUTO) and Kuznetsov (author of LOCBIF). Another, related theme is the computation of normal forms with Mathematica-based software. Partially these activities were also supported by the NWO priority programme 'Nonlinear Systems'.

From the applications outside mathematics we mention the bifurcation analysis of structured populations.

Computation and vizualization of invariant manifolds (Broer, Osinga (Ph.D. student), Vegter). The problem is to develop programs that numerically compute invariant manifolds, using (normal) hyperbolicity. A first contribution to this, by Homburg, Osinga and Vegter [12], computes the stable and unstable manifolds of a stationary point. The corresponding algorithm is based on the existence proofs of both Perron and Hadamard.

Research is in progress concerning a general normally hyperbolic case, where variations of the graph transform are employed. The corresponding algorithm is very suitable for continuation purposes and fills a gap in the

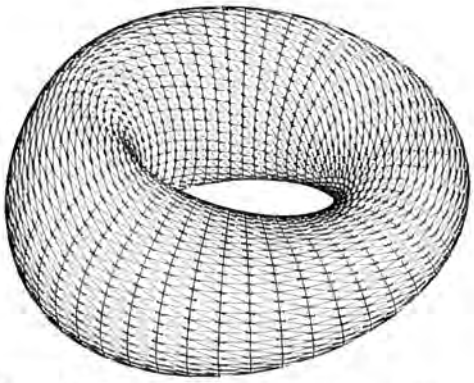


Figure 11. Invariant 2-torus for a fattening of the Thom-automorphism.

existing software. As an example of an invariant 2-torus in a 3-dimensional diffeomorphism, produced by this method, see figure 11.

Since the algorithms described here are based on a theoretical existence proof, analytic error bounds can be obtained while the methods can be used to provide computer assisted proofs of all kinds of dynamical features.

5.4. KAM-theory

Further development of KAM-theory turns out to be of general interest for the research involved. Within the scope of the present programme we mention the above project of the 'skew Hopf bifurcation', where KAM-theory has to be extended to investigate the persistence and bifurcation of certain quasi-periodic attractors. Also similar quasi-periodic bifurcations in conservative systems have to be studied as they occur, for instance, in the rigid body dynamics, see below.

One other project financed by FOM/SMC (via the National Mathematical Physics Community) deals with the Ergodicity Problem mentioned before (Broer, Van Enter, de Jong (Ph.D.-student), Takens, Winnink). Indeed, it considers a concrete infinite-dimensional lattice system as a limit of finite degree of freedom systems, investigating the fate of the quasi-periodic motions in the limiting process.

Again in the context of conservative systems there is a project, financed by Groningen University, dealing with two and three quasi-periodic motions of a rigid body which is a perturbation of the Euler top. (Broer, Cushman (Utrecht University), Haußmann (Ph.D.-student)). One tool for this problem is normal form theory, yielding an approximate system with a 2-torus symmetry. The reduced (slow) system in two-dimensions can be studied in its own right by singularity theory. This leads to quasi-periodic motions with two and three frequencies (including some bifurcations) in the integrable approximation. After this a KAM perturbation theory has to be carried out. See Haußmann [10,11].

Finally we mention the manuscript of a book by Broer, Huitema (PTT-research) and Sevryuk (guest from Russian Academy of Sciences, Moscow). This is a survey of KAM-theory in classes of systems determined by the

preservation of a given structure. Examples are given by the classes of conservative or dissipative systems mentioned before. Another example consists of reversible systems related to a given involution. The involution takes evolutions to evolutions, reversing the time-parametrization. Especially the minimal amount of parameters needed for persistence of quasi-periodic motions is of interest.

5.5. Methods and applications of nonlinear time series analysis
This area, which was not included in the present programme, but in the NWO priority programme 'Nonlinear Systems', is based on concepts from the theory of nonlinear dynamical systems. The idea is the following: chaotic systems behave like random systems, but is it possible to distinguish the two just by observing the dynamics? The answer turns out to be positive, but it requires new methods of time series analysis. Compare Takens [21.22]. These methods are also relevant for systems that are not completely deterministic. At this moment there is a project analysing the statistical aspects of these new methods (Borovkova (Ph.D.-student), Dehling, Takens) and there are two experimental groups applying these methods: the group of Van den Bleek et al. at Delft University of Technology applying this to the problems of design and operation of fluid bed reactors, and the group of De Goede et. al. at Leiden University applying this to physiological time series (EEG and ECG).

REFERENCES

1. J.M. ARMS, R.H. CUSHMAN, M.J. GOTAY (1991). A universal reduction procedure for Hamiltonian group actions. T. RATIU (ed.). *The Geometry of Hamiltonian Systems*, MRSI Workshop proceedings. Springer-Verlag.
2. V.I. ARNOL'D (1983). *Mathematical Aspects of Classical Mechanics*. Springer-Verlag.
3. V.I. ARNOL'D (1983). *Geometrical Methods in the Theory of Ordinary Differential Equations*. Springer-Verlag.
4. M. BENEDICKS, L. CARLESON (1991). The dynamics of the Hénon map. *Annals of Mathematics 133(1)*.
5. H.W. BROER, G.B. HUITEMA, F. TAKENS, B.L.J. BRAAKSMA (1990). Unfoldings and bifurcations of quasi-periodic tori. *Mem. A.M.S. 83(421)*.
6. H.W. BROER, M. LEVI (1995). Geometrical aspects of stability theory for Hill's equations. *Arch. Rat. Mech. An. 131*.
7. H.W. BROER, F. TAKENS (1993). Mixed spectra and rotational symmetry. *Arch. Rat. Mech. An. 124*.

8. R.C. Churchill, M. Kummer, D.L. Rod (1983). On averaging, reduction and symmetry in Hamiltonian systems. *J. Diff. Eqs. 49.*

9. M.J. Feigenbaum (1978). Quantitative universality for a class of nonlinear transformations. *J. Stat. Phys. 19.*

10. H. Hanßmann (1995). Normal forms for perturbations of the Euler Top. W. Langford (ed). *Normal Forms and Homoclinic Bifurcations.* Fields Institute Communications. American Mathematical Society.

11. H. Hanßmann (1995). *Quasi-Periodic Motion of a Rigid Body—A Case Study on Perturbations of Superintegrable Systems.* Ph.D. Thesis. Groningen University, in preparation.

12. A.J. Homburg, H.M. Osinga, G. Vegter (1995). On the computation of invariant manifolds of fixed points. *Z.A.M.P. 46.*

13. I. Hoveijn (1992). *Aspects of Resonance in Dynamical Systems.* Ph.D. Thesis. Utrecht University.

14. I. Hoveijn, M. Ruijgrok (1995). On the stability of parametrically driven coupled oscillators in sumresonance. *Z.A.M.P. 46.*

15. B. Krauskopf (1994). Bifurcation sequences at the 1:4 resonance: an inventory. *Nonlinearity 7.*

16. B. Krauskopf (1995). *On the 1:4 Resonance Problem. Analysis of the Bifurcation Set.* Ph.D. Thesis, Groningen University.

17. E. Lerman, R. Sjamaar (1991). Stratified symplectic spaces and reduction. *Annals of Mathematics 134(2).*

18. W.C. de Melo, S.J. van Strien (1993). *One Dimensional Dynamics.* Springer-Verlag.

19. J. Palis, F. Takens (1993). *Hyperbolicity and Sensitive Chaotic Dynamics at Homoclinic Bifurcations.* Cambridge Studies in Advanced Mathematics, 35.

20. D. Ruelle, F. Takens (1971). On the nature of turbulence. *Comm. Math. Phys. 20* and *23.*

21. F. Takens (1981). Detecting strange attractors in turbulence. *Dynamical Systems and Turbulence. Warwick 1980. LNM 898.* Springer-Verlag.

22. F. Takens (1993). Detecting nonlinearities in stationary time series. *J. Bifurcations and Chaos 3.*

23. R. Thom (1972). *Stabilité Structurelle et Morphogénèse.* Benjamin.

24. M. Viana. Strange attractors in saddle-node cycles: prevalence and globality. *Invent. Math.*, to appear.

CWI

Mathematical Epidemiology of Infectious Diseases

O. Diekmann

1. Introduction

Epidemiology is concerned with patterns in space and time of the occurrence of disease. From the patterns one may infer causes, predict the future and decide about the need for control measures. Many of such inferences require sophisticated statistics. When disease is caused by an infective agent, there is a second way in which mathematics may help to gain insight. Then one can build a mechanistic model for the spread of the agent and use it to disentangle how this spread is influenced by various factors, such as contact structure, population density, incubation period, etc. So one can do thought experiments where real experiments are impossible or unethical. Here the main mathematical tool is the qualitative theory of dynamical systems.

In a period of almost 30 years CWI has been actively engaged in the modelling and analysis of the spread of infectious diseases in structured populations of hosts. In the following we shall, in chronological order, very briefly present the main highlights.

2. Orientation on the classics

In 1971 CWI (then still called MC) started a (national and interdisciplinary) 'Working Group on Biomathematics' (as far as I know, the suggestion to start activities in this area came from F. van der Blij, then a member of the Board of Trustees; it was taken up by H.A. Lauwerier, P.J. van der Houwen, G.M. Willems, two Henker brothers and J. Grasman. This working group

has been a very strong catalyser for the development of mathematical biology in The Netherlands.

The subject of epidemiology was mainly brought in by J.A.J. Metz from the Institute of Theoretical Biology of Leiden University, who himself was inspired by J. Reddingius. Central was the work of W.O. Kermack and A.G. McKendrick in 1927, an early milestone which was so much ahead of its time that forty years later 'generalizations' were published which were actually special cases.

Most likely inspired by R. Ross (who, incidentally, received the Nobel prize for medicine in 1902 for his discovery that malaria is due to the *Plasmodium* parasite, transmitted via mosquitoes, and not due to 'bad air', as the name reflects) Kermack and McKendrick established in great generality the occurrence of a threshold phenomenon: the introduction of an arbitrarily small quantity of the infective agent in a demographically closed population can only trigger an epidemic when a certain compound parameter R_0 exceeds one (in Section 4 below we shall say much more about R_0). From a biological/medical point of view this, and the further characterisation of R_0, is all that matters. From a mathematical point of view, the 'arbitrarily small' is interesting and calls for a singular perturbation analysis. The problem is non-standard because it concerns an infinite-dimensional dynamical system: in fact it requires that one first resolves how one should think about initial value problems for Volterra integral equations of convolution type: once this is settled one can formulate a result in terms of the one-dimensionality of the intersection of an unstable manifold with a cone of positive functions. A second point is that in the introduction phase the deterministic approximation is not warranted, since numbers of infected are not large. The link is via branching processes.

The applied math outlook of the late seventies on the classical mathematical theory of epidemics is nicely summarized in the MC Tract 138 by Lauwerier [3].

3. Spatial spread

Guided by advisor L.A. Peletier (just then moving from Delft University to Leiden University) the applied math department organised in 1976 a colloquium on reaction-diffusion equations (see [1]) in which much attention was given to the then brand-new results of D.G. Aronson and H.F. Weinberger on the asymptotic speed of propagation of disturbances c_0. The idea is simple. A steady state, let's call it 0, is unstable and any (biologically) realizable perturbation, no matter how small, gives rise to a sequence of events (an orbit) which ends in a stable steady state, which we choose to call ∞. Examples include fires (combustion theory), epidemics, rumours and favourable mutant genes. How fast will the transition $0 \rightarrow \infty$ effectively take place? Remarkably, the question becomes more meaningful if

we add a spatial dimension. Then we can look for travelling plane waves, a special kind of self-similar solutions. (In a moving coordinate system the temporal transients look like 'frozen' spatial transitions!) It turns out that travelling plane waves exist for all speeds $c \geq c_0$ for some c_0 and that this minimal speed c_0 is the asymptotic speed of propagation in the sense that, for compactly supported initial disturbances, an observer moving with a speed higher than c_0 will be ahead of the transition, while an observer with a lower speed than c_0 will, eventually, experience state ∞. The following argument explains intuitively why the minimal wave speed equals the 'true' speed. By manipulating the initial condition suitably one can produce travelling waves in much the same way as one can create the illusion of steady movement in an array of electric lights by turning them on and off appropriately. Only one thing can spoil this game: if we try to make the speed too low the inherent 'infection' mechanism of our excitable medium takes over. Therefore this inherent infection speed is exactly the lowest possible wave speed!

The description above makes clear that application of these ideas to epidemic spread is all too natural. And in fact D.G. Kendall had already analyzed a special case (it has been told that Kendall obtained his results much earlier, but that he postponed publication because of the danger that they would be helpful for planning biological warfare). At almost the same time H.R. Thieme and Diekmann independently generalized the results of Aronson-Weinberger and Kendall to the Volterra-Fredholm integral equations describing general epidemic models. But at a meeting organized by the Dutch Society for Theoretical Biology, in which J.C. Zadoks of the Laboratory for Phytopathology of the Agricultural University of Wageningen presented the results of extensive simulations with a model for fungus disease spread by spore dispersal in various crops and formulated directly in computer language, while the present author presented his results in a theorem-proof style, a confusion of tongues of almost Babylonic dimension prevented effective interaction. The abstract results were not at all operational.

The next step was taken at the Institute of Theoretical Biology of Leiden University where F. van den Bosch, in a Ph.D. project guided by Metz, learnt both languages and thereby pulled the communication barrier down. In joint work with Zadoks and Metz he developed mechanistic submodels for spore dispersal, introduced flexible yet parameter sparse kernels for spore production, developed approximation formulae to determine c_0 from such ingredients with a pocket calculator in negligible time and showed that the model predictions match up to simulation studies and agree well with speeds measured in field experiments. All is well that ends well.

But in fact this was not the end. An unexpected follow-up started when the ecologist R. Hengeveld heard about these results at the inaugural ad-

dress in 1986 of the Metz-Diekmann tandem. Hengeveld was collecting and analyzing data on animal range expansions and realized that, with the appropriate interpretation of the ingredients, the results would directly carry over to this context. A fruitful collaboration originated.

4. The basic reproduction ratio R_0

For some time not much epidemiological happened at CWI. But then came AIDS and the modelling of infectious diseases became internationally a hot topic. The CWI group had no ambition to join this trend and continued to concentrate on the population dynamics of structured populations. But gradually it became clear that expertise in this area of structured populations could be used with great advantage when formulating and analyzing complicated epidemic models. And then J.A.P. Heesterbeek embarked upon a Ph.D. project and a colloquium was organized by him, Metz, Diekmann and two visitors, H. Inaba from Japan and M. Kretzschmar from Germany. Most attention was given to the basic reproduction ratio R_0 which, in biological words, is defined as the expected number of secondary cases produced by a *typical* infected individual during its entire infectious period, in a population consisting of susceptibles only. The 'typical' indicates that we take averages when individuals may differ in relevant aspects (e.g. sexual behaviour). Sometimes it is easy to average, sometimes it requires thought. In the present context the dynamics of disease transmission (i.e. the interplay of susceptibility and infectivity) determines how the averaging should be done, viz. by computing the positive eigenvector and eigenvalue of a next-generation operator (recall the Perron-Frobenius theory of positive matrices). All of this is explained in [2].

When $R_0 < 1$ the infective agent cannot invade into a virgin population of susceptible hosts, but when $R_0 > 1$ it can. This is exactly Kermack-McKendrick's threshold condition. When we know, much to our chagrin, by empirical fact that $R_0 > 1$, the threshold condition seems just a somewhat academic instrument to check our model. But once we realize that we need to bring, by control measures, R_0 to a value below one in order to eradicate an agent that is already established, it becomes clear that we can use R_0 as a practical instrument to estimate the effort needed for a successful eradication campaign.

5. Modelling the force of infection

More or less as an outgrowth of the colloquium CWI got a contract to provide mathematical modelling expertise for the project 'Population dynamics of infections' at the Central Veterinary Institute in Lelystad (now ID-DLO) which was started then by M.C.M. de Jong. At first the effort was directed at making the abstract definition of R_0 operational (in much the same way as Van den Bosch had done with the asymptotic speed of spread c_0) by de-

Figure 1. Mathematical models developed at CWI help to understand how farm or colony size affects the severity of an outbreak of a virus disease among pigs or seals. Photo's coutesy ID-DLO Lelystad (left) and IBN-DLO Texel (right).

veloping an algorithm to compute it in a special setting (motivated by the spread of Aujeszky's Disease Virus (ADV) among pigs on farms where the pigs are regularly shifted from one barn to another). A simple question ('When one farm is twice as large as another, what difference does it make for disease transmission?') initiated a new research direction, with field observations, lab experiments and theoretical modelling reinforcing each other.

The *force of infection* is by definition the probability per unit of time for a susceptible individual to become infected. Many viruses are transported from the mucus of one host to that of another by aerosoles when hosts 'meet'.

As a consequence one needs to model the contact process first, and then superimpose the transmission of the virus. The situation resembles that of chemical reactions, where molecules have to come close enough before they can react. Inspired by that similarity, standard deterministic epidemic models are in terms of densities and model the contact process by the law of mass action (which, in the present context, asserts that the force of infection is proportional to the density of infectives).

But in real life one often has to work with numbers, rather than densities. Indeed, both the size of a pig farm and the size of a colony of seals (see also figure 1) is usually expressed as the number of individuals that belong to it, while the density is roughly the same for each farm or each colony (in the latter case the evidence comes from aerial photos of seals sun bathing on sand banks).

It is not difficult at all to formulate and analyze a model in which density is a given constant, while population size is variable (over populations and, possibly, in the course of time). The first test on data from a classical experiment of Greenwood in 1936 (with mice suffering from *Pasteurella muris*, and living in a network of cages that was enlarged when population size increased) was inconclusive: the model with the per capita number of contacts per unit of time independent of population size, and the one for which this number was proportional to population size, could be made to fit the data with roughly the same accuracy. (Incidentally, this work was performed at the Isaac Newton Institute in Cambridge, as part of a special programme on mathematical epidemiology.

This finding prompted ID-DLO to perform experiments with ADV in groups of pigs, a large one in a large stable and a small one in a correspondingly smaller stable. Here the outcome was fortunately very clear: the hypothesis of proportionality of contact rate with population size had to be rejected.

Another convincing argument was found in the data about the spread of *Phocine Distemper Virus* (PDV) during the 1988 epidemic in the coastal waters of Northern-Europe: the final size appeared to be independent of colony size (an observation which had puzzled researchers applying the 'standard' model in which contact rate is proportional to population size.

Thus a combination of modelling considerations, mathematical analysis, experiments and observations helped to disentangle some aspects of the complicated relation between mechanisms at the individual level and phenomena at the population level.

6. Communication and education

It cannot be denied that a gap exists between general abstract mathematical theory and specific concrete real life situations. But one can try to make it smaller by organizing repeated exchanges of information about mo-

tivation, problems, ideas, data, methods, etc. Thus in 1995 a course on epidemic models was organized at CWI by Diekmann, Heesterbeek (now at ID-GLW), De Jong (ID-DLO), Kretzschmar (now at RIVM) and Metz. Participants came from all over the country and had backgrounds covering plant pathology, veterinary science as well as human epidemiology. Their number ranged from 40 at the beginning to 20 at the very end. The plan is to elaborate the notes to a book, hopefully with D. Mollison from Edinburgh as a sixth author and stochastic conscience. This book will then be the culmination of CWI's involvement in the development of mathematical epidemiology during a long period.

References

1. O. Diekmann, N.M. Temme (eds.). (1976). *Nonlinear Diffusion Problems*. MC Syllabus 28. Stichting Mathematisch Centrum, Amsterdam.
2. J.A.P. Heesterbeek (1992). R_0. Ph.D. Thesis, Leiden University.
3. H.A. Lauwerier (1981). *Mathematical Models of Epidemics*. MC Tract 138. Stichting Mathematisch Centrum, Amsterdam.
4. D. Mollison (ed.). (1995). *Epidemic Models: Their Structure and Relation to Data*. Cambridge Univ. Press.

Optimizing Transportation by Polyhedra

A. Schrijver

1. Introduction

Historically, there is a strong interaction between combinatorial optimization and polyhedral methods. The development of the basic polyhedral tool *linear programming* in fact was motivated to a large extent by application to transportation problems.

Linear programming studies minimizing (or maximizing) a linear function $c^T x$ over a given polyhedron P. A *polyhedron* is the solution set of a system $Ax \leq b$ of linear inequalities. Generally, an optimum solution is attained at a vertex of P.

The basic method, the *simplex method*, was already described by Fourier: make a trip along the vertices and edges of the polyhedron, throughout decreasing $c^T x$, until an optimum vertex is reached. What makes the simplex method interesting is that it applies if the polyhedron is given by a set of defining inequalities. Listing all vertices is not necessary; it would also be impracticable for most problems because of the huge number of vertices (while there is a relatively small number of defining inequalities).

The interaction with combinatorial applications originates from the founding fathers of linear programming, L.V. Kantorovich, F.L. Hitchcock, Tj.C. Koopmans, and G.B. Dantzig, who independently designed polyhedral tools during the years 1939-1949. (Dantzig introduced the terms linear programming and simplex method.)

2. Origins of transportation and linear programming problems

In 1939, Kantorovich introduced the idea of linear programming. He gave a wealth of practical applications, which he motivated by the Soviet plan economy.

One of the applications mentioned by him is a problem now known as the *multicommodity flow problem*:

Let there be several points A, B, C, D, E, which are connected to one another by a railroad network (see figure 1). It is possible to make the shipments from B to D by the shortest route BED, but it is also possible to use other routes as well: namely $BC'D$, BAD. Let there also be given a schedule of freight shipments: that is, it is necessary to ship from A to B a certain number of carloads, from D to C' a certain number, and so on. The problem consists of the following. There is given a maximum capacity for each route under the given conditions (it can of course change under new methods of operation in transportation). It is necessary to distribute the freight flows among the different routes in such a way as to complete the necessary shipments with a minimum expenditure of fuel, under the condition of minimizing the empty runs of freight cars and taking account of the maximum capacities of the routes.

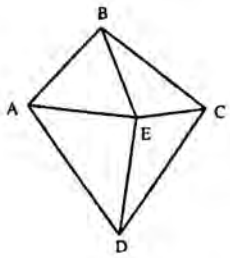


Figure 1.

Independently, at about the same time, Hitchcock at the Massachussetts Institute of Technology introduced and studied the *transportation problem*: given a nonnegative $m \times n$ matrix $C = (c_{i,j})$ and vectors $s \in \mathbb{R}^m_+$ and $d \in \mathbb{R}^n_+$, find a nonnegative $m \times n$ matrix $X = (x_{i,j})$ satisfying

$$\sum_{j=1}^{n} x_{i,j} = s_i \text{ for } i = 1, \dots, m, \tag{2.1}$$

$$\sum_{i=1}^{m} x_{i,j} = d_j \text{ for } j = 1, \dots, n.$$

and minimizing $\sum_{i,j} c_{i,j} x_{i,j}$.

The interpretation is that there are m factories and n customers. Factory i can produce a quantity of s_i of a certain product, while customer j needs a quantity of d_j of this product. The cost of transporting one unit of the product from factory i to customer j is equal to $c_{i,j}$. Then $x_{i,j}$ gives the amount transported from factory i to customer j, at minimum total cost.

Thus the transportation problem is a problem of minimizing a linear function over the polyhedron determined by (2.1) (and by the nonnegativity constraints) — the *transportation polyhedron*. It is a polyhedron in nm-space, and the optimum matrix X can be found by making a trip along the vertices of this polyhedron. Hitchcock showed a simple procedure for doing this.

Independently of Kantorovich and Hitchcock, also Koopmans studied transportation problems. During the Second World War, Koopmans was as a statistician on the staff of the Combined Shipping Adjustment Board, a British-American agency dealing with merchant shipping problems during the Second World War. Influenced by his teacher J. Tinbergen, he was interested in ship freights and capacities. At the Board he studied the assignment of ships to convoys so as to accomplish prescribed deliveries, minimizing empty voyages. Koopmans found his results in 1943, but due to wartime restrictions he published them only after the war. (See figure 2.)

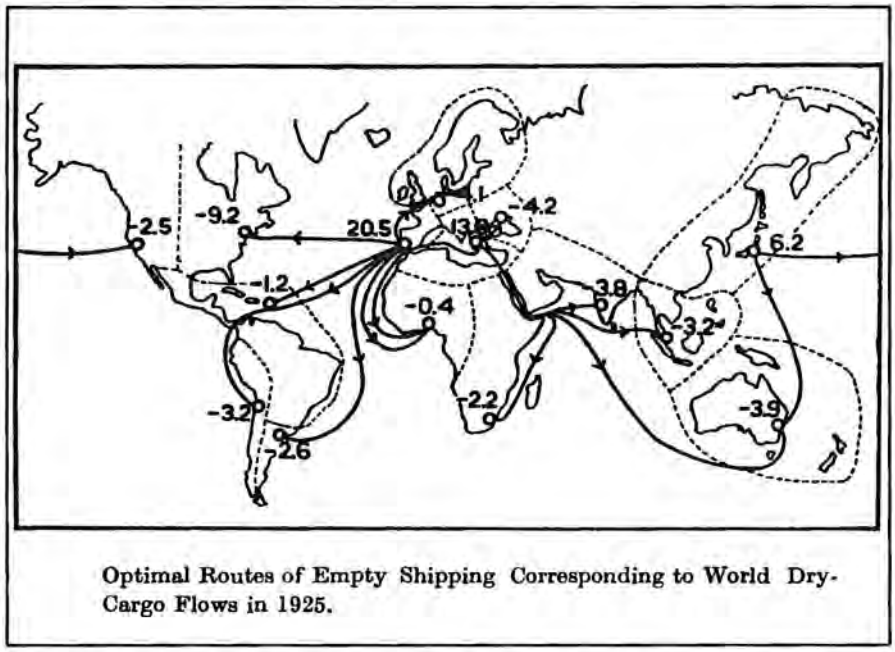
In the second half of the 1940's, the work of Dantzig gave the breakthrough of linear programming, especially due to his description of the simplex method in a compact tableau-form with an easy 'pivoting' rule. Dantzig also observed that if the method is applied to the Hitchcock-Koopmans transportation problem and if the supplies and demands are integer-valued, then there exists an optimum solution X that is integer-valued.

This makes it possible to apply linear programming methods to several other combinatorial optimization problems, in particular to problems where articles are indivisible, like the *optimum assignment problem* — the problem of assigning men (or machines) to jobs so as to minimize costs: given an $n \times n$ matrix $C = (c_{i,j})$, find a permutation π of $\{1, \ldots, n\}$ minimizing $\sum_{i=1}^{n} c_{i,\pi(i)}$. In a different terminology, it asks for a minimum-weight perfect matching in a bipartite graph.

The assignment problem is a special case of the transportation problem, as it is equivalent to minimizing $\sum_{i,j} c_{i,j} x_{i,j}$ over all nonnegative matrices $X = (x_{i,j})$ satisfying

$$\sum_{j=1}^{n} x_{i,j} = 1 \text{ for all } i, \text{ and } \sum_{i=1}^{n} x_{i,j} = 1 \text{ for all } j. \tag{2.2}$$

Nonnegative matrices satisfying (2.2) are called *doubly stochastic*, and the theorem behind is the *Birkhoff-von Neumann theorem*: each doubly stochastic matrix is a convex combination of permutation matrices. That is, each vertex of the polyhedron determined by (2.2) has integer coordinates only.



Optimal Routes of Empty Shipping Corresponding to World Dry-Cargo Flows in 1925.

Figure 2. A map, presented in the 1947 paper by Tj.C. Koopmans, one of the earliest studies on optimization in transportation.

So linear programming automatically gives an integer optimum solution X, and such an integer matrix is a permutation matrix.

3. The traveling salesman problem

Having such efficient methods for the assignment problems, one is tempted to try similar methods to similar problems, and one of the most challenging turned out to be the *traveling salesman problem*: given an $n \times n$ matrix C, find a cyclic permutation π of $\{1, \ldots, n\}$ minimizing $\sum_{i=1}^{n} c_{i,\pi(i)}$. (A permutation is *cyclic* if it has exactly one orbit.)

Solving the traveling salesman problem is not only mathematically intriguing, but also of high practical importance due to its numerous occurrences in practice, in several forms (like in vehicle routing and production planning). M.M. Flood was one of the first studying the traveling salesman problem for practical purposes. He considered the problem in 1937 in relation to the routing of school buses. According to Flood, the idea of using polyhedral methods to solve the traveling salesman problem was

brought to his attention by Koopmans in 1948. Flood next popularized this approach at the RAND Corporation, the intellectual centre of operations research, where several pioneers of linear programming and transportation were employed, and where polyhedral tools were successfully utilized, by Dantzig, L.R. Ford Jr., and D.R. Fulkerson, to solve flow problems coming from routing trains.

If we wish to solve the traveling salesman problem with polyhedral methods, the question arises what are the inequalities describing the *traveling salesman polytope*, that is, the convex hull of the cyclic permutation matrices.

We can take the equalities (2.2) as a basis, but they are obviously not enough, as each noncyclic permutation matrix satisfies (2.2). The noncyclic permutation matrices can be excluded by adding the following *subtour elimination constraints*:

$$\sum_{i\in I, j\notin I} x_{i,j} \geq 1 \text{ for each } I \subseteq \{1,\dots,n\} \text{ with} \qquad (3.1)$$

$$\emptyset \neq I \neq \{1,\dots,n\}.$$

It would be very nice if adding these constraints gives a complete description of the traveling salesman polytope; that is, if the polyhedron determined by (2.2) and (3.1) has integer vertices only. It would enable us to use the simplex method to solve the traveling salesman problem; moreover, it would imply (as we shall see below) that the traveling salesman problem is solvable in polynomial time. This is possible because, although the number of constraints in (3.1) grows exponentially with n, these yet can be checked in polynomial time: given a doubly stochastic matrix X, we can test in polynomial time if it satisfies (3.1) (by a reduction to minimum-capacity cut computations).

However, while the inequalities (3.1) are enough to cut off the noncyclic permutation matrices from the polytope of doubly stochastic matrices, they yet do not yield all facets of the traveling salesman polytope (if $n \geq 5$). There exist doubly stochastic matrices, of any order $n \geq 5$, that satisfy (3.1) but are not a convex combination of cyclic permutation matrices.

This disappointing fact has stimulated a stream of research. In a seminal paper of Dantzig, Fulkerson, and S.M. Johnson (1954) (according to A.J. Hoffman and Ph. Wolfe 'one of the principal events in the history of combinatorial optimization'), several new methods for solving the traveling salesman problem were introduced that are basic in combinatorial optimization up to today.

One of their basic observations is that, although we do not know a full description of the traveling salesman polytope, we obtain a lower bound for the minimum tour length if we optimize over the constraints (2.2) and (3.1).

Figure 3. Optimal world-tour (666 cities).

This lower bound can be calculated with the simplex method, taking the
214 (exponentially many) constraints (3.1) as *cutting planes* that can be added during the process when necessary. In this way, Dantzig, Fulkerson, and Johnson were able to find the shortest tour along cities in the 48 U.S. states and Washington, D.C. (See figure 3.)

This general approach has turned out to be extremely fruitful also in attacking other combinatorial optimization problems. In particular, around 1965, J. Edmonds at the National Bureau of Standards showed the applicability of the method, as an exact method, to many important classes of problems, most prominently matching problems.

4. Complexity

Edmonds also advertized *polynomial-time solvability* as a touchstone of the complexity of a problem. An algorithm is called *polynomial-time* if the num-

ber of steps is bounded by a polynomial in the size of the input. Generally, such an algorithm is fast in practice.

Polynomial-time algorithms for the assignment and transportation problems (the 'Hungarian method' and extensions) were designed in the 1950's and 1960's, but no such algorithm was found for the traveling salesman problem.

Note that checking all possible traveling salesman tours isn't a polynomial-time method, since there are exponentially many $((n-1)!)$ cyclic permutations. (Observe also that the number of feasible (not necessarily optimum) solutions is not a measure for the complexity of a problem: the number of cyclic permutations is smaller than the number of all permutations $(n!)$, but yet selecting an optimum permutation (the assignment problem) turns out to be easier than selecting an optimum cyclic permutation (the traveling salesman problem).)

The general feeling that the traveling salesman problem is much more difficult than the transportation and assignment problem, got mathematical foundation at the start of the 1970's, by the work of S.A. Cook and R.M. Karp on the complexity classification of problems. The introduction of the complexity classes P and NP gave a key to distinguish problems on their complexity. The class P consists of all problems that can be solved in polynomial time; the transportation and the assignment problem belong to this class.

The class NP is potentially much wider than P. It includes all optimization problems with the property that it has an optimum solution the feasibility of which can be checked in polynomial time. About any combinatorial optimization problem belongs to NP, for instance the traveling salesman problem. What Karp showed was that the traveling salesman problem, and many other important combinatorial optimization problems, are the hardest in the class NP; in technical terms, they are NP-*complete*. It means that each problem in NP can be reduced, in polynomial time, to the traveling salesman problem. So if NP≠P then the traveling salesman problem is not solvable in polynomial time.

The definition of NP is very little restrictive, and there is no reason to believe that NP=P. But as yet, no mathematical proof has been found that these classes are really different. Beside being an intriguing mathematical problem, knowing whether NP≠P holds is also of practical importance. If NP=P can be proved, it might imply a revolutionary new algorithm, or, alternatively, it might mean that the concept of polynomial-time algorithm is completely meaningless. If NP≠P can be shown, the proof might give a clue why certain problems are harder than other, and might direct us to attack the kernel of the problems.

Thus the traveling salesman problem is pivotal — if it is polynomial-time solvable, then about any combinatorial optimization problem is polynomial-

time solvable. What impact does it have on the value of the polyhedral method? Maybe there is an alternative method that gives us a polynomial-time algorithm.

Only in 1979 it was shown that the class of linear programming problems belongs to P. This was shown by L.G. Khachiyan in 1979, by adapting the *ellipsoid method* for nonlinear programming. This implies a polynomial-time algorithm for any combinatorial optimization problem if we know all inequalities describing the corresponding polyhedron. But in what sense do we need to know them?

In 1981, M. Grötschel, L. Lovász, and A. Schrijver showed that one does not need an explicit list of inequalities describing the polyhedron. It suffices to be able to solve the *separation problem* for the corresponding polyhedron P in polynomial time: given a vector x, does x belong to P, and if $x \notin P$, find a hyperplane separating x and P.

The separation problem trivially is polynomial-time solvable if we have an explicit list of all inequalities determining P completely written out before us—in that case we can check them one by one; but it is not necessary to have such a list. In fact, what is shown is that the polynomial-time solvability of a combinatorial optimization problem is *equivalent* to the polynomial-time solvability of the separation problem. Thus an appropriate description of the corresponding polytope is necessary and sufficient for the polynomial-time solvability of the optimization problem.

For the traveling salesman problem it means that it is polynomial-time solvable if and only if there is a polynomial-time algorithm checking if a given doubly stochastic matrix belongs to the traveling salesman polytope. Thus the polyhedral approach is in a sense necessary and sufficient.

5. Computational and other work

In any case, the polyhedral method can give very good bounds, especially if we include it in a *branch-and-bound* algorithm. A branch-and-bound algorithm consists of a branched case checking (by setting entries in the variable matrix X to 0 or 1), guided by lower bounds calculated for each case considered. At each iteration, the case with the smallest lower bound is split into two new cases, by setting a new entry to 0 and to 1 respectively. Having good lower bounds that can be calculated fast is essential for the computational behaviour of a branch-and-bound method.

To obtain better lower bounds, more facet-defining inequalities can be added to (3.1), so as to describe the traveling salesman polytope more and more accurate. To shortcut the branching process, heuristics finding and improving tours can be incorporated, so that 'hopeless' branches can be eliminated.

Such methods form the basis for the continuous progress in solving large-scale traveling salesman problems. Around 1980, Grötschel found an op-

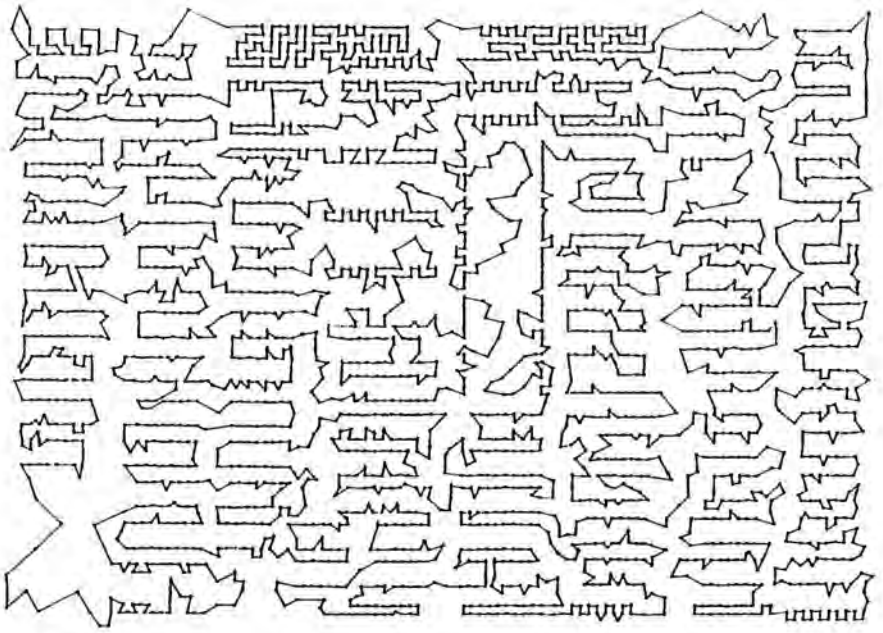


Figure 4. Record optimum tour along 3038 holes in a printed circuit board (1993).

timum tour along 120 cities in the Federal Republic of Germany, and H. Crowder and M. Padberg found one along 318 holes in a certain printed circuit board. Subsequent improvements during the last 15 years have led to the solution of a 7397 city problem by D. Applegate, R.E. Bixby, V. Chvátal, and W.J. Cook in 1994. (See also figure 4.)

At CWI research has been done on the complexity analysis of methods for the traveling salesman and related problems and on practical applications. For Van Gend & Loos, a Dutch transport company, the vehicle-routing system CAR (Computer-Aided Routing) was developed and installed. Important ingredient is a method to solve a traveling salesman problem with time-windows; that is, each 'city' can be visited only during a certain time period. (See figure 6.)

Moreover, research at CWI on polyhedral methods in combinatorial optimization has been pointed to identifying problem classes that are polynomial-time solvable with the polyhedral method. These problems include disjoint paths and trees, in particular in relation to routing wires on a VLSI-circuit.

At a more elementary level, CWI developed polyhedral methods for the problem of the most economical circulation of railway stock, a problem pre-

ride number		2123	2127	2131	2135	2139	2143	2147	2151	2155	2159	2163	2167	2171	2175	2179	2183	2187	2191
Amsterdam	V		6.48	7.55	8.56	9.56	10.56	11.56	12.56	13.56	14.56	15.56	16.56	17.56	18.56	19.56	20.56	21.56	22.56
Rotterdam	A		7.55	8.58	9.58	10.58	11.58	12.58	13.58	14.58	15.58	16.58	17.58	18.58	19.58	20.58	21.58	22.58	23.58
Rotterdam	V	7.00	8.01	9.02	10.03	11.02	12.03	13.02	14.02	15.02	16.00	17.01	18.01	19.02	20.02	21.02	22.02	23.02	
Roosendaal	A	7.40	8.41	9.41	10.43	11.41	12.41	13.41	14.41	15.41	16.43	17.43	18.42	19.41	20.41	21.41	22.41	23.54	
Roosendaal	V	7.43	8.43	9.43	10.45	11.43	12.43	13.43	14.43	15.43	16.45	17.45	18.44	19.43	20.43	21.43			
Vlissingen	A	8.38	9.38	10.38	11.38	12.38	13.38	14.38	15.38	16.38	17.40	18.40	19.39	20.38	21.38	22.38			

ride number		2108	2112	2116	2120	2124	2128	2132	2136	2140	2144	2148	2152	2156	2160	2164	2168	2172	2176
Vlissingen	V			5.30	6.54	7.56	8.56	9.56	10.56	11.56	12.56	13.56	14.56	15.56	16.56	17.56	18.56	19.55	
Roosendaal	A			6.35	7.46	8.50	9.50	10.50	11.50	12.50	13.50	14.50	15.50	16.50	17.50	18.50	19.50	20.49	
Roosendaal	V		5.29	6.43	7.52	8.53	9.53	10.53	11.53	12.53	13.53	14.53	15.53	16.53	17.53	18.53	19.53	20.52	21.53
Rotterdam	A		6.28	7.26	8.32	9.32	10.32	11.32	12.32	13.32	14.32	15.32	16.32	17.33	18.32	19.32	20.32	21.30	22.32
Rotterdam	V	5.31	6.29	7.32	8.35	9.34	10.34	11.34	12.34	13.35	14.35	15.34	16.34	17.35	18.34	19.34	20.35	21.32	22.34
Amsterdam	A	6.39	7.38	8.38	9.40	10.38	11.38	12.38	13.38	14.36	15.38	16.40	17.38	18.36	19.38	20.38	21.38	22.38	23.38

Table 1

Table 2

train number	2123	2127	2131	2135	2139	2143	2147	2151	2155	2159	2163	2167	2171	2175	2179	2183	2187	2191
Amsterdam-Rotterdam		47 340	100 616	61 407	41 336	31 282	46 287	42 297	33 292	39 378	84 527	109 616	78 563	44 320	28 184	21 161	28 190	10 123
Rotterdam-Roosendaal	4 58	35 272	52 396	41 364	26 240	25 221	27 252	27 267	28 287	52 497	113 749	98 594	51 395	29 254	22 165	13 130	8 77	
Roosendaal-Vlissingen	14 328	19 181	27 270	26 237	24 208	32 188	15 160	21 195	23 290	41 388	76 504	67 381	43 276	20 187	15 136			

train number	2108	2112	2116	2120	2124	2128	2132	2136	2140	2144	2148	2152	2156	2160	2164	2168	2172	2176
Vlissingen-Roosendaal			28 138	100 448	48 449	57 436	24 224	19 177	19 184	17 161	19 165	22 225	39 332	30 309	19 164	15 142	11 121	
Roosendaal-Rotterdam		16 167	88 449	134 628	57 397	71 521	34 281	26 214	22 218	21 174	25 206	35 298	51 422	32 313	20 156	14 155	14 130	7 64
Rotterdam-Amsterdam	7 61	26 230	106 586	105 545	56 427	75 512	47 344	36 303	32 283	34 330	39 338	67 518	74 606	37 327	23 169	18 157	17 154	11 143

Figure 5. Commissioned by Nederlandse Spoorwegen, CWI computed an optimum circulation plan for rolling stock on the Amsterdam-Vlissingen line.

Figure 6. Computer-Aided Routing (CAR) is an interactive software package used as a support tool for physical distribution. CAR was jointly designed and developed by CWI and the Dutch road haulier Van Gend & Loos.

sented by Nederlandse Spoorwegen (see figure 5). This leads us back to a multicommodity flow problem, one of the original motivations of Kantorovich. The problem consists of determining the minimum amount of rolling stock to be purchased by NS in order to guarantee a given number of seats in each of the scheduled train legs. If there would be only one type of stock, one could solve the problem directly with linear programming methods, as it would automatically yield integer solutions. The question of NS was to extend the method to the case where units of several types are available, that can be coupled together. The original circulation problem (with lower bounds) then becomes a multicommodity circulation problem. The solutions are restricted to be integer-valued, since one cannot break train-units.

However, in that case, applying linear programming does not automatically give an optimum solution that is integer-valued. Thus we were bound to embed the polyhedral method in a branch-and-bound framework. To make it work, a number of cutting planes had to be added. This gives an algorithm that finds an optimum circulation plan for the Amsterdam-Vlissingen line (with 99 scheduled legs) within a few seconds.

For background information we refer to the books mentioned below.

References

1. M. Grötschel, L. Lovász, A. Schrijver (1988). *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin.

2. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys

(eds.). (1985). *The Traveling Salesman Problem—A Guided Tour of Combinatorial Optimization*, Wiley, Chichester.

3. J.K. Lenstra, A.H.G. Rinnooy Kan, A. Schrijver (eds.) (1991). *History of Mathematical Programming—A Collection of Personal Reminiscences*, CWI/North-Holland, Amsterdam.
4. M.W.P. Savelsbergh (1992). *Computer Aided Routing*, CWI Tract 75, CWI, Amsterdam.
5. A. Schrijver (1986). *Theory of Linear and Integer Programming*, Wiley, Chichester.

Queueing Theory

O.J. Boxma

1. Introduction

Queueing phenomena occur in several real-life situations when resources (machines at a factory, elevators, telephone lines, traffic lights) cannot immediately render the amount or the kind of service required by their users. Also, at byte level in modern data-handling technologies (communication systems, computer networks) queueing phenomena may arise; they are typically less visible but their effects at user level are usually not less serious. Quite often such congestion effects may be adequately studied by mathematical methods from *queueing theory*. Adopting the abstract terminology from queueing theory, the object of study is formulated as a network of service units with customers requiring services at those units. The nature of the arrival processes and service requests is usually such that they have to be represented by stochastic processes. Hence the most important performance measures, like waiting times, workloads and queue lengths, are random variables. Accordingly, the main techniques of queueing theory stem from probability theory.

In Section 2 we discuss some elementary phenomena and results from queueing theory. Section 3 contains a brief history of the past 50 years of queueing theory, with some of the applications that guided its development. Section 4 is devoted to polling systems, a class of queueing systems that recently has received much attention in the literature and at CWI.

2. Elementary queueing theory

The influence of randomness on queueing processes is often remarkably strong, and sometimes at first sight counterintuitive. For example, if a hairdresser spends exactly 12 minutes on each customer, and his customers arrive at intervals of exactly 15 minutes, then no customer has to wait. But if these same customers arrive according to a Poisson process with an arrival rate of 4 customers per hour, then they wait 24 minutes on average! And if, moreover, the service time $\mathbf{S}$ spent per customer is negative exponentially distributed with mean 12 minutes (i.e. the probability that $\mathbf{S}$ exceeds x equals $\Pr\{\mathbf{S} > x\} = e^{-x/12}, x \geq 0$), then the mean waiting time of a customer doubles to 48 minutes.

If, in the latter case, an observer enters the shop at a randomly chosen point in time during a service, he might think that the expected time until completion of that service is one half of 12 minutes, i.e. 6 minutes. But it is easily seen that the probability that $\mathbf{S}$ will exceed $x + y$, given that it has already exceeded x, equals $e^{-(x+y)/12}/e^{-x/12} = e^{-y/12}$, which is the probability that $\mathbf{S}$ exceeds y: this is called the memoryless property of the exponential distribution. Hence the residual time until service completion has exactly the same exponential distribution as $\mathbf{S}$ itself, and its mean is 12 minutes instead of 6. For generally distributed service times, the mean residual service time equals $E\mathbf{S}^2/2E\mathbf{S}$, which exceeds $E\mathbf{S}/2$ when service times are not constant. The intuitive explanation of this phenomenon, the

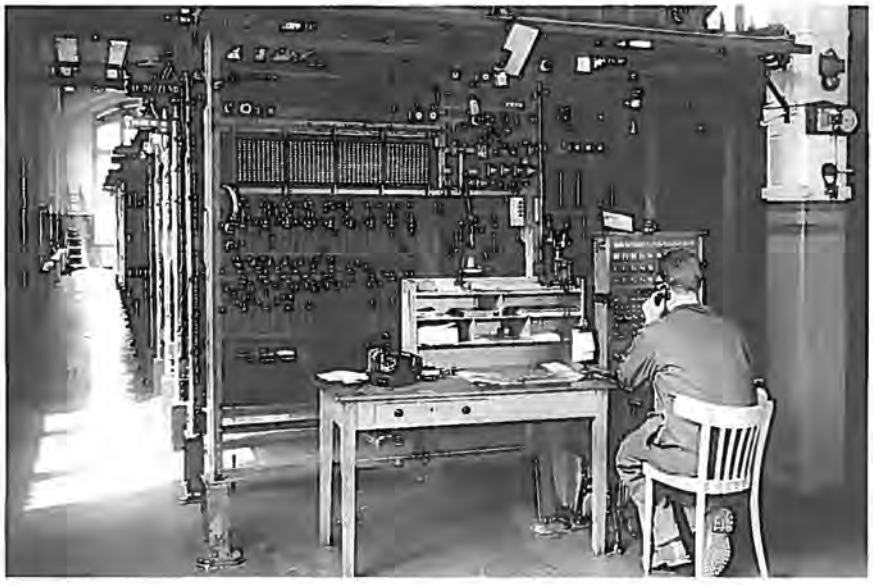


Figure 1. Queueing theory originated early this century in the study of overload in telephone exchanges. (Photo: PTT Telecom.)

'waiting time paradox', is that the observer has a relatively high probability of entering the shop during a relatively long service (the same phenomenon explains why, on average, at a bus stop one has to wait longer than half the interarrival interval indicated by the time table).

The above-mentioned memoryless property is very attractive from a mathematical viewpoint: Given the present, it allows one to disregard the past in studying the future. This is the characteristic of Markov processes, for which a rich literature has been developed in probability theory. At an early stage of queueing theory, in the first half of this century, it allowed the exact analysis of a whole class of 'Markovian' queueing systems. Examples are the loss and delay models developed and analysed by the Danish queueing pioneer A.K. Erlang (1909) with the purpose of dimensioning telephone exchanges (see also figure 1). Another example is the M/M/1 queue (see Box).

The M/M/1 queue

This is a queue with Markovian or memoryless (M) interarrival times and service times, and a single (1) server. The number of customers $\mathbf{X}$ in the M/M/1 queue is also a Markov process, and its steady-state distribution is geometric, i.e., again memoryless: $\Pr\{\mathbf{X} = n\} = (1 - \lambda/\mu)(\lambda/\mu)^n$, $n = 0, 1, \cdots$. Here λ is the arrival rate and μ the service rate. The expected number of *waiting* customers equals $\mathbf{E}\mathbf{X}_w = \mathbf{E}\mathbf{X} - \lambda/\mu = \lambda^2/(\mu(\mu - \lambda))$. Little's formula gives a (very generally valid) relation between the mean number of waiting customers $\mathbf{E}\mathbf{X}_w$ and the mean waiting time $\mathbf{E}\mathbf{W}$:

$$\mathbf{E}\mathbf{X}_w = \lambda \mathbf{E}\mathbf{W}, \tag{2.1}$$

so that for the M/M/1 queue

$$\mathbf{E}\mathbf{W} = \frac{\lambda/\mu^2}{1 - \lambda/\mu}. \tag{2.2}$$

One can now verify that the mean waiting time in the example discussed in the beginning of this section indeed equals 48 minutes.

3. Post-war queueing theory

3.1. Development of queueing theory

The post-war technological revolution and the resulting new attitude towards pure and applied science had a strong influence on the development of queueing theory. Mathematical modelling in economics and management had always been seriously hampered by the difficulties encountered in the

numerical evaluation of the analytical models. The development of powerful computing machinery lowered this 'numerical' barrier rather abruptly. A new discipline evolved: Operations Research, with queueing theory as one of its fastest growing branches.

Not only as a branch of Operations Research, but also as a branch of Applied Probability, queueing theory attracted the interest of professional mathematicians after the Second World War. Around 1950, the mathematical theory of stochastic processes had reached a certain maturity. Brownian motion and noise phenomena, investigated by physicists in the first quarter of the century, and biological processes such as the development of epidemics and the growth of bacteria populations appeared to be accessible for probabilistic modelling. Around 1950, too, monographs on stochastic processes became available, and the appearance of Feller's famous book *An Introduction to Probability Theory and its Applications* turned out to be a landmark in the development of stochastic modelling. Under the influence of Feller's exposition the techniques required for the analysis of stochastic models were systemized, and were investigated on their merits and on their potential for obtaining numerical results. The influence of these developments on queueing theory was strong, the more so since queueing models turned out to be a gratifying testing ground for many techniques developed in subfields of Probability Theory like Renewal Theory, Birth-and-Death Processes, Branching Processes, Fluctuation Theory and, in particular, (semi-)Markov Processes.

Figure 2. D. van Dantzig.

In The Netherlands, D. van Dantzig (one of the founding fathers of SMC, see figure 2) began, shortly after the Second World War, to teach courses in probability and mathematical statistics. His teaching has been very influential on the development of these fields in The Netherlands. In particular he may be considered to have laid the foundations for the strong international position of Dutch applied probabilists. Van Dantzig's interest in the application of mathematics has also been a stimulus to the development of probabilistic Operations Research in The Netherlands.

In 1953, D.G. Kendall published what was to become one of the most influential papers in queueing theory [4]. In it he analyzed the M/G/1 queue—a single server queue with Poisson arrival process and *generally* (G) distributed service times. The queue length process no longer has the Markov property. Kendall showed an interesting way to circumvent that problem. He observed that the queue length process at the successive epochs (points in time) at which a customer leaves the system is again Markovian—a so-called imbedded Markov chain. He was thus able to determine the steady-state distribution of the queue length process at departure epochs; that distribution could subsequently be shown to be equal to the steady-state queue length distribution at an arbitrary epoch. For future reference we mention the steady-state mean waiting time in the M/G/1 queue:

$$\mathrm{E}\mathbf{W} = \frac{\lambda \mathrm{E}\mathbf{S}^2}{2(1-\rho)}; \tag{3.1}$$

here λ is the arrival rate, $\mathbf{S}$ denotes a service time, and $\rho := \lambda \mathrm{E}\mathbf{S} < 1$ is the offered load. One can now verify the 24-minute result from the hairdresser example in Section 2.

3.2. New stimuli for queueing theory

In the midsixties queueing theory got a new stimulus from the fields of computer engineering and computer-communication networks. At that time computer technology had reached a level of development which required a good insight in the data flow inside the computer as well as in computer networks. For the latter in particular, the classical single service facilities did not suffice; networks of service facilities had to be analyzed. For networks of M/M/1-like queues, J.R. Jackson and also W.J. Gordon and G.F. Newell had shown that the explicit expression for the joint distribution of the queue lengths at the various nodes has a *product form* (in some cases it is a product of the marginal queue length distributions). These product-form results turned out to be very useful for the performance analysis of several basic computer systems, such as the central server system and the computer-terminal system. Furthermore, computer technology created new service disciplines, like processor sharing, that also gave rise to some new product-form results. Landmarks are the beautiful studies of F.P. Kelly and of F. Baskett, K.M. Chandy, R. Muntz and F. Palacios. In The Netherlands researchers like J.W. Cohen, A. Hordijk and N.M. van Dijk also made fundamental contributions.

Towards the end of the sixties it was recognized that the availability of the special resources and capabilities of many separate computer facilities could be extended by resource and load sharing; as a consequence, networks of computer systems, or more generally data communication networks, started to emerge. In line with its tradition, queueing theory has responded very

positively to the challenges posed by these new technological developments. Satellite communication led to queueing models for the phenomenon of colliding transmissions; protocols for message transmission in local area networks led to new priority models (like polling models, cf. Section 4); flexible manufacturing and distributed processing gave rise to queueing models with complicated dependencies between the arrival processes of customer streams at various queues, and between their service processes.

The effectiveness of queueing theory in handling such problems may to a large extent be due to the deep understanding that has been acquired for basic queueing models, such as Erlang's loss model, the M/G/1 queue and product-form networks. Accordingly, queueing theory has established itself as an indispensable tool in the design and performance analysis of computer-communication networks. Presently a new key topic is coming to the front: the performance analysis of Broadband Integrated Services Digital Networks (B-ISDN, see also figure 3). Such networks allow the simultaneous transmission of different traffic types (data, video, voice). Traditional arrival processes are inadequate for describing the sometimes bursty nature and long-ranging dependencies of these traffic streams; again this poses new challenges to performance analysts.

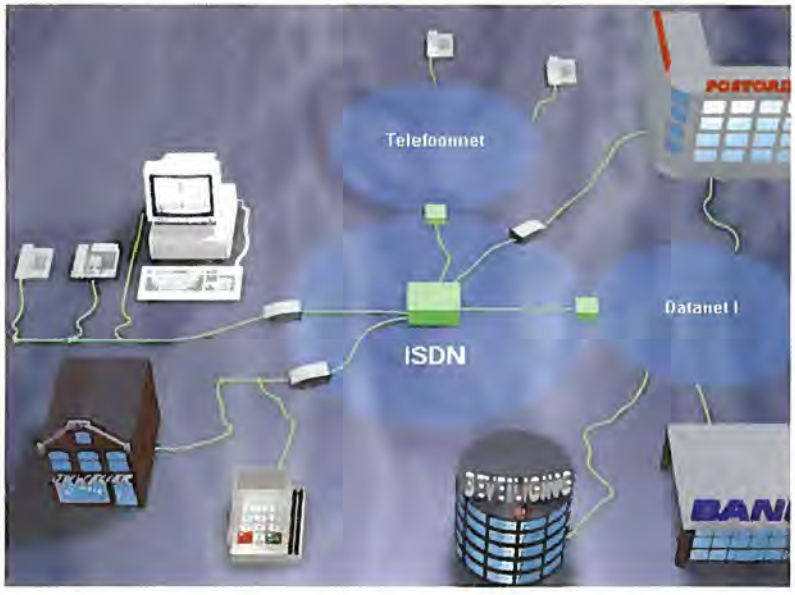


Figure 3. The simultaneous transmission of data, video and voice through broadband networks presents new challenges to queueing theory. (Courtesy PTT Telecom.)

4. CWI-RESEARCH ON POLLING SYSTEMS

4.1. Introduction

While the activities in CWI's research group Analysis and Control of Information Flows in Networks are not restricted to queueing theory (there is also much research in random walks, percolation theory and reliability theory, and in a more distant past many strong contributions have also been made to Markov decision theory), the group's core activity in the last few years has been queueing theory and its application to computer-communication performance analysis. Of the five Ph.D. theses that have been produced in the group in the nineties, two have been devoted to the analysis and optimization of polling systems [1, 3]. Therefore we now pay special attention to this class of queueing systems, starting with a motivating example.

Many communication systems provide a broadcast channel which is shared by all connected stations. When two or more stations wish to transmit simultaneously, a conflict arises. The rules for resolving such conflicts are referred to as 'multi-access protocols'. The token ring protocol is one such protocol, that is being used in many local area networks.

In a token ring local area network, several stations (terminals, file servers, hosts, gateways, etc.) are connected to a common transmission medium in a ring topology. A special bit sequence called the *token* is passed from one station to the next; a station that 'possesses the token' is allowed to transmit messages. After completion of his transmission the station releases the token, giving the next station in turn an opportunity to transmit. This situation can be presented by a so-called *polling model*.

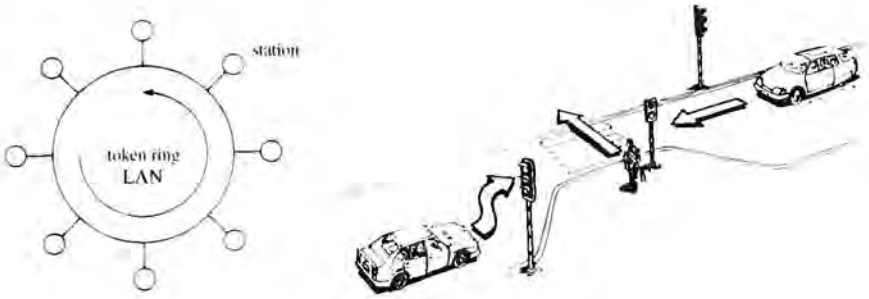


Figure 4. Multi-queueing problems with cyclic service are as common in computer networks as in road traffic situations. CWI has arrived at a conservation law giving insights into average waiting times for all sorts of customers in such systems.

4.2. The polling model

A polling model is a single-server multi-queue model, in which the server attends to the queues in cyclic order. The N queues $Q_1, \ldots, Q_N$ have infinitely large waiting rooms. Arrival times of customers at the queues are usually assumed to occur according to a Poisson process. Service requirements of customers at a queue are independent, identically distributed stochastic variables; the same holds for the switch-over times of the server between queues. Arrival rates, service time and switch-over time distributions may differ from queue to queue.

A polling model describes the behaviour of a token ring local area network in a natural way. The server represents the token-passing mechanism, and the customers represent messages generated at the stations. Many other situations in which several users compete for access to a common resource can also be described by this polling model. Examples are a repairman patrolling a number of machines which may be subject to breakdown, assembly work on a carousel in a production system, a computer with multi-drop terminals, and a signalized road traffic intersection (see figure 4). Depending on the application, various service disciplines at the queues may be considered. Common disciplines are *exhaustive service* (the server continues to work at a queue until it becomes empty), *gated service* (the server serves exactly those customers who were present when he arrived at the queue) and *1-limited service* (the server serves just one customer—if anyone is present—before moving on to the next queue).

Exact results for waiting time distributions are known when the service discipline at each queue is either exhaustive or gated. In a recent CWI report, J.A.C. Resing has shown that a detailed exact analysis (using the theory of multitype branching processes) is possible for the broader class of polling systems for which the service discipline at each queue has a so-called 'branching property'. Hardly any exact results for individual queue lengths or waiting times are known when this property is violated at one or more queues. However, even in such a case there exists a simple expression for a certain *weighted sum* of all the steady-state mean waiting times: L. Kleinrock has shown in 1964 that, when all switch-over times between queues are zero:

$$\sum_{i=1}^{N} \rho_i \mathbf{EW}_i = \rho \frac{\sum_{i=1}^{N} \lambda_i \mathbf{ES}_i^2}{2(1-\rho)}. \tag{4.1}$$

Here $\mathbf{EW}_i$ denotes the mean waiting time at Q_i, λ_i the arrival rate, $\mathbf{S}_i$ the generic service time, and $\rho_i := \lambda_i \mathbf{ES}_i$ the offered traffic load; $\rho = \Sigma_i \rho_i$ denotes the total offered load. This is called a *conservation law*: if the service discipline at a queue is changed, the weighted sum of mean waiting times (the left-hand side of (4.1)) remains the same, although the individual mean waiting times may change. Note that this formula is a generalization of formula (3.1) for the mean waiting time in a single M/G/1 queue.

4.3. Work conservation and work decomposition

The conservation law is a consequence of the 'principle of work conservation'. Suppose that the scheduling policy, i.e., the procedure for deciding at any time which customer(s) should be in service, has the properties that it does not allow the server to be idle when at least one customer is present and does not affect the amount of service given to a customer or the arrival time of any customer. Comparing the sample paths of the 'workload process' for such a system under different scheduling disciplines leads to the observation that *the workload process is independent of the scheduling discipline.*

The principle of work conservation has in the past proven to be very useful. It enables one to analyze the workload process of queueing systems with a highly complicated scheduling discipline as if the scheduling were a relatively simple one, such as the First Come First Served discipline.

For the token ring local area network mentioned above, the time for the token to be passed from station to station is in general not negligible. Correspondingly, in the polling model the time the server needs for switching from station to station has to be taken into account. This fact considerably complicates the analysis: the principle of work conservation is no longer valid, since now the server may be idle (switching), although there is at least one customer in the system. However, under certain conditions there exists a natural modification of the principle of work conservation for polling systems with switch-over times, based on a *decomposition* of the amount of work in the system [2, 3]. This result states that —under certain conditions— the amount of work in the polling system, $\mathbf{V}_{with}$, is in distribution equal to the sum of the amount of work in the simpler 'corresponding' system *without* switch-over times, $\mathbf{V}_{without}$, plus the amount of work, $\mathbf{Y}$, at an arbitrary moment during a period in which the server is switching from one queue to another:

$$\mathbf{V}_{with} =^{(d)} \mathbf{V}_{without} + \mathbf{Y}. \tag{4.2}$$

$=^{(d)}$ denoting equality in distribution.

The work decomposition gives rise to similar expressions for a weighted sum of the mean waiting times as Formula (4.1). We can write

$$\mathrm{E}\mathbf{V}_{with} = \sum_{i=1}^{N} \mathrm{E}\mathbf{S}_i \mathrm{E}\mathbf{X}_{i,w} + \sum_{i=1}^{N} \rho_i \frac{\mathrm{E}\mathbf{S}_i^2}{2\mathrm{E}\mathbf{S}_i} = \sum_{i=1}^{N} \rho_i \mathrm{E}\mathbf{W}_i + \frac{1}{2}\sum_{i=1}^{N} \lambda_i \mathrm{E}\mathbf{S}_i^2.$$

The first relation splits the mean workload into the mean workload of the $\mathbf{X}_{i,w}$ waiting customers and the mean residual workload of the customer in service (if a customer of type i is in service, his mean residual service time equals $E\mathbf{S}_i^2/2E\mathbf{S}_i$, cf. the discussion of the waiting time paradox in the beginning of this essay); the second equality follows from Little's formula $\mathrm{E}\mathbf{X}_{i,w} = \lambda_i \mathrm{E}\mathbf{W}_i$. Taking means in (4.2) now leads to:

$$\sum_{i=1}^{N} \rho_i \mathbf{EW}_i = \rho \frac{\sum_{i=1}^{N} \lambda_i \mathbf{ES}_i^2}{2(1-\rho)} + \mathbf{EY}. \tag{4.3}$$

Denote the mean total switch-over time in one cycle of the server by s, and the second moment by $s^{(2)}$. Evaluating $\mathbf{EY}$ (cf. [2]) yields:

$$\sum_{i=1}^{N} \rho_i \mathbf{EW}_i = \rho \frac{\sum_{i=1}^{N} \lambda_i \mathbf{ES}_i^2}{2(1-\rho)} + \rho \frac{s^{(2)}}{2s} +$$

$$\frac{s}{2(1-\rho)}[\rho^2 - \sum_{i=1}^{N} \rho_i^2] + \sum_{i=1}^{N} \mathbf{EM}_i. \tag{4.4}$$

where $\mathbf{EM}_i$ denotes the mean amount of work in Q_i left by the server upon its departure from that queue (when $s \to 0$, the fraction of visits to Q_i in which the server finds Q_i empty tends to one, and the right-hand side of (4.4) reduces to the right-hand side of (4.1)). Formula (4.4) has been coined a *pseudo-conservation law*. The main difference with Kleinrock's conservation law is that now the weighted sum of mean waiting times *does* depend on the service discipline at each queue, through $\sum \mathbf{EM}_i$. For many service disciplines, amongst which are exhaustive, gated and 1-limited service, we are able to determine the right-hand side of (4.4) explicitly.

It is also possible to extend the work decomposition property and pseudo-conservation law to much more general single-server systems with multiple customer classes [2]. Such pseudo-conservation laws often provide the only information available in polling and multiclass systems with nonzero switch-over times. They are therefore of considerable practical importance. One of the main features of the pseudo-conservation laws is that they are very useful for testing *and* developing approximations for the individual mean waiting times [3]. Such approximations have in turn supplied an approach for solving various optimization problems, like (i) determine the optimal visit times in a cyclic polling model [1] (this problem has been studied in a consultancy for PTT Telecom) and (ii) determine the optimal visit order of the stations when non-cyclic polling is allowed.

The work decomposition property (4.2) relates the workload in polling models with and without switch-over times. Such a simple relationship does not in general exist between the joint queue length processes in both models. However, when all service disciplines have the above-mentioned 'branching property', then one can also find [1] a surprisingly simple relation between the joint queue length processes in both models, at particular imbedded points in time (cf. our earlier reference to [4]!).

Other very recent polling research at CWI has led to the first exact results for polling systems with *multiple servers* [1], and to good rules for the NP-complete problem of the optimal (with respect to a weighted sum of

mean waiting times) probabilistic allocation of several customer types to a collection of parallel servers.

4.4. Epilogue

In Section 4 some emphasis has been put on Ph.D. research. In recent years the emphasis in the group has been shifting somewhat towards postdoctoral research: several postdocs have worked in the group, supported by grants from Shell, PTT Research, Esprit and ERCIM.

Finally it deserves to be mentioned that, since the early eighties, the group has strongly benefitted from the advisorship of J.W. Cohen. Through his research he has contributed, more than anyone else, towards establishing queueing theory as a mature mathematical discipline within Applied Probability. Almost ten years after his retirement, his unflagging energy and love for the mathematical modelling and analysis of congestion phenomena continue to stimulate his environment.

References

1. S.C. Borst (1994). *Polling Systems*. Ph.D. Thesis, Tilburg University.
2. O.J. Boxma (1989). Workloads and waiting times in single-server queues with multiple customer classes. *Queueing Systems 5*, 185-214.
3. W.P. Groenendijk (1990). *Conservation Laws in Polling Systems*. Ph.D. Thesis, Utrecht University.
4. D.G. Kendall (1953). Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. *Ann. Math. Statist. 24*, 338-354.

System Theory – A Brief Exposition

J.M. van den Hof, J.H. van Schuppen

1. Introduction

This paper provides an introduction to system theory for a general readership of mathematicians, engineers, and other scientists. In addition, contributions of CWI's research group System and Control Theory are summarized.

In several areas of the sciences there is a need for mathematical models of phenomena that evolve in time. Such models, called *control system* or *system*, are used for control or for signal processing, and have been formulated, for example, in connection with the movement of a compact disc, the temperature in a glass furnace, the behaviour of an aircraft, the behaviour of an underwater autonomous robot, and the flow of nitrate in the human body.

A control system interacts with its environment by receiving an input signal and providing an output signal. A control system may be described by a differential equation, a difference equation, logical rules, or a combination of these as in a hybrid system. Here we restrict ourselves to system theory. Control theory, with its main concept of feedback, is only marginally touched upon.

The main problem of system theory is *realization*. This is motivated by the problem of *system identification*. The realization problem is to derive for observations a system in a recursive state space representation, explained below, and to classify all systems that represent the same observations. The

system identification problem is to construct from observations a system in a selected model class that approximates the data according to an approximation criterion. An example of such a problem is to construct a model for the flow of nitrate in the human body (see section 4). Realization and system identification theory have been developed extensively for finite-dimensional linear systems and for Gaussian systems. For other classes of systems the results are much less complete.

System theory has been developed by researchers in various disciplines, like engineering, mathematics, and econometrics, and uses many different branches of mathematics, including linear algebra, differential equations, geometry, operator theory, probability, and stochastic processes.

2. History

2.1. Origins

Control and signal processing problems were already intensively studied in engineering and mathematics in the 1940's, when N. Wiener and A. Kolmogorov pioneered an approach to the least-squares prediction problem. In this problem an algorithm, called a filter, is to be derived that on the *basis* of observations predicts a signal. Applications and extensions of this approach followed in the 1950's. Within engineering feedback control was developed for amplifiers and communication equipment. Within mathematics optimal control theory was studied, based on the calculus of variations, following publications of Russian mathematicians led by L.S. Pontryagin. Related developments in linear algebra, stochastic processes, information theory, and communication theory influenced researchers active at that time.

Around 1960 weaknesses and limitations of optimal control and least-squares prediction became clear. A filter that at any time needs an infinite number of past data cannot be implemented on a computer with a finite memory. Researchers in optimal control theory realized that only a limited class of problems can be solved analytically.

Then R.E. Kalman proposed a new problem formulation for control and filtering. If finite memory, however defined, is required for implementation, then why not consider as starting point a control system with finite memory, that interacts with its environment via input and output signals? The definition of a control system is inspired by developments in computer science around 1960 with the concepts of an automaton and of a recursive function, and is based on the concept of state, as used in physics, and on a recursive structure for that state. At any time the current state and the future input of a system uniquely determine the future of the state and the output. Seen in this way engineering models, control systems, and computer algorithms, become analogous objects. System theory aims to study such objects in a unified way. As a consequence there can be a unified approach

Figure 1. R.E. Kalman. (Courtesy Springer-Verlag.)

to problems of control, communication, signal processing, and computing.

Along these lines Kalman solved a least-squares prediction and filtering problem for a control system, the solution of which, now known as the Kalman filter, is widely applied in signal processing and control. The realization problem, inspired by the definition of a control system, is to construct a recursive state space representation from observations of input-output signals. Kalman also showed that an optimal control problem for a linear system and a quadratic criterion can be solved analytically and that the solution is dual to the Kalman filter. The shift from optimal control problems and least-squares prediction for models with infinite memory to systems with finite memory had been shown to work.

With T.S. Kuhn one may speak of a paradigm shift for control and signal processing, with enormous consequences. Control and signal processing problems now yielded solutions with finite memory that could be implemented directly and analyzed explicitly. Engineering modelling and system identification took a new turn. Results from system theory, usually through control and signal processing, are used in research areas including engineering, computer science, technology, economics, and econometrics. By now courses in systems and signals are in the undergraduate curricula of most engineering departments and of mathematics departments, and software packages with algorithms based on system theory are used in industry and in government.

A few lessons can be drawn from the development of system theory. Applications of system theory algorithms provide ample evidence for the usefulness of the concept of a state space representation. A system as mathematical model must be regarded as a representation of observations. System identification must take into account the fact that for a given set of input-output signals there is in general a large equivalence class of models. Optimal control and filtering problems may not admit a solution with finite memory, however it be defined. Solutions with finite memory may be determined by turning the problem formulation around and asking for a realization of the observations in a selected class of systems with finite memory.

2.2. What is a control system?

As an example we introduce the concept of a *time-invariant finite-dimensional linear system* without much attention to mathematical finesse. Consider the system specified by the equations

$$\dot{x}(t) = Ax(t) + Bu(t), \; x(t_0) = x_0, \tag{2.1}$$

$$y(t) = Cx(t) + Du(t). \tag{2.2}$$

where $t_0 \in \mathbb{R}$, $T = [t_0, \infty)$ is called the *time interval* of interest; $x_0 \in \mathbb{R}^n$ the *initial state*; $u : T \to \mathbb{R}^m$ the *input function*; $x : T \to \mathbb{R}^n$ the *state function*; $y : T \to \mathbb{R}^p$ the *output function*; and $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, and $D \in \mathbb{R}^{p \times m}$ are matrices. As mentioned before, the main characteristic of such a system is that at any time the state and the future input uniquely determine the future of the state and of the output. The observations are formed by the input and output functions, or, alternatively, by input-output signals. The system is called *linear* because the output is a linear function of the input and the initial condition, *finite-dimensional* because the state space, $\mathbb{R}^n$, is a finite-dimensional vector space, and *time-invariant* because its response is the same if it starts from the same state at a later time.

The external description of such a system is specified by the following relation for input-output signals:

$$y(t) = Ce^{A(t-t_0)}x_0 + \int_{t_0}^{t} W(t-s)u(s)ds, \tag{2.3}$$

$$W(t) = Ce^{A(t-t_0)}B + D\delta(t-t_0), \quad W : T \to \mathbb{R}^{p \times m}, \tag{2.4}$$

where W is called the *impulse response function* and δ is the Dirac delta function. The observations of the system,

$$\left\{ \begin{pmatrix} u \\ y \end{pmatrix} : T \to \mathbb{R}^{m+p} \mid u, y \text{ satisfy (2.3) for a } x_0 \in \mathbb{R}^n \right\}, \tag{2.5}$$

are also called the *observable behaviour* or *behaviour* of the system. A *time series* is a set of numerical values of input-output signals.

2.3. Realization theory for finite-dimensional linear systems

First realization from the impulse response function W is discussed. Consider the external representation as in (2.3). Through experimentation with a phenomenon an engineer can obtain an estimate of W. The question is then whether there exists a finite-dimensional linear system with matrices A, B, C, D such that (2.4) holds. If so, it is called a *realization* of the given external system description or of the impulse response function. The realization problem also requires the classification of all *minimal* realizations.

i.e., those where the dimension n of the state space is minimal. Other questions include: what is the characterization of the state space description of a system if its external description is either time-reversible, symmetric, or dissipative?

Kalman has derived a necessary and sufficient condition for an impulse response function to have a realization as a finite-dimensional linear system and a reachability and an observability condition for the realization to be minimal. In addition, he has provided a classification of all minimal realizations (see for details the textbook [3] by E.D. Sontag). Parameterizations for the class of minimal realizations were derived later. M. Hazewinkel and Kalman have proven that for multi-input/multi-output systems no continuous parameterization exists.

Secondly realization from input-output signals is discussed. This is very relevant in research areas with short time series or where experimentation is not permitted, such as environmental modelling, biology, economics, and econometrics. This should be contrasted with electrical engineering, where through experimentation one can obtain arbitrarily long time series. In the 1970's this problem was treated by R. Liu and L.C. Suen, and shortly afterwards by E. Emre, L.M. Silverman, and K. Glover with the term 'dynamic covers'.

In the behavioural approach to system theory, proposed by J.C. Willems and developed by him and co-workers, realization from input-output signals is generalized. In this approach the observation vector is not a priori distinguished into an input and output signal. In many engineering problems the distinction is clear because of a causality relation, but in other problems, for example in econometrics (e.g., income and consumption of households), this is often not a priori the case.

2.4. Stochastic realization of stationary Gaussian processes

Kalman also proposed a definition of a stochastic control system. Consider three discrete-time stationary stochastic processes: an input, a state, and an output process. They form a stochastic control system if for all $t \in T$ the conditional probability distribution of the next state and the current output, $(x(t+1), y(t))$, given the past of the state, output, and input process, depends only on the current state and the current input, $(x(t), u(t))$. If the probability distribution is Gaussian or normal and if only the conditional mean of this distribution depends linearly on $(x(t), u(t))$, then the processes satisfy the following relations

$$x(t+1) = Ax(t) + Bu(t) + Mv(t), \quad x(t_0) = x_0, \tag{2.6}$$

$$y(t) = Cx(t) + Du(t) + Nv(t), \tag{2.7}$$

where v is a Gaussian white noise process, i.e., a sequence of independent random variables each of which has a Gaussian probability distribution function. The system specified by this representation is said to be a *Gaussian stochastic control system* and a *Gaussian system* in case there is no input process.

Furthermore, Kalman formulated the weak stochastic realization problem for stationary Gaussian processes. It was motivated by an analysis of the Kalman filter. A stationary Gaussian process is said to have a *stochastic realization* if there exists a Gaussian system such that the output process equals the given process in distribution. P. Faurre and co-workers, in cooperation with Kalman, have given a characterization of a minimal stochastic realization, classified them, and also analyzed a parameterization of the class of stochastic realizations.

A. Lindquist and G. Picci (see [1]), and G. Ruckebusch, have solved the strong Gaussian stochastic realization problem, in which the output process must equal the given process almost surely. This problem is best studied in geometric terms in which a stationary Gaussian process is associated with a subspace of a Hilbert space.

Stochastic realization theory forms the theoretical foundation of signal processing. Prediction problems were treated in the 1940's. By now signal processing includes several techniques based on system theory, such as prediction, filtering, smoothing, interpolation, and image processing.

2.5. Realization theory—extensions

The realization theory formulated for finite-dimensional linear systems and for Gaussian systems has been generalized to many other classes of systems. Only a few of these generalizations will be mentioned below. For each mathematical structure the concept of a system must be defined anew.

Algebraic generalizations are linear systems over modules, rings, and finite fields. Research in these directions was initiated by Kalman with contributions by M.L.J. Hautus, E.W. Kamen, and E.D. Sontag. Linear systems over finite fields are used as mathematical models in coding theory and have recently drawn new interest. The realization problem for systems in algebraic structures as groups, semigroups, and algebras, is essentially the problem of finding irreducible representations of input-output maps. A special case of current interest is realization of positive linear systems that is motivated by, for example, problems in biomathematics, chemical engineering, and economics. The realization problem for this class is unsolved and requires further study of polyhedral cones and positive linear algebra.

Other systems for which the realization problem has been studied include: linear systems with functions in Hilbert spaces (by P.A. Fuhrmann), systems in which the dynamics is specified by polynomials (by Sontag, in cooperation with Kalman), bilinear systems, a system in a differential geo-

metric context (described in terms of vector fields), and specific classes of nonlinear systems (by H. Sussmann, M. Fliess, and B. Jacubczyk) and of mechanical systems, such as Hamiltonian systems. In computer science the concept corresponding to a system is an input-output automaton, a Petri net, or a process algebra. In the automata literature, the realization problem has been solved by A. Nerode. A generalization of the concept of state for systems with functions taking values in arbitrary sets was formulated by Willems in terms of the conditional independence relation for sets. In this definition the current state and the input signal make the past and the future of the state and output signal conditionally independent.

A multi-parameter system is a system in which the time axis has been generalized to an arbitrary index set or to a partially ordered set. A picture may be modeled as a two-parameter system. The concept of state of such a system may be phrased in terms of the conditional independence relation of sets. The realization problem for this class has been studied in connection with image processing.

Stochastic realization theory of Gaussian processes has also been generalized, for example to diffusion processes in analogy with statistical mechanics and quantum mechanics. A finite stochastic system may be defined analogously to a Gaussian system for a finite-valued process with a finite-state Markov process. In signal processing it is called a *hidden Markov model* and in automata theory a *probabilistic automaton*. The stochastic realization problem for this class, already studied in the 1960's, is still unsolved, as is the case for counting and jump processes.

An investigation is needed of the stochastic realization problem for stochastic control systems with partial observations. The concepts of information state and of information system should be studied in the framework of exponential families of distribution functions.

Kalman's definition of a stochastic system can be reformulated in terms of the conditional independence relation of probability theory, stating that at any time the current state and the input process make the past and the future of the state and the output process conditionally independent. Multi-parameter stochastic realization problems in connection with random fields are under investigation.

A generalization in another direction is the factor analysis model. In this model for random variables the factor, corresponding to the state, makes two or more variables conditionally independent. This generalization of the concept of state is very interesting. R. Frisch, who received the Nobel prize in economics, proposed this model as an alternative for the model used in least squares estimation. Kalman has pointed out its relevance for economic modelling and contributed to the associated stochastic realization problem. The problem is unsolved. The stochastic system corresponding to a factor analysis model is termed an *errors-in-variables* model or a dynamic factor

Figure 2. Environmental problems will motivate future research in system identification. (Photo: James Nachtwey— Magnum Photos. Courtesy ABC Press.)

system. The realization problem for this class is studied in system theory and econometrics by G. Picci and M. Deistler.

2.6. System identification

As stated in the Introduction, the system identification problem is to construct from observations a control system in a chosen model class that best fits the observations according to a specified approximation criterion. A procedure for system identification is: (1) Selection of a model class based on a priori information; (2) Input design, experimentation, and data collection; (3) Parameterization of the model class based on realization theory and a check of its identifiability; (4) Approximation, selection of a control system; and (5) Evaluation of the quality of the selected system. The selection of the model class is often based on domain modelling, for example on physical laws, on chemical reaction kinetics, and on economic or physiological modelling. In step (3) of the procedure realization theory is used exclusively. A textbook on system identification is that of L. Ljung [2].

System identification has been well developed for the classes of finite-dimensional linear systems and for Gaussian stochastic systems. For the

approximation criterion use is made of the least-squares criterion or the likelihood function. The problem is largely solved for single-input/single-output linear systems, but still not satisfactory for multi-input/multi-output linear systems. The most effective solutions are based on realization theory. The most promising approach is presently the so-called subspace method, based on stochastic realization theory and numerical linear algebra.

System identification problems for nonlinear systems have been studied for a long time in engineering and econometrics. The relation between realization theory for nonlinear systems and system identification problems for the same class remains to be explored.

3. CWI contributions

3.1. Stochastic realization and system identification

The research by J.H. van Schuppen in stochastic realization theory is motivated by system identification, signal processing, and control for counting and jump processes. The stochastic realization problem for finite-valued processes is investigated in cooperation with G. Picci. The current bottleneck is the characterization of minimal realizations of finite stochastic systems. Solution of this problem leads to a factorization problem for positive matrices. The closely related realization problem for deterministic positive linear systems is currently investigated by J.M. van den Hof and Van Schuppen. The stochastic realization problem in terms of σ-algebras, as well as for the factor analysis model have been treated.

Motivated by the engineering practice of using Gaussian white noise as input signal, a stochastic realization problem for a Gaussian stochastic control system has been formulated and solved. Parameter estimation problems for counting processes were treated by P.J.C. Spreij. Recently A.A. Stoorvogel and Van Schuppen investigated the approximation problem for Gaussian stochastic systems using information theoretic criteria.

3.2. Linear systems

Systems are modeled by a variety of methods including black-box identification and the use of physical laws. The classical input/output framework, see (2.1) and (2.2), which dominates control theory is less appropriate in the modelling context, and has to be replaced by a setting in which all external variables are treated on an equal footing. This point of view, recently emphasized in particular by Willems, leads to new questions for realization theory. First-order representations of other types than the standard input/state/output form (2.1) and (2.2) are used, and one needs to analyze the minimality conditions for such representations. As a basis for the notion of equivalence of representations, the transfer function is replaced by the 'behaviour', which is the set of all trajectories (in some given func-

tion space) admitted by the system equations. A study of minimality and equivalence for general first-order representations of linear systems was undertaken at CWI by J.M. Schumacher together with M. Kuijper and has led to several journal papers and a book. In the approach based on behaviours, the choice of a function space has an impact on the notion of equivalence. The equivalence notion that is obtained from working with the space of so-called 'impulsive-smooth distributions', was studied in a joint effort of A.H.W. Geerts and Schumacher. Recent work of M.S. Ravi and J. Rosenthal in the U.S.A. and of V. Lomadze in the Republic of Georgia has made clear that the set of generalized linear systems obtained in this way provides the long-sought smooth compactification of the class of standard linear systems of a fixed McMillan degree. This issue is currently being further explored in joint work of Ravi, Rosenthal, and Schumacher.

4. System identification of nitrate flow in the human body

4.1. Structural identifiability from input-output signals

In biology and mathematics the class of compartmental systems is frequently used. A physiological model of a living organism may consist of several compartments with more or less homogeneous concentrations of material. The compartments interact by processes of transportation and diffusion. In biology there often is prior knowledge on the structure of the model. Therefore the class of compartmental systems is related to the class of structured linear systems, in which the system is structured by physical laws. Before estimating the parameters, it should be examined whether the parameterization is structurally identifiable, i.e., whether the parameters can in principle be determined *uniquely* from the data. Conditions for structural identifiability from the impulse response follow directly from realization theory. J.M. van den Hof has investigated structural identifiability from input-output signals with unknown initial condition for both finite-dimensional linear systems and positive linear systems.

4.2. Example

As an example of a system identification problem we consider a model for the uptake and dispersion of nitrate in the human body. In the model class four compartments are considered: nitrate (NO_3^-) in the stomach, the body pool, and the saliva, and nitrite (NO_2^-) in the saliva, as shown in figure 3.

The model may be described by the following differential equations:

$$\dot{x}_1 = -K_a x_1 + \frac{b}{V_s} x_3 + u_1, \qquad \dot{x}_2 = K_a x_1 - (K_2 + K_T) x_2 + K,$$

$$\dot{x}_3 = K_2 x_2 - (K_1 + \frac{b}{V_s}) x_3, \qquad \dot{x}_4 = K_1 x_3 - \frac{b}{V_s} x_4.$$

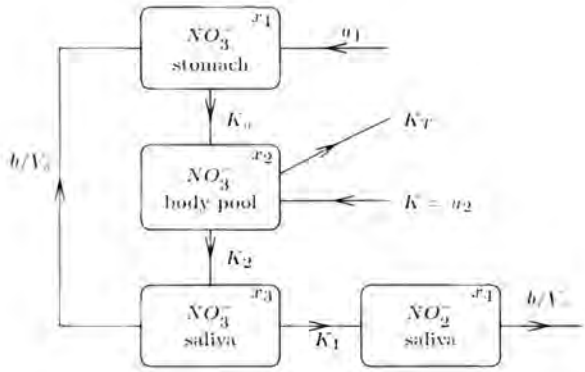


Figure 3. Nitrate model.

in which x_1, x_2, and x_3 denote the amount of NO_3^- in the stomach, the body pool, and the saliva, respectively, and x_4 denotes the amount of NO_2^- in the saliva; u_1 denotes the uptake of nitrate. The remaining variables are constants. The constants K, K_T, and V_d are assumed to be known. The unknown parameters are K_a, K_2, K_1, b, V_s, and the initial condition x_0. One can observe the concentration of NO_3^- in the body pool and the saliva, and the concentration of NO_2^- in the saliva, i.e., we can observe $x_2/V_d, x_3/V_s$, and x_4/V_s. The model has been developed by the National Institute of Public Health and Environmental Protection (RIVM).

The theory developed by Van den Hof for structural identifiability from input-output signals with a nonzero initial condition provides conditions on the inputs u_1 and K such that the unknown parameters K_a, K_2, K_1, b, V_s, and the initial condition x_0 can be uniquely determined from the observations.

5. Concluding remarks

System theory has proven to be extremely useful for engineering, mathematics, and other areas of the sciences, in particular for control and signal processing. The concept of a control system, and the results of realization theory and system identification are widely applied in industry, commerce, and government.

System theory will in the future be motivated by new problems of engineering and the sciences. Solution of these problems will become urgent through the technological development and through the demands for increased living standards. There may also be a shift away from electrical and mechanical engineering to information processing. Realization problems motivated by information processing may therefore receive relatively

more interest. A realization approach is also needed for team and game problems. In such decision and control problems there are two or more decision makers with different observations. System theory has many open problems.

ACKNOWLEDGEMENTS

The authors thank G. Picci and E.D. Sontag for their comments on an earlier version of the paper.

REFERENCES

1. A. LINDQUIST, G. PICCI (1985). Realization theory for multivariate stationary Gaussian processes. *SIAM J. Control & Opt. 23*. 809-857.
2. L. LJUNG (1987). *System Identification: Theory for the User*. Prentice-Hall, Inc., Englewood Cliffs, NJ.
3. E.D. SONTAG (1990). *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer-Verlag, New York.

Bootstrap Resampling

R. Helmers

1. Introduction

B. Efron, who invented the bootstrap in 1979, recently wrote: 'Computer-intensive methods like the bootstrap greatly extend the range of classical methods, and this is the way I believe that they will most dramatically affect 21st century statistics'. The bootstrap is a computer-intensive method for estimating the variability of statistical quantities and for setting confidence regions. The name 'bootstrap' refers to the analogy with pulling oneself up by one's own bootstraps. Efron's bootstrap is to resample the data. Given observations $X_1, \ldots, X_n$ artificial bootstrap samples are drawn with replacement from $X_1, \ldots, X_n$, putting equal probability mass $\frac{1}{n}$ at each X_i. For example, with sample size $n = 5$ and distinct observations X_1, X_2, X_3, X_4, X_5 one might obtain X_3, X_3, X_1, X_5, X_4 as bootstrap sample. In fact there are 126 distinct bootstrap samples in this case.

Bootstrap resampling often gives much better estimates than traditional statistics usually provide us with. The bootstrap can also be an effective tool in many problems of statistical inference, which otherwise would have been too complicated to handle; e.g., the construction of a confidence band in nonparametric regression, testing for the number of modes of a density, or the calibration of confidence bounds. The problem of constructing a confidence band for an unknown 'regression mean' arises, e.g., if one tries to ascertain a trend in annual series of observed (air) temperatures, possibly due to the influence of 'global warming' on such data.

In this paper I will survey recent research at CWI in the general area of bootstrap resampling methods. At the same time research in this area, which takes place at Leiden University, will also be briefly reviewed. Resampling is one of the four selected areas of research in the focus area 'Computationally Intensive Methods in Stochastics' (1993-1998) of NWO. This topic was also the central theme of a two-month research workshop presented (in the summer of 1995) at the Institute of Technology, Bandung, as part of a cooperation project 'Applied Mathematics and Computational Methods' (1995-1999) between The Netherlands and Indonesia, in which CWI is one of the Dutch cooperating Institutes.

2. Efron's nonparametric bootstrap

To begin with I describe Efron's nonparametric bootstrap in a simple setting and address briefly the important question: when does Efron's bootstrap work and when does it fail?

2.1. Description of the bootstrap

Suppose $X_1, \ldots, X_n$ is a random sample of size n from a population with unknown distribution function F on the real line. Let, in addition,

$$\theta = \theta(F) \tag{2.1}$$

denote a real-valued parameter which we want to estimate.

Let $T_n = T_n(X_1, \ldots, X_n)$ denote an estimator of θ, based on the data $X_1, \ldots, X_n$. Our object of interest is the distribution of $n^{\frac{1}{2}}(T_n - \theta)$, i.e., we define

$$G_n(x) = P(n^{\frac{1}{2}}(T_n - \theta) \leq x), \quad -\infty < x < \infty, \tag{2.2}$$

where P denotes 'probability' corresponding to F. Clearly G_n, the exact distribution of $n^{\frac{1}{2}}(T_n - \theta)$, is unknown, because F is not known to us, but we can try to estimate it. The Efron's nonparametric bootstrap estimator (approximation) of G_n is given by

$$G_n^*(x) = P_n^*(n^{\frac{1}{2}}(T_n^* - \theta_n) \leq x), \quad -\infty < x < \infty. \tag{2.3}$$

Here $T_n^* = T_n(X_1^*, \ldots, X_n^*)$, where $X_1^*, \ldots, X_n^*$ denotes an artifical random sample—the bootstrap sample—from $\hat{F}_n$, the empirical distribution function of the original observations $X_1, \ldots, X_n$, and $\theta_n = \theta(\hat{F}_n)$. Note that $\hat{F}_n$ is the random distribution—a step function—which puts probability mass $\frac{1}{n}$ at each of the X_i's $(1 \leq i \leq n)$, sometimes referred to as the resampling distribution. The empirical distribution function $\hat{F}_n$ is illustrated in figure 1. Finally, P_n^* denotes 'probability' corresponding to $\hat{F}_n$, conditionally given $\hat{F}_n$, i.e., given the observations $X_1, \ldots, X_n$. To emphasize the fact that G_n^* is a conditional distribution, one may as well write

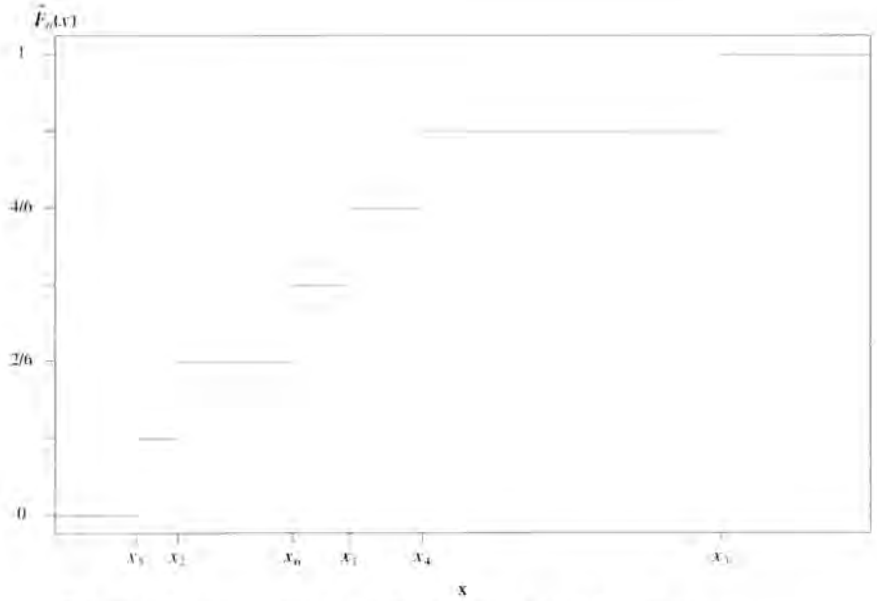


Figure 1. Empirical distribution function based on observations $x_1, \ldots, x_6$.

$$G_n^*(x) = P_n^*(n^{\frac{1}{2}}(T_n^* - \theta_n) \leq x | X_1, \ldots, X_n), \quad -\infty < x < \infty. \tag{2.4}$$

instead of (2.3). Obviously, given the observed values $X_1, \ldots, X_n$ in our sample, F_n is completely known, and – at least in principle – G_n^* is also completely known. We may view G_n^* as the empirical counterpart in the 'bootstrap world' to G_n in the 'real world'. In practice, exact computation of G_n^* by complete enumeration is usually impossible (even in our sophisticated computer age): for a sample $X_1, \ldots, X_n$ of n distinct numbers there are $\binom{2n-1}{n}$ distinct bootstrap samples. For $n = 10$ already near to 100,000 bootstrap samples have to be enumerated, so very soon this method becomes unfeasible and we have to turn to another solution: Monte-Carlo simulation.

In a sense, this boils down to repeatedly drawing a random bootstrap sample from all possible bootstrap samples. We fix a large number B. With the use of the computer, we generate a bootstrap sample and calculate the resulting value of $n^{\frac{1}{2}}(T_n^* - \theta_n)$. By repeating this procedure B times, we obtain B values, say $n^{\frac{1}{2}}(T_{n,1}^* - \theta_n), \ldots, n^{\frac{1}{2}}(T_{n,B}^* - \theta_n)$, which give an accurate Monte-Carlo estimate to the theoretical bootstrap distribution G_n^* of $n^{\frac{1}{2}}(T_n^* - \theta_n)$. Monte-Carlo simulation was of course already well established before the invention of the bootstrap, but it finds a very natural place here. Generating a bootstrap sample amounts to randomly drawing a sample of size n with replacement from $X_1, \ldots, X_n$. The Monte-Carlo procedure introduces a second source of randomness. However, by choosing

B suitable large we can control the Monte-Carlo error and make sure that it is negligible in comparison with the bootstrap approximation error.

2.2. Operation of the bootstrap

When does Efron's bootstrap work? The consistency of the bootstrap approximation G_n^*, viewed as an estimate of G_n, i.e., we require

$$\sup_x |G_n(x) - G_n^*(x)| \to 0, \text{ as } n \to \infty \tag{2.5}$$

to hold, with P-probability one (i.e., for almost all sequences $X_1, X_2, \ldots$), or a slightly weaker version of it, namely that (2.5) holds only in P-probability, rather than P-almost surely, is generally viewed as an absolute prerequisite for Efron's bootstrap to work in the problem at hand. Of course, the assertion (2.5) is only a first order asymptotic result, and the error committed, when the bootstrap is applied in finite samples—say, with sample size $n = 20$—may still be quite large.

In the important special case that $\theta(F) = \mu = \int x dF(x)$, the population mean, and $T_n = \bar{X}_n = n^{-1} \sum_{i=1}^n X_i$, the sample mean, a by now classical result asserts that (2.5) holds true, i.e., Efron's bootstrap works, provided the variance σ^2 of the underlying distribution F is finite. If σ^2 is infinite the situation becomes more complex: it has been proved that Efron's bootstrap still works, provided F is in the domain of attraction of the normal law; otherwise Efron's bootstrap fails.

Bootstrap resampling can also be used to estimate functionals of G_n, e.g., its variance, rather then G_n itself. W.R. van Zwet (1994) has recently studied the performance of Efron's bootstrap estimate of variance for arbitrary symmetric statistics $T_n = T_n(X_1, \ldots, X_n)$ with finite second moment using the Hoeffding decomposition. He showed that Efron's bootstrap will typically work, provided $\sum_{i=1}^n E(T_n|X_i)$, the linear term in the Hoeffding decomposition of T_n, is the dominant one and the higher order terms in the Hoeffding decomposition tend to zero rather fast. The requirement concerning the linear term is also shown to be a necessary condition for the consistency of Efron's bootstrap; otherwise (2.5) generally fails to hold. A specific case for which Efron's bootstrap works—namely Serfling's class of generalized L-statistics—is investigated by R. Helmers, P. Janssen, and R. Serfling (1990).

H. Putter and Van Zwet (1993) (c.f. also chapter 2 of the Ph.D. thesis of Putter (1994)) emphasized the importance of a proper choice of the resampling distribution (not necessarily the empirical distribution $\hat{F}_n$, as in Efron's nonparametric bootstrap). Let $\tau_n(F)$ denote the distribution of a statistical quantity $R_n = R_n(X_1, \ldots, X_n; F)$. If $\tilde{F}_n$ denotes the resampling distribution, $\tilde{F}_n = \tilde{F}_n(X_1, \ldots, X_n)$ being an estimate of F, then the bootstrap estimate of $\tau_n(F)$ becomes $\tau_n(\tilde{F}_n)$. Note that $\tilde{F}_n$ may be very different from the empirical distribution $\hat{F}_n$, e.g., one may consider $\tilde{F}_n = F_{\hat{\theta}_n}$, when

it is a priori known that F belongs to a parametric model $\{F_\theta, \theta \in \Theta\}$, the finite-dimensional parameter θ is estimated by $\hat{\theta}_n = \hat{\theta}_n(X_1, \ldots, X_n)$, a consistent estimator of θ (parametric bootstrap). Putter and Van Zwet (1993) have recently proved a general result concerning the consistency of bootstrap estimates, with general resampling distribution $\hat{F}_n$.

3. Accuracy of bootstrap estimates

3.1. Smooth cases

In the previous section we have seen that Efron's bootstrap is consistent for the case of the sample mean $\bar{X}_n = n^{-1}\sum_{i=1}^n X_i$, provided the underlying distribution F of the observations has a finite second moment. With

$$G_n(x) = P(n^{\frac{1}{2}}(\bar{X}_n - \mu) \leq x) \tag{3.1}$$

and

$$G_n^*(x) = P_n^*(n^{\frac{1}{2}}(\bar{X}_n^* - \bar{X}_n) \leq x) \tag{3.2}$$

we have, with P-probability 1,

$$\sup_x |G_n(x) - G_n^*(x)| \to 0, \text{ as } n \to \infty \tag{3.3}$$

whenever $0 < \int x^2 dF(x) < \infty$. However, the question remains: how well does Efron's bootstrap estimate G_n^* approximates G_n? The answer is that typically the rate of convergence in (3.3) is of the classical order $n^{-\frac{1}{2}}$. The famous Berry-Esseen theorem asserts that the accuracy of the normal approximation is of the same order $n^{-\frac{1}{2}}$, provided $\int |x|^3 dF(x) < \infty$. However, we can easily improve the accuracy of our bootstrap estimate, by first employing 'Studentization'. That is, instead of the statistical quantity $n^{\frac{1}{2}}(\bar{X}_n - \mu)$ and its bootstrap version $n^{\frac{1}{2}}(\bar{X}_n^* - \bar{X}_n)$, we consider the old and famous Student t statistic $n^{\frac{1}{2}}(\bar{X}_n - \mu)/S_n$ and its bootstrap counterpart $n^{\frac{1}{2}}(\bar{X}_n^* - \bar{X}_n)/S_n^*$, with respective distribution functions

$$G_{ns}(x) = P(n^{\frac{1}{2}}(\bar{X}_n - \mu)/S_n \leq x), \quad -\infty < x < \infty, \tag{3.4}$$

and

$$G_{ns}^*(x) = P_n^*(n^{\frac{1}{2}}(\bar{X}_n^* - \bar{X}_n)/S_n^* \leq x), \quad -\infty < x < \infty. \tag{3.5}$$

where $S_n^2 = (n-1)^{-1}\sum_{i=1}^n (X_i - \bar{X}_n)^2$ denotes the sample variance. Note that S_n^{*2} is nothing but S_n^2, with the X_i's replaced by the X_i^*'s. We note in passing that, if F is normal, G_{ns} of course reduces to the well-known Student t distribution with $n-1$ degrees of freedom. In general, however, the exact distribution G_{ns} of Student's t is unknown, but we can try to estimate it, e.g., by using the bootstrap. Similarly, as in (3.3) we have that

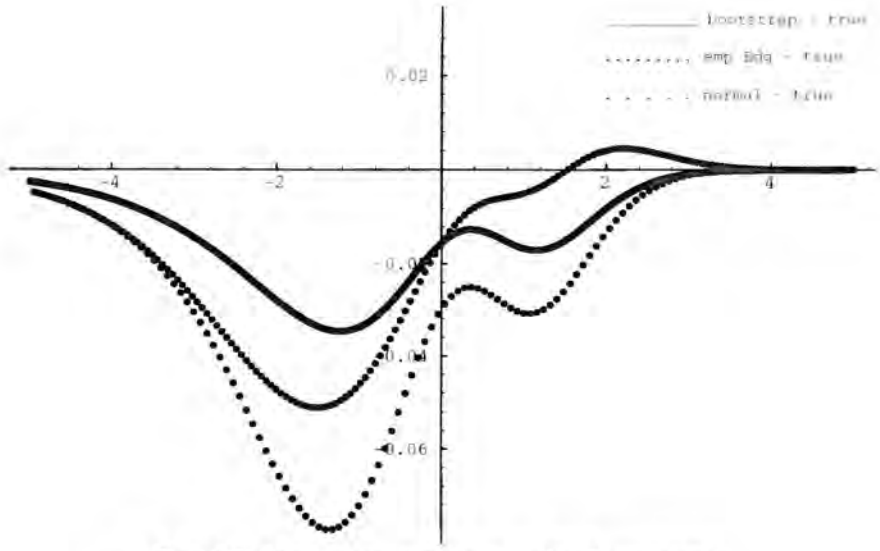


Figure 2. Three approximations; $n = 20$; F exponential.

$$\sup_x |G_{ns}(x) - G^*_{ns}(x)| \to 0, \text{ as } n \to \infty, \tag{3.6}$$

but now the rate of convergence is faster: in fact, one can show that in P-probability

$$n^{\frac{1}{2}} \sup_x |G_{ns}(x) - G^*_{ns}(x)| \to 0, \text{ as } n \to \infty \tag{3.7}$$

under rather weak conditions. Under somewhat more stringent assumptions, one can prove that $\sup_x |G_{ns}(x) - G^*_{ns}(x)|$, the accuracy of the bootstrap approximation, is of the exact order n^{-1} in P-probability. In contrast, the normal approximation for Student's t possesses the classical Berry-Esseen type error of order $n^{-\frac{1}{2}}$.

250

In other words: the bootstrap estimate G^*_{ns} is asymptotically closer to G_{ns} than the standard normal distribution. This 'bootstrap is better than normal' property of Efron's bootstrap for the Student t statistic clearly suggests the benificial effect of 'Studentization' before bootstrapping for this important special case. A Monte-Carlo result, which supports this claim, is presented in figure 2 (borrowed from Putter's thesis (1994)). We consider the special case that F is exponential. First of all, the distribution G_{ns} was approximated by Monte-Carlo using 10^7 samples. Next a sample of size $n = 20$ was drawn from a standard exponential distribution and —based on this sample— the distribution G_{ns} was estimated in three ways, first using the classical normal approximation, secondly using the bootstrap G^*_{ns} (as in (3.5), using Monte-Carlo simulation with $B = 10^6$). With this choice

of B, we are pretty certain that the Monte-Carlo error is negligible. Note that we should take care that too low a choice for B doesn't ruin the second order accuracy of the bootstrap estimate G^*_{nS}. In fact, it is easily checked that this means that the Monte-Carlo error – which is of order $\frac{1}{\sqrt{B}}$ – should be of a smaller order than $\frac{1}{n}$, the accuracy of the bootstrap approximation, i.e., B should be of a larger order than n^2. The third way uses empirical Edgeworth expansion (EEE). To make the differences between these three methods discernible we have plotted for each of the three methods the resulting estimate minus the target distribution G_{nS}. The graph that lies closest to zero corresponds therefore to the best approximation. It is clearly seen that both bootstrap and Edgeworth expansion outperform the normal approximation. The bootstrap performs slightly better than EEE, due to the fact that the bootstrap also implicitly estimates higher order terms in the Edgeworth expansion consistently.

3.2. Non-smooth cases

The above result for Student's t is in fact already known for about 10 years (cf., e.g., the references in [2]) and it is generally viewed as an important argument in favour of Efron's bootstrap. Helmers proved (1991) that the 'bootstrap is better than normal' property also holds true for more complicated nonlinear statistics like Hoeffding's famous class of U-statistics. The extension of the 'bootstrap is better than normal' property to arbitrary Studentized symmetric statistics is still an interesting open problem at present. In any case, however, the quadratic and higher order terms in the Hoeffding decomposition for a symmetric statistic $T_n = T_n(X_1, \ldots, X_n)$ should be of a required order of magnitude, otherwise the speed of bootstrap convergence asserted in (3.7) fails to hold. An important specific example of the latter is the case of the median, and more generally, quantiles. In such 'non-smooth' cases (the parameter of interest, e.g., $\theta = \theta(F) = F^{-1}(\frac{1}{2})$ is a much less smooth functional of F, then the parameter $\theta = \theta(F) = \int x dF(x)$) we have a much slower rate (roughly of order $n^{-\frac{1}{4}}$) of convergence of Efron's bootstrap approximation. In fact, although Efron's bootstrap for the median is consistent, it is worthless in practice, even for a sample size n as large as 100. In the computer calculations that led to figure 3 we have generated a sample of size $n = 100$ from a standard normal distribution. As a result we find that the difference with the true distribution function is maximized at $x = 0.39$; at this point the true distribution equals 0.657 while the bootstrap approximation yields 0.935, which means a relative error of more than 40%. It is well known that the smoothed bootstrap, where resampling is done from a smoothed version of the empirical distribution, results in a better approximation. For the smoothed bootstrap we used a normal kernel and a bandwidth $h = 0.1$, and indeed the smoothed bootstrap seems to perform much better. A more sophisticated choice of kernel and bandwidth will

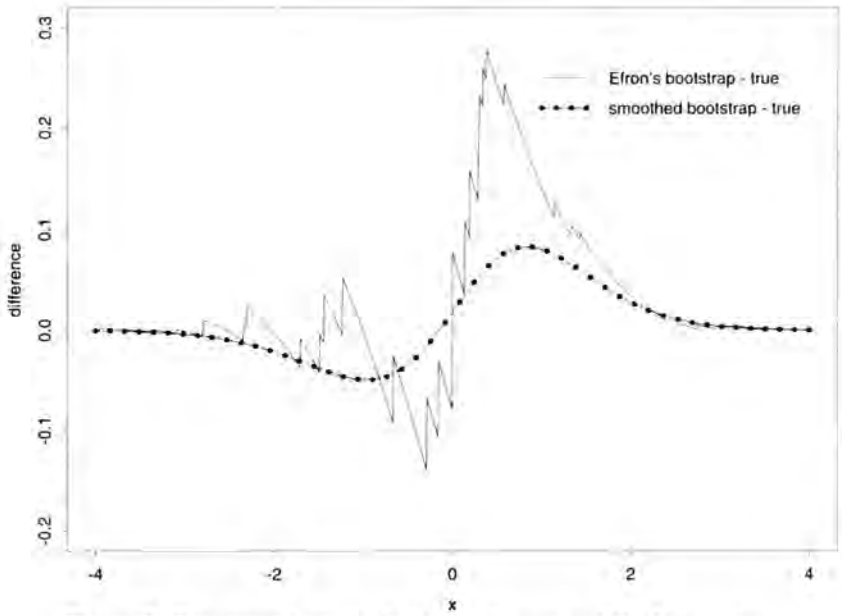


Figure 3. Two approximations for the median; $n = 100$; F normal.

presumably lead to a further reduction of the error of the smooth bootstrap approximation.

Related work for the more general case of U-quantiles can be found in a contribution of Helmers, Janssen and N. Veraverbeke to [4] (cf. Helmers, M. Hušková (1994) for an extension to multivariate U-quantiles). Specific examples of interest of U-quantiles are the well-known Hodges-Lehmann estimator of location, which is given by the median of all pairwise averages, and an estimator of spread proposed by Bickel and Lehmann. Young, p. 392 in a prominent recent review paper [5], feels that 'this sort of work is important', because 'the contexts to which the results apply are highly relevant to precisely the sort of circumstances—when there is limited knowledge about the underlying distribution—for which bootstrap was designed'.

4. Applications

To conclude I briefly discuss two selected topics of current interest in bootstrap theory and its applications: resampling methods for finite populations, and spatial bootstrapping.

Resampling methods for finite populations is an important topic of current interest. Helmers and M.H. Wegkamp considered (1995) the situation where the finite population is viewed as a realization of a certain superpopulation model (heteroscedastic linear regression, without intercept). This

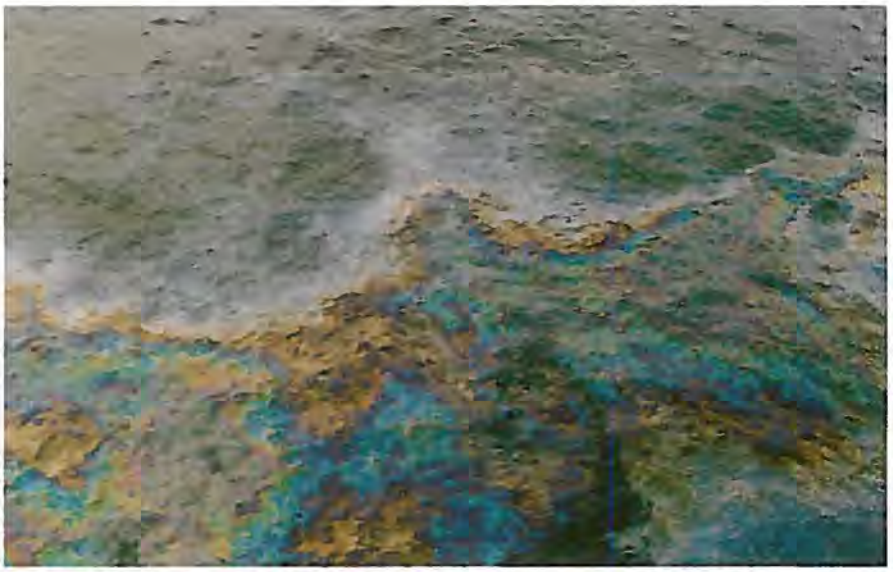


Figure 4. Observed oilspot.

enables us to incorporate auxiliary information (past experience) in the statistical analysis. The authors first came across this problem in a 1994 statistical consultation project at CWI with The Netherlands postal services PTT Post. In this setup a new resampling scheme called 'two-stage wild bootstrapping' is proposed and studied. The basic probabilistic tool we employ in our mathematical analysis is the celebrated Erdös-Rényi central limit theorem for samples without replacement from a finite population.

Bootstrapping with spatial data is very clearly an important area for future work in the research group 'Image analysis and spatial stochastics' of CWI. We briefly describe here a practical application in which spatial bootstrapping is used. In a project commissioned by the North Sea Directorate, Ministry of Public Works the problem is to estimate the intensity of oilpollution in the North Sea. The available real data sets ('marked planar point patterns') consist of the locations and sizes (marks) of the oilspots observed (cf. figure 4) by a surveillance aircraft. A planar inhomogeneous Poisson point process with intensity function $\lambda(\cdot, \theta)$—parameterized by a finite-dimensional parameter θ—was used as a spatial (parametric) model for the locations of (the centres of) oilspots. The parameterization enables one to incorporate the available a priori knowledge about oil pollution, such as the location of sources of oilpollution (i.e. shipping areas or off-shore locations) and the intensity of shipping in various regions. However, nothing

seems to be known about the distribution of the volumes (marks) of oilspots, but we can of course use the sizes of the observed oilspots to estimate it (nonparametric approach). In this setup a simple semiparametric form of spatial bootstrapping was developed in order to estimate the accuracy of the estimated total amount of oilpollution in the North Sea.

5. Acknowledgement

I want to thank H. Putter for his contributions to the present paper.

The interested reader is referred to [1] for an excellent introduction to the bootstrap. Uses of Edgeworth expansions in the mathematical analysis of Efron's bootstrap is the topic of the research monograph [2]. Additional information on the bootstrap may also be found in the proceedings volume [4] and discussion paper [5]. The present article is basically a shortened nontechnical revision of [3]. The latter reference also contains a more complete list of references.

References

1. B. Efron, R.J. Tibshirani (1993). *An Introduction to the Bootstrap*. Chapman and Hall. New York.
2. P. Hall (1992). *The Bootstrap and Edgeworth expansion*. Springer. New York.
3. R. Helmers, H. Putter (1995). Bootstrap resampling: a survey of recent research in The Netherlands, CWI Report BS-R9517. To appear in *Proceedings of the SEAMS Regional Conference on Mathematical Analysis and Statistics*, Yogyakarta, Indonesia, July 10-13, 1995.
4. R. Helmers, P. Janssen, N. Veraverbeke (1992). Bootstrapping U-quantiles. R. LePage, L. Billard (eds.). *Exploring the Limits of Bootstrap*. Wiley. New York.
5. G.A. Young (1994). Bootstrap: more than a stab in the dark? *Statistical Science 9(3)*. 382-415.

Morphological Image Processing

H.J.A.M. Heijmans

1. Introduction

Among the major tasks in the field of image processing and analysis are feature extraction, shape description, and pattern recognition. Such tasks inherently require a geometry-oriented approach as they refer to geometrical concepts such as size, shape and orientation. However, until recently the most important tools in image processing were of a probabilistic and analytic nature, and were based upon, e.g., the correlation of signals and the frequency analysis of the Fourier spectrum.

Mathematical morphology is an approach to image processing which is based on set-theoretical, geometrical and topological concepts, and as such it is particularly useful for the analysis of geometrical structure in an image. In contrast to the traditional approach using Fourier analysis, morphology is highly nonlinear in nature, and poses several challenging mathematical problems. Below we shall briefly describe the historical development of this approach, explain some of its basic techniques, and discuss some recent theoretical developments, with an emphasis on those carried out at CWI.

2. The nature of mathematical morphology?

It is interesting to have a deeper reflection upon the origin and nature of mathematical morphology. What is it? Where does it come from? How does it operate?

The word 'morphology' stems from the Greek words *μορφή* and *λογος*

meaning 'the study of forms'. The term is encountered in a number of scientific disciplines including biology and geography. In the context of image processing it is the name of a specific methodology designed for the analysis of the geometrical structure in an image. It was founded in the early sixties by two researchers at the Paris School of Mines in Fontainebleau, G. Matheron [3] and J. Serra [5], who worked on a number of problems in mineralogy and petrography. Their main goal was the automatic analysis of the structure of images from geological and metallurgic specimens. They were particularly interested in the quantization of the permeability of a porous medium and the petrography of iron ores. Their investigations ultimately led to a new quantitative approach in image analysis, nowadays known as mathematical morphology. During the last two decades, this discipline has gained increasing popularity among the image processing community and has achieved the status of a powerful alternative to the classical linear approach. It has been applied in numerous practical situations, e.g., mineralogy, medical diagnostics, histology, industrial inspection, computer vision and character recognition.

Mathematical morphology has three aspects: an algebraic one, dealing with image transformations derived from set-theoretical and geometrical operations, a probabilistic one, dealing with models of random sets applicable to the selection of small samples of materials, and an integral geometric one, dealing with image functionals. Only the first aspect will be addressed here.

By its very nature, mathematical morphology is set-based, that is, it treats a binary image as a set. The corresponding morphological operators use essentially only four ingredients from set theory: set intersection, union, complementation, and translation. As a result such operators are translation invariant; additionally, they are highly nonlinear.

One of the basic intuitions of mathematical morphology is that the analysis of an image does not reduce to a simple measurement. Instead, it relies on a succession of operators which transform it in order to make certain features apparent. Indeed, a picture usually contains an unstructured wealth of information; in order to analyze it, one has to distinguish meaningful information from irrelevant distortions. One has to extract what is of interest. In practice this amounts to transformations which reduce the original image to a sort of caricature. For example, in optical character recognition, one can simplify the task by first performing a *skeletonization* on a binary digital image representing a typed text, which reduces each connected component to a one-pixel-thick skeleton retaining its shape; this discards all (useless) information about the thickness of characters, and the reduced amount of information contained in such an image makes further recognition steps quicker and easier.

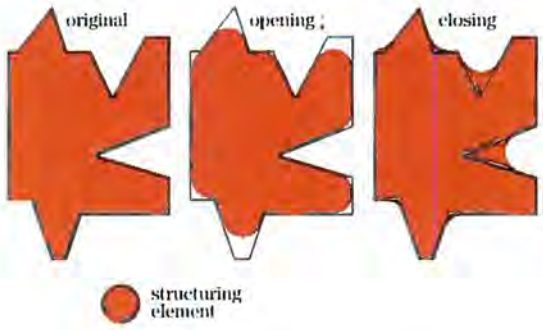


Figure 1. Opening and closing of a polygon by a disk.

3. AN EXAMPLE: OPENINGS

The central idea of mathematical morphology is to examine the geometrical structure of an image by probing it with small patterns, called *structuring elements*, at various locations in the image. By varying the size and shape of the structuring elements, one can extract useful shape information from the image. This procedure results in image operators which are well-suited for the analysis of the geometrical and topological structure of an image.

This is perhaps best illustrated by discussing one operator in more detail, the *opening*, one of the most important operators in daily morphological practice. Restricting to binary (i.e., black-and-white) images modelled by $\mathcal{P}(\mathbb{R}^2)$, the subsets of $\mathbb{R}^2$, we say that the mapping $\alpha : \mathcal{P}(\mathbb{R}^2) \to \mathcal{P}(\mathbb{R}^2)$ is an *opening* if it is

- *increasing*: $X \subseteq Y$ implies $\alpha(X) \subseteq \alpha(Y)$;
- *idempotent*: $\alpha(\alpha(X)) = \alpha(X)$;
- *anti-extensive*: $\alpha(X) \subseteq X$.

We discuss three different types of openings here: the structural opening, the linear opening, and the area opening.

The *structural opening* requires a structuring element $A \subseteq \mathbb{R}^2$. It is the union of all translates of A which are contained inside X:

$$X \circ A = \bigcup \{A_h \mid h \in \mathbb{R}^2 \text{ and } A_h \subseteq X\}.$$

Here A_h denotes the translate of A along the vector h. This opening is illustrated in figure 1, along with its *negative*, the structural closing by A, which is essentially an opening of the background. A closing operator, say β, is increasing, idempotent, and extensive (i.e., $X \subseteq \beta(X)$).

The *linear opening* uses a (finite or infinite) collection of bounded line segments L_i, $i \in I$, with different directions for structuring elements. It is defined by

$$\alpha_L(X) = \bigcup_{i \in I} X \circ L_i.$$

Finally, the *area opening* uses the notion of (arc-)connected component. Let $S \geq 0$ be a real number, then $\alpha_S(X)$ comprises all components of X with area larger than S. The three different openings are illustrated in figure 2.

Openings are used for different purposes, such as image filtering (see section 5). Here we discuss a different application, the computation of size distributions.

Consider the family of structuring elements rB, the spheres in $\mathbb{R}^2$ centered at the origin and with radius $r > 0$. The family of structural openings $\alpha_r(X) = X \circ rB$ satisfies the following semigroup property:

$$\alpha_r \alpha_s = \alpha_s \alpha_r = \alpha_r \quad \text{if } r \geq s.$$

This is due to the fact that a larger ball can be obtained as a union of smaller ones. This semigroup property forms the basis for a formal definition of a *size distribution*. The openings α_r formalize the intuitive idea of the sieving of a binary image according to the size and shape of grains within the image. As the mesh size of the sieve (the radius r) is increased, more of the image grains will fall through the sieve and the residual area of the filtered (sieved) image will decrease monotonically. These residual areas form a size distribution, called *granulometric size distribution*, that is indicative of the image structure. Its derivative is a density function, called the *granulometric size density*.

The opening transform of a binary image $X \subseteq \mathbb{R}^2$ is a function $F : \mathbb{R}^2 \to \mathbb{R}_+$ whose value $F(h)$ at the point h represents the radius r of the largest sphere which contains h and fits entirely inside X. Its histogram corresponds with the granulometric size density. See figure 3 for an illustration.

4. Complete lattice framework

Although, originally, mathematical morphology was developed for binary images, from the very beginning there was a need for a more general theory. Such a theory should be powerful enough to handle different object spaces such as the closed subsets of a topological space, the convex sets of a (topological) vector space, and grey-scale images.

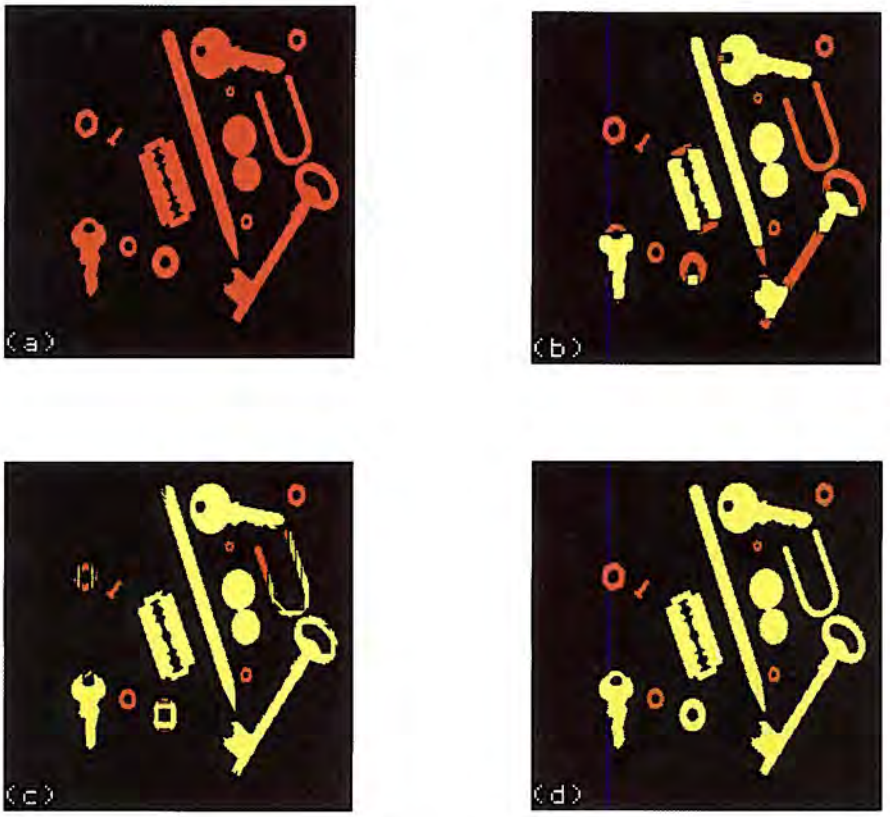


Figure 2. Three different openings. (a) binary input image (foreground in red); (b) the structural opening by a 7×7 square (yellow); (c) linear opening using four line segments (horizontal, vertical, and diagonal) with length 15; (d) area opening with $S = 256$.

Besides this enormous variation in object spaces there is yet another generalization which is quite important. It is, namely, by no means obvious why morphological operators have to be translation invariant. In radar imaging, for example, rotation invariance is more appropriate. Furthermore, there are a number of situations where perspective transformations enter naturally. Think, for instance, of the problem of monitoring the traffic on a highway with a camera at a fixed position. It is obvious that in such a configuration the detection algorithms should take into account the distance between the camera and the object (e.g. a car).

Only recently has it been realized that complete lattices are the right mathematical framework for a general theory of morphology: see [1, 2, 4,

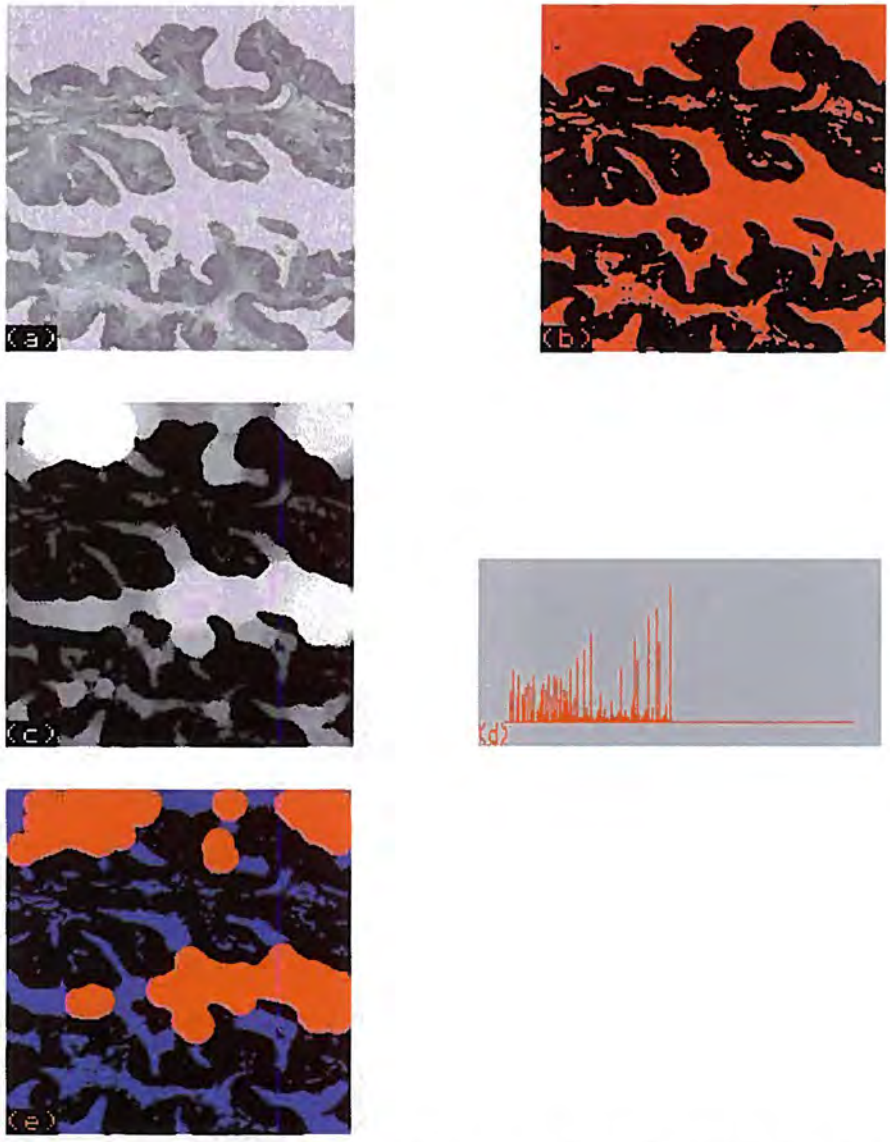


Figure 3. Opening transform and size distribution. (a) Grey-scale image; (b) binary image obtained by thresholding; (c) its opening transform; (d) the histogram of grey-values of the opening transform (which corresponds with the granulometric size density); (e) the binary image (red), obtained by thresholding the opening transform. In this particular case the 5-7-11-chamfer metric has been used as an accurate discrete approximation of the continuous Euclidean distance.

6]. The main motivation for this generalization is that it unifies a number of particular examples into one abstract mathematical framework; they help to prevent the periodic 'reinvention of the wheel' which happens too often in applied mathematics and engineering, where 'new ideas' are sometimes particular cases of 'old ideas' in pure mathematics. A second motivation intimately connected to the previous one is that an abstract approach provides a deeper insight into the essence of the theory (which assumptions are minimally required to have certain properties?) and links it to other, sometimes rather old, mathematical disciplines.

The mathematical morphology research group at CWI has made a substantial contribution to the development of the complete lattice framework for morphology [1].

5. MORPHOLOGICAL FILTERS

5.1. Introduction

Another class of problems dealt with at CWI concerns the construction of morphological filters. One goal of image filtering may be the enhancement of the visual quality of a distorted image. More frequently, however, its goal is to make the image more suitable for subsequent image processing tasks, such as segmentation.

In mathematical morphology a 'filter' is an operator which is increasing and idempotent. Idempotence seems a sensible requirement for a filtering operation as it characterizes the successive stages of a series of transformations in image analysis. Indeed, if an operation is idempotent, then there is no point in repeating it, and so we must do something else, i.e., go to another stage. Conversely, a stage must produce a clear result, and not stop halfway.

Given a filtering operator ψ which is not idempotent, one often applies it until the result does not change anymore. This corresponds to a conditional loop, such as '**while** ... **do** ...' ; provided such a loop eventually terminates, it implements an idempotent operation. However, in general there is no guarantee of convergence. At CWI a theory has been developed which says under what sort of conditions on the operator ψ, iteration leads to idempotence. This theory covers all the interesting cases occurring in practice.

5.2. Alternating sequential filters

By means of illustration we describe one family of morphological filters in more detail, namely the alternating sequential filters based on rank-order. This class was 'invented' recently by the author. To define it, we need to introduce some notation and terminology. Denote by $\mathrm{Fun}(\mathbb{Z}^2)$ the class of grey-scale functions $F : \mathbb{Z}^2 \to \{0, 1, \ldots, N\}$, where N is an integer.

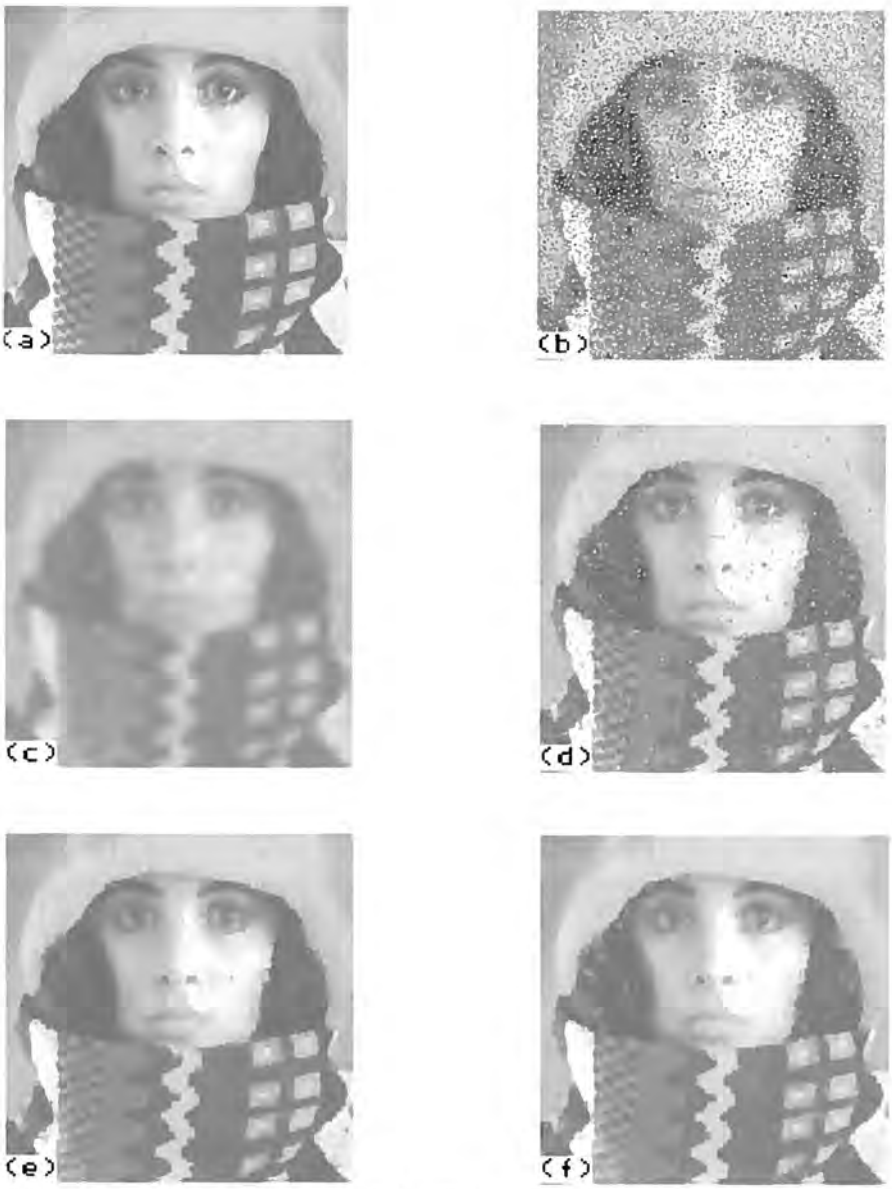


Figure 4. Morphological filtering: (a) undistorted image; (b) distorted version of (a) in which about one-third of the pixels are affected by noise; this image is used as input image F; (c) Gaussian filtered version of F; (d) median $\rho_5(F)$; (e) and (f) are the alternating sequential filtered images $(\beta\alpha)_9(F)$ and $(\alpha\beta)_9(F)$, respectively. Note that the linear Gaussian operator blurs the image, whereas the morphological operators are able to remove noise without blurring the image.

Let α_k, $k = 1, 2, \ldots, n$, be a sequence of openings such that

$$\alpha_n \leq \alpha_{n-1} \leq \cdots \leq \alpha_1.$$

For example, α_k may be the structural opening with a $(2k+1) \times (2k+1)$ structuring element. Dually, let β_k, $k = 1, 2, \ldots, n$, be a sequence of closings such that

$$\beta_n \geq \beta_{n-1} \geq \cdots \geq \beta_1.$$

Denote by $(\alpha\beta)_n$ the composition

$$(\alpha\beta)_n = \alpha_n\beta_n\alpha_{n-1}\beta_{n-1}\cdots\alpha_1\beta_1.$$

The composition $(\beta\alpha)_n$ is defined analogously. Now the following result holds.

Proposition. *Under the given assumptions, the operators $(\alpha\beta)_n$ and $(\beta\alpha)_n$ are morphological filters on $\mathrm{Fun}(\mathbb{Z}^2)$, i.e., both operators are increasing and idempotent.*

The filters $(\alpha\beta)_n$ and $(\beta\alpha)_n$ are called *alternating sequential filters*. One particular example will be discussed here. Consider the points $0, p_1, p_2, \ldots$ origin; define the rank operator ρ_k (where $k = 1, 2, \ldots, 9$) as follows: sort, for a given input image F and pixel $x \in \mathbb{Z}^2$, the values $F(x), F(x+p_1), F(x+p_2), \ldots, F(x+p_8)$ in decreasing order and take as output $\rho_k(F)(x)$ the value at the k'th position. The operator ρ_1, which returns as output the maximum of the values $F(x+p_i)$, is called *dilation*, and is denoted by δ. Dually, ρ_9, which returns the minimum is called *erosion*, and is denoted by ε. The operator ρ_5 is called the *median operator*. It is evident that

$$\rho_1 \geq \rho_2 \geq \cdots \geq \rho_9.$$

The operator $\alpha_k = \mathsf{id} \wedge \delta\rho_k$, where id is the identity operator ($\mathsf{id}(F) = F$) and $\wedge$ denotes the (pointwise) minimum, is an opening, called *rank-max opening*; see [4] or [1]. It follows that α_k is a decreasing sequence and that $\alpha_1 = \mathsf{id}$.

Dually, the operator $\beta_k = \mathsf{id} \vee \varepsilon\rho_{10-k}$ is a closing, the *rank-min closing*. The sequence β_k is increasing, and $\beta_1 = \mathsf{id}$.

The proposition stated above implies that the compositions $(\alpha\beta)_k$ and $(\beta\alpha)_k$ are morphological filters. In figure 4 one can see that these filters are eminently suited for noise cleaning.

REFERENCES

1. H.J.A.M. HEIJMANS (1994). *Morphological Image Operators*, Academic Press, Boston.

2. H.J.A.M. Heijmans, C. Ronse (1990). The algebraic basis of mathematical morphology—part I: Dilations and erosions. *Computer Vision, Graphics and Image Processing 50*. 245-295.
3. G. Matheron (1975). *Random Sets and Integral Geometry*. John Wiley and Sons. New York.
4. C. Ronse. H.J.A.M. Heijmans (1991). The algebraic basis of mathematical morphology—part II: Openings and closings. *Computer Vision, Graphics and Image Processing 54*. 74-97.
5. J. Serra (1982). *Image Analysis and Mathematical Morphology*. Academic Press. London.
6. J. Serra (1988). *Image Analysis and Mathematical Morphology. II: Theoretical Advances*. Academic Press. London.

Numerical Algorithms for Transport-Chemistry Problems

J.G. Verwer

1. Introduction

Ever increasing emissions of pollutants in the atmosphere, ground water and surface water, as a consequence of the growing population and economic activities, are affecting our environment. The negative effects become more and more noticeable: smog in urban regions (see also figure 1), ground water contamination, the growth of algae in surface water, and even atmospheric climate change has to be feared. Since physical experiments are in general too costly, or even impossible, mathematical simulations become increasingly important to study the long term effects of these emissions and to predict the efficiency of policy decisions to reduce the effects.

Once emitted, the pollutants will be transported by wind or water and dispersed by molecular diffusion and turbulence, while at the same time complex chemical reactions take place. The mathematical description of these processes is given by a large system of time dependent 3-dimensional partial differential equations of the advection-diffusion-reaction type involving the pollutants and all other substances that play a role in the reactive chain. For the numerical solution of these systems the availability of efficient numerical algorithms and the access to large computers is crucial. In fact, computer capacity is, and will remain, a critical factor. Though computer power continues to expand, the computational requirements for high resolution transport models with full chemistry are still out of range for many applications.

Figure 1. Smog above Rotterdam Harbour (Photo: Geosens).

Numerical analysis can assist the development of new models by providing more efficient numerical methods, tailored to the application at hand, and the full exploitation of the evolving computer architectures, such as advanced multi vector processors and, in the near future, massively parallel processing systems.

Below the numerical aspects will be outlined and discussed. Thereafter
266 a short description will be given of three projects presently carried out at CWI concerning transport of chemically reactive substances. These projects are part of CWI's research programme Mathematics & the Environment.

2. NUMERICAL RESEARCH

The basic mathematical equations describing transport and chemistry consist of a system of s partial differential equations for the unknown concentrations $c_k(x,t)$, $k = 1,\ldots,s$, which depend on time t and space $x = (x_1, x_2, x_3)$ in a 3-dimensional domain Ω. The equations, derived from mass balances, are given by

$$\frac{\partial}{\partial t}c_k(x,t) + \sum_{i=1}^{3}\frac{\partial}{\partial x_i}\left(u_i(x,t)c_k(x,t)\right) =$$

$$\sum_{i=1}^{3} \frac{\partial}{\partial x_i} \left(d_i(x,t) \frac{\partial}{\partial x_i} c_k(x,t) \right) + f_k(x,t,c_1(x,t),\ldots,c_s(x,t)),$$

with suitable initial and boundary conditions. The quantities $u_i(x,t)$ represent the velocities of the transport medium, such as water or air. These are either given in a data archive or computed alongside with a Navier-Stokes or hydrodynamical shallow water program. The diffusion coefficients $d_i(x,t)$ are constructed by the modellers and include also parametrizations of turbulence. The final term $f_k(x,t,c(x,t))$, which gives a coupling between the various substances, describes the nonlinear chemistry and also emissions (sources) and depositions (sinks). In actual models these equations are augmented with other suitable sub-grid parametrizations and coordinate transformations.

The number of substances s can be large, for example up to 100 in current atmospheric models. Also the spatial domain Ω can be very large, which implies that many grid points are needed for a proper spatial resolution, say hundred thousand to millions. Moreover, often one is interested in long term effects, so that the equations have to be integrated over long time intervals.

The huge size of the problems makes them difficult to implement, even on modern supercomputers. In order to obtain numerical solutions, without using excessive computer time or memory, fast and efficient numerical algorithms are crucial. The numerical results should be reliable and the numerical errors should be well below the errors introduced by physical approximations in the model. These demands cause a number of outstanding numerical challenges, some of which are discussed below.

2.1. Efficient chemistry schemes

In many applications the chemistry is stiff, which means that the reactions take place on very different time scales. To obtain accurate solutions it would be unnecessary to represent the very fast reactions, but for numerical stability this is required when using explicit methods. For this reason, such equations are commonly handled with implicit methods, where at each time step a system of nonlinear algebraic equations arises. Usually these implicit relations are solved with a Newton type iteration, which requires calculation of Jacobians and the solution of large linear systems, with much computer storage needed.

A way to overcome the problem of implicitness has been developed at CWI [4]. Instead of using a Newton iteration, one employs a Gauss-Seidel iteration. By using the special form of the chemical equations this leads to numerical schemes that are essentially explicit but still have the favourable stability properties of implicit schemes. Recent comparisons have shown that this approach can lead to codes which are for practical accuracy de-

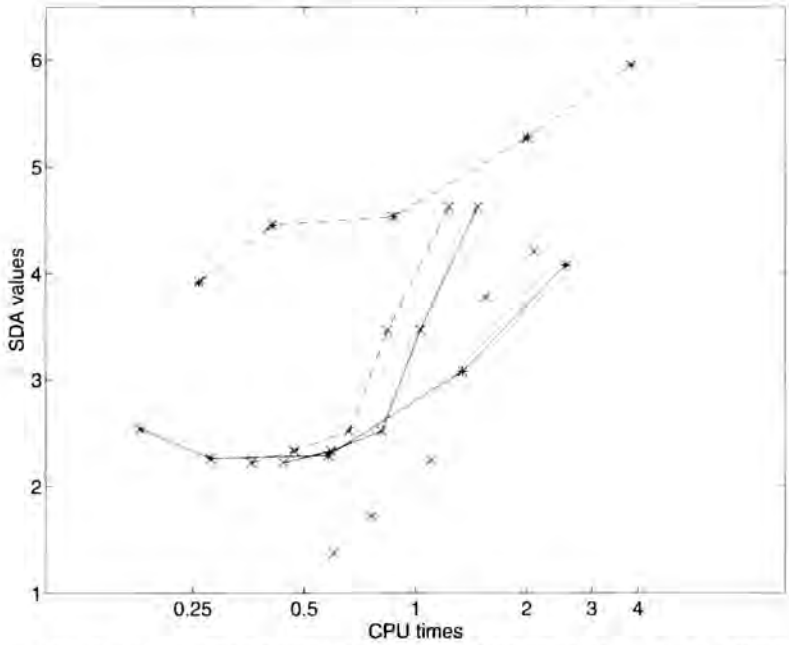


Figure 2. Results for TWOSTEP1 (*, solid), TWOSTEP2 (*, dashed), VODE1 (×, dotted), VODE2 (x, solid), VODE3 (x, dashed).

mands at least competitive with modern codes for general stiff problems, such as VODE, and for problems where the number of chemical substances is not too large significant gains in computing time (CPU) have been achieved.

An example is presented in figure 2 where the number of significant digits (SDA $= -\log_{10}(\text{error})$) is plotted against the CPU times for several versions of the codes TWOSTEP and VODE applied to a chemical ozone system with 15 substances; TWOSTEP is based on the BDF2 method with Gauss-Seidel iteration, whereas VODE uses a range of BDF methods with Newton iteration. The versions TWOSTEP1 and VODE1 are used as black-box solvers. In TWOSTEP2 some substances are grouped together which are known a-priori to have strong interactions. In VODE2 an analytic expression for the Jacobians is provided (calculated with the formula manipulation program MAPLE) and VODE3 uses also a manual reordering of the equations to exploit sparsity patterns for the linear algebra. In this comparison TWOSTEP1 needs less CPU time to reach the 1% error level than its black-box counterpart VODE1. The code TWOSTEP2 gives here a much better accuracy for given CPU time than the other VODE schemes.

2.2. Operator splittings

The different parts of the equations can be solved most efficiently by different numerical methods, but the different modules then have to be combined after each time step. This will lead to additional numerical errors, the so-called splitting errors. If the equations are strongly advection dominated, as they usually are in practice, these splitting errors can be strongly reduced, with little computational work, by solving the chemistry equations along the transport paths (characteristics).

It is still an open question which parts of the equations should be treated separately. In general, it is expected that numerical errors can be reduced by the simultaneous solution of processes with comparable time scales. In a recent research report (supported by the CRAY Research Grant Program) it has been shown that chemistry and vertical diffusion can be solved simultaneously in an efficient way by using a Gauss-Seidel approach, similar as for the chemistry only, combined with tri-diagonal solvers for the vertical diffusion.

2.3. Advection transport schemes

The accurate solution of advective transport is still a difficult numerical problem, due to the following requirements:

- The equations are mass conservative and a numerical method should mimic this behaviour.
- Concentrations are nonnegative, of course, but to maintain this property in the numerical solution the standard discretizations in space of order 2 or higher cannot be used.
- Strong gradients may be present in the concentration profiles. These gradients should not lead to oscillations (wiggles) but they should also not be smeared out by the numerical method.

A good compromise seems to be a combination of accurate spatial discretizations and flux limiters, together with explicit Runge-Kutta time integration, see [1]. However, if the advection is coupled with diffusion coefficients that are not small or if the grid spacing is fine in certain regions, then numerical stability will necessitate very small time steps and this will cause a degradation of efficiency.

Several ways to avoid such small time steps have been examined at CWI. With moderate diffusion coefficients one can use a Hopscotch type splitting to solve advection and diffusion simultaneously [3]. With small grid spacings in certain regions a dimension splitting approach can be advantageous, where the multi-dimensional advection problem is replaced by a series of 1-dimensional problems, which can be solved easily without any time step restriction for stability [2].

2.4. Local grid refinements

On the spatial domain there will be regions with smooth solutions and little chemical activity as well as regions with strong gradients and large chemical activity, caused by strong local emissions for instance. In the latter regions a high resolution is needed to avoid large numerical errors, whereas in the other regions less resolution would lead to better efficiency. This can be accomplished with local grid refinement where either the user specifies the high-resolution regions or where such regions are computed adaptively using suitable monitor functions. The use of local refinements for regional air-pollution models is being examined at present in the project EUSMOG.

2.5. HPCN: high performance computing and networking

The very large number of equations needed for an accurate description of transport and chemistry in many applications requires full use of advanced supercomputers, such as the CRAY C90, with possibilities of vector and parallel processing. At present, models are frequently simplified by restricting the number of chemical substances and spatial resolution in view of available computer capacity and speed. With the fast growth of the available meteorological and physical data, for example through remote sensing, it is expected that in the near future massively parallel computer systems will become increasingly important for the realization of new models.

The full use of such computer systems will force numerical analists to rethink the setup of algorithms and implementations. Although the design of massively parallel architectures is still in motion, numerical research in this direction is necessary to anticipate the future developments. At CWI a pilot project has started which examines the possibilities of the CRAY T3D system for use in regional atmospheric models.

3. Projects at CWI

At present three projects are carried out at CWI concerning transport of pollutants with various application backgrounds.

3.1. EUSMOG

The project EUSMOG is carried out in close cooperation with scientists of the Air Laboratory of RIVM – the Dutch National Institute of Public Health and Environmental Protection. RIVM also gives financial support for the project.

The goal of this project is to develop fast numerical algorithms for the prediction of smog episodes. The model uses a 4-layer parameterization in the vertical direction and a chemical ozone model with 15 chemical substances. The numerical research in this project is mainly devoted to the development of new chemical schemes and the use of local grid refinements

to represent local emission sources and to follow strong gradients in space and time.

Figures 3 and 4 are produced with CWIROS, the code which is being developed at CWI for summer and winter smog. Figure 3 gives a 5-day prediction of sulfur-dioxide SO_2 concentrations over Europe using emission data and meteorological input parameters from RIVM. Figure 4 gives the computational grid at ground level used at the end of the prediction. The grid refinement procedure is fully automatic and the grid adjusts itself to the solutions. Since the grid is strongly refined at small regions only, this procedure is much more efficient than using overall refinements.

3.2. TRUST

In the TRUST project we study the simulation of transport and bio-chemical interaction of salinity, pollutants, suspended material (such as sediment or mud), etc. in shallow seas. These seas play an important role as a link between land and ocean and usually have a highly productive ecosystem. As an example, we may consider the North Sea, which is partly surrounded by highly industrialized countries; as a consequence, it is subject to a large number of effects, such as natural sewage in urban sewer outlets, pollution by rivers, recreation, heavy transport, and, in this example, also the exploration of minerals (gas and oil), and fishery. Moreover, the interaction with the ocean and the atmosphere has a significant influence on the ecosystem. Although we are still far away from the final goal, i.e., a thorough understanding, and control, of the marine ecosystem dynamics in shallow seas, the TRUST project tries to contribute in pushing the problem one step further into the direction of its final solution. The research at CWI concentrates on the development of fast, parallel algorithms for the time integration of the advection-diffusion-reaction equations [3]. In contrast to air-pollution models, the chemical reactions in water are usually quite slow, implying that the reaction terms do not need special concern, and can be treated by standard explicit methods. For the advection-diffusion part, we developed a so-called Hopscotch method, which seems to be very promising, both with respect to numerical properties as well as with respect to vectorization/parallelization capabilities. An other point of interest in this context is that various regions require different spatial grid resolution. For example, to capture the local phenomena in coastal regions and estuaries, a much finer grid is needed. This is clearly illustrated in figure 5, showing the distribution of suspended material in the North Sea. Since the information shown in this figure is very costly and time-consuming to obtain, it is clear that efficient numerical simulations are of utmost importance. In the TRUST project, there is a close co-operation with the research groups at the TUD (Delft University of Technology), RIKZ (National Institute for Coastal and Marine Management), and Delft Hydraulics. Financial support is obtained from the CRAY

Figure 3. SO_2 concentrations over Europe, 5-day prediction.

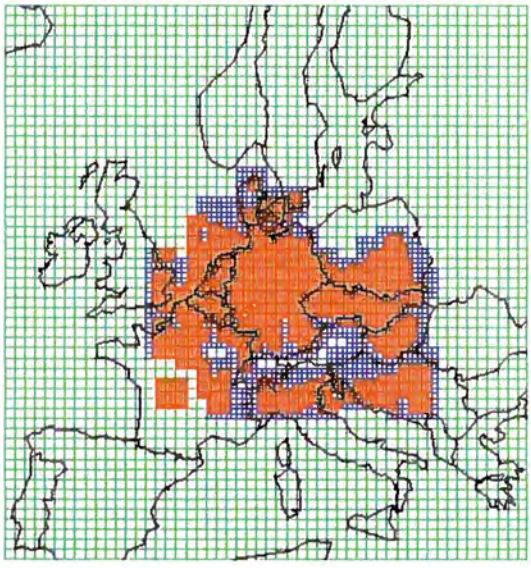


Figure 4. Horizontal grid at day 5 in calculation of Figure 2.

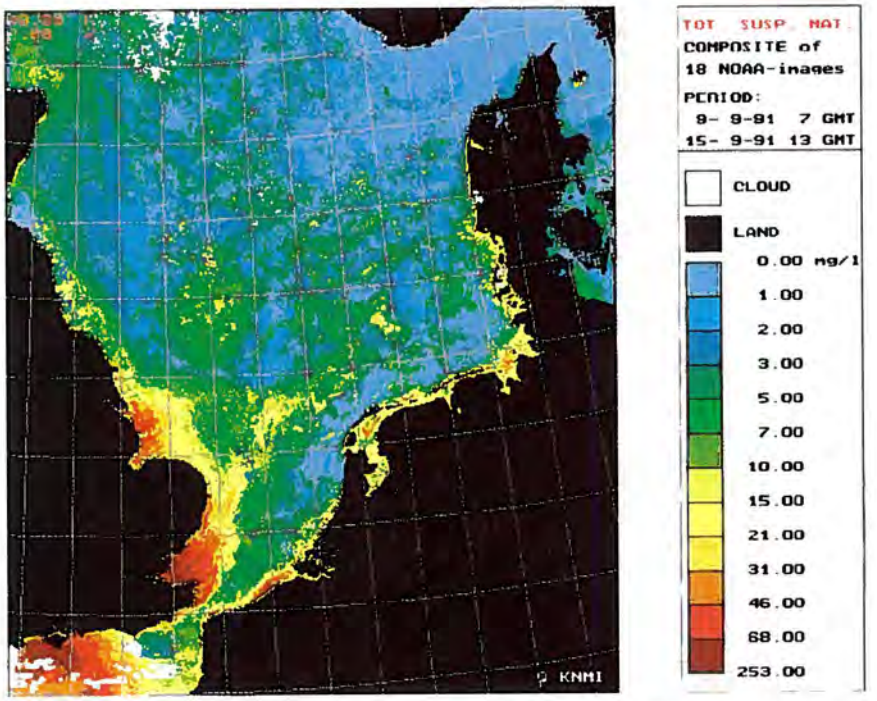


Figure 5. Remote sensing observations of the North sea area show that suspended fine-grained particles are predominantly concentrated in relatively narrow zones along the coastal boundaries.

Research Grant Program and the NOWESP project, which is part of the MAST II program of the EU.

3.3. CIRK

CIRK is a joint research programme of CWI, IMAU (Institute for Marine and Atmospheric Research) , RIVM (National Institute of Public Health and Environmental Protection) and KNMI (Royal Netherlands Meteorological Institute). The goal of the programme is to develop a model for global transport and chemistry of trace constituents in the troposphere, which should improve present models by using more advanced numerical methods and more refined parameterizations of physical and chemical processes. The development of such global models, which cover the whole troposphere and part of the stratosphere, is becoming increasingly important for atmospheric

scientists to study the long-term effects of regional pollution sources.

The research at CWI is financially supported by RIVM, and is directed to numerical problems associated with advection on spherical geometries, fast chemical solvers, computational aspects of operator splitting, and vectorization and parallelization aspects. This work is combined with the research at IMAU, RIVM and KNMI concerning, among other things, troposphere-stratosphere exchanges and sub-grid parameterizations for vertical transport and cloud systems.

Below some of the relevant references are given. More references for the projects EUSMOG and CIRK can be found at the WWW site 'http://www.cwi.nl/cwi/projects/nw1.4.html'.

REFERENCES

1. W. HUNDSDORFER, B. KOREN, M. VAN LOON, J.G. VERWER (1995). A positive finite-difference advection scheme. *Journal of Computational Physics 117*, 35-46.
2. W. HUNDSDORFER, E.J. SPEE (1994). *An Efficient Horizontal Advection Scheme for the Modelling of Global Transport of Constituents*, CWI Report NM-R9416, to appear in *Monthly Weather Review*.
3. B.P. SOMMEIJER, J. KOK (1995). *A Vector/Parallel Method for a Three-Dimensional Transport Model Coupled with Bio-Chemical Terms*, CWI Report NM-R9503.
4. J.G. VERWER (1994). Gauss-Seidel iteration for stiff ODEs from atmospheric chemistry. *SIAM J. Sci. Comput. 15*, 1243-1250.

Multigrid, Semi-Refinement and Fluid Flow

P.W. Hemker

1. Introduction

One of the continuous, major challenges in numerical analysis is the fast solution of partial differential equations (PDEs). Many different types of such equations exist, appearing in many areas of science and technology, e.g., when fluid flow problems have to be computed.

When PDEs are solved numerically, they have to be discretized, i.e. their solution, which is a set of functions defined over an area, is characterized by a set of N numbers, and the original differential equations are transformed into a system of N algebraic equations in which these N numbers are the unknowns. The difficulty is that, for an accurate approximation of the solution, the number N should be chosen large, which results in a large size of the system of algebraic equations.

For problems in which the solution is a function over a two-dimensional domain the size of the system can already be very large, if the solution is a function of three space variables, the size can be enormous.

To solve these large systems of equations, special techniques have been developed. Among these, the *multigrid method* is optimal in the sense that it is the only known approach by which the amount of computational work to solve the algebraic system is only proportional to N, the number of unknowns. For all other methods the amount of work grows faster than directly proportional with N.

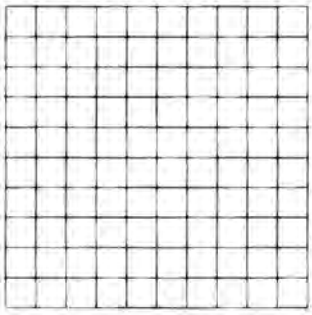


Figure 1. The domain Ω, discretized by a uniform mesh.

2. EXAMPLE

2.1. The problem

The basic model problem to demonstrate the value of multigrid has always been the Poisson equation on a square. This is a typical example for a general elliptic boundary value problem.

$$-\Delta u = f \quad \text{in} \quad \Omega = (0,1)^2\,; \quad u = 0 \quad \text{on} \quad \partial\Omega\,. \tag{2.1}$$

A uniform $n \times n$-mesh is placed over Ω, as shown in figure 1. This means that $n+1$ equidistant mesh-lines are drawn in the horizontal, and the same number in the vertical direction. The distance between the mesh-lines is called the mesh-width, $h = 1/n$.

The grid points are x_{ij}, where $0 \le i, j \le n$. If we want to approximate the solution of (2.1) numerically, discretisation is applied to (2.1) to get a set of linear equations:

$$AU = F\,, \tag{2.2}$$

where $F_{(i-1)(n+1)+j} = h^2 f(x_{ij})$, and A is a block tridiagonal matrix with a special structure. The system has $N = (n+1)^2$ equations and the same number of unknowns.

The element $U_{(i-1)(n+1)+j}$ of the solution vector U in the system of equations (2.2) represents the approximate solution of equation (2.1) at the point x_{ij}, i.e. $U_{(i-1)(n+1)+j} \approx u(x_{ij})$. The accuracy of this approximation depends on the type of discretisation and on h, the width of the mesh applied. This means that the approximation becomes more accurate if more mesh-lines are introduced. Typically, for a simple discretisation method, the error in the solution, $|U_{(i-1)(n+1)+j} - u(x_{ij})|$, is proportional to h^2. If higher accuracies are required, smaller values of h may be needed, i.e. a large number of

mesh-points may be necessary. Such large numbers of mesh-points give rise to very large systems (2.2), and the techniques used to solve such systems of moderate size (e.g., Gauss elimination) cannot be applied because the number of arithmetic operations (additions and multiplications) to compute the solution by these methods is proportional to N^3.

2.2. The solutions

For large systems of type (2.2), Gauss elimination would take too much time, even on present day's fastest computers, and different methods are used, that take advantage of the special properties of such equations. One such special property is, e.g., that although the system has $N = (n+1)^2$ unknowns, in each equation only a limited number of unknowns is involved, typically less than 10. This means that in the matrix A in (2.2) most entries are equal to zero, which reduces the amount of work considerably; but also other special properties of the matrix A can be used. All these special methods to solve discretized PDEs are iterative methods, where a first guess of the solution is improved step by step in an iteration process.

Until the sixties, simple relaxation methods were very popular. Here, all separate equations in (2.2) are scanned one by one, and each time when an equation is visited, the corresponding unknown is updated, based on the present information about the other unknowns. Such an approach was first mentioned by C.F. Gauss, in a letter to Chr.L. Gerling (26 December 1823), where he mentions: 'Das indirekte Verfahren lässt sich halb im Schlafe ausführen, oder man kann während desselben an andere Dinge denken'.

Later, in the seventies, more efficient iterative methods, based on the construction of Krylov spaces, appeared, such as the preconditioned conjugate gradient method, GMRES or—a more recent development—Bi-CG-Stab. Nowadays, these methods are the most popular ones to solve the very large systems. One reason is that these methods are relatively easy to implement in a computer program.

However, to restrict the amount of work to $\mathcal{O}(N)$, we have to resort to multigrid (MG) methods. These methods have a more complex structure. Invented in the sixties, they got the full attention of the numerical community not before 1980. A pioneering paper in the late seventies, [1], started the interest, and at present the multigrid method is well-accepted and in many fields it is successfully applied [3], see also figure 2.

3. MULTIGRID AND SEMI-COARSENING

The principle behind the MG technique is the fact that simple relaxation techniques only efficiently reduce the high-frequency errors on a mesh, and that the low-frequency errors can better be reduced by a discrete equation on a related coarser mesh with essentially less mesh points. (These errors may be compared to the disturbances of a quiet water surface, which in general

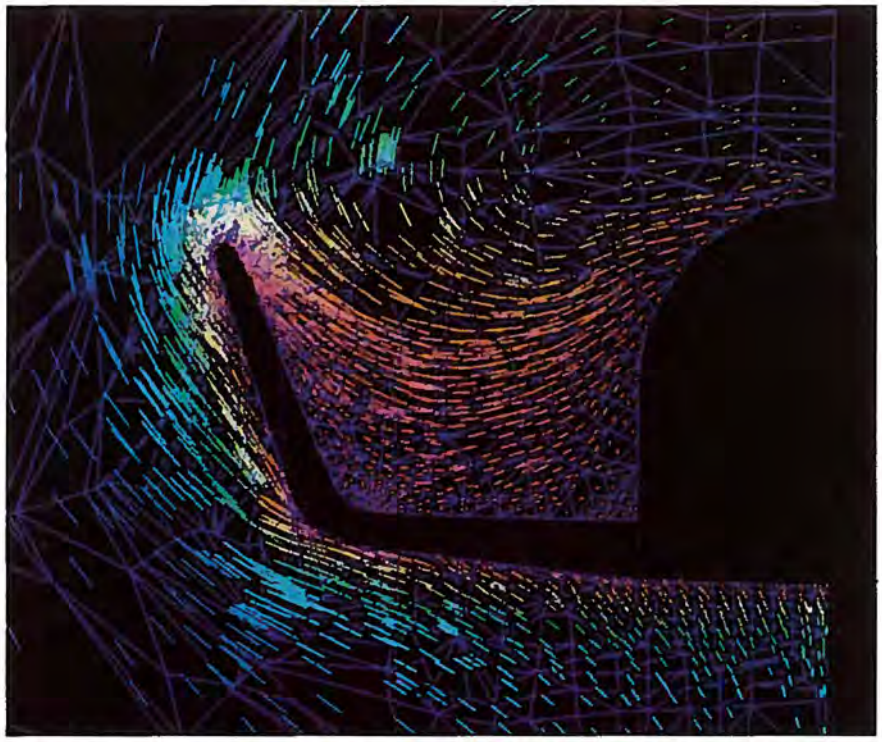


Figure 2. Streamline distribution around a half cross-section of the Hermes spaceplane. Applying multigrid methods, CWI studied numerical solutions of equations describing the flow around heavier-than-air craft as a part of the European space programme. Photo: Dassault.

consist of a spectrum ranging from short to long waves.) Now the MG method uses this principle recursively to solve the problem on the coarser meshes (see figure 3). All computational work together (on the coarse and the fine meshes) to solve the differential problem as accurate as is possible on the finest mesh (with N mesh points) is still $\mathcal{O}(N)$.

It is well known how multigrid methods can be used for two-dimensional (2D) problems, and that the same techniques can be used for three-dimensional (3D) problems as well. One may even point to the fact that the total amount of work on the coarse grids is relatively smaller in the 3D-case, than in the 2D-case. However, the reverse side is that only a relatively small amount of error components can be annihilated by these coarse grid corrections. Still today, the consequence is that in the 3D-case powerful relaxation methods are required to reduce the total error with a sufficient efficiency.

One possible relaxation procedure is alternating plane-relaxation, in which all planes in the cube are visited by different orderings, and where for each

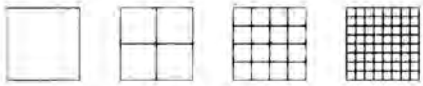


Figure 3. A classical sequence of grids in two dimensions.

plane a 2D sub-problem is solved (by a 2D MG method). This procedure is not very attractive, because there are many possibilities to order the planes in the cube, and a choice has to be made how these planes have to be visited. For a general problem such a choice is artificial, and the one choice may be better for the one problem while another choice can be advantageous in another situation. Such 3D-methods are also hard to vectorize or to parallelize so that we may have little advantage of new computer architectures.

However, there exists an alternative. Already for 2D fluid flow problems it had become clear that it is sometimes better to generate coarser grids, not by taking together a 2×2 set of four small cells to form one bigger cell, but to take together only 2 cells, so that a coarser mesh is obtained with a different mesh-size ratio. This is the principle of *semi-coarsening*. Here also we have the argument that the semi-coarsening is direction-dependent, and that there are more ways to assemble pairs of cells to form the coarse grids. But in the general, problem-independent case we may apply both semi-coarsenings at the same time. In that case the fine grid has two corresponding coarse grids. Now we have to study how the corrections from both coarse grids can cooperate to yield a good coarse grid correction for the solution on the finer grid.

We can approach the same technique from the other side. We may start with a coarse grid and make finer and finer grids, each time by halving grid cells into two finer cells (see figure 4). This principle of refinement can also be applied in three dimensions. In this case the number of possible grid refinements is even larger (see figure 5).

This approach of semi-refinement can be very powerful if it is combined with adaptive meshing, i.e., in all meshes only those cells are created that really contribute best to the reduction of the total error. Here the idea of hierarchical basis plays an important role, in order to combine the function approximations on the different grids into a single, unique representation.

4. Research at CWI

At CWI in recent years different applications have been studied, where MG was used to solve the discrete equations. Until 1994 all this work was concerned with problems in two space dimensions. Since 1984 fluid flow problems have been a major area of interest for MG investigations. First the

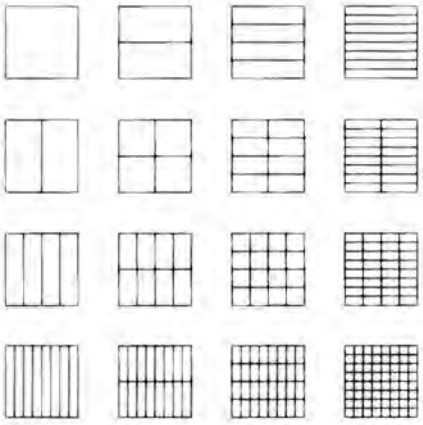


Figure 4. A family of semi-refined grids in two dimensions.

steady state Euler equations for inviscid compressible flow were considered. The equations differ essentially from (2.1) because the Euler equations are not elliptic, but – as time-dependent equations – hyperbolic. The approach to solve these is: to base the MG procedure on a finite-volume upwind discretisation. Later the same approach was extended for the compressible Navier-Stokes equations. In the steady case, these equations are elliptic, but for high Reynolds numbers (the interesting case in aerodynamical applications) they are singularly perturbed. This implies that thin layers may appear in the solution and that outside such layers the solution behaves very much like the Euler equations.

Steered by a contract with the European Space Agency, from 1988 to 1991, research was done for the special case of hypersonic flows. For a more recent contract, with a scientific board of the European Union, we studied structured adaptive methods for compressible flows. In these adaptive methods the refinement of the grids depends (automatically) on the features of the solution that is computed. Some examples will be given below. For more details we refer to [2].

At this moment the research is directed to the use of MG methods for 3D problems. A data structure has been developed to allow a well-structured coding of the self-adaptive semi-refinement algorithms. Procedures for self-adaptive representation of 3D-functions are now available, and the first experiments with 3D fluid flow problems have been carried out. In the mean time theoretical developments are continuing. Lately, Fourier analysis was used to compute the convergence rate of semi-coarsened MG algorithms. The computed rates were confirmed by numerical experiments.

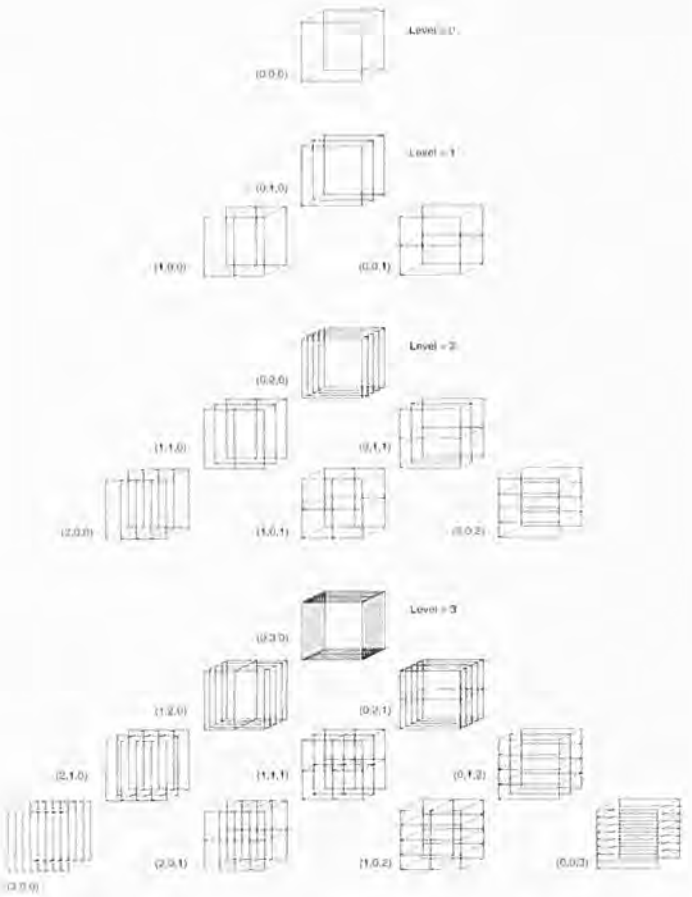


Figure 5. Semi-refinement of the cube. Grids on levels 0, 1, 2, and 3.

5. Fluid flow problems

If the MG method is used, starting from a coarse mesh and constructing finer and finer meshes, this procedure can be continued until the approximate solution is sufficiently accurate. If the solution is not very regular, some parts of the domain will need more refinement than other parts. How far the refinement should be continued depends on the solution itself, and can be decided during the computation. In this way self-adaptive algorithms are developed, that minimize the number of gridpoints required, for a given precision. At CWI such algorithms have been studied in detail by H.T.M. van der Maarel. Algorithms have been developed for the Euler equations for compressible, inviscid flow, and also for the compressible

Navier-Stokes equations. An example of a solution of the Euler equations on a self-adaptive grid is given in figure 6, where the adapted grid and isolines for the Mach number distribution are shown for the flow over a NACA0012 airfoil.

Figure 6. Euler solution for flow over an airfoil: the regular adaptive grid (left), and Mach number distribution (right).

The computation with the Navier-Stokes equations can be seen as an extension with diffusive terms of the same procedure for the Euler equations. Apart from the additional diffusive flux computations and boundary conditions, all techniques for the Navier-Stokes equations are the same as for the Euler flow computations. The convective part of the numerical flux is the same. For Navier-Stokes computations the flux is extended with a diffusive part, involving shear stresses and temperature gradients. The convective flux may be computed with either $\mathcal{O}(h)$ or $\mathcal{O}(h^2)$ accuracy. The diffusive part is always computed with $\mathcal{O}(h^2)$ accuracy.

Because the difference scheme for the diffusive part of the equations extends over a different set of grid cells compared to the convective part, much attention has to be paid to the different possible grid structures encountered in an adaptively refined Navier-Stokes grid.

The multigrid smoother approximately inverts the first-order accurate convective operator, extended with the second-order accurate diffusive terms. As with the Euler equations, the equations resulting from the second-order accurate discretisation are solved by defect correction.

An example is the computation of a flow along an adiabatic flat plate (see figure 7). The plate is located at $y = 0$, $-0.5 < x < 0.5$, and the Reynolds number (Re) equals 100. The result shown has been obtained with the first-order accurate discretisation and a refinement criterion based on the the cross-flow gradient of the velocity component times the mesh-width (the

Figure 7. Navier-Stokes solution over a flat plate, Re=100, regular adaptive grid (above), and velocity in the x-direction (below).

undivided difference). Figure 7 shows the resulting grid when refinements are introduced where the undivided difference is larger than 0.1 ρ (above), and an iso-line plot of the computed velocity in the x-direction (below).

References

1. A. Brandt (1977). Multi-level adaptive solutions to boundary value problems. *Math. Comput. 31*, 333-390.
2. P.W. Hemker, B. Koren, W.M. Lioen, M. Nool, H.T.M. van der Maarel (1995). Review of an accurate and efficient multigrid method for steady gas dynamics problems. *Computational Fluid Dynamics Review 1.*
3. P.W. Hemker, P. Wesseling (eds.) (1993). *Multigrid methods IV*, volume 116 of *International Series of Numerical Mathematics*, Basel. 1994. Birkhäuser Verlag. Proceedings of the Fourth European Multigrid Conference, held in Amsterdam.

Computational Number Theory

H.J.J. te Riele

1. Introduction

Natural numbers — the numbers by which we count — have always struck the imagination. Problems involving natural numbers often are simple to state, but their solution may require a lot of mathematical creativity and ingenuity. A classical example is the problem of perfect numbers which was studied already by the ancient Greeks: nowadays, it has become a standard problem for testing the accuracy and reliability of new computers and software. (See also figure 1.) Daily life, even in antiquity, is unthinkable without counting, so the scientific discipline now known as *Number Theory* finds its roots in practical problems of every day.

In number theory, one studies the properties of natural and rational numbers and the solution of equations by such numbers. Some typical questions are: what are the divisors of a given number and how many are there? How many prime numbers (i.e., numbers > 1 only divisible by 1 and themselves) are there below a given bound x? Is there an $n > 2$ for which the equation $x^n + y^n = z^n$ has a solution in rational numbers?

Many problems can be solved in a step-by-step way, i.e., with the help of an *algorithm*. Loosely speaking, an algorithm is a set of arithmetic rules which yields, when applied to a prescribed input, a definite output in a finite number of computational steps. The invention of mechanical and electronic computers meant a huge step forward for the study of algorithms. These machines, if programmed correctly, do not make mistakes nor lose concen-

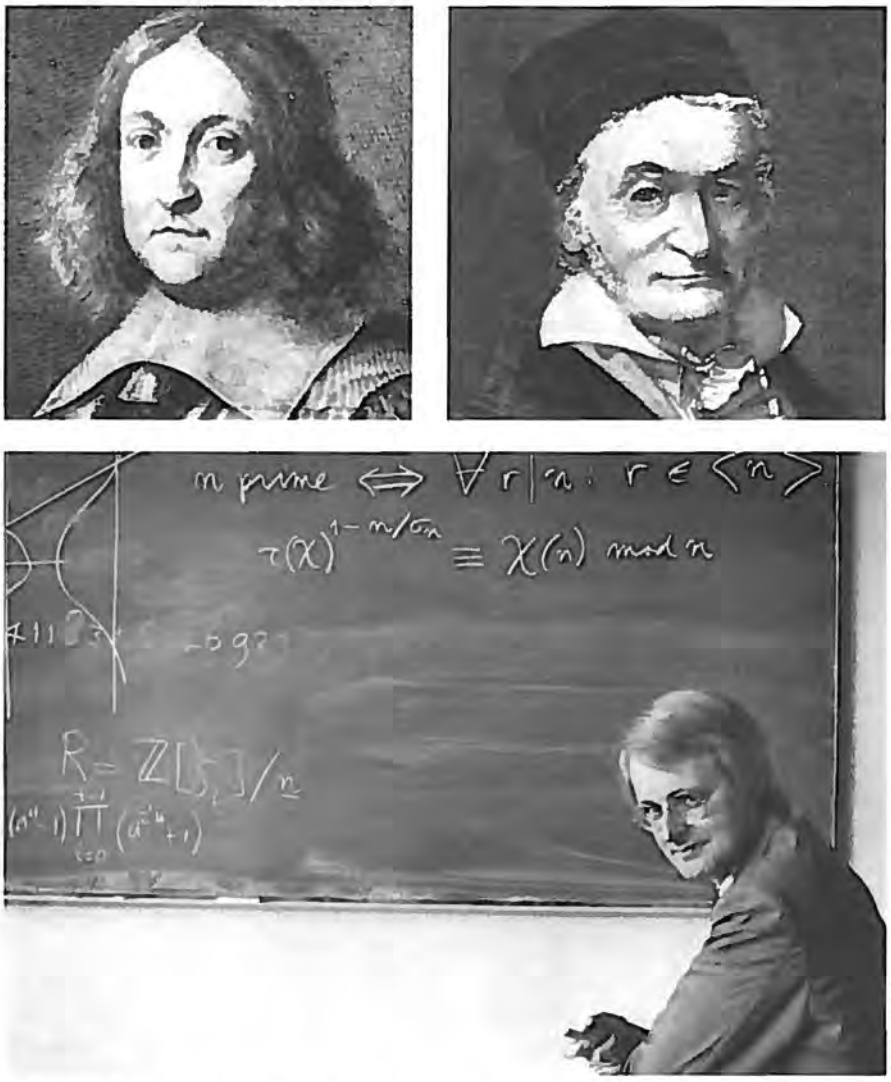


Figure 1. Factorization of numbers has fascinated mathematicians from ancient Greece through to the present day. Three prominent representatives are: Pierre de Fermat (1601-1665, above left), C.F. Gauss (1777-1855, above right), and Hendrik W. Lenstra, Jr. (b. 1949, below).

tration, so they are perfectly suited to serve as modern slaves for the tedious and time-consuming work of the computational number-theorist. In addition, many known ingenious algorithms for answering number-theoretic

questions would not have been invented *without* computers. For example, modern algorithms for finding the prime factors of *large* numbers, are very inefficient for *small* numbers, and it is even practically impossible to apply them without the help of fast electronic computers. The advent of vector computers in the eighties, and parallel computers in the nineties has stimulated the study of algorithms for such architectures. Several number-theoretic problems, like those where one wants to find numbers with a special property, are well-suited for treatment with the help of such vector and/or parallel algorithms.

These developments have given rise to the birth of 'Computational (or Algorithmic) Number Theory'. Here, the computer is a tool for experimentation and for testing hypotheses, and it is a stimulus for the development of ever more efficient algorithms, which can lead to new insight and new mathematical results.

2. Global developments in computational number theory

Before the advent of fast electronic computers, *tables* were an important aid to number-theorists. Nowadays, it is much more efficient to save a *computer program* or *implemented algorithm* and to quickly generate the tables or individual table entries each time they are needed. Collections of such (sub)programs are available now in several *computer algebra packages* like PARI, MAPLE and MATHEMATICA. They enable the researcher to perform arithmetic calculations (in arbitrary precision) on mathematical objects such as numbers, vectors, matrices, algebraic numbers and finite fields, and to perform *symbolic* computation like integration, differentiation and formal series expansion.

The design, implementation, and analysis of efficient algorithms for solving number-theoretic problems has been the main activity of researchers in computational number theory. As an illustration, we will briefly describe here three major algorithmic developments, namely, in factoring, primality testing and lattice basis reduction. A survey of modern factoring methods can be found in [2]. An excellent historical survey of the computational history of factoring and primality testing from 1750 to about 1950, i.e., *before the era of electronic computers* is presented in [5]. Old and modern primality tests are treated in [1]; this book also treats algorithms for lattice basis reduction. An excellent textbook on algorithmic *algebraic* number theory is [3].

2.1. Factoring

An important stimulus for the study of *factoring* algorithms was the discovery by R.L. Rivest, A. Shamir and L. Adleman, in 1978, of a public-key encryption scheme, now known as RSA. It is based on the (presumed) difficulty of decomposing a given large number into prime factors. For the cur-

rently best known factoring algorithms practical experience suggests that factoring indeed is a hard problem, although nothing has been proved so far.

The best known factoring algorithms try to find integers x and y with the property that

$$x^2 \equiv y^2 \bmod n, \tag{2.1}$$

where n is the number to be factored (and known to be composite). For such x and y, the number $d = \gcd(x - y, n)$ is easily computed by a well-known algorithm of Euclid, and d is a proper divisor of n in at least half the number of cases for which (2.1) holds. So if we have no success, we try to find another pair x, y. To find a congruence of the form (2.1), one tries to collect *many* congruences of the form $x_i^2 \equiv a_i \bmod n$, where the a_i only have prime factors in a given set $\mathcal{F}$, which is called the *factor base*. If we have succeeded to find more such congruences (also called *relations*) than there are different primes in the factor base, we can combine them with the help of linear algebra techniques (Gaussian elimination or iterative methods), to find a congruence of the form (2.1). There are several methods to find the above relations. One is based on the computation of the continued fraction of $\sqrt{n}$ and another is based on efficient sieving techniques for finding values of quadratic polynomials which only consist of prime factors in the set $\mathcal{F}$.

Two important algorithmic discoveries have effectuated a jump in the size of the numbers which can be factored within a reasonable time on a modern computer: the quadratic sieve method (QS) published in its modern form in 1985 by C. Pomerance (but with main ideas going back to M. Kraitchik in 1926), and the elliptic curve method (ECM) published in 1987 by H.W. Lenstra, Jr. ECM is suitable to find factors up to 35–40 decimal digits of large numbers. Its complexity, as conjectured theoretically, and as observed in the experiments, depends primarily on the *size* of the smallest prime factor p of the number n which we wish to factor. The complexity of the quadratic sieve method depends on the size of n, and not on its prime factors. It is still the method by which the largest numbers (not of a special form like $a^n \pm b$ where a and b are small compared to $a^n \pm b$) have been factored. The present world record is the so-called RSA–129 number, a number of 129 decimal digits. In 1977 Rivest et al. challenged the public to factor this number. They estimated that the required running time, using the best algorithms and machines available in 1977, would be 40 quadrillion ($= 10^{15}$) years. It was factored only seventeen years later, in April 1994, with a variation of the quadratic sieve method after an eight-month worldwide computing effort organized by D. Atkins, M. Graff, A.K. Lenstra, and P. Leyland. Also CWI has contributed idle workstation cycles to this result. RSA-129 turned out to be the product of two primes, one of 64 and one of 65 digits, and it is a typical example of a key used in the RSA

public-key encryption scheme. ECM and QS nicely complement each other: one usually tries ECM first in order to find factors less than 25–30 decimal digits. If one is lucky, larger factors are sometimes found: the world record is a prime factor of 42 decimal digits. In the next step QS is tried, provided that the number to be factored is small enough: popularly spoken, ECM finds smaller factors of larger numbers, QS finds larger factors of smaller numbers.

A third method, called the Number Field Sieve (NFS) and published in 1993 by J.M. Pollard, and in refined form by J.P. Buhler, H.W. Lenstra, Jr., and C. Pomerance, is expected to be more efficient for general numbers than the quadratic sieve, and it is the subject of intensive current research to find out where the cross-over point between NFS and QS lies. (See also figure 2.)

The size of the numbers which could just be factored at a given time with the available algorithms and computer technology was about 25 decimal digits in 1967, 40–50 in 1974, 70–80 in 1987, 100 in 1990, and 120–130 at present. This illustrates the rapid developments, both in algorithms and in hardware, if we realize that for the best known factoring methods the computational effort roughly *doubles* if the number to be factored grows with 2–3 decimal digits.

2.2. *Primality testing*

Before we are going to try to factor a number n, how do we know that n indeed is composite? Tests for compositeness are based on the 'Little Theorem' of Fermat which states that if p is a prime number, and a is a positive integer such that $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \bmod p$. So if we find for some b with $\gcd(b, n) = 1$ that $b^{n-1} \not\equiv 1 \bmod n$, then n cannot be prime, and we can attempt to factor n. If the test yields $\equiv 1 \bmod n$, we can not be sure that n is prime since the *converse* of Fermat's Little Theorem does not hold. However, in most such cases n indeed is prime and exceptions are very rare. The simplest way to rigorously prove primality of n is to show that it has no divisors $\leq \sqrt{n}$. For small n this method works on a modern PC or workstation, but for larger n (consisting of more than 15 decimal digits, say) the number of operations becomes too large. Until 1980, the available primality tests were based on the knowledge of the prime factors of $n - 1$ or $n + 1$ and became impractical for numbers of more than 100 decimal digits. A breakthrough came when Adleman, Pomerance, and R. Rumely found a test that was efficient for much larger numbers. This was simplified and improved by H. Cohen, and H.W. Lenstra, Jr. An efficient implementation was written by H. Cohen and A.K. Lenstra, with the help of D.T. Winter at CWI. With this program, it was possible in 1986 to prove primality of numbers up to 300 decimal digits in a few minutes CPU-time. At present, one is able to prove primality of general numbers with more than

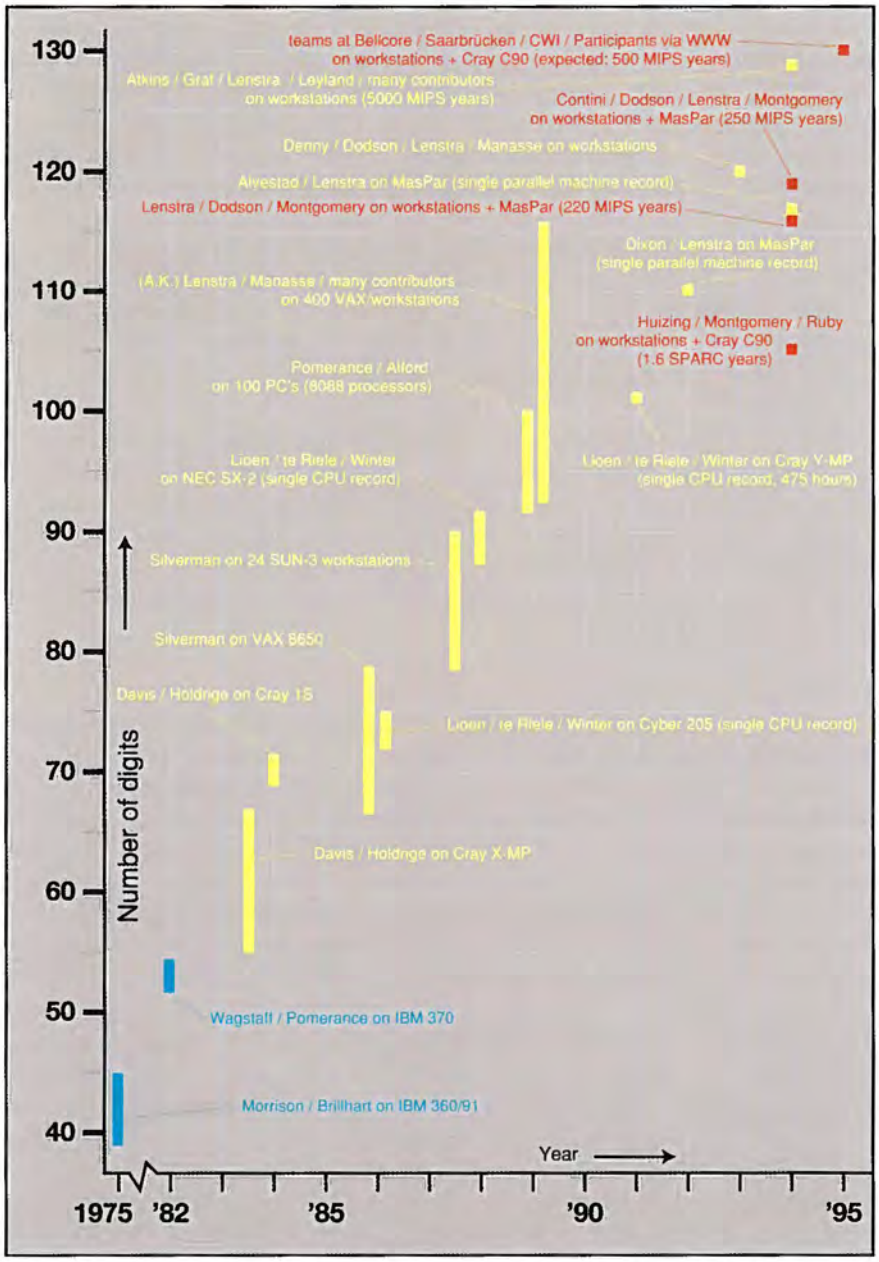


Figure 2. History of factoring records obtained with general-purpose factoring methods: Continued Fraction (blue), Quadratic Sieve (yellow), and General Number Field Sieve (red).

1000 decimal digits, thanks to algorithmic and implementational results of A. Atkin, F. Morain, W. Bosma, and M.-P. van der Hulst. For numbers of a special form, like the *Mersenne numbers* $2^p - 1$, special primality tests are known. In fact, the largest known prime number is the Mersenne prime $2^{859433} - 1$, a number of 258,716 decimal digits. It was discovered by D. Slowinski and P. Gage in 1994 with the help of the so-called Lucas-Lehmer test, which reads as follows: define the sequence $\{u_i\}$ by: $u_0 = 4$ and $u_{i+1} = u_i^2 - 2$ $(i = 0, 1, \ldots)$; then $n = 2^p - 1$ is a prime if and only if p is a prime > 2, and if n divides u_{p-2}.

2.3. Lattice basis reduction

The third problem we mention here is that of finding *small vectors in lattices*. In 1982, A.K. Lenstra, H.W. Lenstra, Jr., and L. Lovász published their so-called 'lattice basis reduction' algorithm. It computes from an arbitrary basis of a lattice in $\mathcal{R}^m$ a so-called reduced basis which has certain nice properties (its vectors are nearly orthogonal). The algorithm has many important applications in a variety of mathematical fields, like the factorization of polynomials, public-key cryptography, extracting the square-root of extremely large algebraic numbers (one crucial step in the Number Field Sieve factoring algorithm), and the disproof of the Mertens conjecture (discussed in the next section). For some applications, this algorithm in fact is a very efficient multi-dimensional continued fraction algorithm by which one is able to find simultaneous approximations of vectors of real numbers by vectors of rational numbers with the same denominator. This problem occurs frequently in number theory.

3. Computational number theory at CWI

A recent survey of the research in computational number theory at CWI in the past 25 years is presented in [4]. We restrict ourselves here to giving a concise description of the results obtained with respect to the Riemann hypothesis, the Mertens conjecture, and the problem of factoring large numbers. The computational number theory group at CWI presently consists of H.J.J. te Riele (project leader), W.M. Lioen and D.T. Winter (scientific programmers), and H. Boender and R.-M. Huizing (junior researchers at CWI and Leiden University). In 1993–1994, P.L. Montgomery was a visiting researcher in the group. J. van de Lune (senior researcher, retired in 1993) was the initiator of the computational work on the Riemann hypothesis, and of other projects of the group like the work on the Goldbach conjecture. Close cooperation exists with the number theory group of R. Tijdeman in Leiden.

3.1. The Riemann hypothesis

Consider the function $\zeta(s) = \sum_{n=1}^{\infty} n^{-s}$, where $s = \sigma + it$ is a complex variable. If $\sigma > 1$, then the series converges, so that $\zeta(s)$ is properly defined there. By using a technique now known as analytic continuation, Riemann showed in 1859 that there is a unique function which coincides with $\zeta(s)$ for $\sigma > 1$, and which is analytic in the *whole* complex plane, except at the point $s = 1$ (where the function has a pole of order 1). This function is known as the *Riemann zeta function*, and it plays a prominent role in prime number theory. It is known to have infinitely many complex zeros in the so-called *critical strip* $0 \leq \sigma \leq 1$, and in an eight-page paper which appeared in 1859, Riemann wrote that it is very likely that all these zeros lie *on* the line $\sigma = \frac{1}{2}$. So far, nobody has been able to (dis)prove this assertion, which is known now as the *Riemann hypothesis*.

What is the relation between the Riemann hypothesis and prime number theory? Let $\pi(x)$ denote the number of primes $\leq x$. As early as in 1792 or 1793, C.F. Gauss conjectured that the density of the prime numbers close to x is approximately equal to $1/\log x$, and that the so-called logarithmic integral

$$\mathrm{li}(x) = \int_2^x \frac{dt}{\log t} \tag{3.2}$$

is a good approximation of the function $\pi(x)$. Extensive numerical computations by A.M. Odlyzko suggest that the error in this approximation is proportional to $\sqrt{x}$: for $x = 10^{12}, 10^{14}, 10^{16}, 10^{17}, 10^{18}$ we have

$$(\pi(x) - \mathrm{li}(x))/\sqrt{x} = -0.038, \ -0.031, \ -0.032, \ -0.025, \ -0.022,$$

respectively. The truth of the Riemann hypothesis implies that

$$\pi(x) = \mathrm{li}(x) + \mathcal{O}(x^{1/2} \log x) \quad \text{as} \quad x \to \infty.$$

What is known about the location of the complex zeros of $\zeta(s)$? Massive numerical computations carried out by Van de Lune, Te Riele, and Winter at CWI in 1983–1984 on a CDC Cyber 750 computer, and on a CDC Cyber 205 (one of the first vector computers in The Netherlands), have *proved* that the first 1.5×10^9 complex zeros of $\zeta(s)$ are *all simple* and lie on the line $\sigma = \frac{1}{2}$. The amount of CPU-time used was about 1000 (low-priority) hours on both machines. This extended similar computational work by R.P. Brent at the Australian National University in Canberra for the first 156,800,001 complex zeros. Extensive computations by Odlyzko have shown later that the Riemann hypothesis holds for long sequences of consecutive zeros with rank in the neighbourhood of 10^{18}, 10^{19}, and 10^{20}. Table 1 gives the successive published records in proving that the first n complex zeros of the Riemann zeta function satisfy the Riemann hypothesis.

Investigator	n
Gram (1903)	10
Backlund (1914)	79
Hutchinson (1925)	138
Titchmarsh (1936)	1,041
Turing (1953)	1,104
Lehmer (1956)	25,000
Meller (1958)	35,337
Lehman (1966)	250,000
Rosser, Yohe, Schoenfeld (1969)	3,500,000
Brent (1979)	81,000,001
Brent, Van de Lune, Te Riele, Winter (1982)	200,000,001
Van de Lune, Te Riele, Winter (1986)	1,500,000,001

Table 1. Numerical verification of the Riemann hypothesis for the first n complex zeros.

3.2. The Mertens conjecture

The *Mertens conjecture* is a statement about the so-called Möbius function

$$\mu(n) := \begin{cases} 1, & n = 1, \\ 0, & \text{if } n \text{ is divisible by the square of a prime number,} \\ (-1)^k, & \text{if } n \text{ is the product of } k \text{ distinct primes.} \end{cases}$$

Based on numerical data concerning the function

$$M(x) = \sum_{1 \le n \le x} \mu(n),$$

F. Mertens stated in 1897 that the inequality

$$|M(x)| < \sqrt{x}, \quad x > 1,$$

is 'very probable'. This is now known as the *Mertens conjecture*.

The *size* of $M(x)$ is closely related to the location of the complex zeros of the Riemann zeta function. In fact, it is not too difficult to show that the *boundedness* of $M(x)/\sqrt{x}$ implies the truth of the Riemann hypothesis. For all values of $M(x)$ which have been computed explicitly, the Mertens conjecture is true. In 1994, Lioen and Van de Lune at CWI established that $-0.513 < M(x)/\sqrt{x} < 0.571$ for $200 < x \le 1.78 \times 10^{13}$. Their computations consumed about 400 CPU-hours on a Cray C98 super vector computer. Nevertheless, serious doubts concerning the truth of the Mertens conjecture were raised already in 1942 by A.E. Ingham, who showed that it is possible to prove the existence of certain large values of $|M(x)|/\sqrt{x}$ *without the need to explicitly compute* $M(x)$. In order to find such large values, one has to solve a so-called simultaneous inhomogeneous Diophantine approximation

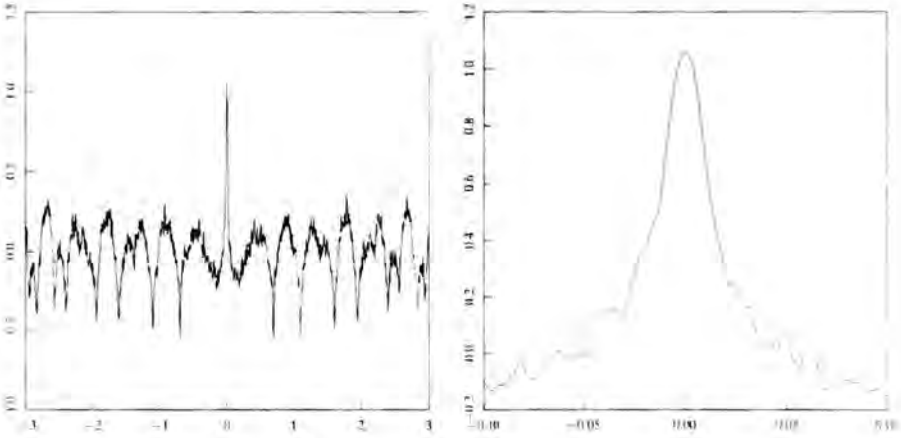


Figure 3. Graph of the function $h(y_0 + t)$ for $t \in [-3, +3]$ (left), with enlargement of its central part (right).

problem. Using the algorithm of Lenstra, Lenstra, and Lovász, mentioned in the previous section, Odlyzko and Te Riele in 1985 found a disproof of the Mertens conjecture. This required to find a value $y = y_0$ for which a certain function $h(y)$ (which we shall not give explicitly here) assumes a value > 1. The value found (see also figure 3) was:

$$y_0 = 1.4594\,5289\,6805\,9299\,8046\,7903\,6163\,0399\,7841\,2740\,0591\,9997\,8973\,8039\,9659\,6076\,2.521505.$$

Unfortunately, this disproof is ineffective: only the *existence* of an x where $|M(x)|/\sqrt{x} > 1$ was proved. In 1987 however, J. Pintz gave an explicit (huge) upper bound, proving that $|M(x)|/\sqrt{x} > 1$ for $x \leq \exp(3.21 \times 10^{64})$. For the computation of this upper bound, Pintz used 100-digit accurate values of the first 2000 complex zeros of the Riemann zeta function, and 28-digit accurate values of the next 12950 complex zeros, as computed earlier by Te Riele for the *ineffective* disproof.

3.3. Factoring large numbers

At CWI much time and effort has been spent on the efficient implementation of the quadratic sieve method on large vector mainframes like the CDC Cyber 205, the NEC SX-2, and the Cray Y-MP and Cray C98 vector computers.

In the course of years, various new factorization records have been established by the CWI Computational Number Theory group. These, and

many other factored numbers were contributions to the so-called Cunningham Table (a table of known factors of numbers of the form $a^n \pm 1$, initiated in 1925 by A.J.C. Cunningham and H.J. Woodall) and to an extension of this table.

In Table 2 we give some figures about record factorizations found at CWI on vector computers. All results were obtained on *one* processor of the vector computer listed. On the Cray Y-MP we *could* have used four CPUs, thus reducing the sieving time by a factor of about four, since the most time-consuming steps of the quadratic sieve algorithm are almost perfectly parallelizable.

year	machine	size of numbers (decimals)	sieving time (hours)	Gaussian elim. time (seconds)	approximate order of sparse system
1986	Cyber 205	72	4.3	21	6,070
		75	12.2	37	7,400
1988	NEC SX-2	87	30	200	18,800
		92	95	700	24,300
1991	Cray Y-MP	101	475	1800	50,200

Table 2. Record factorizations with QS on vector (super)computers.

The latest records were obtained in the summer of 1994 with the help of the Cray C98 at SARA (The Academic Computing Centre Amsterdam), and many workstations in a collaboration between Oregon State University and CWI: a 162-digit Cunningham number was factored with the 'Special Number Field Sieve' (SNFS, for which the number n to be factored has the form $n = a^m \pm b$, a and b being small compared to n), and a 105-digit number was factored with the 'General Number Field Sieve' (GNFS, for which no special form of n is known). One month after the latter result was obtained, A.K. Lenstra, B. Dodson, and Montgomery cracked a 116-digit partition number with GNFS. On November 26, 1994 S. Contini, Dodson, A.K. Lenstra, and Montgomery completed the factorization of a 119-digit cofactor of the 123-digit partition number $p(13171)$ into two primes of 52 and 67 digits using GNFS. From the time they used (about 250 mips years) they estimate that this is about 2.5 times less than what they would need to factor a number of comparable size with the quadratic sieve method.

Montgomery and Huizing factored several other numbers with SNFS (of 98, 99, 106, 119, 123, 135, and 137 decimal digits) including some *more* and *most* wanted Cunningham numbers (i.e. difficult numbers in the Cun-

ningham table, not yet factored) using a new algorithm of Montgomery for computing the square root of the product of many algebraic numbers, and his new iterative block Lanczos algorithm for finding dependencies in large sparse matrices over GF(2). Huizing also factored 87-, 97-, and 107-digit numbers with GNFS.

Currently, most factorization research at CWI aims at contributing to the Cunningham tables. In the first update to the extended table, issued in September 1994, all the composite numbers with less than 86 decimal digits were completed. This bound has been raised in May 1995 to 90 decimal digits.

References

1. H. Cohen (1993). *A Course in Computational Algebraic Number Theory*, Volume 138 of *Graduate Texts in Mathematics*, Springer–Verlag, Berlin.
2. P.L. Montgomery (1994). A survey of modern integer factorization algorithms. *CWI Quarterly, 7(4)*.
3. M. Pohst, H. Zassenhaus (1989). Algorithmic algebraic number theory. *Encyclopedia of Mathematics and Applications*, Cambridge University Press, Cambridge.
4. H.J.J. te Riele, J. van de Lune (1994). Computational number theory at CWI in 1970 – 1994. *CWI Quarterly, 7(4)*.
5. H.C. Williams, J.O. Shallit (1994). Factoring integers before computers. W. Gautschi (ed.). *Mathematics of Computation 1943–1993: a Half-Century of Computational Mathematics*, 481-531. Proceedings of Symposia in Applied Mathematics, American Mathematical Society.

Semantics

J.W. de Bakker

1. History

Programs are written in a programming language, and serve as a means to instruct a computer to perform a given task. As linguistic entities, programs have form and meaning. In the specification of their form, one employs syntactic rules, usually in the form of some grammatical formalism. In semantics, one is concerned with defining meanings of programs in terms of a mathematical model. For $\mathcal{L}$ a programming language (the reader may think of PASCAL, ML or PROLOG as typical examples), one looks for a set $\mathcal{P}$ of mathematical objects and a meaning function $\mathcal{M} : \mathcal{L} \rightarrow \mathcal{P}$ (*) such that, for each $s \in \mathcal{L}$ (each program), one determines its meaning $\mathcal{M}(s) = p$ as object in $\mathcal{P}$. One advantage of having such a meaning function is that we get a notion of equivalence of programs: two programs s_1, s_2 are equivalent if they have the same meaning, i.e. $\mathcal{M}(s_1) = \mathcal{M}(s_2)$. For example, two procedures may be seen as equivalent if they compute the same function (even though they may be 'programmed' in different ways). In general, the design of semantic models is a rather demanding endeavour. Programming languages are complex entities, and so are the computations specified by the programs of the language. Accordingly, ever since the advent of high-level programming languages (say from 1960 onwards), a rich body of methods and tools has been developed to be used for this purpose.

In the period 1960 to 1970, the emphasis was on the use of general techniques from the theory of computability for the formal definition of (syn-

tax and) semantics of programming languages, e.g., based on (generalized) Markov algorithms, or Van Wijngaarden's two level grammars. Indeed, thanks to the universality of these definitional systems, it was not surprising that complete formal definitions could be given. What was lacking in these definitions was sufficient abstraction from representational (and often arbitrary) detail. Sheer symbol manipulation was often the prevalent approach.

1.1. Denotational semantics

Owing to the pioneering work of D.S. Scott around 1970, the study of semantics returned to the treasured principles of mathematical logic, viz. (i) definitions should be *compositional* (a classic principle due to G. Frege) and (ii) definitions should clearly separate the linguistic realm from the mathematical structure(s) (the domain(s) of interpretation) to which the linguistic constructs are mapped (the $\mathcal{P}$ from definition (*) above). These principles are fundamental for the style of so-called denotational semantics, which has remained one of the major methodologies in semantics till the present day. An even more seminal contribution of Scott was the design of semantic models for the lambda calculus—and many more related languages—couched in the framework of a general theory of (lattice-theoretic) domain equations. Jointly with his coworker, the late C. Strachey, Scott laid the foundations for the semantic analysis of a host of—mostly sequential—programming languages.

1.2. Structural operational semantics

Subsequently, extensions of the general theory were proposed by G.D. Plotkin, especially to cover as well the notions of nondeterminacy and parallelism. Around 1980, two further innovative developments took place. Firstly, the notion of so-called *structural operational semantics* (SOS) was introduced by Plotkin. Its origin can be traced back to automata theory: a transition system $(S, A, \rightarrow)$ consists of a set of *states* S, a set of *actions* A, and a transition relation $\rightarrow \subseteq S \times A \times S$. In automata theory, one would write $\delta(s, a) = s'$, in the SOS-oriented semantics the same fact is written as $s \xrightarrow{a} s'$ (†). Plotkin's idea was to instantiate the abstract set of states S to a concrete set, viz. the set $\mathcal{L}$ of statements in a programming language, and to read (†) as: statement s performs an a-step and then turns into the statement s'—which may, in turn, make a b-step, etc. The formalism of transitions such as (†) turned out to be especially fruitful in the study of *concurrency*, initiated around 1980 in the work of R. Milner on CCS (*A Calculus for Communicating Systems*) and C.A.R. Hoare on CSP (*Communicating Sequential Processes*).

1.3. Algebraic semantics

We next discuss two related areas which have been of prime importance in the history of semantics. Firstly, so-called *algebraic semantics* has gained a central status—as third methodology—alongside the methods of denotational and operational semantics. Here, the theory is built on the foundations of universal algebra (such as the notions of initial and final algebra) and equational logic. Algebraic semantics has turned out to be quite valuable, in particular for a logical underpinning of abstract data types (abstract versions of the data structures of programming). In addition, there are deep connections with the theory of rewriting, the theory of concurrency, and with (the vast variety of) specification formalisms.

1.4. Program logics

A second area of research neighbouring on that of semantics is the theory of program correctness, verification and transformation. Historically, this work dates back to Floyd's inductive assertion method (1967), Dijkstra's structured programming and weakest preconditions (early seventies), and Hoare's axiomatic method for simple sequential languages (1969). Though partly more of a logical/syntactic flavour, this area exploits semantic modelling in the investigation of the soundness of formal systems to prove program correctness or to deduce program transformations. Also, the steps prescribed in refining a program from an abstract specification to an executable—and hopefully efficient—implementation require semantic justification. So much for the history of semantics. Some evidence for the world-wide recognition of the developments sketched above may be inferred from the fact that five of the pioneers named above (E.W. Dijkstra, Hoare, R.W. Floyd, Scott, Milner) are recipients of the Turing award of the American Association for Computing Machinery (the Turing award being the Nobel prize of the computer science profession).

2. Current developments

All four areas listed above—denotational semantics, operational semantics, algebraic semantics, program logics—are topics of vigorous current activity. During the last two decades a mathematical discipline called 'category theory' has become increasingly important in semantical investigations in computer science. (This also holds for other areas, like specification or type theory.) Category theory provides an elementary foundational language in which the basic concepts of mathematics can be expressed, not in terms of membership like in set theory, but in terms of 'arrows' (or 'morphisms') between 'objects'. The basic idea is to describe mathematical entities not as what they are made of, but as how they behave. For example, set-theoretically a product consists of a set of pairs, whereas category-theoretically a product is an object with two projection arrows which be-

have in a certain 'universal' way. In category theory one does not 'open' the things under investigation, but one describes them 'from the outside'. More strongly: in category theory one gives specifications instead of implementations. This categorical perspective is fruitful in computer science, where many black boxes occur, of which it is not known what precisely is inside.

Interesting applications of semantics are being developed in the design of semantics driven implementations. *Abstract interpretation* is used as a technique to investigate those properties of programs which may be derived from their 'execution' in restricted—mostly finite—models, e.g. to ascertain termination properties (so-called strictness analysis). SOS-style semantic specifications are at present investigated in a language independent fashion, e.g. by analyzing the feasibility of 'automatically' deriving a denotational semantics or a system of (equational) axioms from a given SOS definition.

Semantics is partly driven by its intrinsic foundational questions, and partly by external developments such as technological advances and associated software innovations. The scene of programming language design has expanded considerably in the decade of the 1980s. The group of the traditional imperative languages (ALGOL, PASCAL), together with an occasional functional language (LISP), formed the starting point of a rapidly growing variety of programming paradigms. Languages for concurrency played a central role in the 1980s. Next, the field of functional languages gained in impact, with the language ML as, possibly, the most influential contemporary representative. *Logic programming* (LP) is an area of much current interest, not in the least thanks to the influential Japanese fifth generation project. One relatively fresh protagonist on this scene is the paradigm of object-oriented (OO) languages, a belated offspring of the 1960s language SIMULA. Smalltalk and C^{++} are more contemporary instances of OO languages. One of the difficult issues at present is how to give a complete semantical account of concurrent object-oriented programming. This involves a combination of two levels: There are objects, which are collections of 'small' programs acting on a local state, specified by a class, and there is on a global level a 'pool of objects', in which one can have (concurrent) interaction via sending of messages.

All these language families pose their own problems and often require 'special-purpose' mathematical tools. For example, functional languages rely heavily on the (theory of the) typed lambda calculus, and LP is a programming variant of Horn clause logic, itself a version of resolution logic (which is, in turn, a way of viewing first order predicate logic).

Finally, we here draw attention to the growing interest for the interface between the semantics of programming languages and that of natural languages as studied in computational linguistics. Semantics has grown in the 1980s, both in depth and in width, facing ever new challenges to assimilate the continuous stream of foundational insights and technological advances.

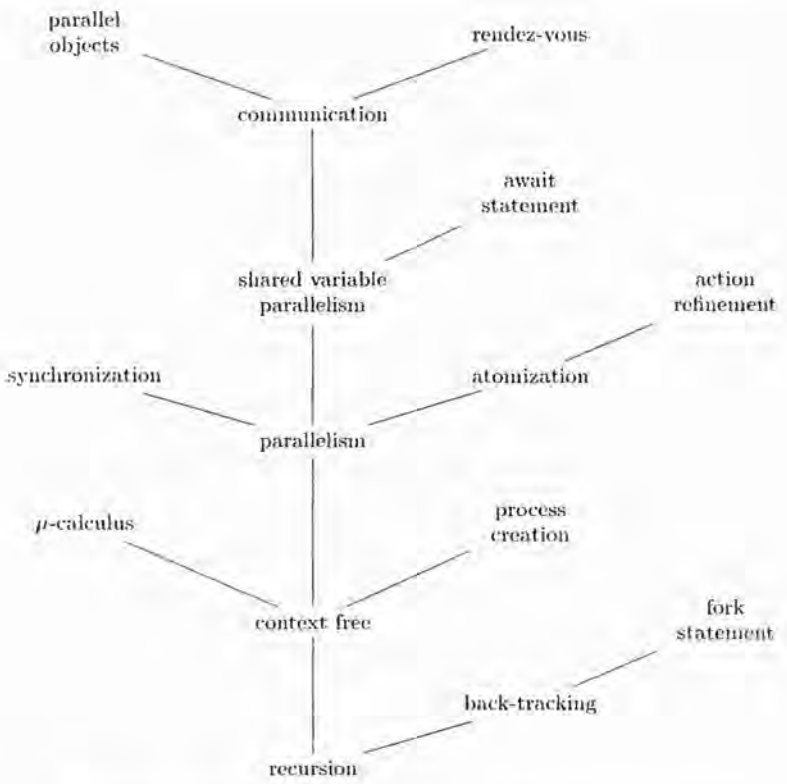


Figure 1. A selection of control flow notions as studied in ref. [4].

3. Contributions of CWI

3.1. Research topics

Parallelism or *concurrency* has been a major focus of semantic research at the CWI in the past decade. An overview is contained in the collection of reprinted papers [2]: [4] is an advanced text/monograph presenting a comprehensive survey of our work since the early eighties (see also figure 1). Characteristic for a good deal of our approach is, on the one hand, the reliance on topological structures in the semantic modelling, and on the other, the large variety of forms of parallelism considered. Not only the more traditional concurrency in an imperative setting, but also parallel versions of LP and OO have been studied in depth. In the period under review a total of nine Ph.D. theses have been completed on the theory of parallel processes in relation to the design and semantics of parallel languages

according to the styles of imperative, dataflow, LP and OO programming, with one further thesis on the proof theory for parallel OO. At present, the main topics in our research are (i) algebraic and coalgebraic approaches to transition systems; (ii) category-theoretic investigations in comparative domain theory; (iii) generalized finiteness conditions in topological models; (iv) predicate—versus state—transformations as theoretical underpinning for a study of refinement; (v) semantics of higher-order and object-oriented processes. In line with the general development mentioned above, category-theoretic tools play an important role in much of this research. For example, it has turned out to be useful to describe transition systems in terms of *coalgebras*, which formally are defined as the dual of algebras. Also observational equivalences, such as the widely used notion of *bisimulation*, can be described in coalgebraic terms. In this manner, a theory of coalgebras is being developed along the lines of (but dual to) universal algebra. This theory seems to have promising applications, for instance, in the semantic description of object-oriented languages.

In comparative domain theory, one of the main issues has been to reconcile the use of metric spaces and partial orders (and their corresponding Hausdorff and non-Hausdorff topologies). Lawvere's view of metric spaces as so-called *enriched categories*, already developed in the early 1970's, offers the right context for this problem. It has led not only to a unification of both theories, but also to new insights concerning, for instance, powerdomains and topology (see also figure 2).

3.2. International and national cooperation

A substantial part of the CWI research in this field over the years has been embedded in international or national collaborative projects. In the first category, we participated in the ESPRIT sponsored project Parallel Architectures and Languages (1984-1989, see [1] for a selection of its results on semantics) and the ESPRIT Basic Research Action Integration—integrating the foundations of functional, logic and object-oriented programming (1989-1992). Currently, our foundational work is supported by the SCIENCE-MASK project—Mathematical Structures in Semantics for Concurrency, and a (national) SION project entitled 'Non-well-founded sets in the semantics of programming languages'. Nationally, we have collaborated for many years with the groups led by G. Rozenberg (Leiden University) and W.P. de Roever (Eindhoven University of Technology), first in the SION-sponsored National Concurrency Project (LPC, 1984-1988), and next in the NFI-project REX—Research and Education in Concurrent Systems, 1988-1993. REX has funded a series of international schools/workshops; [3] contains the proceedings of the 1992 meeting on semantics. Presently, we are involved in a SION funded collaborative project entitled HOOP—Higher Order and Object-Oriented Processes—with as partners the CWI group in

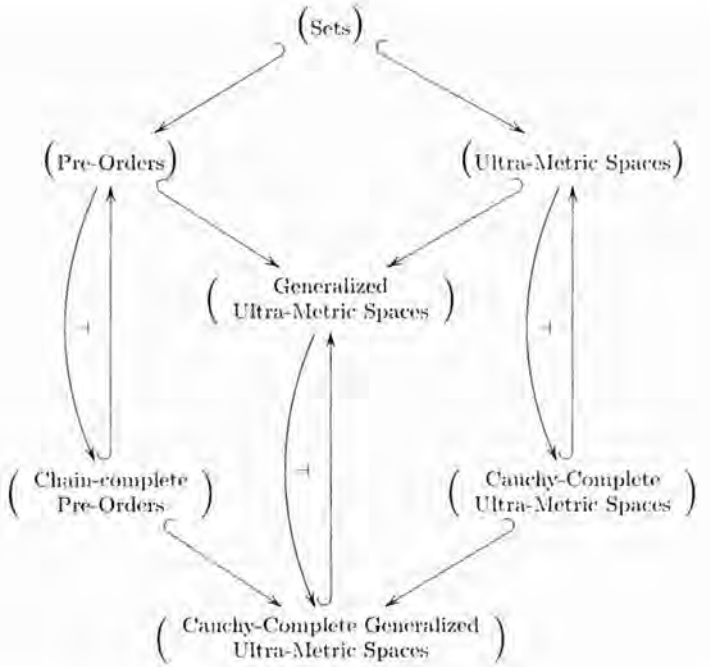


Figure 2. Categories of basic mathematical structures as used in denotational semantics, with some (adjoint) functors between them.

semantics. Leiden University (J. Engelfriet, G. Rozenberg) and Eindhoven University of Technology (J.C.M. Baeten).

References

1. J.W. de Bakker (ed.). (1989). *Languages for Parallel Architectures: Design, Semantics, Implementation Models*. Wiley.
2. J.W. de Bakker, J.J.M.M. Rutten (eds.). (1992). *Ten Years of Concurrency Semantics*, selected papers of the Amsterdam Concurrency Group. World Scientific.
3. J.W. de Bakker, W.P. de Roever, G. Rozenberg (eds.). (1993). *Semantics: Foundations and Applications*. Springer Lecture Notes in Computer Science Vol. 666.
4. J.W. de Bakker, E.P. de Vink (1995). *Control Flow Semantics*. MIT Press.

Hybrid Systems

F.W. Vaandrager

1. Introduction

Hybrid systems are systems that intermix discrete and continuous components, typically computers that interact with the physical world. Due to the rapid development of processor and circuit technology, such systems are becoming more and more common in all application domains, ranging from avionics and process control to robotics and consumer electronics. The specification, design and analysis of hybrid systems require a synthesis of ideas, concepts, mathematical theories and tools that are currently spread over distinct disciplines, most notably computer science and control theory. The interest in the formal treatment of digital systems that interact with an analog environment is certainly not new, but received a new impetus by the extension of formal models from computer science with real-time around 1991 and by the explosion of embedded applications of computer technology within our society. Today, the study of hybrid systems has grown into an area of research with rapidly increasing popularity.

In this contribution, we will briefly sketch the development of this field. Also, we will discuss recent work on hybrid systems in the group Concurrency and Real-time Systems at CWI.

Figure 1. The 3D Biped robot, developed at MIT, hops, runs and performs tucked somersaults. (Courtesy MIT Leg Laboratory.)

2. Motivation

2.1. Embedded computer systems

With the decrease in the size and price of computing elements, more and more computers are used within real-world technical applications such as in avionics, process control, robotics, telecommunications and consumer products.

In all these 'embedded' applications it is software that determines to a large extent the functionality of the products and that offers the required dynamics and flexibility. This makes the construction of large real-time embedded computer systems one of the most challenging tasks facing the computer science community, if not the engineering community as a whole.

Characteristic of embedded computer systems is that they are *reactive*: they accept stimuli from the outside world and react to those stimuli. This means that one can *only* design and reason about the correctness of these systems if one takes the behaviour of the outside world into account. As

an example, consider a computer controlling a chemical plant. Regularly, the control program reads sensor data, such as temperature and pressure, of the plant. Based on these data, the computer may decide to turn on a heating system, switch off a pump, etc. When a dangerous situation arises, for instance the pressure in a tank gets too high, the computer has to initiate appropriate action, like opening a valve. In order to formally prove correctness properties like 'No explosion will ever occur', the underlying mathematical model needs to contain information about the way in which the chemical reactions take place, the pressure in the tank evolves, etc.

Traditionally, computer scientists take a discrete view of the world: they assume that both a computer system and its environment can be modelled as a discrete event system. This assumption is justified and has proved to be very successful within application areas such as operating systems and computer networks. However, we see more and more applications, in particular in the area of embedded computer systems, in which modelling the environment as a discrete process greatly distorts reality and may lead to unreliable conclusions. Examples include the temperature in a thermostatically controlled room, hopping robots (see figure 1), and intelligent vehicle and highway systems (IVHS) utilizing 'platooning' technique. For these applications, the continuous models for the real-world developed by physicists and control engineers are typically more compact, more tractable and more accurate than the discrete approximations computer scientists tend to come up with. Since designing and reasoning about software for reactive systems is only possible if one has a model for both the software and the environment in which it operates, this provides a real practical motivation for the development of a comprehensive theory of *hybrid systems*, i.e., systems consisting of a non-trivial mixture of discrete and continuous components.

2.2. The need for a theory

The need for such a theory has also been identified by the control community. Inherently unstable applications such as flight by wire of an unstable aircraft cannot be controlled by a classical control system employing a single control rule. The only solution, which is the one being implemented today for such systems is a hybrid system, in which a digital controller keeps switching between different continuous control laws or control modes very rapidly.

The real-world is usually modeled by control engineers as a continuous dynamical system, described in the language of functional analysis. Computer scientists investigate the dynamics of discrete systems, described in the language of mathematical logic. Since the construction of large embedded computer systems will always involve representatives of both cultures, a comprehensive theory of hybrid systems is required to avoid a Tower of Babel. Such a theory must comprise a synthesis of ideas, concepts, mathe-

matical theories and tools that are currently spread over several disciplines, most notably computer science and control theory, but also electrical engineering, mechanical engineering, physics and others.

Even though hybrid applications have been built in practice since the beginning of the fifties, it is only since a few years that the mathematical study of hybrid systems is starting to receive the attention that it deserves. Until the end of the seventies, most research on program verification was devoted to the analysis of programs for 'autistic' (i.e. nonreactive) sequential batch computers. The eighties have witnessed a revolution in the formal methodology for the specification, verification, and development of reactive programs (and more general reactive systems). Once the concept of a reactive system was well understood, it turned out not to be so difficult to bring quantitative, real-time aspects into the picture. Even though the development of reliable real-time systems will remain an important research topic for many years to come, several important theoretical results concerning real-time systems were obtained by 1991. Around that time, the first two important workshops on hybrid systems were being organized in Ithaca, NY, USA and Lyngby, Denmark, and triggered a rapidly increasing interest in this new area, both in computer science and in applied mathematics.

Research on hybrid systems is part of the general effort to apply formal methods in software development. Formal methods have been under development since the mid-1960s, but it is in the last decade that significant developments have evolved, and over the last few years interest in formal methods has grown phenomenally. Highly publicized accounts of the application of formal methods to a number of well-known systems, such as the Darlington Nuclear Facility and Airbus, have helped to bring the industrial application of formal methods to a wider audience (see also figure 2). Today use of formal methods is required in the construction of software and digital hardware for certain critical systems. A recent paper by W.W. Gibbs [2] in Scientific American presents plenty of well-laid-out arguments and experi-

ences of formal methods. For an overview of the field we refer to the World Wide Web page

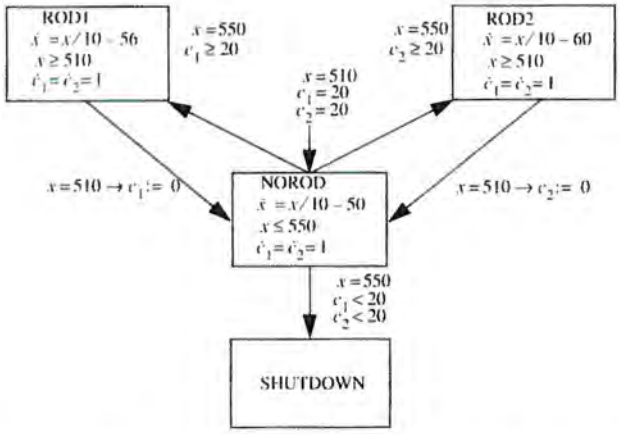
`http://www.comlab.ox.ac.uk/archive/formal-methods.html`,

which contains numerous pointers to other electronic archives throughout the world.

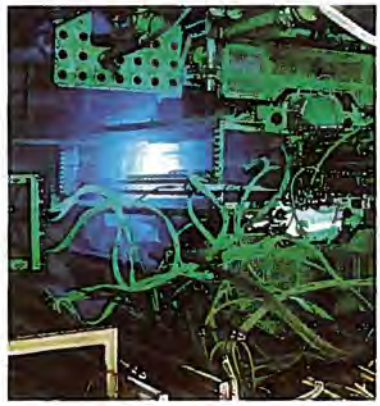
3. Research on Hybrid Systems

3.1. Semantic models and logics

Several semantic models for hybrid systems have been proposed in the literature. Good starting point for reading are the Springer LNCS volume [3] and the recent Special Issue of Theoretical Computer Science [4]. Even



A hybrid automaton for a simple temperature control system. The system controls the coolant temperature in a reactor tank by moving two independent control rods. The temperature of the coolant is represented by the variable x. Initially x is 510 degrees, both rods are outside the reactor core and the system is in state NOROD. In this state the temperature rises according to the differential equation $\dot{x} = x/10\text{-}50$. If the temperature reaches 550 degrees, three things can happen: either rod 1 is put into the reactor core and the automaton moves to state ROD1, or rod 2 is put into the reactor core and the automaton moves to state ROD2, or a complete shutdown occurs and the automaton moves to state SHUTDOWN. Mechanical factors make that a rod can only be placed in the core if it has not been there for at least 20 seconds. Shutdown will only occur if no rod is available. In the automaton, variables c_1 and c_2 are used to measure the times elapsed since the last use of rod 1 and rod 2, respectively. The initial values of both c_1 and c_2 are set to 20 seconds, so that initially both rods are available. Since they represent perfect logical clocks, the first derivatives of c_1 and c_2 with respect to time are always equal to 1. Control rod 1 refrigerates the coolant according to the differential equation $\dot{x} = x/10\text{-}56$; control rod 2 has a stronger effect and refrigerates the coolant according to the differential equation $\dot{x} = x/10\text{-}60$. If the temperature has decreased to 510 degrees, the system moves back from state ROD1 or ROD2 to state NOROD, and variable c_1 or c_2, respectively, is reset to 0. The correctness property for this system is not difficult to prove and says that the SHUTDOWN state cannot be reached. (This example is from Leveson et al.)



Core of High Flux Reactor in Petten.

Figure 2.

though there are many differences between the approaches, there is a surprising consensus that the behaviour of a hybrid system is best represented as an alternating sequence of instantaneous 'discrete' actions and 'continuous' phases during which time advances. Within the continuous phases the system's variables vary continuously according to a control law, fixed for the interval. Via the discrete actions the system can move to a new state, where a new control law becomes valid.

This view of hybrid systems is highly compatible with the earlier computer science semantic models for real-time systems: roughly speaking, the only difference is that where in real-time models the only available information about the continuous phases is the amount of time that passes, the hybrid models allow one to use differential equations, etc., to specify how precisely the system variables evolve in a continuous phase. Due to this similarity, many semantic models for timed systems generalize smoothly to hybrid systems. In several cases it is even possible to use verification techniques for timed systems directly for hybrid systems. Also in the area of specification logics many ideas generalize from timed to hybrid systems and logics such as TCTL, ICTL, TLA and the calculus of durations are being used in both settings.

3.2. Verification

However, there is a problem. Because the expressivity of the hybrid system models is enormous, the verification problem for these models is intrinsically difficult, even under severe restrictions. Typically, verification problems that are decidable in polynomial time for untimed systems (represented as finite transition systems), become exponentially hard for timed systems, and undecidable even for simple classes of hybrid systems. Much work remains to be done to identify restricted classes of hybrid systems that on the one hand are sufficiently expressive to model realistic applications in the area of embedded systems, but on the other hand can be analyzed by algorithmic means. A very promising line of research here is the work on (semi) decision procedures and tools for (subclasses of) the so-called *linear hybrid automata* of R. Alur et al. Two prototype tools have been developed, the KRONOS tool in Grenoble and the HYTECH tool at Cornell. Using these tools, their implementors have been able to verify automatically dozens of toy examples proposed in the literature on hybrid systems, as well as some practical examples from industry. The key idea behind these tools is to represent infinite sets of states as convex polyhedra in multi-dimensional space and to use standard data structures and geometric algorithms for polyhedral manipulation to do reachability analysis and model checking.

Despite the recent successes, computer aided verification techniques for linear hybrid automata will never become the solution to all industrial verification problems: they perform well in the case of small tricky circuits like

the Philips audio control protocol analyzed at CWI, but are not designed to face the immense timing problems that arise in larger applications, such as control systems for sonar applications, air-traffic, etc. Nevertheless, one can envisage two important uses of this work in a large-scale application. First, the knowledge about (linear) hybrid automata will be useful in a mathematical formulation of the problem. Second, there will be isolated situations where the tools themselves can be applied.

Clearly, the work on semantic models, logics and verification methods for hybrid systems is just starting. In the case of untimed discrete event systems, a rich body of closely related theories has been created during the last twenty years involving temporal and modal logics, assertional verification methods, process algebras, and tools for computer aided verification. Almost all of this work still needs to be lifted to (or at least related to) the setting of hybrid systems.

3.3. Perspective from control theory

Besides computer science also control theory plays an important role in the theory of hybrid systems, and thus we see for instance at CWI that both the Computer Science group Concurrency and Real-time Systems, and the Mathematics group System and Control Theory have become interested in hybrid systems, and benefit from each others expertise. It is interesting to note that the questions asked in control theory are quite different from those asked in computer science. Whereas in computer science there is much emphasis on verification, control theory concentrates more on synthesis: it aims at finding synthesis procedures for a supervisor that forces a discrete event system such that it satisfies prespecified control objectives. A well-known theory of 'supervisory control' for untimed discrete event systems has been developed by P.J. Ramadge and W.M. Wonham. There are a few applications of this theory, but more experience must be gained and the model needs to be refined before it will become really useful in practice. Interesting approaches to the control of timed systems are proposed by O. Maler, A. Pnueli and J. Sifakis, and by H. Wong-Toi and G.J. Hoffman.

In traditional control system theories stability is an important performance criterion. Here stability means that for the controlled system, small changes in input and relevant parameters yield small changes in output. All theories of stability of continuous systems are topologically based. In the design of discrete event systems it is very difficult to come up with a similar notion of stability: in software engineering it is well known that replacing a '.' by a ';' can have a dramatic impact on the behaviour of almost any program. Kolm's theory of declarative control is an attempt to define a notion of stability for hybrid systems, using non-Hausdorff subtopologies of the usual topologies for continuous systems.

Networks of continuous devices have been studied by control engineers

for a long time and some hybrid system models build directly on this long tradition. It is important to relate these models to the automaton based models proposed by computer scientists.

3.4. Specification and implementation of embedded systems

Virtually all of the specification languages that are currently used by computer scientists to formally specify software systems have a discrete event semantics and are not directly suitable for the formal specification and analysis of hybrid systems. (Examples of such languages are VDM, Z, COLD-1 and LOTOS.) A possible exception is the language Funmath, a declarative formalism for describing systems with both analog and digital components, that has been developed in Nijmegen by R.T.G.M. Boute and his team.

If one leafes through a document with a state-of-the-art specification of an embedded software system, one encounters a mixture of architecture diagrams, programming text, flow charts, transition tables, diagrams by electrical engineers with IC's, transistors, resistors, etc., and diagrams made by mechanical engineers in which the mechanical parts of the system are described. The relations between the various parts of the specification are only described by informal text, there is no such thing as a common semantic framework for all the design notations that are used. This is undesirable, since in the design of embedded systems hardware and software are more and more viewed as interchangeable: often mechanical and electronic implementations of new and improved functions are replaced by software solutions (anyone can think of dozens of examples in the area of consumer electronics). The theory of hybrid systems aims at providing a semantic basis for a new generation of wide-spectrum formal specification languages in which *all* relevant elements in a design of an embedded software system can be described formally in an integrated way. In order to be accepted by software and control engineers in industry, these new languages should contain (close variants of) design notations that are currently used as sub-

languages. The role of hybrid system theory will mainly be one of glue by which different design notations can be formally related.

It is well known that the formalization step is a major source of errors in the design of critical software: therefore an important consideration in the design of a specification language is the readability of the expressions written in it. From this perspective, the work on graphical specification languages for hybrid systems is quite important.

4. Work at CWI

Central to the approach of the Concurrency and Real-time Systems group at CWI is the I/O automata model of N.A. Lynch and M. Tuttle. Lynch and F.W. Vaandrager extended this model for reactive systems to the setting of real-time and hybrid systems. Below we will first briefly outline the main

features of this extension and then discuss a practical application.

4.1. Timed transition systems

In the theory of reactive systems, a central role is played by the notion of a *transition system (TS)*. A TS consists of a set of *states*, a subset of *initial* states, a set of *(discrete) actions*, and a set of *(discrete) transitions*, which are triples

$$s \xrightarrow{a} s'$$

specifying that from state s the system can evolve to state s' by the instantaneous occurrence of the action a. A run of a TS starts in an initial state. The system jumps from state to state via instantaneous transitions, and in between these transitions, it can remain arbitrarily long in any state.

At the lowest level, timed and hybrid systems can be described by TS's with as an additional component a collection of *time transitions*, which are triples

$$s \xrightarrow{d} s'$$

specifying that from state s the system can evolve in a positive, real-valued amount of time d to state s'. In the model of *timed transition systems (TTS)* of Lynch and Vaandrager, two axioms are imposed on time transitions. The first axiom states that if there are time transitions $s \xrightarrow{d} s'$ and $s' \xrightarrow{d'} s''$, there exists a time transition $s \xrightarrow{d+d'} s''$. The second axiom, which is a bit more involved to state, postulates that for each time transition $s \xrightarrow{d} s'$ there exists a *trajectory*, a function w that specifies an intermediate state for each intermediate point in time, such that $w(0) = s$, $w(d) = s'$, and for all $t, t' \in [0, d]$ with $t < t'$,

$$w(t) \xrightarrow{t'-t} w(t').$$

Thus a trajectory describes *how* the system evolves from s to s'. A run of a TTS consists of a sequence of two-phase steps. The first phase of a step corresponds to a continuous state transformation described by a trajectory. In the second phase the state is submitted to a discrete change taking zero time described by a discrete transition.

We can add more structure to timed transition systems by defining states to be pairs $(\vec{x}, \vec{y})$ of a vector $\vec{x}$ of *discrete variables* and a vector $\vec{y}$ of *continuous variables*. In a discrete transition both the discrete and continuous variables can be changed. However, in a time transition only the continuous variables may change. In the timed automata model of Alur and Dill, which is widely used for the description and analysis of real-time systems, all time transitions are of the form

$$(\vec{x}, \vec{y}) \xrightarrow{d} (\vec{x}, \vec{y} + d),$$

where $\vec{y} + d$ is the vector obtained by adding d to each of the variables in vector $\vec{y}$. In this model, the continuous variables behave as perfect logical *clocks*, whose values increase with the same rate as time. In the timed transition systems associated to the more general models of hybrid automata (see the paper of Alur et al. in [4]), the way in which the continuous variables change can be specified via differential equations. In figure 2, an example is presented of a hybrid automaton modelling the temperature control system of a nuclear reactor.

In [1], D.J.B. Bosscher, I. Polak and Vaandrager develop a language for the specification of linear hybrid automata, and define the semantics of this language via a translation to timed transition systems.

4.2. An application

In [1], the theoretical work on linear hybrid systems has also been applied to solve a problem from Philips. This application will be briefly discussed below.

Fully fledged computer networks are standard features in today's consumer electronics, like the Philips 900 audio system (see also figure 3). These networks make it possible for the different devices to talk to each other, and to offer a series of new, useful services to the consumer. A consumer can for instance wake up the whole system by touching a single button: there is no need to switch on the tuner first, then the CD player, then the amplifier, etc. Instead the system will do this job by broadcasting a 'wake up' message over the network. The main technical difficulty in building the network for the Philips 900 audio system was that it had to be cheap: consumers are only willing to pay a tiny bit more for the additional services provided by the network. In fact, the only additional hardware that Philips needs to implement in the network consists of a few transistors, resistors, etc., for the bus interface. The software runs on microprocessors that have to be present anyway. Because the clocks of these microprocessors drift, and because sometimes the programs dealing with the network have to wait for other programs that run on the same microprocessors, the network protocol has to deal with a significant uncertainty in the timing of events. In fact, Philips allows for a tolerance of 1/20 on all the timing.

At CWI, correctness was proved of part of the Easylink real-time protocol used by Philips to achieve reliable communication between the devices despite this very large timing uncertainty. The protocol is modeled as a linear hybrid automaton, with continuous variables to represent the drifting logical clocks of the sender and receiver in the protocol. Formally, the drifting is expressed by the requirement that the first derivative of the clock variables is in the interval $[1 - \mathsf{T}, 1 + \mathsf{T}]$, where T is the tolerance on the timing. Correctness of the protocol has been proved if the tolerance is less

Figure 3. Philips 900 audio system. (Photo: courtesy n.v. Philips Industrial Activities Leuven.)

than 1/17. This value is larger than the tolerance of 1/20 that is allowed by Philips. A counterexample shows that the protocol fails for tolerances greater or equal to 1/17.

In order to manage the complexity of real world applications, mechanical support is absolutely essential. Therefore, much of the research effort on hybrid systems is currently devoted to the development of mechanical tools that support specification and verification. At CWI, W.O.D. Griffioen has succeeded in mechanically checking the complete verification of the audio control protocol using the general purpose theorem proving tool LP. An impressive complementary result has recently been obtained by P.-H. Ho and Wong-Toi from Cornell University. Based on the CWI modelling of the Philips protocol, they verified an instance fully automatically using the HYTECH symbolic model checker, and also synthesized automatically the maximum clock drift of 1/17. Independently, C. Daws and S. Yovine from Grenoble have also verified the protocol fully automatically using the KRONOS tool.

References

1. D.J.B. Bosscher, I. Polak, F.W. Vaandrager (1994). Verification of an audio control protocol. H. Langmaack, W.-P. de Roever, J. Vytopil (eds.). *Proceedings of the Third International School and Symposium on Formal Techniques in Real Time and Fault Tolerant Systems*, Lübeck, Germany, September 1994. LNCS 863, Springer-Verlag, 1994. Full version available as Report CS-R9445, CWI, Amsterdam, 170-192.
2. W.W. Gibbs (1994). Software's chronic crisis. *Scientific American*, 72-81.
3. R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel (eds.) (1993). *Hybrid Systems*, LNCS 736. Springer-Verlag.
4. A. Pnueli and J. Sifakis (eds.) (1995). *Special Issue on Hybrid Systems of Theoretical Computer Science 138(1)*, Elsevier Science Publishers.

Generating Interactive Programming Environments

J. Heering, P. Klint

1. Introduction

During the past ten years considerable progress has been made towards the automatic generation of interactive programming/development environments on the basis of a formal definition of some programming or specification language. In most cases, research has focused on the functionality and efficiency of the generated environments. These are the key quality factors which will ultimately determine the acceptance of environment generators. Only marginal attention has been paid to the actual development process of formal language definitions. Assuming the quality of automatically generated environments to become satisfactory within a few years, the cost of developing formal language definitions will then become the next limiting factor determining the ultimate success and acceptance of environment generators.

We will briefly sketch the design and implementation of a *meta-environment* (a development environment for formal language definitions) based on the formalism ASF+SDF and some of its applications.

2. Background—The Centaur system

A programming environment is a coherent set of interactive tools for developing and executing programs or specifications in some formal language. Well-known examples of such tools are syntax-directed editors, interpreters, debuggers, code generators, and prettyprinters. Programming environments

have been generated automatically for languages in such diverse application areas as programming, formal specification, proof construction, text formatting, process control, and statistical analysis. All projects in this area are based on the assumption that major parts of the generated environments are language independent and that all language dependent parts can be derived from a suitable language definition.

An example of a general architecture for programming environment generation is the CENTAUR system which was developed mainly by INRIA (France) in the ESPRIT GIPE project (1985-1993) in which CWI and the University of Amsterdam participated. This is a set of generic components for building environment generators. The kernel provides a number of useful data types but does not make many assumptions about, for instance, the language definition formalism itself. It has been extended with compilers for various language definition subformalisms as well as with several interactive tools. As such CENTAUR is an extensible toolkit rather than a closed system. We used it to build the ASF+SDF Meta-environment.

3. THE ASF+SDF META-ENVIRONMENT

The ASF+SDF Meta-environment [5] is a development environment for formal language definitions and an associated programming environment generator built on top of CENTAUR. Our research, which was part of the GIPE project mentioned before, went through three phases:

- Design of an integrated language definition formalism (ASF+SDF).
- Implementation of a generator for interactive programming/development environments given a language definition written in ASF+SDF.
- Design and implementation of an interactive development environment for the ASF+SDF formalism itself.

The result is the *Meta-environment* mentioned in section 1 in which language definitions can be edited, checked and compiled just like programs can be manipulated in a *generated environment*, which is an environment obtained by compiling a language definition. Note that 'compiling a language definition' and 'generating an environment' are synonymous in our terminology. Both the generator itself and the Meta-environment have been implemented on top of the CENTAUR system.

Figure 1 shows the overall organization of our system. First of all, we make a distinction between the *Meta-environment* and a *generated environment*. In the Meta-environment we distinguish:

- A language definition (in ASF+SDF) consisting of a set of modules `M1...Mn`.

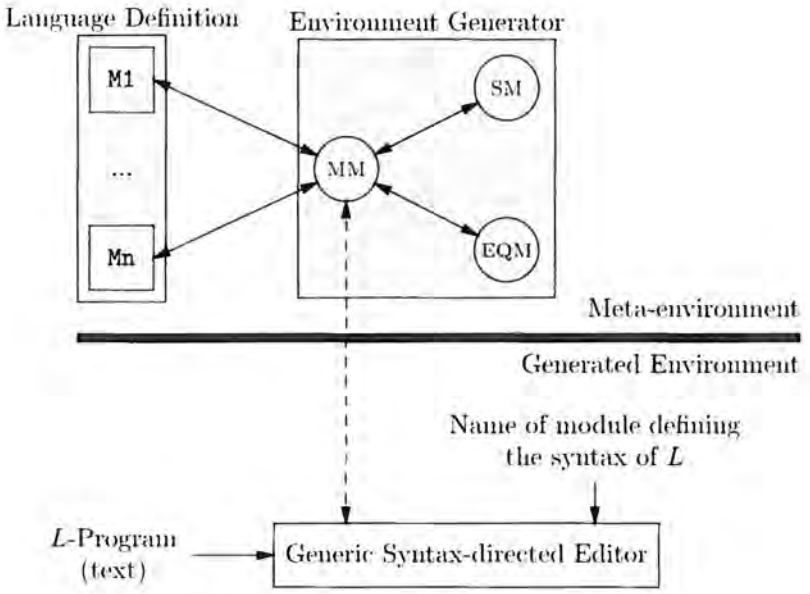


Figure 1. Global organization of the ASF+SDF Meta-environment.

- The environment generator itself, which consists of three components: a Module Manager (MM) controlling the overall processing of the modules in the language definition, the Syntax Manager (SM) controlling all syntactic aspects, and the Equation Manager (EQM), taking care of all semantic aspects of the language definition.

The output of the environment generator is used in conjunction with GSE (Generic Syntax-directed Editor), a generic building block which we use in generated environments. GSE not only supports text-oriented and syntax-oriented editing operations on programs but can also be extended by attaching 'external tools' which perform operations on the edited program such as checking and evaluation. The main inputs to the Generic Syntax-directed Editor are:

- A program text P.
- The modules defining the syntax of P.
- Connections with external tools.

As both the syntax description of P and the definition of external tools may be distributed over several modules, we are faced with the problem of managing several sets of syntax rules and equations simultaneously. One of the major contributions of the ASF+SDF Meta-environment is that the

system is so interactive and responsive that users are completely unaware of the fact that each modification they make to their language definition has major impacts on the generated environment. For instance, the presence of a parser generator is completely invisible to the user. As a consequence, the system is also accessible to 'naïve' users who have no previous experience with tools like scanner and parser generators. Important factors are: (1) an internal syntax tree representation ('term'—see next section) and a prettyprinter for the language are derived automatically from the language definition; (2) after parsing, syntax trees are built automatically; (3) the generated scanner, parser, tree constructor and rewrite system are interfaced automatically. To summarize, several parts of the generated implementation are derived from the language definition, and the system takes care of the interfacing of *all* components of the generated environment.

The implementation of the ASF+SDF Meta-environment is based on *lazy/incremental* program generation [3].

4. TERM REWRITING

Central to our approach is the fact that we represent everything (i.e., programs and specifications being edited) as uniform tree structures which we call *terms*. All operations on programs—like checking and compiling—are expressed as operations on their underlying term representation. These operations have to be defined in the language definition and their execution is based on *term rewriting*. Given some initial term t_0, an attempt is made to apply a rule in the specification and transform the initial term into a new term t_1. This process is repeated until a term t_n is obtained to which no further rule is applicable. This is the *normal form* of the initial term t_0.

Clearly, efficient term rewriting is essential to us and we approach this problem from several angles. First, by investigating how rewrite rules can be translated directly to C programs. This would enable elimination of much of the overhead of term rewriting (in particular the search for matching rules) by performing an extensive static analysis of the given set of rules. A first prototype of this approach, the ASF2C compiler, has been completed and yields a speed improvement of a factor of 50–100 over our current, more interpretive, approach. Secondly, we have investigated *incremental rewriting*, a technique where previous runs of the rewriting engine on the same, or a slightly modified, term are being reused to avoid rewriting steps. This method is important for speeding up interactive tools that operate on terms. A typical example is an interactive typechecker operating on a program being edited by a user.

The fact that we base our computations on term rewriting gives us some interesting possibilities which can be exploited in the generated environments. One of them is *origin tracking* [7], which establishes links between subterms of the normal form t_n and the corresponding subterms (origins)

of the initial term t_0. This is vital information for interactive tools like error reporters (to associate an error message with a part of the source program) and animators (to visualize the statement we are currently executing). Generalizations of origin tracking (i.e., *dependence tracking*) permit the formulation of program slicing in the context of term rewriting. This may be useful in systems for interactive program understanding and reverse engineering.

5. Current research

The ASF2C compiler already mentioned above has demonstrated the potential of compiling algebraic specifications to efficient code. Its redesign, which is currently in progress, will introduce further optimizations and reduce the memory requirements of the generated code. Since the compiler has itself been specified in ASF+SDF, it also benefits from these improvements. Other extensions involve selective outermost rewriting [4], and the use of narrowing for simulating input/output.

In cooperation with J. Field (IBM T.J. Watson Research Center) work is in progress on optimizing compilers. The basic idea is to translate the source language, e.g., C, to an intermediate language called PIM. All further optimizations can be expressed as symbolic manipulations on the intermediate PIM representation of the program. These manipulations have been defined using ASF+SDF and are based on the ω-completion of algebraic specifications as described in [2].

In close cooperation with the University of Amsterdam (Programming Research Group) various extensions of the Meta-environment are being developed, e.g., generation of prettyprinters and documentation tools, visual editors, and the integration of parsing and rewriting. As a step towards reengineering the current implementation of the Meta-environment, a component interconnection architecture called ToolBus was developed in which all direct communication between components ('tools') is forbidden. Instead, all such communications are controlled by a process-oriented script that formalizes all the desired interactions between tools. No assumptions are being made about the implementation language or execution platform of each tool: tools may be implemented in different languages and may run on different computers. By adopting this approach we hope to make the implementation more flexible and manageable and to facilitate connecting externally developed software.

6. Applications

Although originally designed as a generator for *programming* environments, it has turned out there are many other areas where the ASF+SDF Meta-environment can be applied. These range from general system design and the specification of environments for various languages to specific areas like

query optimization, hydraulic simulation, and application generators. We sketch three applications in some detail:

- In the context of ESPRIT project COMPARE (1991-1995), which aimed at the construction of optimizing compilers for parallel architectures, we designed a specification formalism fSDL for defining the intermediate data representations in compilers. In addition, using ASF+SDF we constructed a generator that compiled these specifications into C.

- In cooperation with a Dutch bank, we designed a specification language for financial products. Given such a product definition, appropriate (Cobol) code can be generated to include the information related to the product instance in the company's information system. In this way the time needed to construct software for new products can be reduced from months to days.

- In close cooperation with P.D. Mosses (Aarhus, Denmark), we constructed an interactive system to support the development of specifications written in *Action Semantics*, a formalism for defining the semantics of (programming) languages. It is currently being used for defining the semantics of ANDF (Architecture Neutral Definition Format), an exchange format for compiled programs.

Other applications using the ASF+SDF Meta-environment include:

- Automated induction proofs: D. Naidich (University of Iowa), T.B. Dinesh (CWI).

- Category theory: S. Vigna (University of Milano).

- Program transformations: M.G.J. van den Brand (UvA), H. Meijer (KUN).

- Message Sequence Charts: E.A. van der Meulen (UvA), S. Mouw (TUE).

- π-calculus: A. van Deursen (TUE).

- μCRL: J.A. Hillebrand (UvA), J.F. Groote (UU).

A survey of recent work can be found in [6]. Our earlier work on algebraic specifications can be found in [1].

ACKNOWLEDGEMENTS

The following persons made contributions to this project: H.C.M. Bakker, J.A. Bergstra, M.G.J. van den Brand, A. van Deursen, N.W.P. van Diepen, T.B. Dinesh, C. Dik, H. van Dijk, J.J. Ganzevoort, P.R.H. Hendriks, J.F.Th. Kamperman, A.S. Klusener, J.W.C. Koorn, M.H. Logger, E.A. van der Meulen, P.A. Olivier, J.G. Rekers, M. Res, F. Tip, S. Üsküdarli, A. Verhoog, E. Visser, S. van Vlijmen, P. Vriend, H.R. Walters, A. van Waveren.

REFERENCES

1. J.A. BERGSTRA, J. HEERING, P. KLINT (1989). *Algebraic Specification*. ACM Press Frontier Series. The ACM Press in cooperation with Addison-Wesley.
2. J. HEERING (1986). Partial evaluation and ω-completeness of algebraic specifications. *Theoretical Computer Science 43*, 149-167.
3. J. HEERING, P. KLINT, J. REKERS (1994). Lazy and incremental program generation. *ACM Transactions on Programming Languages and Systems 16(3)*, 1010–1023.
4. J.F.TH. KAMPERMAN, H.R. WALTERS (1995). Lazy Rewriting and Eager Machinery. JIEH HSIANG (ed.). *Rewriting Techniques and Applications, 6th International Conference (RTA-95), Lecture Notes in Computer Science*. Vol. 914. Springer-Verlag, 147-162.
5. P. KLINT (1993). A meta-environment for generating programming environments. *ACM Transactions on Software Engineering and Methodology 2(2)*, 176-201.
6. M.G.J. VAN DEN BRAND, A. VAN DEURSEN, T.B. DINESH, J.F.TH. KAMPERMAN, E. VISSER (eds.) (1994). *ASF+SDF'95*, a workshop on Generating Tools from Algebraic Specifications. Tecnical Report P9504. Programming Research Group, University of Amsterdam.
7. A. VAN DEURSEN, P. KLINT, F. TIP (1993). Origin tracking. *Journal of Symbolic Computation 15*, 523–545.

Rewriting

I. Bethke, J.W. Klop

1. Introduction

Many computations can be modeled as step-by-step transformations or rewriting of a string of symbols (words, expressions, terms), intending to reach some final result as an answer (a *normal form*). Such a rewrite step, to be perceived as an atomic computation step, consists of replacing part of the expression by a simpler part, according to the rules of some *rewriting system*. E.g., in arithmetic: $(3 + 5) \cdot (1 + 2) \rightarrow 8 \cdot (1 + 2) \rightarrow 8 \cdot 3 \rightarrow 24$.

The study of rewriting systems belongs to the area of symbolic computation. The main applications of rewriting are in the fields of abstract data types and algebraic specifications, automated theorem proving, functional programming and logic programming. Rewriting can be studied at several levels. In this informal survey, we aim to give an impression of several of these levels, roughly in order of increasing complexity. In our choice of topics, we have put an emphasis on subjects that CWI has contributed to during the last decade. Especially we mention conditional rewriting, higher-order rewriting, infinitary rewriting and term graph rewriting.

2. Abstract rewriting

The simplest level is *abstract rewriting* which consists essentially of the study of one or more binary relations on some set of abstract objects. Figure 1 displays such an abstract reduction system or ARS. The arrows give the binary rewrite relation on the four objects a, b, c, d. Thus we have *successful*

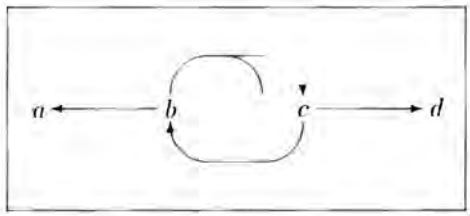


Figure 1.

terminating rewritings such as $b \to c \to d$, but also *unsuccessful* infinite rewritings $b \to c \to b \to c \to \cdots$. The elements a and d, from which no further step is possible, are called *normal forms*.

3. STRING REWRITING

A more concrete form of rewriting is that of *string rewriting*. As an example, consider the following interesting puzzle, posed by H. Zantema (Utrecht University): Given is the string rewrite rule

$$0011 \to 111000.$$

An application of the rule consists in replacing in some 0, 1-string an occurrence of a substring 0011 by 111000. For example, we may rewrite

$$\underline{0011}1111 \to$$
$$1110\underline{0011}11 \to$$
$$111011\underline{00011} \to$$
$$11101110111000$$

from where no further rewriting is possible; so the string is a normal form. The reader may enjoy herself with discovering that any rewrite sequence using this rule must terminate eventually - that is, the rule has the *termination property*. The proof is non-trivial.

4. FIRST-ORDER TERM REWRITING

The next level of rewriting, next in order of increasing complexity, is that of *first-order term rewriting*. Whereas strings (or words) over some alphabet are rather poorly structured carriers of information, first-order terms are a very general medium for carrying information, and this notion together with its semantics as given by A. Tarski has turned out to be extremely fruitful and permeates much of mathematical logic and computer science of this century. We introduce the notion of a first-order rewrite system by the example in Table 1.

These four rewrite rules specify elegantly addition `A` and multiplication `M` on natural numbers $0, \mathtt{S}(0), \mathtt{S}(\mathtt{S}(0)), \ldots$. Using these rules we compute

r_1	$\mathtt{A}(x, 0)$	$\longrightarrow$	x
r_2	$\mathtt{A}(x, \mathtt{S}(y))$	$\longrightarrow$	$\mathtt{S}(\mathtt{A}(x, y))$
r_3	$\mathtt{M}(x, 0)$	$\longrightarrow$	0
r_4	$\mathtt{M}(x, \mathtt{S}(y))$	$\longrightarrow$	$\mathtt{A}(\mathtt{M}(x, y), x)$

Table 1.

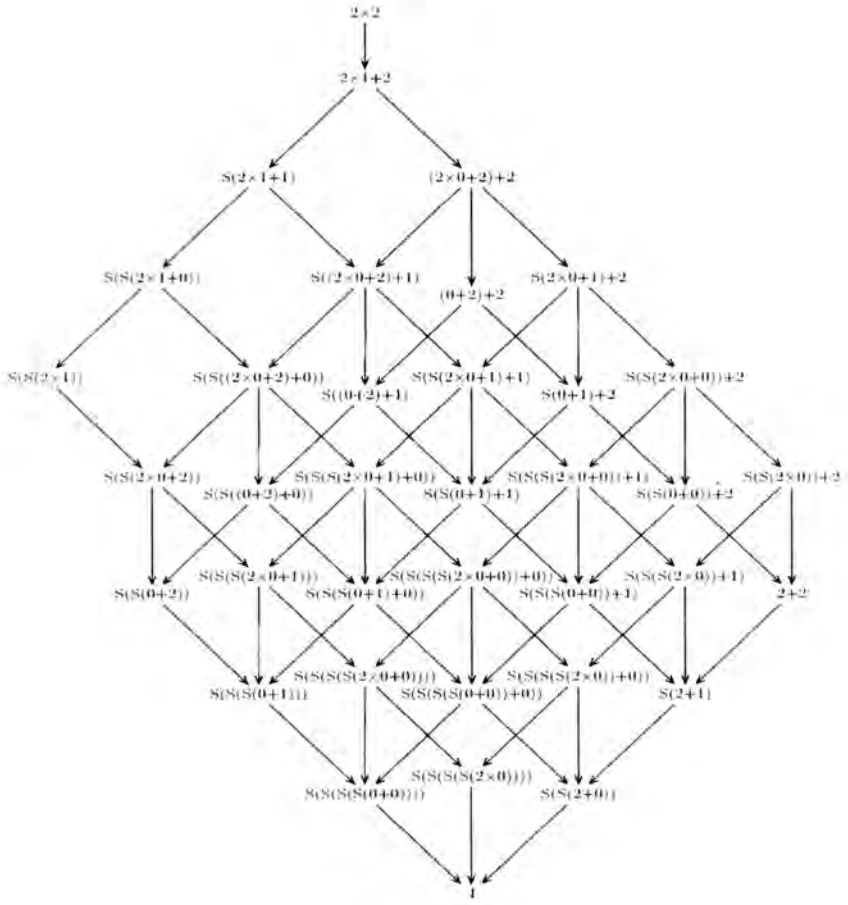


Figure 2.

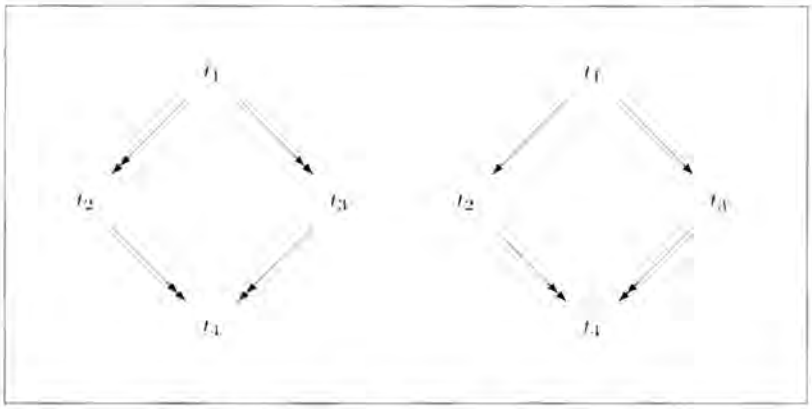


Figure 3. Confluence and weak confluence property.

$2 \times 2 = 4$ as displayed in figure 2, where the usual *infix* notation for `A` and `M` is employed. An inspection of figure 2, containing all possible computations or rewrite sequences of 2×2 to 4, is enough to wonder why all computations indeed yield the same final result or normal form. In other words why the rewrite system is *confluent* (see figure 3). There a double-headed arrow $\twoheadrightarrow$ denotes a sequence of rewrite steps of arbitrary length (possibly 0). Fortunately, the system for arithmetic is confluent as the rules have the technical property of *orthogonality* (they are independent of each other in the sense that applying one rule does not destroy the possible application of another rule.).

Figure 3 displays the confluence property, also called *Church-Rosser property*, which is next to the termination property the most fundamental property in rewriting. It guarantees the uniqueness of normal forms. The weaker property of *weak confluence* is useful to prove confluence, but actually not enough: turning back to the ARS in figure 1 we see that this rewrite system is weakly confluent, but not confluent. Every pair of divergent single *steps* can be joined again (by arbitrary long rewrite sequences), but not every pair of divergent rewrite *sequences* can be made to converge again. (E.g., the end points of the pair $b \leftarrow a$ and $b \rightarrow c \rightarrow d$ cannot come together any more.) Actually, the weak confluence property suggests that one can obtain confluence by repeatedly *tiling the plane* with tiles as in the figure for weak confluence. But this will not succeed always, as the tiling procedure might go on indefinitely, and diverge to yield some fractal-like picture as in figure 4.

But looking at this infinite *rewrite diagram*, it is easily and rightly conjectured that in the presence of the termination property we will have success with this tiling procedure.

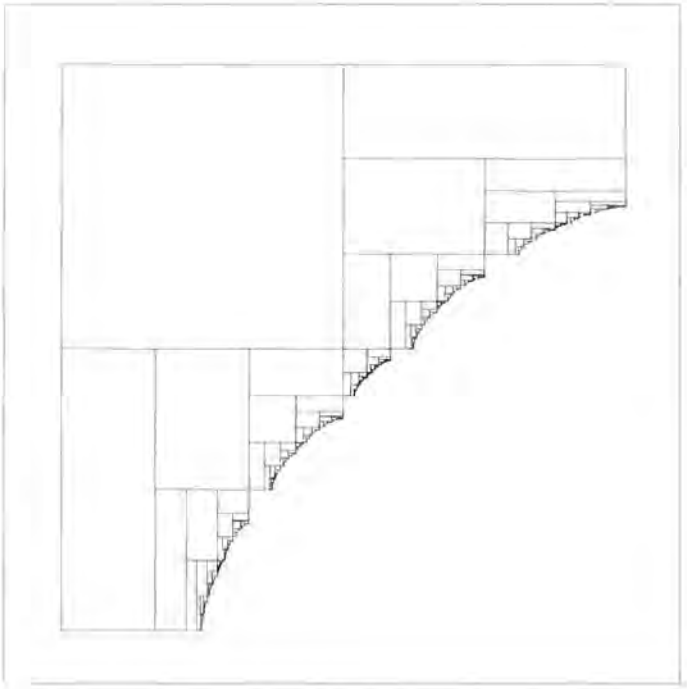


Figure 4.

5. Combinatory Logic

Actually, we do not need to devise special-purpose rewrite systems such as the one above in Table 1—there is a universal, general-purpose rewrite system, discovered in 1924 by M. Schönfinkel, called *Combinatory Logic*. Just as lambda calculus it is one of the perennial gems that mathematical logic has contributed to computer science. Combinatory Logic consists of the three rewrite rules in Table 2.

Here S, K, I are the basic constants and x, y, z are variables for terms. It is understood that a part of a term, built from S, K, I and matching the left-

r_1	$(((\mathtt{S} \cdot x) \cdot y) \cdot z)$	$\rightarrow$	$((x \cdot z) \cdot (y \cdot z))$
r_2	$((\mathtt{K} \cdot x) \cdot y)$	$\rightarrow$	x
r_3	$(\mathtt{I} \cdot x)$	$\rightarrow$	x

Table 2.

hand side of one of these rules, may be replaced by the corresponding right-hand side. The binary operator · is called *application*; often its notation is suppressed. Thus we have, e.g., the two step rewrite sequence

$$(((\mathtt{SK})\mathtt{I})\mathtt{I}) \to ((\mathtt{KI})(\mathtt{KI})) \to \mathtt{I}$$

which cannot be prolonged, since the final term I is irreducible (a normal form). Not all terms in CL can be rewritten to a normal form: for instance $((\mathtt{SI})\mathtt{I})((\mathtt{SI})\mathtt{I})$ cannot.

6. Conditional rewriting

There are several ways to enhance, refine, or generalize first-order rewriting. One of them is *conditional rewriting*, an example of which is given in Table 3.

This system computes the greatest common divisor of natural numbers (generated by 0 and successor S) using the two conditional rewrite rules r_8 and r_9. The intended meaning of such conditional rewrite rules is that their application is only allowed if the condition to the right of $\Longleftarrow$ is fulfilled. Here a circularity is apparent: the conditions are stated themselves in terms of the rewrite relation $\to$ that they help to define. But a little bit of theory shows that this circularity is not harmful at all but quite innocent. Theory also has established (in an observation of J.A. Bergstra) that the conditional format is indeed strictly more powerful than the unconditional first-order scheme: some natural data types can be specified with a conditional rewrite system, but cannot without.

A different enhancement of first-order rewriting is to impose a certain order on the rewrite rules, with the intention that a rule which is higher in the order will be the preferred one to apply in case of choice. Such systems are called *priority rewrite systems*; their actual definition and semantics is

r_1	$0 < 0$	$\to$	$\mathtt{false}$		
r_2	$0 < \mathtt{S}(x)$	$\to$	$\mathtt{true}$		
r_3	$\mathtt{S}(x) < 0$	$\to$	$\mathtt{false}$		
r_4	$\mathtt{S}(x) < \mathtt{S}(y)$	$\to$	$x < y$		
r_5	$\mathtt{S}(x) - \mathtt{S}(y)$	$\to$	$x - y$		
r_6	$0 - x$	$\to$	0		
r_7	$x - 0$	$\to$	x		
r_8	$\mathtt{gcd}(x, y)$	$\to$	$\mathtt{gcd}(x - y, y)$	$\Longleftarrow$	$y < x \to \mathtt{true}$
r_9	$\mathtt{gcd}(x, y)$	$\to$	$\mathtt{gcd}(x, y - x)$	$\Longleftarrow$	$x < y \to \mathtt{false}$
r_{10}	$\mathtt{gcd}(x, x)$	$\to$	x		

Table 3.

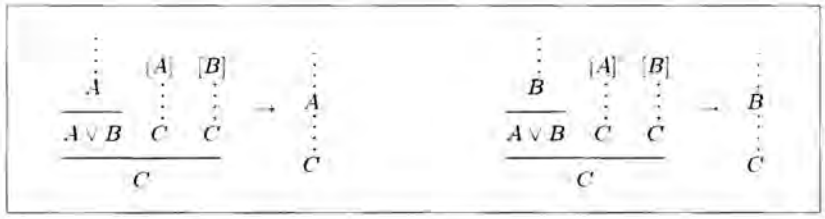


Figure 5.

technically difficult. An interesting and wide open area of investigation is given by the combination of the two last features, priorities and conditions.

7. HIGHER-ORDER REWRITING

A vast generalization is obtained when we go to *higher-order rewriting*. Here an essentially new feature is encountered (as compared to first-order rewriting): that of the bound variable, already well known in first-order predicate logic in quantified assertions as $\forall x\,\phi(x)$ (all x have property ϕ) and $\exists x\,\phi(x)$ (there exists an x with property ϕ).

The paradigm rewrite system of higher-order rewriting is another classical gem: lambda calculus. But higher-order rewriting has a wider scope and also includes rewrite systems appearing in Proof Theory such as the one in figure 5. These rewrite rules 'normalize' proofs in Natural Deduction by cutting away superfluous detours. The rules take in linear notation written in the formalism of *Combinatory Reduction Systems* (which constitutes one specific format for higher-order rewriting) the form displayed in Table 4. Here the alphabet of the Combinatory Reduction System consists of two unary function symbols `inl` and `inr` (for introduction of disjunction) and a ternary function symbol `el` (for elimination of disjunction).

8. INFINITARY REWRITING

For practical purposes one is often more interested in infinite objects than finite terms and their normal form. Such infinite objects can be given by a recursive ('circular') definition such as

$$\begin{array}{lcl} \texttt{el}(\texttt{inl}(Z), [x]Z_0(x), [y]Z_1(y)) & \rightarrow & Z_0(Z) \\ \texttt{el}(\texttt{inr}(Z), [x]Z_0(x), [y]Z_1(y)) & \rightarrow & Z_1(Z) \end{array}$$

Table 4.

```
#letrecones = 1 :: ones;;
```

which denotes the infinite sequence of ones, $111\cdots$, written in CAML, a modern functional programming language. (A note on syntax: # and :: denote the CAML prompt and list constructor, respectively, ;; represents the end of a sentence.)

The crucial manoeuvre to get infinite rewriting off the ground, is the formulation of the right notion of converging rewrite sequences. Namely, we have rewrite sequences which may take more than ω steps, where ω is the ordinal just after the natural numbers. So we need to know what is the *limit* of a rewrite sequence at limit ordinals λ. It turns out that the right notion of convergence towards a limit term is the one where not only an increasing part of the term is 'crystallized out', but also in this process the depth of the rewrite activity tends to infinity at every limit ordinal, ω, $\omega.2$, $\omega.3$, $\cdots$, ω^2, $\cdots$. Figure 6 pictures this situation.

Infinitary rewriting is a point of view that can be applied to first-order rewriting, but also to the higher-order rewrite system of lambda calculus. Figure 7 displays a rewrite sequence of length $\omega+\omega$ involving infinite lambda terms. Infinitary lambda calculus has an important theoretical application: namely that of providing a semantics for cyclic lambda graph rewriting, discussed below.

9. Term graph rewriting

In recent years attention has been given to a generalization of term rewriting called *graph rewriting*. The main idea, arising from the need for efficient implementation of term rewriting, is not to duplicate subterms when rewriting, but to share subterms by using pointers to just one copy. Also cyclic graphs are allowed. Thus the *fixed point combinator* Y, embodying the possibility

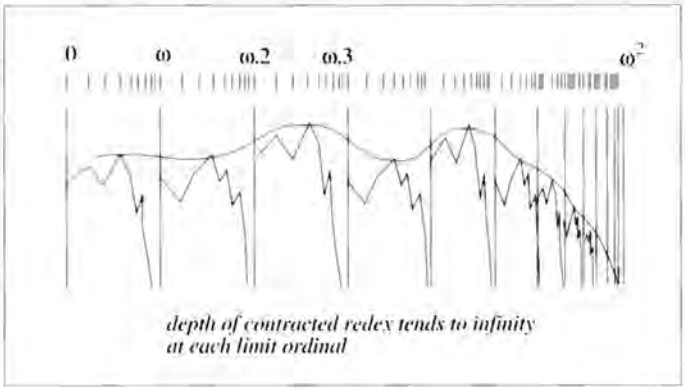


Figure 6.

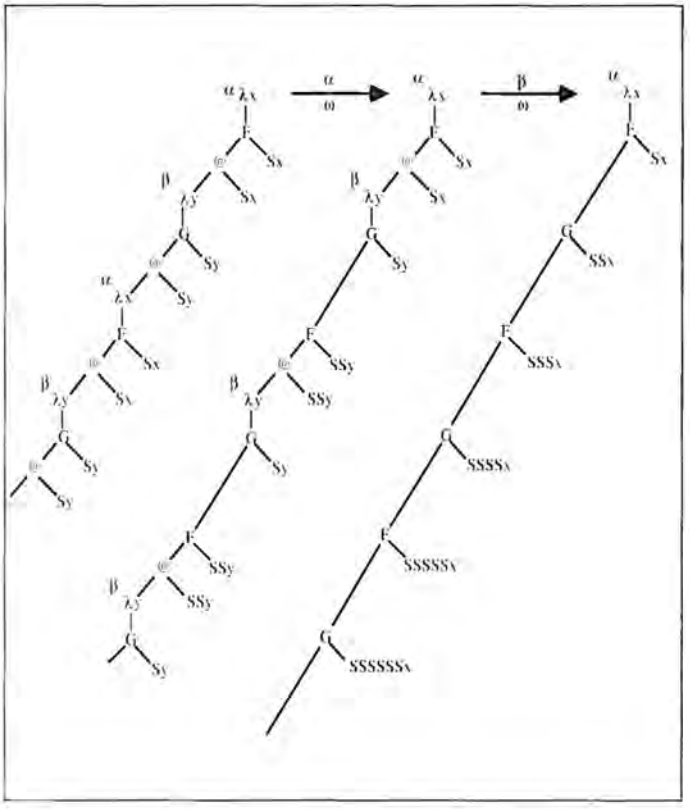


Figure 7.

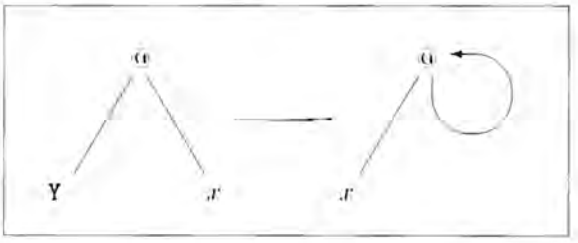
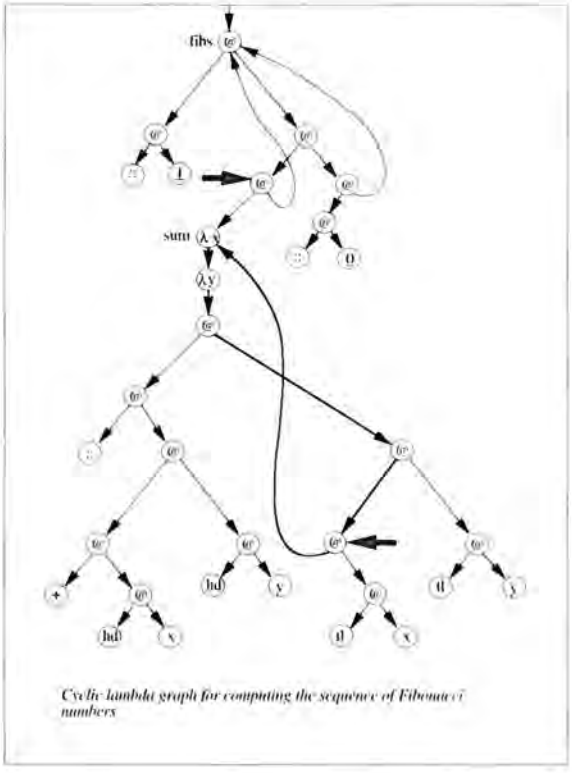


Figure 8.

of recursive definitions by means of its typical rewrite rule $\mathtt{Y} \cdot x \to x \cdot (\mathtt{Y} \cdot x)$, can be implemented in an elegant way as in figure 8. (Note that repeated application of the rewrite rule $\mathtt{Y} \cdot x \to x \cdot (\mathtt{Y} \cdot x)$ leads to the *infinite* term $x \cdot (x \cdot (x \cdot (x \cdot \ldots$, which is finitely presented by the right-hand side of the graph rewrite rule in the figure, where @ stands for $\cdot$.)

10. Cyclic lambda graphs

Not only in the realm of first-order terms cycles are important, also for lambda calculus they constitute a useful new level of rewriting. While already occurring in the practice of functional programming, the theory of cyclic lambda calculus or as we prefer to say, lambda calculus with explicit recursion, is only in development since very recent years. As an example, consider the CAML specification of the sequence of Fibonacci numbers $1, 1, 2, 3, 5, 8, \cdots$.



Cyclic lambda graph for computing the sequence of Fibonacci numbers

Figure 9.

```
# let rec fibs = 1 :: sum fibs (0::fibs);;
# let rec sum = fun x y → (hd x + hd y) :: sum (tl x) (tl y) ;;
```

Graphically, this is a cyclic lambda graph as in figure 9. (The heavy arrows point to the roots of the two 'redexes' that are present in this graph ('redex = reducible expression'). An understanding of explicit recursion in lambda calculus serves to clarify the important programming concepts of 'let' and 'letrec'.

ACKNOWLEDGEMENTS

Many thanks to Z.M. Ariola, F. van Raamsdonk for helping out with the production of this paper in various ways, and to A. Middeldorp for kindly making figure 2 (from his Ph.D. thesis) available to us.

REFERENCES

1. Z.M. ARIOLA, J.W. KLOP (1994). Cyclic lambda graph rewriting. *Proc. Ninth Symposium on Logic in Computer Science (LICS'94)*, Paris, France, 416-425.
2. J.A. BERGSTRA, J.W. KLOP (1986). Conditional rewrite rules: confluence and termination. *J. Comput. System Sci. 32(3)*, 323-362.
3. J.R. KENNAWAY, J.W. KLOP, M.R. SLEEP, F.J. DE VRIES (1995). Transfinite reductions in orthogonal term rewriting systems. *Information and Computation 119(1)*, 18-38.
4. J.W. KLOP (1992). Term rewriting systems. S. ABRAMSKY, D. GABBAY, T. MAIBAUM (eds.). *Handbook of Logic in Computer Science*, volume II. Oxford University Press, 1-116.
5. J.W. KLOP, V.VAN OOSTROM, F.VAN RAAMSDONK (1993). Combinatory reduction systems: Introduction and survey. *Theoretical Computer Science 121(1-2)*, 279-308. A Collection of Contributions in Honour of Corrado Böhm on the Occasion of his 70th Birthday, guest eds. M. Dezani-Ciancaglini, S. Ronchi Della Rocca and M. Venturini-Zilli.

A Tour of Algorithmics

P.M.B. Vitányi

1. Introduction

Computer science distinguishes itself among most other sciences in that it does not deal exclusively with a given reality (like physics) nor does it deal exclusively with a man-made ideal construct (like mathematics). Instead, computer science deals with invented ideal constructs, some aspects of which have to be realized in the physical world. Like the human body there is a physical shape and mental operations being performed in this physical substrate. The physical body corresponds with the physical architecture of the computing device, and the mental equipment and operations correspond to the various algorithms the device executes. The notions of *algorithm* and *architecture* are basic to all computer programming, and so are the *complexity* issues which arise in this context. Of course, algorithms need to be expressed in particular programming languages, and checked for correctness, much like the mind and body of a human need to be checked on appropriate function. The role of medication is filled by logics-based semantics and related fields in computer science and is not dealt with here.

2. A Brief History

The word 'algorithm' itself derives from the 9th century Persian Abu Ja'far Mohammed ibn Musa al-Khowārizmī (native of Khowārizm, today the small city of Khiva in the former Soviet Union). He is the author of a celebrated book which preserved large parts of mathematics from antiquity through

Figure 1. Part of the 9th century Arabic work 'al-jabr wa l-muqabala' by al-Khowârizmî, dealing with the solution of the quadratic equation $x^2 + 21 = 10x$.

the dark ages. Incidentally, the word 'algebra' is derived from the book's title *Kitab al jabr wa l-muqabala* which means 'Rules of restoration and reduction'. In the middle ages there was a fierce struggle between 'abacists' who calculated on the abacus or counting board, and the 'algorists' computing with pencil and paper using algorithms for addition, subtraction, multiplication and division following the teaching of al-Khowarizmi.

In his famous address to the International Mathematical Congress in 1900, D. Hilbert proposed twenty-three mathematical problems as a programme to direct the mathematical efforts in the twentieth century. The tenth problem asks for an algorithm which, given an arbitrary Diophantine equation, produces either an integer solution for this equation or indicates that no such solution exists. (In the 1970s Yu.V. Matijasevich showed that no such algorithm exists.)

The idea of a completely mechanical procedure, an *algorithm*, to find solutions to mathematical questionsgoes back at least to Hilbert. In 1931 K. Gödel proved that not every true mathematical statement is provable in a finitely axiomatized system of mathematics. With the purpose of identifying fundamental ideas immanent in Gödel's proof, in 1936 A.M. Turing exhibited an exceedingly simple type of hypothetical machine and gave a brilliant demonstration that everything that can be reasonably said to be computed by a human computer using a fixed procedure can be computed by such a machine. As Turing claimed: any process which can be naturally called an effective procedure is realized by a Turing machine. This is known as *Turing's Thesis*. Over the years, all serious attempts to give precise yet intuitively satisfactory definitions of a notion of 'effective procedure' have turned out to define essentially the same class of processes. (In his original paper, Turing established the equivalence of his notion of 'effective procedure' with A. Church's notion of 'effective calculability'.)

Church's Thesis states that, in this sense, there is an objective notion of effective computability independent of a particular formalization. According to Gödel: 'With this concept one has for the first time succeeded in giving an absolute definition of an interesting epistimological notion, i.e., one which does not depend on the formalism chosen. In all other cases treated previously...one has been able to define them only relative to a given language, and for each individual language it is clear that the one [definition] thus obtained is not the one looked for... This situation [according to Church's Thesis] is some kind of miracle.'

Thus, theoretically any formally computable function is computable by a laptop computer with indefinitely expandable memory or by a Turing machine, however clumsy the latter may be. But a computation that takes 2^n steps on an input of length n would not be regarded as *practical* or *feasible*. No computer would ever finish such a computation in the lifetime of the universe even with n merely 1000. Computational complexity theory tries to identify problems that are feasibly computable.

> If we have 10^9 processors taking 10^9 steps/second, then we can execute $3.1 \times 10^{25} < 2^{100}$ steps/year.

This shows that in practice the relevant question is whether a computation is also *feasible*. To characterize this notion of feasibility J. Edmonds in 1965 proposed a classification of computational problems in terms of polynomial time bounds on the length of the computation.

A problem is in the compexity class P if it can be solved (the answer is 'yes' or 'no') in time polynomial in the input length, and in NP if it can be solved by a so-called 'nondeterministic algorithm' in polynomial time. Informally speaking, P is the set of 'yes-no' problems where it is easy to find the answer (easy: doable by a deterministic Turing machine in polynomial time), and NP the set where it is easy to show that the answer is 'yes'.

Normally, we do not ask questions unless we can easily recognize the good answer. NP is about those questions that we are likely to want answers to.

The question: $P = NP?$ is possibly the most important problem in computer science if not in mathematics. Attempts to resolve this question have thus far mainly led to reformulations or reductions. For example, the difficulty of the entire problem class NP has been reduced to a nucleus of so-called 'NP-hard' problems (S. Cook and R. Karp, 1971; L. Levin, 1973).

These notions are tied to sequential computation such as performed by a Turing machine or a Von Neumann architecture computer. However, in the past years hopes have emerged that nonclassical or nonstandard physical realizations of computers may have different properties that may help in beating the NP barrier.

Numerous computer developments together with an ever-increasing complexity of the problems handled by computers, produce challenging demands

requiring the invention of new architectures for emergent computer technologies and more efficient algorithmic designs. Research questions cover the design, construction and use of hardware, as well as applications. Solutions to these problems are sought via improved networks and parallel architectures, partially through exploitation of opportunities arising in novel applications of physics phenomena, in combination with efficient algorithms. For an overivew, see for example D. Knuth's *The Art of Computer Programming* Series published by Addison-Wesley. In this article we trace CWI-based research in this area.

3. Past Highlights

3.1. Machine complexity

In machine complexity we are interested in the variation of computing power resulting from variation of machine parameters. Three well-known open basic problems were resolved as follows.

1. It is possible to real-time simulate a fixed finite number of independent counters on-line by a one tape Turing machine, [4].

2. It requires n^2 steps to simulate n steps of a k-tape Turing machine by a one-tape Turing machine, and many such results, were discovered independently by M. Li, W. Maass and P.M.B. Vitányi. This matches the trivial upper bound and improves the previous best lower bound of $n\sqrt{\log n}$ by almost an order of magnitude.

3. For over 30 years it was conjectured that two heads on the same work tape (of a Turing machine) are more powerful in real time than two work tapes with one head each. Vitányi published in 1984 a preliminary lemma, aimed at eventually proving this result, which was accomplished very recently by T. Jiang, J. Seiferas, and Vitányi.

3.2. Computational number theory

A.K. Lenstra (then at CWI), H.W. Lenstra, Jr., and L. Lóvasz [1] showed that factorization of polynomials over the radicals into irreducible factors can be performed in deterministic polynomial time. Later, the link between cryptography and computational number theory was pursued at CWI by E. Kranakis (who authored the first monograph on public key cryptosystems while at Yale University), and by D. Chaum, who founded CWI's group on cryptographic research. (See also H.J.J. te Riele's article in this volume.)

3.3. Distributed and parallel computing

Around 1985 attention shifted from sequential computing to distributed and parallel computing. Distributed computation is related to the emergence of

computer networks: computer applications moved from single stand-alone mainframes to multiple communicating local workstations. Parallel computation arose from the quest of fundamentally improving the speed of sequential computation by using multiple processing units. Both fields generate questions of architecture of physical interconnects and topologies, and concurrent algorithms for control of interprocessor communication and applications. We give a selection of CWI related research.

1. The common approach towards synchronicity issues of multiprocessor systems was to assume that either the processors were totally synchronized or totally unsynchronized. We pioneered an approach in between: 'Archimedean time systems', which is more realistic in terms of real-time issues.

2. Many communication issues in multicomputer systems such as mutual exclusion, name server, load balancing, data integrity, voting systems, and so on, have a mutual underlying core which was identified and analyzed by S.J. Mullender and Vitányi as 'distributed match-making'.

3. A basic primitive for asynchronous interprocess communication was identified by L. Lamport as wait-free read-write shared register. He constructed the single user case. For multiple users the problem of implementing such shared memory primitives from basic available electronic components or software components becomes very difficult and possibly *a priori* impossible. Vitányi and B. Awerbuch developed the appropriate theory and gave a basic implementation now known as the Vitányi-Awerbuch register, [6]. To settle the theoretically interesting question whether such a construction can exist using only a bounded number of control bits, after several published erroneous solutions by several researchers, M. Li and Vitányi (later joined by J.T. Tromp) gave the first uncontested solution.

4. In a sequential computation one can safely ignore many physical aspects of the underlying computer system and analyse the computational complexity of a program in a purely logical fashion. This is not the case in nonsequential computation. Moreover, nonclassical or nonstandard physical realizations of computers may have totally unexpected properties. A popular model to analyse parallel algorithms is the parallel random access machine (PRAM), where many processors can read and write a single shared memory in unit time per operation.

   In fact, optimality of PRAM algorithms may be misleading, because in any physically realizable machine architecture a much simpler and unsophisticated algorithm may outperform the optimal PRAM algorithm. Do networks help with this problem? We can simulate PRAMs

fast by networks of processors communicating by message passing at the cost of a multiplicative slowdown square logarithmic in the number of processors n for simulation on a $\log n$-dimensional hypercube. However, this does not solve the problem mentioned above, since the hypercube nodes need to be order $n^{1/3}$ apart for the majority of pairs (see below). Together it turns out that rather than saving time, the simulation costs at least a logarithmic in n factor more time than the original. R. Landauer at IBM T.J. Watson Research Labs has emphasized that 'information is physical'. So is communication. We have analyzed the real physical aspects of proposed computer architectures through a sequence of papers debunking many popular misconceptions about models for parallel computations.

4. Kolmogorov Complexity and the Incompressibility Method

In parts of the research mentioned above, we and our collaborators developed a new mathematical proof technique now known as the *incompressibility method* – a basic technique such as the 'pidgeon hole' argument, 'the counting method' or the 'probabilistic method'.

The new method is based on so-called Kolmogorov complexity, a modern notion of randomness proposed by A.N. Kolmogorov in 1965 to quantify the randomness of individual objects in an objective and absolute manner. This is impossible by classical probability theory (a branch of measure theory satsifying the so-called Kolmogorov axioms formulated in 1933). Likewise, the Kolmogorov complexity of an object is a form of absolute information of the individual object. This is not possible to do by C. Shannon's information theory, since the latter is only concerned with the average information of a random source.

After we pioneered several successful applications of Kolmogorov complexity in the theory of computation, the general pattern of the incompressibility method emerged. It is a sharper relative of classical information theory and yet satisfies many of the laws of classical information theory – although with a slight error term.

Applications of Kolmogorov complexity by us and others have been given in a plethora of areas, including the theory of computation, inductive reasoning, formal language theory, computational learning theory, combinatorial theory, randomness, Gödel style incompleteness results, graph theory, Kolmogorov 0-1 Laws, theory of parallel and distributed computation, average complexity, sorting, string matching, routing in computer networks, circuit theory, complexity of tapes, stacks, queues, complexity of parallel random access machines, in physics of computation, information distance (for example in pattern recognition) and so on. A comprehensive account of both theory and applications is given in the (text)book [2].

5. Current and Future Directions

Current research at CWI is in the direction of machine learning and physical aspects of computation, while continuing work in communication infrastructure in parallel and distributed computation. The incompressibility method and other compression based techniques are used throughout.

5.1. Computational Machine Learning

It is not always realized that most of traditional statistics is about computational learning. It is always involved with algorithms to obtain the general from the particular. A novel approach in statistical learning is based on the minimum description length of hypotheses and data together—one way to express the so-called MDL principle.

In our work we follow the thread of inductive inference and pac learning and, from the other end, we examine an approach related to statistics, Bayesian reasoning, and the principle of 'minimum description length' or 'MDL' for short. It appears to us that the future of computational machine learning will involve combinations of these approaches coupled with guaranties with respect to used time and memory resources. It is clear that computational learning theory will move closer to practice and the application of principles such as MDL requires further justification. Building on our earlier work, as reported in the textbook [2], we can justify certain applications of MDL via the Bayesian approach as follows.

Figure 2. A.N. Kolmogorov (1903-1987) at the 1954 International Mathematical Congress in Amsterdam.

A general task of statistical learning is to select the most plausible hypothesis in the light of experimental evidence. The classic method to do so is Bayes' rule. The problem with applying Bayes' rule is that one first requires the prior probabilities of the possible hypotheses. Unfortunately, it is often impossible to obtain these.

One way out of this conundrum is to require inference of hypotheses to be completely data driven. The MDL approach embodies this idea. This approach is currently widely and successfully used in many diverse applications. MDL is usually presented as justified in and of itself by philosophical persuasion. As one of the founders, C. Wallace, remarked at an AAAI meeting at Stanford University in 1990, 'the most surprising thing about MDL is that no monster has yet sprung from the woods'. That is, the principle has not yet known spectacular failures in practice. It is important for theoretical foundation and for practical application that a firm basis for the principle is established.

In [3] we supply a rigorous justification for MDL from first principles, identify similarities and differences with Bayesian inference, and give a comparison of pac learning criteria and MDL algorithms for the practical topic of decision tree learning.

5.2. Computational Linguistics

Given a body of text (a *corpus*), we want to automatically derive a grammar for it. While systems doing so will produce unfamilar grammars for, say, natural language texts, in other contexts we do not know or care what is 'natural'. For example, the grammatical rules of entries in the *Oxford English Dictionary* are poorly described and unknown. Yet automatically extracting rules from existing texts yields a grammar which can monitor correct format of new entries. In the *Human Genome Project* an enormous corpus of genetic data has been collected and is available in data banks. Extracting a grammar from these data can be used to validate or reject hypotheses on this material. The approach we use is primarily based on statistics of pattern conjunctions. This is used to generate the grammatical syntax rules.

Generally, the main problem of such approaches is how to judge relative goodness of alternative possibilities and similarly when to stop complicating the grammar to obtain a better fit with the data. We anticipate that using the MDL principle in this essentially data-driven process is the right thing to do. Our initial results seem to confirm this idea.

5.3. Multiple Computing Agents

Computational approaches based on the biology of the 'brain' and 'evolution' comprise research in areas of neurocomputing and adaptive computing. The employed programming techniques are referred to as 'multiple

computing agents'. Such approaches turn out to be very useful to apply to ill-defined problems which can only with great difficulty be expressed in conventional algorithmics, such as for example problems of computer vision and speech recognition.

Neurocomputing deals with the design, analysis and application of networks built from artificial neurons. Here the goal is not to imitate the human brain but to design a functional computing engine.

Genetic algorithms solve optimization problems in a way which is based on natural selection in biology. A population of possible solutions (programs) must converge in a short time to yield the best (or a very good) solution. Even more than in neurocomputing, we can talk about 'automatic programming' in this setting. The program develops by itself, governed by 'evolutionary fitness and selection' (therefore, one also talks about 'evolutionary' or 'genetic' programming). This approach is successful in contexts where it is almost impossible to write explicit programs. This is very often the case in practice. Genetic programming appears to be a popular approach with software houses, since the result is an optimal program which can be 'explained' to the client, rather than a set of optimal weights in a neural network which may perform great but have no direct intuitive explanation. At CWI we have developed a pilot implementation of novel evolutionary programs based partially on the genetic programming paradigm, with more capabilities than hitherto known (FALS), which will be extensively tested on real problems.

Automatic programming with genetic algorithms and the like for providing near-optimal solutions to ill-defined problems is widely used by software developers in commercial applications. For example, it is used by banks to judge credit card applications and by KLM to predict career planning of its pilots.

The method has become so popular because it is relatively easy to program and leads in practice almost invariably to very good performance. The reasons for this are mathematically still poorly understood. Because of its immense commercial impact it is paramount that the mathematical underpinnings of this discipline are discovered and performance guaranties can be given, and, moreover, that parameters which speed up convergence and improve quality of solutions are identified.

Our investigation in the underlying mathematical theory is based on Markov processes and focuses on the analysis and exploitation of 'rapid mixing' properties. Preliminary results point at a mode of operation where many short runs of the genetic algorithm are more likely to yield an optimal program than one very long run – in contrast to current usage. Another direction of work concerns the speed-up of convergence to high quality solutions by application of the Bayesian approach and the MDL principle.

5.4. Routing

In computer networks routing of messages (much like email over internet) is a vital item. As networks grow larger, routing information present at each particular site increases to unmanageable size. Clearly, it is sufficient to maintain a routing table at each node which says over which adjacent node a message to a target node must be routed. However, such a method requires routing tables in all nodes of size $n \log n$ and the question is how to route messages using as compact as possible routing tables. A typical method is *interval routing*. Adjacent nodes are first ordered (for example lexicographically by name), and then a set of intervals on the set of nodes, say $\{1, \ldots, n\}$, is assigned to each such adjacent node. The intervals are chosen such that together they cover $\{1, \ldots, n\}$ completely. To route a message to any target node, we first look for an interval containing the target node, and then route the message to the adjacent node associated with the interval. Clearly, for certain networks (like trees) interval routing can be very efficient in the sense of saving a lot of bits in the description of the routing information.

We use Kolmogorov complexity to determine the optimal space used by routing tables in communication networks for both worst-case static networks and on the average for all static networks. This resolves the problem of the necessary and sufficient size of all routing tables together for unrestricted routing schemes. Similarly we determine the optimum routing table size for shortest path routing on almost all graphs (the Kolmogorov random graphs which constitute a fraction of at least $1 - 1/n^3$ of all graphs). We prove that $\Theta(n^2)$ bits are sufficient and necessary for the total size of the routing tables. We show that this implies the same optimum for the average of all graphs. It turns out that our methods are applicable to many different current models used in applied routing.

5.5. Quantum Coherent Computation

New computation devices increasingly depend on particular physical properties. The theory of computation is thus becoming an increasingly interdisciplinary subject, because of the need to understand and apply physical laws in computational considerations.

Quantum coherent computation (QCC) is a new field of research that has attracted considerable attention over the last 10 years (see for example the article by S. Lloyd in the Scientific American, October 1995). Recent evidence that the proposed coherent quantum computers may be intrinsically much faster than classical computing devices makes their technological development of great economic interest. QCC may contribute to solving standing open problems in computation theory as well as increase our understanding of quantum phenomena.

The ultimate limits of the speed of computation are determined by the

energy dissipation per unit area per unit time, which increases with the density of switching elements in the computing device. Linear speed up by shortening interconnects on a two-dimensional device is attended by cubing the dissipated energy per area unit per second, and dissipation on this scale in the long run cannot be compensated for by cooling. The dissipated energy per bit operation at room temperature has decreased from 10^{-2} Joule in 1940 to 10^{-17} Joule at present, and extrapolation of current trends shows that within twenty years a further reduction below the thermal noise level of 10^{-21} Joule is required. Reducing the energy dissipation of a computation is very relevant in the development of massive multiprocessing architectures [5], where the interconnect volume essentially is a square or cube power of the processor volume. The requirement of (almost) dissipationless computation has led computer scientists to consider 'reversible' computation, for which there is no lower bound on the energy dissipation. It has been established that the functionality of reversible classical logical gates forms a proper subclass of the functionality of quantum logical gates. An efficient methodology of quantum gate construction for n-bit unitary transformations is still a subject of current research.

Since R. Feynman's statement that quantum mechanics does not impose a physical limitation on computation, the question whether quantum-based computation can be more efficient than classical (probabilistic) computation, has subsequently become an object of intensive research. This has led to a *nonclassical* emergent possible computer technology (quantum coherent computation or QCC).

The QCC approach will partially alleviate the interconnect problem (above) because a large number of different computation paths can be simultaneously followed (with appropriate probability amplitudes, to be sure) by the same single physical apparatus requiring but a tiny amount of physical space. This is the substance of Feynman's dictum 'there is room at the bottom' in the context of his proposal of QCC. Of course, since the different computation paths of a quantum computation cannot communicate as is often a main feature in a parallel distributed computation, it is only a very special type of room which is available at the bottom.

In 1994 P. Shor showed a remarkable result, which suddenly made the physical realization of a quantum computer extremely interesting. Shor provided a quantum algorithm that could do integer factoring (or discrete logarithm) with a bounded probability of error in polynomial time. On a classical computer with currently known methods the determination of prime factors can be an exceedingly difficult (exponential time) problem, although verification is trivial. This asymmetry is the basis of modern cryptography and is used to obtain secret codes used in your bank card and to transfer diplomatic messages between embassies.

Physical implementation of quantum algorithms, like Shor's, will however

face great problems. The power of quantum computation is related to the use of coherent quantum superpositions in the computation. This coherence, however, can be destroyed by the inevitable coupling to the 'environment'. W. Unruh has calculated that QC'C computations using physical realizations based on spin lattices will have to be finished in an extremely short time. For example, for factoring a 1000 bit number in square quantum factoring time we have to perform 10^6 steps in less than the thermal time scale h/kT which at 1 K is of order 10^{-9} seconds.

Another problem is *error correction*: measurements to detect errors will destroy the computation. A novel partial method for error correction has been suggested by A. Berthiaume, D. Deutsch and R. Josza.

It will require an exploration of several candidate physical systems to obtain a more precise idea of the possibilities and limitations of the physical implementation of a quantum computer.

References

1. A.K. Lenstra, H.W. Lenstra, Jr., L. Lovász (1982). Factoring polynomials with rational coefficients. *Math. Ann. 261*, 515-532. See also: A.K. Lenstra, H.W. Lenstra, Jr. (1990). Algorithms in number theory. In: *Handbook of Theoret. Comp. Sci.*, J. van Leeuwen (ed.), MIT Press/Esevier, Amsterdam.
2. M. Li, P.M.B. Vitányi (1993). *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York.
3. M. Li, P.M.B. Vitányi (1995). Computational Machine Learning in Theory and Praxis. In: *Lecture Notes in Computer Science, Vol. 1000*, Springer Verlag, Heidelberg. (Invited paper)
4. P.M.B. Vitányi (1985). An optimal simulation of counter machines. *SIAM Journal on Computing 14*, 1-33. See also: J. Seiferas, P.M.B. Vitányi (1988). Counting is easy. *J. Assoc. Comp. Mach. 35*, 985-1000.
5. P.M.B. Vitányi (1995). Physics and the New Computation, Prague, August 1995, Proc. 20th Int. Symp. Math. Foundations of Computer Science, MFCS'95, *Lecture Notes in Computer Science, Vol 969*, Springer-Verlag, Heidelberg. (Invited paper)
6. P.M.B. Vitányi, B. Awerbuch (1986). Atomic shared register access by asynchronous hardware. *Proceedings 27th Annual IEEE Symposium on Foundations of Computer Science*, Errata, FOCS'87, 233-243.

An Electronic Wallet for Digital Money

R. Hirschfeld

1. Background

The last quarter century has witnessed explosive growth in the technology for automated handling of information. This has resulted in many conveniences, but has also introduced new dangers, such as increasing opportunities for unauthorized access to sensitive or personal data, and for tampering with such data. With the advent of electronic commerce has come the need for electronic money, i.e., a digital representation of cash. Electronic money introduces additional associated security problems, such as forgery of money, respending the same money, and invasion of privacy of people's spending habits. The field of cryptography has addressed the problems of authentication and data security in general and of the security of electronic money in particular.

CWI has been conducting research in cryptography for over ten years, and is an internationally recognized leader in the areas of digital signatures and public-key cryptographic protocols. Early emphasis was placed on the security of ordinary users of automated systems, and particularly on the protection of individual privacy. CWI coordinated the European RACE project RIPE, which developed and evaluated a collection of cryptographic primitives. The company DigiCash, founded by D. Chaum and specializing in systems for consumer payments, has its roots in the CWI crypto group. In recent years, research has focused both on proofs of security of crypto-

Partner	Description	Country
CWI	National research institute for mathematics and computer science	The Netherlands
CardWare	consultant for the financial industry on electronic payment technologies	United Kingdom
IFS	social research institute	Germany
Gemplus	smart card manufacturer	France
DigiCash	software and hardware designer	The Netherlands
Ingenico	point-of-sale terminal manufacturer	France
Siemens	industrial electronics manufacturer	Germany
SEPT	national telecommunications and postal research institute	France
Katholieke Universiteit Leuven	university	Belgium
Royal PTT Nederland, PTT Research	national telecommunications	The Netherlands
SINTEF DELAB	university research institute	Norway
Aarhus University, Mathematics Institute	university research institute	Denmark
University of Hildesheim, Informatics Institute	university research institute	Germany

Figure 1. The CAFE Consortium.

graphic protocols and on techniques for achieving secure protocols that can be efficiently implemented and practically realized.

The project CAFE, which is carried out by a consortium of thirteen European institutions (see figure 1) and is funded by the European Commission's ESPRIT programme, has applied modern cryptographic techniques to produce a highly secure but also open and flexible system for consumer payments using electronic money. As a leader in theoretical research on electronic money and as coordinator of the CAFE project, CWI has played a major role in the development of the protocols used by the system.

2. Electronic payments

2.1. Introduction

Europe leads the world in the introduction of smart cards, wallet-size cards with embedded computer chips. French bankcards and the telephone cards of several countries are chip-based. These early chip cards do not really

merit the name 'smart' —they contain memory chips and work in essentially the same way as magnetic stripe cards except that they are more difficult for a counterfeiter to overwrite. More recent systems incorporate microprocessor chips, so that the card can participate actively in transaction protocols. This allows the implementation of digital signature techniques, upon which electronic money is based.

An application such as a telephone card, in which the card issuer is the same as the service provider (so no clearing is necessary), and all points of payment are online, does not really require a sophisticated microprocessor on the card. But the availability of smarter cards has led to the introduction of prepaid electronic purse systems, in which value is stored on a card and can be used for payment at a variety of shops and other service providers. Because these transactions can be completed offline, they are suitable for small value payments for which cash is traditionally used; credit and debit cards require costly online authorization, which is infeasible for low-value payments. Electronic purse systems are undergoing trials in several European countries; the Dutch banks have recently introduced one in The Netherlands based on a system developed and currently under trial in Belgium.

2.2. Security

The security of electronic purse systems is based on digital signatures, a cryptographic method of certifying the origin of a digital message. Messages (which can represent banknotes or card balances) are signed by an algorithm that uses a secret key provided by the signer, and authentication is performed by another algorithm that also uses a key. In most of the electronic purse systems underway, the signing key and the authenticating key are the same. Because the authenticating key present at the point of sale could also be used to sign messages (i.e., create money), it is protected against discovery and possible misuse by storing it in a tamper-resistant hardware module. This reduces the flexibility of the system; it is difficult to combine multiple issuers of electronic money into the same system, and security modules cannot be given out indiscriminately, but only to service providers who can be trusted not to try to compromise them.

An alternative to symmetric systems that use the same key to create and to verify a signature is public-key digital signatures. In a public-key system, the signing and authenticating keys are different, and no knowledge of the signing key can be obtained from the authenticating key. Such an asymmetric system is ideal for an open and interoperable electronic purse environment, because the signing key need be known only to the issuer, and the authentication key can be made public and need not be protected in any way.

Despite their advantages, public-key signatures are not yet widely used in

One of the fundamental notions of modern cryptography is that of a one-way function—a function that is efficient to compute but impractical to invert. For example, it is easy to multiply large integers, but is thought to be difficult to factor a large integer product into its constituent prime factors. Similarly, modular exponentiation is relatively efficient, but its inverse, the discrete logarithm, remains intractable. Although no proof of the difficulty of either of these number-theoretic problems is known, they have thus far resisted all attempts at efficient solutions. Many cryptographic protocols are based upon assumptions of their intractability.

A related notion is that of a one-way trapdoor function, which is normally difficult to invert, but which becomes easy to invert if some additional information is known (this is the trap door). This additional information can form the secret key (or part of the key) in a cryptosystem. An example is the well-known RSA cryptosystem (named after its inventors: R.L. Rivest, A. Shamir, and L. Adleman, see figure 2), which is based on the observation that if $n = pq$, the product of two large primes, then computing powers mod n is easy but computing roots mod n seems difficult unless the factors p and q are known, in which case it is easy. Cryptosystems were originally developed for encryption rather than signing of messages. In general, a public-key cryptosystem consists of a public encryption algorithm E and a secret decryption algorithm D with the property that for a message m, $D(E(m)) = m$. Each person has their own pair of encryption and decryption algorithms. To send a secret message, the sender encrypts it using the recipient's E (which is publicly available), and then only the recipient can decrypt it, using his own secret D.

Figure 2. The inventors of the RSA cryptographic method. From left to right: R.L. Rivest, A. Shamir, L. Adleman. Photo courtesy RSA Data Security Inc.

Digital signatures turn this situation around. The secret algorithm D is used to sign messages, and the public algorithm E is used to authenticate them. The property required is that $E(D(m)) = m$. To sign a message, the signer applies her own secret algorithm D, and then anybody can verify it using her publicly available E. The RSA cryptosystem can be used for both encryption and for digital signatures because it has the property that $D(E(m)) = m = E(D(m))$, since encryption and decryption are the inverse operations of raising to a power and extracting a root.

Commercially, the most widely-used cryptosystem is the Data Encryption Standard (DES). This is a symmetric algorithm and is unsuited for public-key use, but has the advantage that it does not rely on time-consuming modular arithmetic. As hardware support for these operations becomes more widely available and inexpensive, however, this is becoming less of a consideration.

Figure 3. The CAFE infrared wallet (left) and card (right). Courtesy CAFE Project.

commercial systems. This is because they require more elaborate computation, and until recently performing this computation on a smart card chip has been too slow or too expensive. But with the development of specialized cryptographic smart card chips, which implement complex operations needed for public-key signatures in hardware, public-key electronic purses are now feasible. The CAFE project has developed a public-key system for electronic purses, and, more generally, for electronic wallets, which combine an electronic purse with other applications, such as digital passports, driver's licenses, house keys, etc.

In addition to a smart card, CAFE has developed a hand-held wallet that communicates with payment terminal via infrared (much like a television remote control (see figure 3), except bidirectional) and allows consumers to confirm payments with their own device and to complete the payment without the wallet ever leaving their hands.

3. CWI's role

CWI was the coordinating partner of the CAFE project, and as such was responsible for the overall management of the project. In addition, CWI played a major role in the design of the protocols used in the system, drawing

on its many years of active research experience in digital signatures and electronic money.

In collaboration with the other protocol partners, CWI worked on all aspects of the cryptographic protocols developed for CAFE, including not only the fundamental protocols for secure transactions (withdrawal, payment, deposit, etc.), but also currency exchange, tolerance of loss and faults, key management, and other related items.

CWI has also applied its special expertise on privacy protection and user-moderated transactions in the design of the wallet protocols and the provision of untraceability as a system option.

4. Future plans

In 1995, the CAFE system began a trial on the premises of the European Commission in Brussels. If successful, this will expand to include other EU institutions in other cities and perhaps the surrounding communities. Although the currency exchange mechanism developed by the project is very general, the trial will focus on the introduction of an electronic ECU. Users will load their cards with any combination of their home currency and ECU, and (in the trial) they will be able to spend their home currency only in their home country, but the electronic ECU at a CAFE terminal in any country. The results anticipated include an evaluation of the technology, surveys of user opinions, and a cost assessment and business case analysis of the system.

After the end of the CAFE project, the trial will be expanded as part of a separate project to the premises of some of the sponsoring financial institutions, in different countries. This will provide a real test of the international capabilities of the system. A follow-up project with large-scale pilots of both the electronic purse and other applications is in the proposal stage.

CWI is also participating in a new European project called SEMPER, which is developing secure mechanisms for payments and other marketplace activities on computer networks, both the Internet and advanced high-speed networks. This project will include trials focused on multimedia applications.

By continued involvement in development projects, the CWI crypto group enriches its research scope. At the same time, research continues into the theoretical underpinnings of these applied systems, which is really the group's fundamental core.

Interactive Mathematical Books

A.M. Cohen, L.G.L.T. Meertens, S. Pemberton

1. Introduction

Recent developments in information technology—hypertext techniques, multimedia facilities, CD-ROMs, networking—as well as the increasing affordability of sufficiently powerful computers with high-resolution graphical displays, have greatly improved the possibilities of using computers for making knowledge accessible. Nevertheless, the prevalent way of learning about some advanced subject is still through the reading of books, whether or not aided by a human teacher who serves as an expert ready to explain those bits that you did not grasp from reading the book.

In classical Computer-Aided Instruction the assumption is that there is a given syllabus, a well-delineated body of knowledge, that students are supposed to master, whether motivated or not. Accordingly, there is generally a fixed route through the material, with strong emphasis on testing if the student has indeed mastered a morsel of knowledge, typically with a dialogue controlled by the automated instructor.

In contrast, our assumption is that the reader, in consulting an interactive book, *is* motivated to learn. Thus, the *book* metaphor is essentially more apt than the *instructor* metaphor. In a paper book, the reader can just browse, instead of reading it from cover to cover, consult it in any haphazard order, skip any sections, examples, exercises, etc., and come back to any point at any time. To call an interactive system an interactive *book*, not only should there be content prepared by an author for readers, but we also hold

the tenet that *all interaction is initiated, and in general controlled, by the reader.*

Interactive books are not superior to more traditional books in all aspects. For the time being, they will be less easy to carry around. The resolution of affordable screens is an order of magnitude less than that of high-quality printing, which makes it tiring to read the fine print in mathematical formulae like '$n^{p'}$'. Also, the initial investment in acquiring the necessary equipment is sizable, and the initial production cost is appreciably higher (although, on the other hand, the unit reproduction cost of the book texts themselves is very low).

2. An example

A book's interactive potential has to be substantive before it offsets the disadvantages. As an example of the kinds of possibilities an interactive book can offer let us consider a cookbook (most of the sophisticated interactive 'features' mentioned below for this cookbook can be found in one or more commercially available interactive-cookbook programs).

2.1. Navigation support

Navigation comprises 'standard' hypertext facilities, which can be used both for unfolding/folding portions of a hierarchically structured text (as in what is called *outline processing*) and for excursions to related subjects, as well as browsing and search facilities.

Unfolding/folding can be used for quick navigation through a table of contents. In a cookbook, clicking on the first item of the following list

- Meats
- Fish and Shellfish
- Vegetables

will cause this line to 'open up', so that the screen displays

- Meats
  - * *Beef*
  - * *Pork*
  - * *Veal*
  - * *Lamb*
  - * *Poultry*
- Fish and Shellfish
- Vegetables

Repeating this process on a well-organized table can quickly lead to any desired destination.

A beginning chef might not know the meaning of some technical term, say '*sauté*'. Clicking on such a term will display an explanation. An alternative possibility is to 'macro expand' the sentence in which the term occurs. For example, '*Sauté the slices of liver*' is expanded into '*Cook the slices of liver in a small amount of cooking fat or oil until brown and tender, using a skillet on medium fire and stirring well*'.

Note that the extent of navigation facilities need not be confined to the book itself: the book proper may be a coordinated collection of nodes in a larger web of active knowledge. This is of obvious relevance to mathematics (as well as many other fields of knowledge).

2.2. Annotation facilities

As in a paper book, readers may want to highlight portions, or to jot down remarks, add references, and so on. Unlike a paper book, these are reversible actions, so that the reader need not feel the anxiety of possibly spoiling the book. Possible annotations in a cookbook could be a suggestion which wine to serve with a dish, or marks for excellent or disappointing dishes. Cooks can also add their own favorite recipes to the cookbook, right where they belong in its organization.

2.3. Tailoring to user need

Given the interests and needs of the reader, the book can present different versions of its text. An interactive cookbook can give the measures in recipes in pints and ounces, or in liters and grams, according to user preference. It can also present the actual quantities needed for the expected number of people, where a paper cookbook leaves it to the chef to do a multiplication and division, the latter by a number you can never find in the book when you need it. Likewise, the number of calories (or joules) can be computed and presented. The recipes presented can be filtered on the availability of ingredients or the time available for cooking. The book can even turn into a specialized cookbook for certain dietary restrictions: a vegetarian cookbook by omitting meat-based recipes and doing some standard substitutions; or a cookbook trimmed down to low-calorie recipes for people who are minding their waist.

There are more exotic possibilities. Given a menu (a collection of recipes, one for each course) the book can present a merged and sorted shopping list. Also, it can merge the preparation steps of the recipes for the various dishes into one time axis and produce one combined list of instructions, so that a cook can work in parallel on several dishes without ever having to jump to and fro between different recipes.

2.4. Expert assistance

An interactive cookbook can also assist a cook more actively. While going through the actual preparation, it can highlight the next step to be done. When the reader clicks the DONE button, it highlights the following step. In addition, it can serve as a timer and tell our cook when to lower the heat, etc. If things do not go quite as smoothly as planned, it can rearrange the steps on the time axis in the culinarily most appropriate way. Finally, if something goes really wrong, it could offer expert advice on the best way to save the meal—with '*Call* 1-800-DIAL-A-PIZZA' as the last resort.

Expert assistance can also be helpful in composing a menu. In its simplest form, the user composes a tentative menu, whereupon the book reacts with an evaluation, using basic rules about the desirable variation in a menu as well as 'common-sense' and user-supplied constraints. In a more advanced setup expert assistance can be a truly interactive affair, with the book also suggesting modifications, and using the user reaction to those to adjust the constraints.

This does not exhaust the possibilities; neither in general (for example, we have not mentioned setting bookmarks), nor those specific to cookbooks (for example, planning ahead to use parts of a preparation for a later day, or giving advice what to do with leftovers), and, in fact, the possibilities are limited more by our imagination than by technical constraints.

3. Interactive mathematical books

A mathematical textbook is not a cookbook. Yet, the reader will have few problems in transforming many of the examples given above of how interactivity can be helpful in a mathematical context. Although the examples were made concrete with subject-specific detail, they reflect more general principles of the kind of support that is helpful to human beings when approaching a complex task, in particular in an area that they have not yet mastered. Thus, it might seem that any mathematical aspects are only in the content of the book, and not in the medium, just as for traditional books.

Yet, there is something unique to the mathematical world: it is a formal world that does not of itself refer to the outside world. Any such references, as in mathematical physics, are externally imposed interpretations on symbols that by themselves have no meaning grounded in reality.

To learn to cook, the aspirant cook has to get hands-on experience and stir the pots and pans. No amount of reading, however interactive, can fully replace that experience. In mathematics, the 'real' world is the mathematical world 'in here': if some piece of mathematics can be done at all, it can—in principle—be done here and now, in the medium of the book. This opens unique possibilities for active reader involvement. For instance:

- *Navigation support.* Half of mathematics consists of giving definitions, and the other half of applying definitions. So the number of defined notions that are used is consistently high, and, more than in any other subject, the precise formulation of the definition counts. Thus, the ability to look up the definition of a mathematical term or symbol at the click of a button is especially important. Defined notions usually have arguments. For example, 'is continuous' takes a function as argument, as well as some arguments that are usually implicit, namely the topologies of the domain and codomain. It should be possible to view an unfolded application with its actual arguments in place—including the implicit ones, which may become important if the definition is unfolded further.

  Often not all mathematical knowledge used will be contained in the current interactive book. Thus, next to references to traditional texts, there is a clear value in allowing links to other interactive documents, mathematical encyclopedias in electronic form, etc., which will increase as more mathematical texts become available on-line, for instance via the World-Wide Web (T.J. Berners-Lee, R. Cailliau and J.-F. Groff [2]—see for instance figure 1.

  When studying a large theory it is very easy to get lost. In such cases a reader can be greatly helped when provided with an easily navigable presentation of the mathematical structure of that particular theory, which is basically a graph containing the definitions, lemmas, and theorems and their interrelations. This graph can be automatically extracted from the text if all hyperlinks are in place, and properly annotated as to their nature. Obviously, such a facility is not only useful for not getting lost; it may also be very helpful for authors, or for researchers who want to generalize a theory to a related area of mathematics.

- *Annotation facilities.* For mathematics it is important that reader annotations not be limited by the size of the margin. Likewise, they need not be confined by the limitations of a purely textual rendering. Whenever the reader can ask the book to perform certain computations on mathematical objects contained in the original text of the book (say, compute the dimension of a given Lie algebra), this should be equally possible on mathematical objects occurring in annotations added by the reader. Thus, annotations are not second-class citizens, but may have recourse to the full spectrum of interactive capabilities.

- *Tailoring to user need.* An interactive book can allow readers to change the mathematical notations used to preferred notations, for

Figure 1. A WWW page that could serve as an external hyperlink from an encyclopedia entry on classical scientific literature.

example because of familiarity. The book can then also accept these notations for user-supplied input.

For advanced readers the book could hide from view elementary sections or expositions, give only informal and intuitive versions of the proofs, and skip elementary or add more advanced examples. It could also be possible for a reader to ask for a view on the book tuned to a specialized subject treated in it (for example, free Lie algebras), in which case it only presents those parts that are relevant to this subject.

More generally, there may be several linear routes through an essentially non-linear document, and the best choice may be determined by the interest and expertise of the reader. For example, a reader conversant with the topic of a book, consulting it as a reference on a specific issue, has other needs than the reader who is gradually learning about an entirely new subject, and this difference in needs can be reflected in a different presentation of the material. It is important here that the choices are under the control of the reader, and that locally other choices can be made than globally, for example for the professional reader who wants to take a quick 'refresher course' in some specific part of a generally familiar area. The concept of routes and its implications for interactive books is worked out in more detail in [4].

- *Expert assistance.* For interactive mathematical books, the recent and rapid development of high-quality computer-algebra packages is of paramount importance. Provided the infrastructure needed to connect the two is present, a computer-algebra package can bring life to the examples and exercises given in the book. Instead of one static example with predefined data, we can have *active examples*: the reader can change the data and immediately see the effect on the results.

  In exercises, a book can provide hints and access to the computing machinery needed: tedious calculations can be left to a computer-algebra engine, and the reader can concentrate on understanding the essence of the exercise. Furthermore, the book can (for some exercises) check the reader-supplied answer by simply computing it and, if the given answer is wrong, present a related exercise, which is recomputed for the given context.

  Readers can use the book as a platform to perform their own mathematics within the context of the book. They will be able to insert solutions to exercises, and to invoke algorithms from computer-algebra systems and other external applications transparently, meaning that

the book takes care of the translation between the application's input and output format and the book's internal format.

It is this promising integration of mathematical textbooks with recent developments in computer science, and especially computer algebra, that instigated us to start the ACELA project (A.M. Cohen and L.G.L.T. Meertens [1]).

By taking Lie algebras as the subject matter for our first prototype, we hope to establish convincing evidence that an interactive book offers the reader a more active involvement than is possible with a paper book, a better guided tour than is provided by a computer-algebra package, and a more rewarding experience than obtained by using these two together but separately, as independent entities.

4. Architecture

In designing and implementing the prototype system, we can distinguish between a subject-independent part—the kernel system—and a subject-dependent part—in our case, mathematics, and more specifically, Lie algebras.

The kernel system needs content, like a record player needs a record: in this case, the content is the text of the book. For a proper separation of concerns, we wish to consider the 'truly' subject-dependent part as belonging to the content. So the content of a book can consist, next to a text, also of algorithmic entities: author-supplied *methods* to work on the types of objects occurring in the text. When the book 'record' is loaded into the system 'player', these methods are loaded as well. Part of the project is indeed devoted to creating powerful implementations of various algorithms for Lie algebras.

Still, we must take the future mathematical content into account right from the start, also for the design of the kernel system. The non-trivial demands posed by the requirement that mathematics can be brought to life would be virtually impossible to meet by afterthought modifications.

A simple example of this is the user control over the mathematical notation. This requires a strong logical separation of content structure and visual presentation, something that is missing in almost every mathematical-document preparation system. For example, although the *presentation* structure of $f(x)$ is juxtapose(identifier('f'), parenthesis(identifier('x'))), its *content* structure is apply(function=identifier('f'), argument=identifier('x')). If content structure and presentation structure are not kept separate in the software *ab initio*, it is practically impossible to pry them apart later. Such a separation is only possible if it is catered for in the architecture itself.

More generally, the desire to achieve transparent interoperability with external engines requires a flexible juggling with various representations

for the same (mathematical) object. The actual representation rules are, of course, part of the (algorithmic) content of the book, but it would be awkward, to say the least, if the infrastructure for handling pluriform representations and interposing the right transformations is not provided by the kernel system.

Amongst aims of the planned system are:

- *Interoperability with external engines.* For the direct purpose of having a prototype system, we could resort to an architecturally less pleasing design and create an *ad hoc* interface to a Lie-algebraic engine. However, our plan is to approach the interoperability issues in a more general way: a further goal of the ACELA project is to establish a test ground for the transparent integration of computer algebra, document processors, and proof checkers, using a variety of external engines.

- *User interaction.* The infrastructure provided by the book for the interaction with external engines will also be used for the interaction with the user. Thus, the user can use familiar presentation-oriented notation while entering formulae, which are translated by the book into content-oriented structures.

  We want the reader of the book to get into it with a minimum learning effort, and in general that all access to external engines is mediated transparently by the book.

  'Transparently' means here that the user interface keeps the same 'look and feel', whatever the engine used. To achieve this we use the interaction paradigm developed in the Views project as described in [3], specifically with the aim of creating a unified interaction model for the interoperability of varying applications (see figure 2). The conceptual model underlying the interaction paradigm of Views is that all interaction is achieved by the user by editing documents. These documents may be structured, and contain mathematical objects. The notion of editing comprises ways to enter and change text, as well as facilities for moving around and searching – in short, navigation.

  An essential part of the Views interaction paradigm is that the semantics of the system as it presents itself to the user is based on defined logical relationships between various documents. If one is modified by the user, the system recomputes and updates the other. A basic example of this is a spreadsheet. A more involved example can be the relationship between a query on a database and the result of that query. By editing the query, the result is automatically updated. The same holds when the database is modified. Exercises and active examples can easily be created with this mechanism.

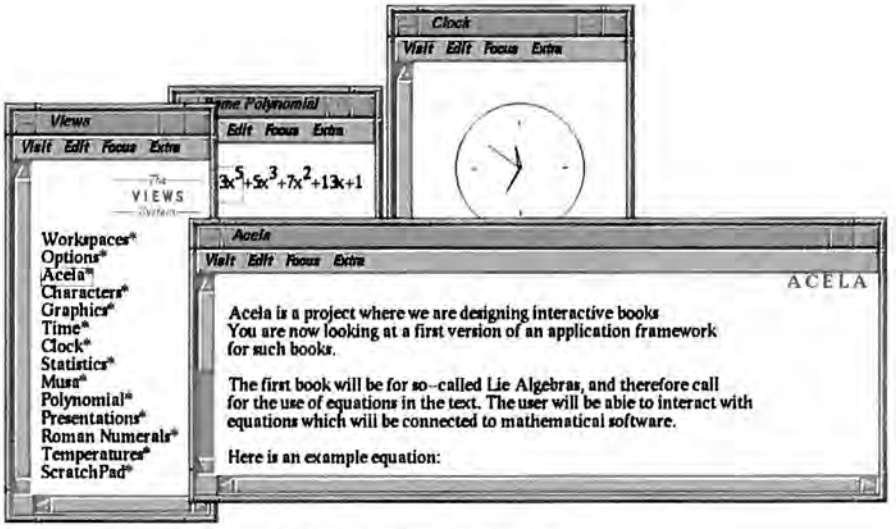


Figure 2. Sample Views session.

Similarly, the visual presentation of formulae is determined by one or more 'notation' sheets, which are style sheets for the type or sub-types of 'formula'. By editing these sheets, the user can exercise control over the notation.

Since all users are familiar with the basic edit paradigm and edit mechanisms, getting to work with the intended interaction paradigm needs hardly any learning.

- *Authoring.* The prototype system is not only intended to be for readers; the same system will be used by authors. Thus, there will be no distinction between a 'reading' system and an 'authoring' system. Obviously, authors have to be able to read what they have just written, and interact with it just like prospective readers could do, and readers, while creating annotations, become authors. The only hard difference is that, presumably, arbitrary readers cannot *change* the content of the book they are reading, but only change its *appearance* and further add annotations, while an author must be able to change the contents. However, we envisage the addition of a number of tools that, although accessible to readers, are primarily intended as facilities for authors creating a complete book text. For example, a constraint on a route through the text is that the sections may only depend on previous sections, and authors will be served with tools for managing routes

that keep track of such constraints. As for reading, the interaction paradigm for authoring will be that of editing text. The additional tools will facilitate structuring and restructuring, creating hyperlinks as well as the dependencies that are used for setting up routes, and interactively plugging in interfaces to external engines, together with monitoring and debugging aids.

5. Conclusion

The design we have set forth in this paper is ambitious, and having some priorities among them is not a luxury. One priority is that we strive for getting the things that we do right. As we see it, the architectural decisions made now will have a strong impact on the long-term potential of the system, and ill-chosen decisions will eventually emerge as unsurmountable limitations, but given the right architectural set-up, further enhancements in functionality will always be possible.

None of the individual aims we have is by itself unique, in the sense that some product exists that achieves it. At the same time, each existing product—in its current incarnation—has limitations that make it unsuitable as a platform for realizing the interactive book we have in mind. The main contribution of our project in this context would be in showing how the various desiderata can be combined gracefully.

Any form of interoperability can ultimately only be successful through the use of a common standard, whether used at the scale of a single project (requiring some form of adaptation for externally supplied software), or adopted at a wider range. System-level standards for document-centred computing are now emerging and will be watched closely. An essential requirement for us, though, is that such standards be both platform-independent and vendor-independent.

References

1. A.M. Cohen, L.G.L.T. Meertens (1995). *The ACELA Project: Aims and Plans*. CWI, Amsterdam. Available on-line:
`http://www.cwi.nl/cwi/projects/acela/papers/Aims+Plans.html` .
2. T.J. Berners-Lee, R. Cailliau, J.-F. Groff (1992). The World-Wide Web. *Computer Networks and ISDN Systems 25*, 454-459.
3. L.G.L.T. Meertens, S. Pemberton (1992). *The Ergonomics of Computer Interface—Designing a System for Human Use*. Report CS-R9258, CWI, Amsterdam. Available on-line:
`http://www.cwi.nl/ftp/CWIreports/AA/CS-R9258.ps.Z` .
4. O. Weber (1994). *Routes through the maze*. ACELA working document, CWI, Amsterdam. Available on-line:
`http://www.cwi.nl/cwi/projects/acela/papers/maze.ps.Z` .

Logic Programming

K.R. Apt

1. Introduction

Logic programming (in short LP) is a simple, yet powerful formalism suitable for programming and for knowledge representation. It was introduced in 1974 by R. Kowalski. LP grew out of an earlier work on automatic theorem proving based on the resolution method. The major difference is that LP can be used not only for proving but also for computing. In fact, LP offers a new programming paradigm, which was originally realized in Prolog, a programming language introduced in early seventies by a group led by A. Colmerauer.

After an initially slow start LP grew twenty years later to an impressive field in computer science, in which by now a couple of thousand articles have been published. Recently, *The Journal of Logic Programming* celebrated its tenth year anniversary. A couple of annual conferences are nowadays taking place and interest in the subject does not seem to be waning. On the contrary. The logic programming paradigm has inspired the design of new programming languages, like CHIP and Gödel, which have been successfully used to tackle various computationally complex problems. These languages attempt to overcome a number of Prolog deficiencies, visibly awkward use of arithmetic, ad hoc control features and lack of types.

One of the reasons for this interest in LP is its simplicity combined with versatility. LP strongly relies on mathematical logic which developed its own methods and techniques and can provide a rigorous mathematical frame-

work for LP. In many cases these methods have to be fine tuned and appropriately modified to be useful in LP. It should be added here that some basic concepts of LP, like unification, were developed earlier by computer scientists working in the field of automated reasoning.

Efficient implementation of Prolog and its extensions, development of appropriate programming methodology and techniques, that aim at better understanding of the logic programming paradigm, and finally design of various successors and/or improvements of Prolog turned out to be an exciting and highly non-trivial field calling for new solutions and fresh insights.

Prolog was originally designed as a programming language for natural language processing. But it soon turned out that other natural applications for the logic programming paradigm exist. Current applications of LP involve such diverse areas as molecular biology, design of VLSI systems, representation of legislation, and option trading. These applications exploit the fact that knowledge about certain domains can be conveniently written down as facts and rules which directly translate into executable logic programs.

These three aspects of LP—theory, programming and applications—grew together and often influenced each other. This versatility of LP makes it an attractive subject to study and an interesting field to work in.

2. Declarative programming

LP allows us to write programs and compute using them. There are two natural interpretations of a logic program. The first one, called a *declarative interpretation*, is concerned with the question *what* is being computed, whereas the second one, called a *procedural interpretation*, explains *how* the computation takes place. Informally, we can say that declarative interpretation is concerned with the *meaning*, whereas procedural interpretation is concerned with the *method*.

These two interpretations are closely related to each other. The first
368 interpretation helps us to better understand the second and is a major reason why LP is an attractive formalism for programming. The fact that when designing a logic program one can rely on its declarative interpretation explains why LP supports *declarative programming*. Loosely speaking, declarative programming can be described as follows. *Specifications*, when written in an appropriate format, can be used as a *program*. Then the desired conclusions follow *logically* from the program. To compute these conclusions some *computation mechanism* is available.

Now 'thinking' declaratively is in general much easier than 'thinking' procedurally. So declarative programs are often simpler to understand, develop and modify. In fact, in some situations the specification of a problem in the appropriate format forms already the algorithmic solution to the problem. In other words, declarative programming makes it possible to write exe-

cutable specifications. It should be added however, that in practice the programs obtained in this way are often inefficient, so this approach to programming has to be coupled with appropriate use of program transformations and various optimization techniques.

This dual interpretation of logic programs also accounts for the double use of LP — as a formalism for programming and for knowledge representation, and explains the importance of LP in the field of artificial intelligence.

3. Declarative programming in Prolog

Prolog differs from LP in several small, but important aspects. In particular, both the selection rule ('choose left first') and the search strategy (depth first search) are fixed.

To illustrate the declarative programming in Prolog we now present three examples. These were on purpose chosen short and simple.

Example 1: a sequence problem

Consider the following problem: arrange three 1's, three 2's, ... three 9's in sequence so that for all $i \in [1,9]$ there are exactly i numbers between successive occurrences of i. Figure 1 shows the program that solves this problem in Prolog and the output showing all 6 solutions.

We see that the Prolog solution to the problem is an almost literal formalization of its formulation.

Example 2: typing of lambda terms

Consider the typed lambda calculus and Curry's system of type assignment (see H.B. Curry and R. Feys [4]). It involves statements of the form $x : t$ which should be read as 'term x has type t'. Finite sequences of such statements often called *environments* are denoted below by E. The following three rules allow us to assign types to lambda terms:

$$\frac{x : t \in E}{E \vdash x : t}$$

$$\frac{E \vdash m : s \to t \ , \quad E \vdash n : s}{E \vdash (m\ n) : t}$$

$$\frac{E \ , \ x : s \vdash m : t}{E \vdash (\lambda x.m) : s \to t}$$

These rules directly translate into the Prolog program given in figure 2. For the sake of this program lambda terms are encoded as first-order terms. To this end the unary function symbol `var` and two binary function symbols, `lambda` and `apply` are used. The lambda term x is translated to the term

Arrange three 1's, three 2's, ... three 9's in sequence so that for all $i \in [1..9]$ there are exactly i numbers between successive occurrences of i.

```
% sequence(Xs)  —  Xs is a list of 27 elements.
sequence([_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]).

% question(Ss)  —  Ss is a list of 27 elements forming the desired sequence.
question(Ss) —
   sequence(Ss),
   sublist([1,_,1,_,1], Ss),
   sublist([2,_,_,2,_,_,2], Ss),
   sublist([3,_,_,_,3,_,_,_,3], Ss),
   sublist([4,_,_,_,_,4,_,_,_,_,4], Ss),
   sublist([5,_,_,_,_,_,5,_,_,_,_,_,5], Ss),
   sublist([6,_,_,_,_,_,_,6,_,_,_,_,_,_,6], Ss),
   sublist([7,_,_,_,_,_,_,_,7,_,_,_,_,_,_,_,7], Ss),
   sublist([8,_,_,_,_,_,_,_,_,8,_,_,_,_,_,_,_,_,8], Ss),
   sublist([9,_,_,_,_,_,_,_,_,_,9,_,_,_,_,_,_,_,_,_,9], Ss).

% append(Xs, Ys, Zs)  —  Zs is the result of concatenating the lists Xs and Ys.
append([], Ys, Ys).
append([X | Xs], Ys, [X | Zs]) — append(Xs, Ys, Zs).

% sublist(Xs, Ys)  —  Xs is a sublist of the list Ys.
sublist(Xs, Ys) — append(_, Zs, Ys), append(Xs, _, Zs).

| ?- question(Ss).

Ss = [1,9,1,2,1,8,2,4,6,2,7,9,4,5,8,6,3,4,7,5,3,9,6,8,3,5,7];

Ss = [1,8,1,9,1,5,2,6,7,2,8,5,2,9,6,4,7,5,3,8,4,6,3,9,7,4,3];

Ss = [1,9,1,6,1,8,2,5,7,2,6,9,2,5,8,4,7,6,3,5,4,9,3,8,7,4,3];

Ss = [3,4,7,8,3,9,4,5,3,6,7,4,8,5,2,9,6,2,7,5,2,8,1,6,1,9,1];

Ss = [3,4,7,9,3,6,4,8,3,5,7,4,6,9,2,5,8,2,7,6,2,5,1,9,1,8,1];

Ss = [7,5,3,8,6,9,3,5,7,4,3,6,8,5,4,9,7,2,6,4,2,8,1,2,1,9,1];
```


Figure 1. The sequence problem.

```
curry(E, var(X), T) ← member([X, T], E).
curry(E, apply(M, N), T) ← curry(E, M, S → T),
                           curry(E, N, S).
curry(E, lambda(X, M), S → T) ← curry([[X, S] | E], M, T).

member(X, [Y | Xs]) ← X ≠ Y, member(X, Xs).
member(X, [X | Xs]).
```

Figure 2. The type assignment program.

`var(x)`, the lambda term $(m\ n)$ to the term `apply(m, n)`, and the lambda term $\lambda x.m$ to the term `lambda(x, m)`. The subtle point is that according to Prolog convention, lower case letters stand for constants, so `var(x)` is a ground term (i.e. a term without variables), etc. For example, the lambda term $\lambda x.(x\ x)$ translates to `lambda(x, apply(var(x), var(x)))`.

Now, the program in figure 2 can be used to compute a type assignment to a lambda term, if such an assignment exists, and to report a failure if such an assignment does not exist. To this end, given a lambda term s, it suffices to use the query `curry([], t, T)`, where `t` is the translation of s to a first-order term.

The problem of computing a type assignment for lambda terms was posed and solved by Curry [4]. It is considered to be an advanced topic in the theory of lambda calculus and the foundations of functional programming. The solution in Prolog given in figure 2 is completely elementary.

Example 3: temporal reasoning

In [5] S. Hanks and D. McDermott discussed a simple problem in temporal reasoning, a branch of non-monotonic reasoning. It became known in the literature as the 'Yale Shooting Problem'. Hanks and McDermott's interest in this problem arose from the fact that apparently all theories about non-monotonic reasoning, when used to formalize this problem, led to too weak conclusions. The problem has been extensively discussed in the literature and several solutions to it have been proposed. In Hanks and McDermott [5] some of these solutions are discussed and critically evaluated.

We present here a particularly simple solution to the above problem by means of logic programming. First, let us explain the problem. Consider a single individual who in any situation can be either `alive` or `dead`, and a gun that can be either `loaded` or `unloaded`. The following statements are stipulated:

1. At some specific situation s_0 the person is alive.

2. The gun becomes loaded any time a `load` event happens.

$$holds(alive, s_0).$$
$$\forall s\ holds(loaded, results(load, s)).$$
$$\forall s\ (holds(loaded, s) \rightarrow ab(alive, shoot, s) \wedge holds(dead, result(shoot, s))).$$
$$\forall f \forall e \forall s\ ((holds(f, s) \wedge \neg ab(f, e, s)) \rightarrow holds(f, result(e, s))).$$

Figure 3. The Yale Shooting Problem: the original formulation.

3. Any time the person is shot with a loaded gun, he becomes dead. Moreover, the fact of staying alive is abnormal with respect to the event of being shot with a loaded gun.

4. Facts which are not abnormal with respect to an event remain true.

To formalize these statements Hanks and McDermott [5] used so-called *situation calculus* in which one distinguishes three entities: facts, events and situations, denoted respectively by the letters f, e, s, and the function *result* such that for an event e and a situation s the term $result(e, s)$ denotes the situation resulting from the occurrence of e in s. These four statements lead to the four formulas given in figure 3.

The problem was to find a way of interpreting these formulas so that statements like

$$holds(dead,\ result(shoot,\ result(wait,\ result(load,\ s_0))))$$

could be proved. Here *wait* is a new event whose occurrence is supposed to have no effect on the truth of the considered facts.

The solution to the Yale Shooting Problem in Prolog is completely straightforward and shown in figure 4. This program is an almost literal translation of the above formulas to Prolog syntax. (To enhance readability we use in it the list notation `[e | s]` of Prolog instead of $result(e, s)$ and denote the initial situation by the empty list `[ ]`.)

In contrast to the solutions in other formalisms, the Prolog solution can

```
holds(alive, []).
holds(loaded, [load | Xs]).
holds(dead, [shoot | Xs]) ← holds(loaded, Xs).
ab(alive, shoot, Xs) ← holds(loaded, Xs).
holds(Xf, [Xe | Xs]) ← ¬ ab(Xf, Xe, Xs), holds(Xf, Xs).
```

Figure 4. A program solving the Yale Shooting Problem.

be used not only to model the problem but also to compute answers to the relevant queries. For example, we have

```
| ?- holds(dead, [shoot, wait, load]).

yes

| ?- holds(dead, [wait, load]).

no
```

Also, using the theory of logic programming, it is possible to provide a natural semantics to this program in the form of a unique model, which admits several natural characterizations and allows us to predict the correct answers to the queries. In [6] E. Marchiori studied an extension of Prolog with so-called constructive negation. This allowed her to deal with more complex queries which Prolog does not handle correctly, like `holds(alive, [X, Y])`. By means of constructive negation it yields two answers `X` $\neq$ `shoot` and `Y` $\neq$ `load`, which is justified from the semantic point of view.

4. PROGRAM VERIFICATION AND PROLOG

The usual way of explaining that a program is correct is that it meets its specifications. This statement has a clear intention but is somewhat imprecise so we shall be more specific in a moment. Correctness of programs is important, both from the point of view of software reliability as from the point of view of software development. Program verification is the formal activity whose aim is to ensure correctness of programs. It has a history spanning a quarter of the century.

In the case of logic programming the declarative interpretation reduces the issue of program correctness to an analysis of the program from the logical point of view. In this analysis the computation mechanism can be completely disregarded. This is an important reduction which significantly simplifies the task of program verification.

In the case of Prolog it is natural to base the program verification on the theory of logic programming. Because of the differences between Prolog and logic programming this theory has to be appropriately modified and revised. Moreover, due to several 'non-declarative' features of Prolog, a declarative interpretation of Prolog programs is, to say the least, problematic. To cope with this problem we determined in our studies a large subset of Prolog and showed that for programs written in this subset it is possible to reason about their correctness by a combination of syntactic analysis and declarative interpretation.

In our approach we dealt with various program properties which are crucial for ensuring proper functioning of these programs. In particular, we

considered:

- Termination.

  This means that the program under consideration should terminate for the appropriate queries.

- Partial correctness.

  This means that the program under consideration should deliver correct answer for the appropriate queries.

- Absence of run-time errors.

  In the case of Prolog these are:

  - absence of the so-called occur-check problem (explained below),
  - absence of errors in presence of arithmetic expressions.

The resulting framework is simple to use and readily applicable to most of the well-known Prolog programs. Moreover, several aspects of the proposed methods can be automated.

4.1. Termination

As an example we explain here the approach to termination of simple Prolog programs due to K.R. Apt and D. Pedreschi [1].

We need some introductory notions. We assume here that all programs and queries are written in a fixed, 'universal', language defined by, say, a Prolog manual.

Definition

- A *level mapping* is a function $|\ |$ from ground atoms (i.e. atomic formulas with no variables) to natural numbers.

- An atom A is *bounded w.r.t.* $|\ |$, if $|\ |$ is bounded on the set of all ground instances of A.

- A clause c is *acceptable w.r.t.* $|\ |$ *and* an interpretation I, if

  - $I \models c$ (I is a model of c),
  - for all ground instances $A \leftarrow \mathbf{A}, B, \mathbf{B}$ of c such that $I \models \mathbf{A}$ $|A| > |B|$.

- A program is *acceptable w.r.t.* $|\ |$ *and* I, if every clause of it is.

Then the following result holds.

Theorem. Suppose that

- P is acceptable w.r.t. $|\ |$ and I,
- A is bounded w.r.t. $|\ |$.

Then all Prolog computations of A w.r.t. P are finite.

Let now `ls(.)` (for *listsize*) be a function from ground terms to natural numbers defined by induction as follows:

$$\texttt{ls}([x|xs]) = \texttt{ls}(xs) + 1,$$
$$\texttt{ls}(f(x_1, \ldots, x_n)) = 0 \text{ if } f \neq [\,.\,|\,.\,].$$

To see a simple use of the above theorem consider the program of figure 1. It is easily seen to be acceptable w.r.t. the level mapping defined by:

$$|\texttt{question(xs)}| = 50,$$
$$|\texttt{sequence(xs)}| = 0,$$
$$|\texttt{sublist(xs,ys)}| = \texttt{ls(xs)} + \texttt{ls(ys)} + 1,$$
$$|\texttt{append(xs,ys,zs)}| = \min(\texttt{ls(xs)}, \texttt{ls(zs)}),$$

and any model I of it such that for a ground term `s`

$I \models$ `seq(s)` iff `s` is a list of 27 elements.

Note that `question(Ss)` is bounded w.r.t. $|\ |$, so we conclude that all Prolog computations of `question(Ss)` are finite.

A natural modification of this approach to programs that use negation can be used to deal with termination of the Prolog programs given in figures 2 and 4.

4.2. Occur-check problem

The occur-check is a special test used in the unification algorithm, a cornerstone of Prolog's computation mechanism. In most Prolog implementations it is omitted for efficiency reasons. This omission affects the unification algorithm and introduces a possibility of divergence. This is obviously an undesired situation.

In our work (see Apt and A. Pellegrini [2]) we provided easy to check syntactic conditions which can be verified mechanically and which imply that the occur-check can be safely omitted for a given program and query. For example, the programs given in figures 1 and 4 are safe from the occur-check problem for the queries used.

In contrast, the program given in figure 2 leads to problems. In particular, for the query `curry([], lambda(x, apply(var(x), var(x))), T)` the omission of the occur-check causes divergence. In our studies we showed how this problem can be taken care of by means of a simple program transformation which inserts calls of the built-in unification predicate (with the

occur-check test) into the program text. In the case of the program given in figure 2 it suffices to modify the second and the last clause as follows:

```
curry(E, apply(M, N), T)  ← curry(E, M, S → T),
                            curry(E, N, Z),
                            Z =oc S.

member(X, [Z | Xs])  ←  Z =oc X.
```

Here '$=_{oc}$' is the unification predicate with the occur-check test. The resulting program can then be used for the queries of interest. For example, in the case of the term $\lambda x.(x\ x)$ we use the query `curry([], lambda(x, apply(var(x), var(x))), T)`. This query finitely fails:

```
| ?-  curry([],  lambda(x, apply(var(x), var(x))), T)

no
```

which confirms that the original lambda term has no type assignment.

4.3. Delay declarations

One of the striking features of logic programs is that they can be easily parallelized. For example, by adding to the program of figure 2 the so-called delay declaration:

```
DELAY append(_, _, Z) UNTIL nonvar(Z).
```

we obtain a program with a large degree of parallelism. The above declaration defers the selection of the `append`-atoms until their last argument is not a variable. By default, no restrictions are imposed on the selection of other atoms.

So the use of delay declarations replaces the Prolog selection rule by a non-deterministic selection rule which dynamically determines which atoms can be selected. In the executions of the resulting programs dynamic networks of processes are created that communicate asynchronously by means of multiparty channels. In the case of the program of figure 1 up to nineteen processes can be created during its executions.

The delay declarations allow us to impose a synchronization on the actions of a logic program in a concise way. Programs augmented by the delay declarations can be translated in a straightforward way into other concurrent languages based on the logic programming paradigm.

In our recent publications, we showed how correctness of such parallel programs can be established by a natural modification of the methods originally developed for Prolog programs.

```
rel sequence:  array [1..27] of [1..9].
sequence(A) ← ∀I ∈ [1..9] ∃J ∈ [1..25-2I]
    (A[J] = I, A[J+I+1] = I, A[J+2I+2] = I)).
```

Figure 5. A simple solution to the problem from figure 1.

4.4. Language extensions

In our recent work [3] we studied language extensions which involve iteration and arrays. Iteration is implemented by means of bounded quantification. We noticed that the use of iteration within the logic programming paradigm often leads to substantially simpler programs which are closer to specifications and are guaranteed to terminate.

As an example consider the solution to the problem from figure 1 given in figure 5. It is very close to the problem specification and much simpler than the one given in figure 1. The range `J ∈ [1..25-2I]` comes from the requirement that the indices `J, J+I+1, J+2I+2` should lie within `[1..27]`.

References

1. K. R. Apt. D. Pedreschi (1993). Reasoning about termination of pure Prolog programs. *Information and Computation 106(1)*. 109–157.
2. K. R. Apt. A. Pellegrini (1994). On the occur-check free Prolog programs. *ACM Toplas 16(3)*, 687–726.
3. K. R. Apt (1995). Arrays, bounded quantification and iteration in logic and constraint logic programming. M. Alpuente Frasnedo. M. I. Sessa (eds.). *1995 Joint Conference on Declarative Programming (GULP-PRODE '95)*. University of Salerno. Italy. Invited Lecture. 19-35.
4. H.B. Curry. R. Feys (1958). *Combinatory Logic, Volume I, Studies in Logic and the Foundation of Mathematics*. North-Holland. Amsterdam.
5. S. Hanks. D. McDermott (1987). Nonmonotonic logic and temporal projection. *Artificial Intelligence 33*. 379-412.
6. E. Marchiori (1995). On termination of general logic programs w.r.t. constructive negation. *The Journal of Logic Programming*. Accepted for publication.

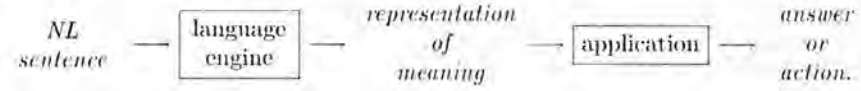
On the Borderline of Logic, Language and Computation

D.J.N. van Eijck

1. What is Computational Linguistics?

Formal language theory, as a branch of computer science, is concerned with the study of formal languages such as *Pascal*, *Prolog*, *LISP*, and various other formalisms designed for human–computer interaction. It has become clear in the past two decades, however, that the tools developed for the analysis of formal languages can fruitfully be applied to the study of natural languages such as English, Russian or Dutch. Recently, a new branch of linguistics has emerged which uses insights from formal language theory, empirical linguistics, and logic, with the overall aim to implement natural language understanding systems on computers. This new discipline is called computational linguistics.

Here is a schematic view of a typical system for human–computer interaction by means of natural language:



A system which supports voice interaction would have extra components for speech recognition and generation. In the scheme above these components are omitted. If written input is assumed, a natural language engine for English should be able to recognize a substantial fragment of grammatical

written English sentences, and generate their literal meanings; it should also be able to generate grammatical English expressions on the basis of symbolic inputs from an application program. In the scheme it is assumed that the answers given by the system are displayed as English expressions on the screen. The language engine should contain an extensive lexicon for English, a set of grammar rules for a considerable fragment of English, and a set of translation rules matching the grammar rules for translating the English expressions into unambiguous formal expressions that represent their meanings and that can be handled by the application [1]. Typical applications are expert systems or knowledge bases.

2. Fragments of English

Probably the best way of forming an accurate picture of what is involved in natural language processing is to develop a toy natural language application. In this contribution we aim to give a basic idea of some aspects of designing a system for natural language understanding by developing a very simple toy fragment of natural language. The method of fragments has been first advocated in the work of R. Montague [5]. Our fragment will be a lot simpler still than the simplest Montague grammar fragment. We want to be able to process sentences like the following:

A woman smiles. (2.1)

John loves a woman. (2.2)

If a woman smiles, John loves her. (2.3)

These examples may seem embarrassingly simple, but they are not quite as trivial as may appear at first sight. Example (2.3) exhibits a logical puzzle that has bothered natural language researchers for a long time. If one compares examples (2.1) and (2.2) with (2.3) then it appears that the first two can be understood as statements about a particular woman, while the third seems to express a general statement about women. This poses a genuine problem for natural language understanding, for the process of building meaning representations for sentences has to proceed in a *compositional* fashion: the meaning of a complex expression of natural language is built from the meanings of its components. This compositionality requirement is one of the cornerstones of the enterprise of building meaning representations in a systematic way.

As the example sentences indicate, indefinite noun phrases such as *a woman* seem to require a meaning representation as existential expressions when they appear in simple contexts and a representation as universal expressions when they appear in the antecedents of *if then* contexts. A first attempt at solving this problem would use ordinary predicate logic. Take example (2.4).

If a woman smiles, John smiles. (2.4)

Here one can translate the noun phrase *a woman* using an existential quantifier and still get the right meaning: $\exists x(Wx \wedge Sx) \rightarrow Sj$. To the logician it is clear immediately that this expresses a universal statement: the existential quantifier occurs in a negative position, so the translation is equivalent to $\forall x((Wx \wedge Sx) \rightarrow Sj)$. Unfortunately, this observation does not work for example (2.3). A straightforward translation of (2.3) in predicate logic would yield $\exists x(Wx \wedge Sx) \rightarrow Ljx$. But this cannot be correct, for the variable x in Ljx is left unbound.

A systematic solution of this unbound variable problem becomes possible if one translates to a representation language where variable binding proceeds in a dynamic fashion, as in imperative programming languages; such a move was first proposed in [2]. To make this work one has to replace existential quantifiers with indeterministic instructions for storing values with specified properties in the memory locations associated with variables. So instead of $\exists x(Wx \wedge \ldots)$ one writes $\eta x : Wx; \ldots$, with the intended meaning: 'fill location x with (a representation of) an object satisfying W and proceed with the ... processing'. The switch to dynamic interpretation will make it necessary to relate dynamic meaning representations to the old fashioned static representations that can be expressed in predicate logic, for example. This problem has been addressed in the research in natural language analysis at CWI [4].

What we will do in the remainder of this contribution is give a very rudimentary sketch of the syntactic processing of natural language by means of a tool called *categorial grammar*. We will then show how the syntactic analysis can be used to build meaning representations in a dynamic representation language. Finally, we will discuss the problem of relating the dynamic representations to static representations phrased in ordinary predicate logic.

3. Categorial Grammar with Features

A categorial grammar is a grammar combined with a lexicon in such a way that the lexical information comprises virtually all the information one needs for syntactic processing. These grammars are called categorial because they proceed by assigning *categories* to expressions. Simple categories such as S for 'sentence', CN for 'common noun' and IV for 'intransitive verb' are taken as basic.

For our fragment we have the following expressions in basic categories: *smiles:IV*, *man:CN* and *woman:CN*. Next, a proper name can be viewed as an expression which combines with an intransitive verb to its right to form a sentence; in categorial notation: S/IV. So for our fragment: *John:S/IV*. The

indefinite article combines with a common noun to form a noun phrase: noun phrases, as we have seen, have category *S/IV*, so we have: *a:(S/IV)/CN*. Transitive verbs combine with noun phrases to form intransitive verbs, which gives: *loves:IV/(S/IV)*. Finally, the sentential operator *if* takes an antecedent sentence and forms an expression which combines with a consequent sentence to form a new sentence, which gives: *if:(S/S)/S*.

The basic strategy for putting categorial expressions together is very simple: put an expression of category CAT_1/CAT_2 in front of an expression of category CAT_2 to form a new expression of category CAT_1. Thus, putting *John:(S/IV)* in front of *smiles:IV* gives *John smiles : S*, and so on.

Of course, this grammar is too simple as it stands. For example, the pronoun *her* will have the same category as any other noun phrase, namely *S/IV*, and this would enable the derivation of *her smiles : S*. To remedy this, it is customary to enrich the categories with *feature information*. For instance, if one assumes that a noun phrase has features for case, gender, number and a coreference index, the category for *her* could look like *(S/IV):[-nom,f,sg,213]*, to indicate that its case is not nominative, its gender is feminine, its number is singular, and it has coreference index 213 (which means that it is intended to be linked to an antecedent which also has index 213). This information can be used to enrich the category of noun phrases to force agreement in case and number with an intransitive verb phrase, as follows : *(S/IV:[Case,Number]):[Case,_ ,Number,_]*. The upper case letters are used to indicate feature constraints: _ indicates that any value will do for the feature at that position. If an item of this category combines with an *IV* with given features for case and number, then these features should agree. Enriching the category of *walks* to *IV:[nom,sg]* now blocks the derivation of *her walks*. It is possible to encode very complex and detailed information in syntactic features.

Toy Categorial Grammar

John$_i$	(S/IV[Case,sg]):[Case,sg,m,i]
her$_i$	(S/IV:[-nom,sg]):[-nom,f,sg,i]
smiles	IV:[nom,sg]
loves	(IV:[nom,sg]/(S/IV)):[-nom,_ ,_ ,_]
man	CN:[sg,m]
woman	CN:[sg,f]
a$_i$	((S/IV):[_ ,sg,Gender,i]/CN:[sg,Gender]
if	(S/S)/S

In the next section we will present a logical perspective on the process of parsing. In Section 5 we will see how a procedure for building meaning representations can be hooked to a categorial grammar.

4. Parsing as Deduction

We can look at the process of parsing natural language sentences as a kind of deduction. A rule of inference of a parsing algorithm has the following general form:

$$\frac{A_1 \quad \cdots \quad A_n}{B} \quad \textit{side conditions on } A_1, \ldots, A_n, B \tag{4.1}$$

A deduction system is given by a set of such rules, plus a set of axioms. A derivation of a formula B from premisses $A_1, \ldots, A_n$ is defined as usual: a sequence of formulas with B at the end, and each member of the sequence is either an axiom or the result of applying a deduction rule to previous members of the sequence. In the parsing application, we assume that formulas may refer to positions in the input string, and that derivation rules may mention grammar rules in their side conditions. In the case of parsing categorial grammars, items have the form $[X, i, j]$, where X is a grammar category and i, j refer to positions in the input string. The intended meaning is: category X spans word sequence $w_i, \ldots, w_j$ in the input string. Axioms have the form $[X, i, i+1]$, where X is a category assigned by the lexicon to word w_{i+1} in the input string. If the input string has length n, the parsing goal has the form $[S, 0, n]$. The inference rule looks as follows:

$$\frac{[X/Y, i, j] \quad [Y, j, k]}{[X, i, k]} \tag{4.2}$$

John loves a woman. (4.3)

The proof that (4.3) is a sentence of the fragment is given in the box below. More information on parsing as deduction can be found in [6].

Proof of sentence in the fragment.

1.	[S/IV, 0, 1]	axiom
2.	[IV/(S/IV),1 , 2]	axiom
3.	[(S/IV)/CN,2 , 3]	axiom
4.	[CN, 3, 4]	axiom
5.	[S/IV, 2, 4]	rule application on 3, 4
6.	[IV, 1, 4]	rule application on 2, 5
7.	[S, 0, 4]	rule application on 1, 6

5. Building Meaning Representations

If a categorial grammar is given, meaning representations for the expressions recognized by the grammar can be given using the tools of lambda abstraction. The semantic operation corresponding to the syntactic combination of an expression of category CAT_1/CAT_2 and one of category CAT_2

will be the functional application of a typed lambda expression corresponding to the functor expression to a typed lambda expression corresponding to the argument. The basic categories provide the clue for the types of the translations: S expressions should translate as formulae, IV and CN expressions as one-place predicates. Thus, an appropriate translation for *woman* is $\lambda x.Wx$, which does indeed denote a one place predicate. Similarly for IV expressions: *smiles* can be translated as $\lambda x.Sx$. Expressions of category S/IV must be lambda expressions that can take things like $\lambda x.Sx$ as arguments, so an appropriate translation for *John* would be $\lambda X.Xj$, where X is a variable for one-place predicates. Given that we want to translate *a woman* as $\eta x : Wx; \ldots$, an appropriate translation for this noun phrase is $\lambda Y.\eta x : Wx; Yx$, which means that *a* can be translated as $\lambda X\lambda Y.\eta x : Xx; Yx$, and so on. If one wants to use the noun phrase indices to establish links of pronouns to their antecedents, individual variables with the same index must be used in the translations. These considerations are all taken into account in the example grammar with semantic component below.

Toy Categorial Grammar with Semantic Component

John$_i$	(S/IV[Case,sg]):[Case,sg,m,i]	$\lambda X.(\eta v_i : v_i = j; Xv_i)$
her$_i$	(S/IV:[-nom,sg]):[-nom,f,sg,i]	$\lambda X.Xv_i$
smiles	IV:[nom,sg]	$\lambda x.Sx$
loves	(IV:[nom,sg]/(S/IV):[-nom,_,_,_]	$\lambda\mathbf{X}\lambda x.\mathbf{X}(\lambda y.Lxy)$
man	CN:[sg,m]	$\lambda x.Mx$
woman	CN:[sg,f]	$\lambda x.Wx$
a$_i$	((S/IV):[_,sg,Gender,i]/CN:[sg,Gender]	$\lambda X\lambda Y.(\eta v_i : Xv_i; Yv_i)$
if	(S/S)/S	$\lambda p\lambda q.(p \Rightarrow q)$

Our fragment enables us to construct meaning representations for the example sentences we started out with. The representation for *John smiles* becomes $\lambda X.(\eta v_i : v_i = j; Xv_i)(\lambda x.Sx)$, which reduces in two steps to $\eta v_i : v_i = j; Sv_i$. The representation for *John loves a woman* is a fairly complex expression which reduces in several steps to $\eta v_i : v_i = j; \eta v_k : Wv_k; Lv_iv_k$. We have assumed that the indices of subject and object are different; in fact, a procedure for checking co-indexings should be invoked to rule out all co-indexings with clashes in the gender or number feature. The representation for *If a woman smiles, John loves her*, in the reading where the pronoun is linked to *a woman*, will, after several reductions, boil down to $(\eta v_i : (Wv_i; Sv_i) \Rightarrow \eta v_k : (v_k = j; Lv_kv_i))$. Our final problem is to make sense of this representation.

6. Axioms for Dynamic Interpretation

The dynamic interpretation strategy treats meaning representations for natural language as imperative programs. This entails that the tools for anal-

ysis of imperative programming languages can be put to use. In particular, an axiom system for dynamic interpretation can be given in terms of Hoare style pre- and postconditions of programs [3]. A convenient way to express such conditions is by means of dynamic logic [?]. Axioms for dynamic interpretation have natural language meaning representations as modalities. $\langle\pi\rangle\phi$ expresses that ϕ holds in some π output state of the current state; $[\pi]\phi$ expresses that ϕ holds in every π output state of the current state. We can find the static meaning of a representation program by checking the conditions under which it will terminate successfully. Some examples of the dynamic logic axioms involved are given below; more details can be found in Van Eijck's contribution in [4].

Axioms for Dynamic Interpretation

$$
\begin{array}{lcl}
\langle R(t_1 \cdots t_n)\rangle\phi & \leftrightarrow & Rt_1 \cdots t_n \wedge \phi \\
\langle t_1 = t_2\rangle\phi & \leftrightarrow & t_1 = t_2 \wedge \phi \\
\langle \pi_1;\pi_2\rangle\phi & \leftrightarrow & \langle\pi_1\rangle\langle\pi_2\rangle\phi \\
\langle \pi_1 \Rightarrow \pi_2\rangle\phi & \leftrightarrow & [\pi_1](\pi_2\rangle\top \wedge \phi \\
\langle \eta x : \pi\rangle\phi & \leftrightarrow & \exists x\langle\pi\rangle\phi
\end{array}
$$

Using the axioms, the static meaning of the representation (6.4) can be derived by a simple calculation.

$$\eta v_i : (Wv_i; Sv_i) \Rightarrow \eta v_k : (v_k = j; Lv_k v_i). \tag{6.4}$$

This turns out to be (6.5), which is indeed an appropriate meaning representation for example sentence (2.3).

$$\forall v_i(Wv_i \rightarrow (Sv_i \rightarrow \exists v_k(v_k = j \wedge Lv_k v_i))). \tag{6.5}$$

7. Conclusion

Natural language understanding research at CWI concentrates on theoretical issues and emphasises the use of tools from programming language analysis for the analysis of natural language. It is expected, however, that the insights thus gained will greatly facilitate the task of building practically useful natural language interfaces in the not too distant future.

References

1. H. Alshawi (ed.) (1992). *The Core Language Engine*. MIT Press. Cambridge Mass. Cambridge, Mass., and London, England.

2. J. BARWISE (1987). Noun phrases, generalized quantifiers and anaphora. P. GÄRDENFORS (ed.). *Generalized Quantifiers: linguistic and logical approaches*. Reidel, Dordrecht. 1-30.

3. J. VAN EIJCK, F.J. DE VRIES (1992). Dynamic interpretation and Hoare deduction. *Journal of Logic, Language, and Information 1*. 1-44.

4. J. VAN EIJCK, A. VISSER (eds.) (1994). *Logic and Information Flow*. MIT Press, Cambridge Mass.

5. R. MONTAGUE (1973). The proper treatment of quantification in ordinary English. J. HINTIKKA E.A. (ed.). *Approaches to Natural Language*. Reidel, 221-242.

6. S.M. SHIEBER, Y. SCHABES, F.C.N. PEREIRA (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming 1(2)*. 3-56.

Computational Steering

R. van Liere, J.J. van Wijk

1. Introduction

The standard cycle in simulation is to prepare input, execute a simulation, and to visualize the results. Performing these activities simultaneously realizes greater insights and higher productivity. This is the underlying idea of *Computational Steering*: researchers change parameters of their simulation on the fly and immediately receive feedback on the effect. The development of a dedicated user-interface for a simulation is a time-consuming process, which requires the expertise of specialists on user-interfacing and computer graphics. Hence, CWI's Computational Steering project aims to develop an environment in which researchers themselves can develop and use interfaces to their simulations. The project generates considerable synergy between the system developers and researchers in a wide variety of disciplines. The project's output consists of research results in scientific visualization, tools such as CSE (Computational Steering Environment), and several applications in collaboration with researchers. The latter include a study of the atmosphere of the planet Venus (astrophysics, Vrije Universiteit Amsterdam, see below), energy research in connection with, e.g., ceramic foam burners, windmills, and high energy physics (Netherlands Energy Research Foundation ECN), and ball-bearings (the Swedish firm SKF).

2. Background

Scientific Visualization has been a separate research area since the publication of an influential report by the US National Science Foundation in 1987 [1]. Many new methods, techniques, and packages have been developed over the past several years; but most of these are confined to post-processing of data-sets. The usual assumption is that all data is generated first, after which the researcher iterates through the remaining steps of the visualization pipeline (selection, filtering, mapping, and rendering) to achieve insight in the generated data. Hence, there is only limited interaction with the simulation.

Tracking is the first step to increasing interaction with the simulation. After each time-step of the simulation the resulting data for that time-step is sent into the visualization pipeline and can be inspected. If the researcher considers the results invalid, the simulation can be stopped at an early stage, and restarted with a different set of input parameters. The next step, *Computational Steering*, goes a lot further, and can be considered as the ultimate goal of interactive computing. Computational steering enables the researcher to change parameters of the simulation while the simulation is progressing.

As an example, R.E. Marshall of the Ohio Supercomputer Center has applied computational steering to the study of a 3D turbulence model of Lake Erie. His conclusions were that: 'Interaction with the computational model and the resulting graphics display is fundamental in scientific visualization. Steering enhances productivity by greatly reducing the time between changes to model parameters and the viewing of the results.'

3. Requirements

Although computational steering is an attractive concept, implementation is cumbersome. A researcher will need to ask a specialist in user-interfaces and visualization to develop a suitable tool for analyzing the simulations' output data. And in the weeks or months that takes, the chances are high that researcher's interests have shifted. Moreover, the use of computational steering will introduce new research questions, which prompt modifications of the tool. Hence, researcher and specialist will need to collaborate for an extended period.

The Computational Steering project started up in 1992 as a joint project between CWI and the Netherlands Energy Research Foundation ECN. The aim is to develop insights, methods, techniques, and tools that enable researchers to apply computational steering. The current focus is on development of a Computational Steering Environment (CSE) that encourages exploratory investigation by the researcher of a simulation.

What are the requirements for such an environment? The first set of requirements follows directly from the definition of computational steering.

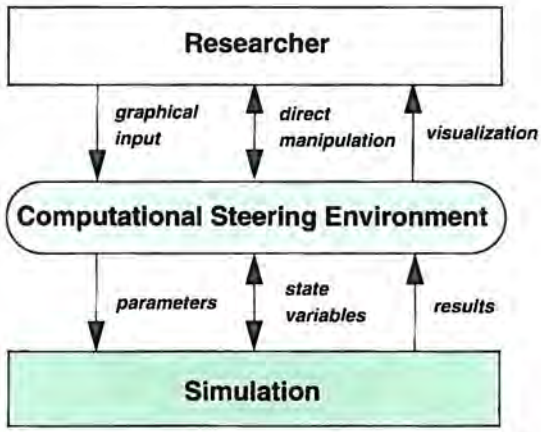


Figure 1. Data flow between researcher, CSE, and simulation.

Researchers must be enabled to change parameters and to visualize results. The simulation must be enabled to read these parameters and to write the results. Certain variables fall both in the input and output categories. For instance, in a time-dependent simulation the state variables are calculated via integration, but occasionally the user might want to change the current value (position, velocity, etc.). This implies that the CSE must support direct manipulation: the user can interact with graphics objects that are also updated by the simulation. (See also figure 1.)

Second, the environment should enable researchers to use computational steering without help from visualization experts. In the process of gaining insight via computational steering, a researcher typically wants to look at and to control other, possibly new, variables, and to visualize them in various ways. This implies that it should be easy to define and refine an interface, as well as to connect a simulation with the environment, and to control variables through the interface.

4. Architecture

Given the set of requirements, how can a solution be realized? The architecture of the environment is centered around a data manager acting as a blackboard for communicating values, and satellites that produce and visualize data [3]. The purpose of the data manager is twofold. First, it manages a database of variables. Satellites can create, open, close, read, and write variables. For each variable the data manager stores a name, type, and value. Variables can be scalars or arrays, in which case the number and

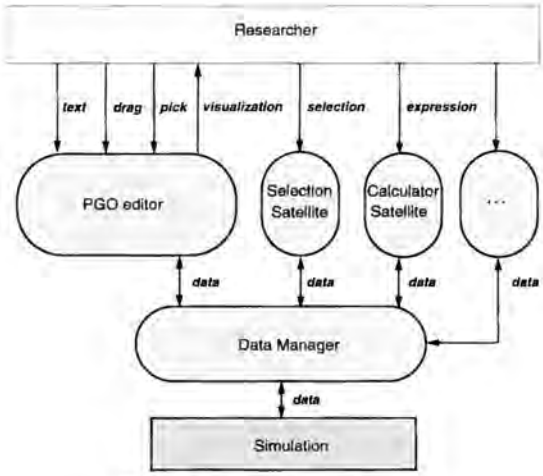


Figure 2. The CSE architecture.

size of the dimensions is also stored. Array sizes can change dynamically during the lifetime of the variable. Second, the data manager acts as an event notification manager. Satellites can subscribe to events that represent state changes in the data manager. (See also figure 2.) Whenever such an event occurs, the satellite will receive an event from the data manager. For example, if a satellite subscribes to mutation events on a particular variable, the data manager will send a notification to that satellite whenever the value of the variable is mutated.

High level libraries enable access to application interfaces to this environment. Simulations can easily connect with the data manager by simply declaring relevant variables.

390 A large suite of general purpose satellites is available for standard visualization tasks. For example, data can be logged, sliced, transformed, and calculated. However, the most important satellite is the PGO editor, a general purpose graphics editor for input and visualization of data.

5. Parameterized Graphics Objects

The PGO editor is an interactive graphics, MacDraw-like, editing tool [2]. The central concept for the graphics editor is the Parameterized Graphics Object (PGO) : an interface is built up from graphics objects whose properties are functions of data in the data manager. Users sketch an interface and bind the graphics objects to variables by parameterizing geometry and attributes with data in the data manager. Simulations may drive the interface by mutating the data bound to the graphics objects. Similarly,

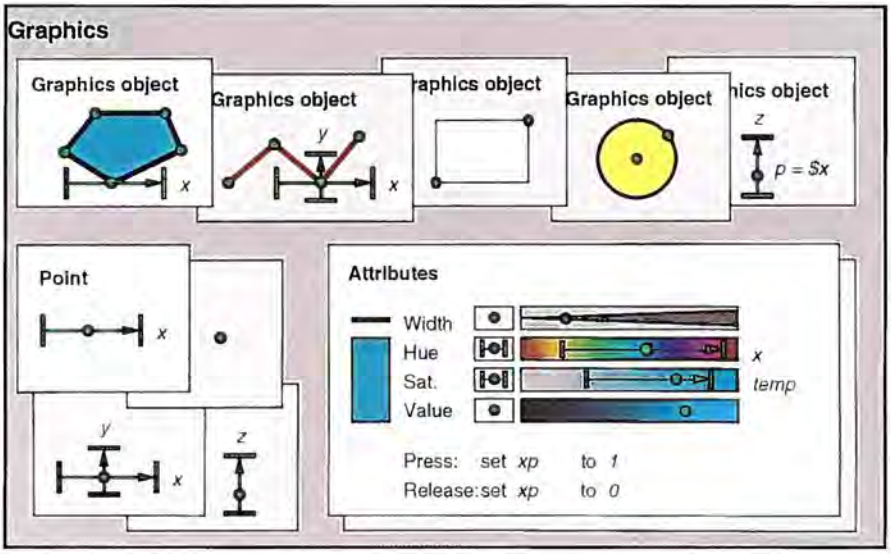


Figure 3. Parameterized Graphics Objects.

users may drive the simulation by interacting with graphics objects. Hence, a two-way communication between graphics and data in the simulation is supported. Since the PGO satellite is an interactive graphics editor, users may incrementally define the interface or change bindings. (See also figure 3.)

The graphics editor has two modes: specification and application, or shorter, *edit* and *run*. In edit-mode, the researcher can create and edit graphics objects. The geometry of the objects is defined by points and degrees of freedom. Degrees of freedom are parameterized to values of variables in the data manager. Furthermore, attributes of objects, such as color and linewidth, can also be parameterized.

In run-mode, a two-way communication is established between the researcher and the simulation. Data is retrieved from the data manager and mapped onto the properties of the graphics objects. Text can be entered, and objects can be dragged and picked, which is translated into changes of the values of variables in the data manager. Hence, there is automatic support for the usually time-consuming and error-prone direct manipulation of graphics objects.

6. Application

Several applications are underway in the Computational Steering project. Here we briefly describe one of them. Researchers at the department of

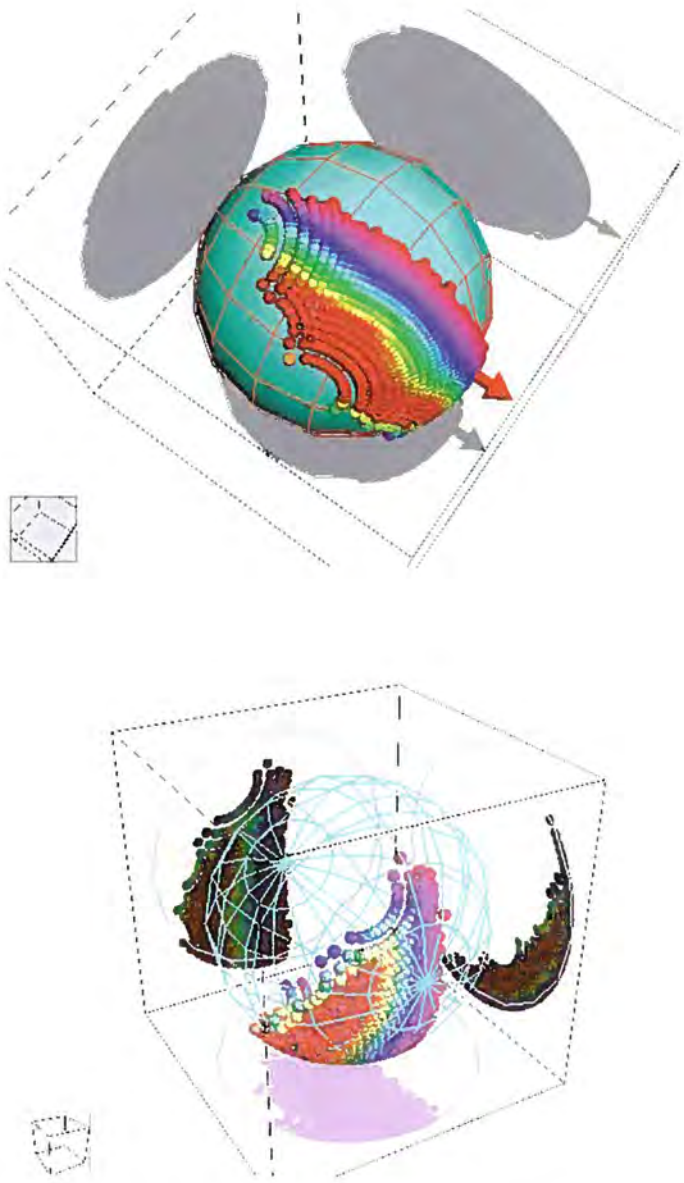


Figure 4. Two visualization snapshots, using 3D-PGO, of intensity measurements on the surface of the planet Venus, made during one day by the Pioneer satellite. The balls represent the degree of polarization.

astrophysics of the Vrije Universiteit study the structure of the atmosphere of Venus (e.g. to determine the vertical and horizontal size distribution of Sulphuric Acid droplets in the atmosphere) by comparing their computational models with measurements taken by the Pioneer satellite. Pioneer traces the atmosphere of Venus and takes measurements of polarization of the reflected sun light which is related to the structure of atmosphere. However, comparison between the measurements and the computational models is made very difficult because they are not independent: the measurements are used by the models to estimate model parameters in one dimension (a particular wavelength) prior to comparison of results in another dimension (another wavelength).

Lacking appropriate tools, the VU researchers were forced to reduce their original goal of modelling the complete Venus atmosphere to finding those points on the planet where the model parameter values are closest to the measurements and determine those values. Off-the-shelf software tools have enabled them to make many temporary and small conclusions based on calculations at relatively few points in the atmosphere.

Researchers at CWI are applying general computational steering concepts to provide more global and better insight to this application. Figure 4 shows CSE snapshots of the visualization of the intensity measurements from the Pioneer.

The CSE allows direct manipulation of modelling/simulation parameters at interactive speeds. While this type of interaction is a very important capability, current research is focussed on the development of higher level input and navigation techniques. These techniques will be developed and packaged as satellites.

References

1. B. McCormic, T. Defanti, M. Brown (1987). Visualization in scientific computing. *Computer Graphics 22(6)*. (SIGGRAPH '88). 103-111.
2. J. Mulder, J.J. van Wijk (1995). 3D computational steering with parameterized graphics objects. *Proceedings Visualization '95*. IEEE Computer Society Press, Los Alamitos, CA.
3. J.J. van Wijk, R. van Liere (1994). *An Environment for Computational Steering*. CWI Report CS-R9448, CWI, Amsterdam. *Proceedings of the Dagstuhl Seminar on Scientific Visualization*, 23-27 May 1994, Germany.

Architectures for Human-Computer Communication

A.A.M. Kuijk

1. Introduction

Human-Computer Communication deals with efficient transfer of information between humans and computers and with information structures that tie in with human conceptual abilities. The human visual system is a most powerful image cognition machine that is able to perceive, analyze, classify and evaluate a great deal of information in real-time. Therefore, interactive computer graphics can significantly enhance our ability to understand data, to perceive trends, and to visualize real or imaginary objects.

The design and development of commercially available graphics systems has primarily been driven by the availability of cost-effective hardware technology. A more proper strategy would also take into account user requirements: i.e., interaction mechanisms provided by graphics user interfaces influence the design of the underlying graphics system architecture. The design of a graphics system that strictly adheres to this concept leads to several challenging research issues.

We shall briefly describe the basics and research issues of interactive computer graphics. Next we will outline the research activities carried out at CWI.

2. Graphics and interaction

A graphics system provides facilities to

- specify (or model) a scene in terms of a set of logical graphical elements:

- specify parameters according to which the scene is transformed into an image;
- handle graphical input.

The scene is mapped onto the display- or image-space. This mapping (also known as rendering) includes both geometric and attribute transformations. The result, a set of display primitives which describe an image, is stored in a refresh buffer. The type of display primitives in which the image is described depends on the type of display technology in play (e.g., a set of pixels for a raster display or a set of line drawing instructions for a vector display).

A logical model of the graphics system envisions the mapping of a scene onto the display space as a stepwise process in which primitives travel through a pipeline of functional modules (see figure 1). This model distinguishes several logical representation levels of the scene that exist in the image synthesis pipeline and the operations applicable on these representations. Each module of the pipeline performs an elementary graphical operation on all passing primitives. Graphical input originates from data that comes in via physical input devices. The raw, device dependent input data, are converted into a set of input primitives. This conversion is handled completely within a logical input model so that all types of physical input devices and associated user actions are encapsulated within this logical model.

Actions involved in generating graphical feedback upon user input include:

- handling of the logical input data;
- handling of the input primitives;
- possible updates of some application data;
- traversing (part of) the graphics pipeline;
- update of the refresh buffer.

Response on user input is a key determining factor of the quality of the user interface. A system's response time depends on several factors. Obvious factors are the amount of processing involved in the feedback-loop and the raw performance of the computing resources. Other factors are operating system related dependencies such as interrupt handling, context switching and the like.

3. Research issues

Effective interactive computer graphics applications require considerable computing resources to guarantee a sufficiently fast response. In the last decade, we have witnessed a remarkable improvement of computing power due to improved processor technologies. This evolution can be expected

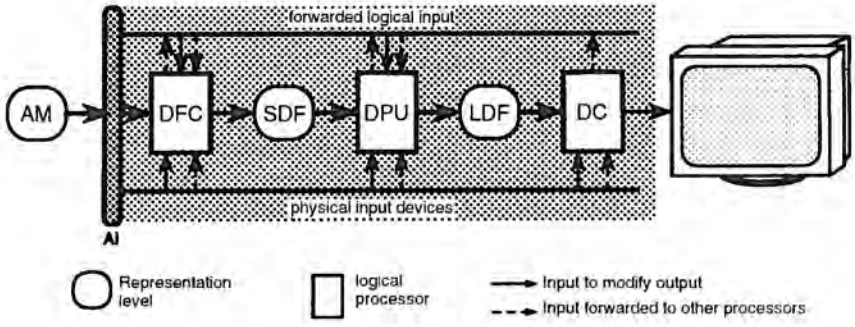


Figure 1. Logical model of image generation

to continue. However, to satisfy the need for higher image quality, scene complexity and interactivity, experts in the field estimate that four to five orders of magnitude more processing power than available in present day processors is necessary. Such a gap can only be bridged by making use of highly parallel multiprocessor systems.

The inherent nature of graphics algorithms makes that they are well suited to be implemented on multiprocessor systems. In increasing levels of complexity, the computing tasks can be organized based on pixels, vertices, polygons, patches, objects or frames. Furthermore, the image generation pipeline consists of a number of clearly separable tasks. As a result graphics algorithms can be mapped onto numerous multiprocessor configuration alternatives.

Many solutions have been proposed to distribute the work involved in fast generation of high quality images across multiprocessor systems [1]. The image generation process itself is a pipeline of several sequential processes which automatically suggests functional subdivision (see figure 2 on the left). Subdivision of the image generation pipeline in smaller tasks can be done up to a limited number of steps only, so that the maximum degree of multiprocessing by means of pipelining only is limited. Also data dependent operations make it hard to balance the load for a purely pipelined architecture. Image-space partitioning, i.e., subdivision of the image in small parts that are handled by separate subsystems (see figure 2 in the middle), implies that all primitives have to be processed by all processors (i.e., the system is object-serial). Hence the throughput is limited by the processor speed. Object-space partitioning, i.e., each object is handled by one of the processors of the system (see figure 2 on the right), results in multiple pixel

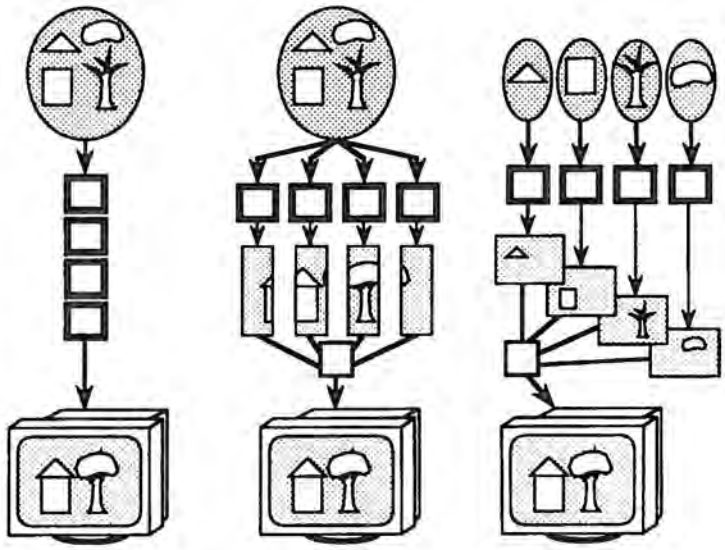


Figure 2. Subdivision strategies.

streams that have to be combined (i.e., the system is pixel-serial). A simple solution implies synchronization which reduces efficiency. An asynchronous solution requires extra memory and composition hardware. Practical solutions in both commercial and academic systems are often mixtures of several of these three 'pure' subdivision strategies.

Presently there exist two mainstream approaches to make raster graphics systems more viable for interaction:

- Perform each step involved in the graphical feedback loop as quickly as possible by pushing the hardware limits to the maximum, viz., running many processors per given task in the image synthesis pipeline.
- Restructure the functional model to reduce the effort needed to complete the graphical feedback loop, viz., looking at image synthesis from a fresher perspective.

A common characteristic of the first approach is to identify and isolate a simple (subset of) operation(s) and map it, frequently in a conceptually simplistic manner, to hardware.

The second approach is no different from the first one in terms of its goal, i.e., produce a responsive raster graphics system. Yet, the methodology is

quite different. In this case, one analyzes interaction tasks and then tries to develop original data structures and devise new architectural organizations which guarantee that for all interaction tasks representations of the proper level are at hand. Only then one maps expedient tasks into hardware as much as this is justified.

4. Research at CWI

The computational complexity community has long ago come to know that the laws of parallel computation are qualitatively different from that of the sequential computation; algorithms do not always smoothly translate from uniprocessor to multiprocessor architectures. We believe that without clarifying the algorithmic improvements, brute-force mappings of existing graphical algorithms into hardware will introduce only temporary speed-ups and these improvements will be nullified in time by growing user demands. The real solution to the hard problems of computer graphics will come, in our view, from a direction which considers the intrinsic difficulty of user driven problems from a computational standpoint. Therefore, research at CWI adhered to the above mentioned second approach: first examine the structure of the image synthesis pipeline in relation with interaction requirements, and only then try to push the hardware limits to the maximum where this is needed [4].

We observed that in an interactive computer graphics application a user interacts with a three-dimensional model (or object) at several levels of abstraction. For an efficient support of interactive editing and incremental updates a raster independent representation of the three-dimensional model should be immediately available at each of these levels. This forms the basis of a lean, yet flexible, computation model.

As a consequence of this raster independent object-space paradigm, the research at CWI has focussed on explicit identification of all visible surfaces, shading methods, rasterization hardware and adaptive rendering.

4.1. Visible surfaces

A fundamental step in generating images of 3D scenes is clipping and hidden surface removal: the resulting image exists of (parts of) surfaces that are visible from a certain position in space. Several types of algorithms exist to tackle this classical computer graphics problem [3]. In most graphics systems hidden surface removal is supported by hardware that checks the visibility on pixel level (the so-called z-buffer algorithm). However, one of the levels of abstraction with which the user interacts most frequently is the level which contains only all visible surfaces. Such a level is not available in a pixel-based z-buffer architecture.

Explicit identification of all visible surfaces implies that the visibility calculation takes place in object-space. Interactive applications involve incre-

mental picture changes. The research at CWI resulted in a hidden surface removal algorithm which includes a set of logical operations on 3D objects. These operations can be used to add and delete individual objects so that incremental changes affect only those objects of which the visibility is changed. Thus a firm basis for interaction and animation is established. Our algorithm operates on a pre-sorted representation of objects and a geometry-based data structure to store these objects. This specific representation of objects reduces the complexity of both the hidden surface removal and the scan-conversion process. The data structure reduces the search space for geometry-based object identification and facilitates data distribution for a multiprocessor implementation.

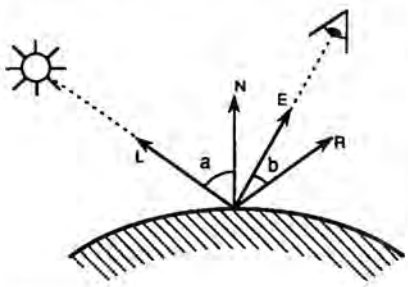


Figure 3. Vectors involved in Phong's illumination model.

4.2. Shading methods

By looking around us in the real world we observe the result of rather complicated physics: the interaction of photons with the inhomogeneous entities that make up the physical environment. This reality is far too complicated to simulate accurately. Therefore computer generated images are produced using a simplified illumination model that describes the interaction between light and the elements of the simulated 3D environment.

Most popular shading methods are based on the illumination model developed by B.T. Phong [2] that has the potential to produce remarkably realistic results, in spite of its simplicity. The model incorporates ambient, diffuse and specular components. The intensity vector I is calculated using the expression

$$I = I_{\text{amb}} + \sum_{\text{sources}} I_{\text{light}} \cdot ((N \cdot L) + (E \cdot R)^n).$$

In this expression I_{amb} represents the amount of energy of the indirect light cast upon the surface area by the environment. I_{light} is the intensity of the light source. **N** is the surface normal. **L** is the direction of the light source. **E** is the direction of the viewpoint. **R** is the direction of reflection and n is a coefficient which relates to the reflectivity of the surface. The vectors **N**, **L**, **E** and **R** are normalized (see figure 3).

The Phong shading method, known since 1975, based on this illumination model involves calculation of the intensity across polygons (a graphics area primitive) based on interpolated vectors on a per-pixel basis. Due to the costs involved (which includes renormalization of interpolated vectors and calculation of the above expression for each individual pixel), this method

Figure 4. Examples of the most advanced shading methods presently used in interactive computer graphics.

is not suited for high speed rasterization. We developed a similar shading method in which interpolation of vectors involved in the calculation is interpreted as rotations. Spherical trigonometry then leads to a linear expression of the angle between the vectors along a scanline. The outcome is a parameterized piecewise quadratic expression for each intensity term. These terms can be handled directly by forward differencing. The image quality obtained is virtually the same as the quality of Phong shaded images.

4.3. Rasterization hardware

A display controller handles the refresh process of a graphics display device. We developed a display controller which reads 'area drawing instructions' and which in real-time produces scanline based video signals to control the electron beam which scans the display area. The path of the electron beam prescribes the organization of the scan-conversion process (see figure 5). The vertical phase of the scan-conversion process involves calculation of the

intersection of objects with a horizontal scanline and determination of the colour function along that scanline. The actual rasterization takes place in the horizontal phase of the scan-conversion process. As a result of our research on illumination models, this can be reduced to relatively simple operations repeated numerous times. For tasks like this we opted for a custom designed high speed 36-bit forward differencing engine, implemented as a highly pipelined systolic array.

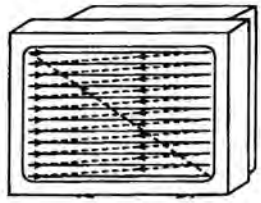


Figure 5. Scanpath of raster display device.

A working prototype system (figure 6) has been built to provide further insight into the operation of the technologically challenging part of the system: the custom VLSI Difference Engine. This prototype produces pictures on a CRT display directly from instructions and without buffering images in a frame buffer.

The Difference Engine, which was developed as a very specialized pixel generator is really very general: it can handle any order forward differences with 36-bit numerical accuracy and an 11ns cycle time. The spline interpolation goes with constant cost independent of span length. Since the Difference Engine can interpolate any spline (polynomial) curve, any signal that is expressed in terms of a spline basis can be reconstructed. Not only that, the architecture with its accumulator allows one to sum over incrementally generated output so that the splines can be summed over different scales to produce the final image to any required accuracy. The reconstruction time depends not on the spacing of the knots in the splines (the lengths of the interpolation spans) but only on the number of knots. An image can be decompressed even at video rates provided that the number of knots is less than the number of pixels to be generated (by some fixed overhead per scanline).

This has opened the way for using this hardware also to reconstruct images that have been coded with a wavelet transform [5]. The wavelet transform is a multiresolution description of the image that can be decoded to yield more and more accurate reconstructions of an image. The transform also precisely locates high-frequency features in space and low-frequency signals in the frequency domain. In fact it is often argued that wavelet transforms perform better than the discrete cosine transform advocated by the JPEG standard, it fits in better with human perceptual aptitudes and is a more compact coding.

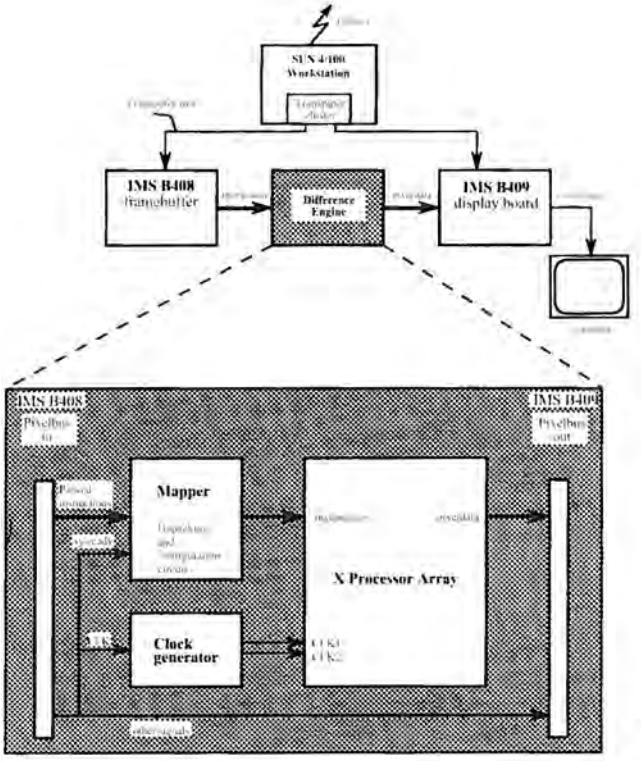


Figure 6. Difference Engine embedded in prototype.

4.4. Adaptive rendering

Generating a physically perfect image would by far exceed the processing power of any state of the art supercomputer. Due to this, a whole scale of rendering models emerged, each with a different level of approximation of the 'physical correct' image. Based on requirements of a specific application, one of these rendering models can be selected. If the system is tuned for worst case situations the efficiency will be far from optimal. On the other hand if the system is optimized for 'normal' situations its worst case behaviour may be unacceptable.

Adaptive image generation is a means to adjust the image generating process to the possibilities of a particular moment. The quality of the image, and thus the cost of rendering, is related to the time available between successive updates. Ideally, this results in the best possible image at any time. There are two distinct aspects that determine the cost of the image generation process: the quality of the rendering process, and the quality of

the object representation. These can be varied according to the need.

For a study on adaptive rendering we implemented an environment to test a rule-based system (ADMIRE). This rule-based system serves to optimize the performance of interactive graphics applications by dynamically selecting rendering algorithms, data structures and level of detail on a per-object basis.

5. Conclusion

A thorough rethink of interactive graphical workstations from a user point of view has uncovered a range of novel approaches to system architectures. CWI has built an integrated set of solutions based on these insights that span conceptual, software and hardware solutions. Interesting spin-offs in image compression are also being exploited.

References

1. S. Molnar, H. Fuchs (1990). Advanced raster graphics architecture. *Computer Graphics: Principles and Practice*. The Systems Programming Series, Addison-Wesley Publishing Company, Reading, Massachusetts, 855-920.
2. B.T. Phong (1975). Illumination for computer generated images. *Communications of the ACM 18(6)*, 311-317.
3. I.E. Sutherland, R.F. Sproull, R.A. Schumacker (1974). A characterization of ten hidden-surface algorithms. *Computing Surveys 6(1)*, 1-55.
4. A.A.M. Kuijk, E.H. Blake, P.J.W. ten Hagen (1992). *An Architecture for Interactive Raster Graphics*, CWI Report CS-R9229.
5. P.C. Marais, E.H. Blake, A.A.M. Kuijk (1993). A spline-wavelet image decomposition for a difference engine. *CWI Quarterly 6(4)*, 335-362.

Coordination of Cooperative Agents

F. Arbab

1. Introduction

The ever decreasing costs and sizes of processors, their ever increasing speeds, faster and wider-band-width communication links, and global networks have made the potential of applying the computational power of several (even hundreds and thousands) of processors to a single application, a reality. Conceptually, the significance of the availability of so many processors to work on an application goes beyond performance issues.

The mere idea of allocating more than one *worker* to the same task immediately opens up a new problem solving paradigm, and simultaneously, presents a new challenge. The paradigm is *concurrency*, and the challenge is *coordination*.

We consider the problem of coordination of very large numbers of concurrent active entities that must cooperate with each other in the context of a single application. We give a brief problem description and history in section 2. In section 3 we distinguish between *communication* and *cooperation*, and show the need for a coherent model and language to describe the cooperation protocols of active entities in massively concurrent systems. Next two models are compared, the currently most widely used *Targeted-Send/Receive*, or TSR model and a new coordination model, *Idealized Worker Idealized Manager* (IWIM) model. A specific coordination language [1,2], called **MANIFOLD**, that is based on the generic model proposed in section 3 is described in section 4. Some of the interesting features

of MANIFOLD are shown through examples in this section. The conclusion of the paper is in section 5.

2. Concurrent programming

Concurrency is about the expression of a computation as a set of concurrent activities. As such, it is a problem solving or a programming paradigm. Parallelism is about allocating more resources to carry out a given computation. As such, it is a method for realizing a solution, i.e., to carry out a computation.

Of course, the study and the application of concurrency in computer science has a long history. The study of deadlocks, the dining philosophers, and the definition of semaphores and monitors were all well established by the early seventies. However, it is illuminating to note that the original context for the interest in concurrency was somewhat different than today in two respects:

- In the early days of computing, hardware resources were prohibitively expensive and had to be shared among several programs that had nothing to do with each other, except for the fact that they were unlucky enough to have to compete with each other for a share of the same resources. This was the *concurrency of competition*.

- The falling costs of processor and communication hardware only recently dropped below the threshold where having very large numbers of 'active entities' in an application makes sense. Thus, it is no more unrealistic to think that a single application can be composed of hundreds of thousands of active entities. This is the *concurrency of cooperation*. Compared to classical uses of concurrency, this is a jump of several orders of magnitude in numbers, and in our view, represents (the need for) a qualitative change.

Theoretical work in this area, e.g., π-calculus or process algebra, is still too fundamental to be used directly in large concurrent applications. On the other hand, the tried and true models of cooperation, such as client-server, barrier synchronization, etc., that are used in practical applications of today are simply a set of ad-hoc, special-case templates; they do not constitute a coherent paradigm for the definition of cooperation protocols.

We believe there is a clear need for programming models that explicitly deal with concurrency of cooperation among very large numbers of active entities that comprise a single application. Such models cannot be built as extensions of the sequential programming paradigm. Because such applications can be distributed over a network, we believe such models cannot be based on synchronous models of concurrency.

3. Communication vs. cooperation models

It is important to distinguish between the conceptual model describing the cooperation of a number of concurrent processes in an application, and the underlying model of communication on top of which such cooperation is implemented [3].

The primary concern in the design of a concurrent application must be its model of cooperation: how the various active entities comprising the application are to cooperate with each other. Eventually, a communication model must be used to realize whatever model of cooperation application designers opt for, and the concerns for performance may indirectly affect their design. Nevertheless, it is important to realize that the conceptual gap between the system supported communication primitives and a concurrent application must often be filled with a non-trivial model of cooperation.

There is no paradigm wherein we can systematically talk about cooperation of active entities, and wherein we can compose cooperation scenarios. Consequently, programmers must directly deal with the lower-level communication primitives that comprise the realization of the cooperation model of a concurrent application. Because these primitives are generally scattered throughout the source code of the application and are typically intermixed with non-communication application code, the cooperation model of an application generally never manifests itself in a tangible form – i.e., it is not an identifiable piece of source code that can be designed, developed, debugged, maintained, and reused, in isolation from the rest of the application code.

3.1. The TSR model of communication

A common characteristic of most flavours of the message passing model of communication is the distinction between the roles they assign to the two active entities involved in a communication: the sender and the receiver. A sender s typically sends a message m to a receiver r. The send operation is generally targeted to a specific (set of) receiver(s). A receiver r, on the other hand, typically waits to receive a message m from any sender, as it normally has no prior knowledge of the origin of the message(s) it may receive. We use the term *Targeted-Send/Receive*, or TSR, to refer to the communication models that share this characteristic.

Consider the following simple example of a concurrent application where the two active entities (i.e., processes) p and q must cooperate with each other. The process p at some point produces two values which it must pass on to q. The source code for this concurrent application looks something like the following:

process p:	process q:
compute m1	receive m1
send m1 to q	let z be the sender of m1
compute m2	receive m2
send m2 to q	compute m using m1 and m2
do other things	send m to z
receive m	
do other computation using m	

The first significant point in the above listing is that the communication concerns are mixed and interspersed with computation. This decreases the understandability, the maintenability and re-usability of the cooperation model.

The second significant point to note is the need of specifying a target for the send and the asymmetry between send and receive operations. Targeted send strengthens the dependence of individual processes on their environment. This too diminishes the reusability and maintenability of processes. In this model, debugging and proving programs correct are also not trivial: a process is not a well-encapsulated concept into this model, it sometimes needs the existence of some other valid processes to be valid.

3.2. The IWIM model of communication

In the following, we consider an alternative generic model of communication that, unlike the TSR model, supports the separation of responsibilities and encourages a weak dependence of workers (processes) on their environment. We refer to this generic model as the *Idealized Worker Idealized Manager* (IWIM) model.

The basic concepts in the IWIM model are *processes*, *events*, *ports*, and *channels*. A process is a *black box* with well-defined ports of connection through which it exchanges *units* of information with the other processes in its environment. A port is a named opening in the bounding walls of a process through which units of information are exchanged using standard I/O type primitives analogous to read and write. Without loss of generality, we assume that each port is used for the exchange of information in only one direction: either into (input port) or out of (output port) a process. We use the notation $p.i$ to refer to the port i of the process instance p.

The interconnections between the ports of processes are made through channels. A channel connects a (port of a) producer (process) to a (port of a) consumer (process). We write $p.o \rightarrow q.i$ to denote a channel connecting the port o of the producer process p to the port i of the consumer process q.

Independent of the channels, there is an event mechanism for information

exchange in IWIM. Events are broadcast by their sources in their environment, yielding an *event occurrence*.

The IWIM model supports *anonymous communication*: in general, a process does not, and need not, know the identity of the processes with which it exchanges information. This concept reduces the dependence of a process on its environment and makes processes more reusable.

A process in IWIM can be regarded as a worker process or a manager (or coordinator) process. The responsibility of a worker process is to perform a (computational) task. A worker process is not responsible for the communication that is necessary for it to obtain the proper input it requires to perform its task, nor is it responsible for the communication that is necessary to deliver the results it produces to their proper recipients. In general, *no process in* IWIM *is responsible for its own communication with other processes*. It is always the responsibility of a manager process to arrange for and to coordinate the necessary communications among a set of worker processes.

There is always a bottom layer of worker processes, called *atomic workers*, in an application. In the IWIM model, an application is built as a (dynamic) hierarchy of (worker and manager) processes on top of this layer. Note that a manager process may itself be considered as a worker process by another manager process. Let us reconsider the example in subsection 3.1., and see how it can be done in the IWIM model. A new process c responsible to facilitate the communication has been created. The source code looks something like the following:

process p:	process q:	process c:
compute m1	read m1 from input port i1	...
write m1 to output port o1	read m2 from input port i2	create the channel p.o1 →q.i1
compute m2	compute m using m1 and m2	create the channel p.o2 →q.i2
write m2 to output port o2	write m to output port o1	create the channel q.o1 →p.i1
do other things		...
read m from input port i1		
do other computation using m		

In this example, the responsibility of the coordinator process c is, very simple: perhaps, it first creates the processes p and q, establishes the communication channels defined above, and then may wait for the proper condition (e.g., termination of p and/or q) to dismantle these channels and terminate itself.

Nevertheless, moving the communication concerns out of p and q and into c already shows some of the advantages of the IWIM model. The processes

p and q are now 'ideal' workers. They do not know and do not care where their input comes from, nor where their output goes to. They know nothing about the pattern of cooperation in this application: they can just as easily be incorporated in any other application, and will do their job provided that they receive 'the right' input at the right time. The cooperation model of this application is now explicit: it is embedded in the coordinator process c. If we wish to have the output of q delivered to another process, or to have yet another process deliver the input of p, neither p nor q, but only c is to be modified.

The process c is an 'ideal' manager. It knows nothing about the details of the tasks performed by p and q. Its only concern is to ensure that they are created at the right time, receive the right input from the right sources, and deliver their results to the right sinks. It also knows when additional new process instances are supposed to be created, how the network of communication channels among processes must change in reaction to significant event occurrences, etc. (none of which is actually a concern in this simple example).

Removing the communication concerns out of worker processes enhances the modularity and the re-usability of the resulting software. Furthermore, the fact that such ideal manager processes know nothing about the tasks performed by the workers they coordinate, makes them generic and re-usable too. The cooperation protocols for a concurrent application can be developed modularly as a set of coordinator processes. It is likely that some of such ideal managers, individually or collectively, may be used in other applications, coordinating very different worker processes, producing very different results; as long as their cooperation follows the same protocol, the same coordinator processes can be used. Modularity and re-usability of the coordinator processes also enhances the re-usability of the resulting software.

4. MANIFOLD

In this section, we briefly introduce **MANIFOLD**: a coordination language for managing complex, dynamically changing interconnections among sets of independent, concurrent, cooperating processes [2], which is based on the IWIM model, described in subsection 3.2.

The **MANIFOLD** system consists of a compiler, a run-time system library, a number of utility programs, and libraries of built-in and predefined processes [4]. A **MANIFOLD** application consists of a (potentially very large) number of processes running on a network of heterogeneous hosts; some of which may be parallel systems. Processes in the same application may be written in different programming languages. Some of them may not know anything about **MANIFOLD**, nor the fact that they are cooperating with other processes through **MANIFOLD** in a concurrent application.

The library routines that comprise the interface between **MANIFOLD** and processes written in other languages (e.g., C), automatically perform the necessary data format conversions when data is routed between various different machines.

4.1. Processes

The atomic workers of the IWIM model are called atomic processes in **MANIFOLD**. Any operating system-level process can be used as an atomic process in **MANIFOLD**. However, **MANIFOLD** also provides a library of functions that can be called from a regular C function running as an atomic process, to support a more appropriate interface between the atomic processes and the **MANIFOLD** world. Atomic processes can only produce and consume units through their ports, generate and receive events, and compute. In this way, the desired separation of computation and coordination is achieved.

Coordination processes are written in the **MANIFOLD** language and are called manifolds. The **MANIFOLD** language is a block-structured, declarative, event driven language. A manifold definition consists of a header and a body. The header of a manifold gives its name, the number and types of its parameters, and the names of its input and output ports. The body of a manifold definition is a block. A block consists of a finite number of states. Each state has a label and a body. The label of a state defines the condition under which a transition to that state is possible. It is an expression that can match observed event occurrences in the event memory of the manifold. The body of a simple state defines the set of actions that are to be performed upon transition to that state. The body of a compound state is either a (nested) block, or a call to a parameterized subprogram known as a *manner* in **MANIFOLD**. A manner consists of a header and a body. As for the subprograms in other languages, the header of a manner essentially defines its name and the types and the number of its parameters. A manner is either atomic or regular. The body of a regular manner is a block. The body of an atomic manner is a C function that can interface with the **MANIFOLD** world through the same interface library as for the compliant atomic processes.

4.2. Streams

The asynchronous communication channels in **MANIFOLD** are called *streams*. A stream has an infinite capacity that is used as a FIFO queue, enabling asynchronous production and consumption of units by its source and sink. The sink of a stream requiring a unit is suspended only if no units are available in the stream. The suspended sink is resumed as soon as the next unit becomes available for its consumption. The source of a stream is never suspended because the infinite buffer capacity of a stream is never filled.

Note that as in the IWIM model, the constructor of a stream between two processes is, in general, a third process. Stream definitions in **MANIFOLD** are generally additive. This means that a port can simultaneously be connected to many different ports through different streams. The flows of information units in streams are automatically replicated and merged at outgoing and incoming port junctions, as necessary. Thus, a unit placed into a port that is connected to more than one outgoing streams is duplicated automatically, with a separate copy placed into each outgoing stream. Analogously, when a process attempts to fetch a unit from a port that is connected to several incoming streams, it obtains the first unit available in a non-empty incoming stream, selected non-deterministically.

4.3. Events

In **MANIFOLD**, once an event is generated by a process, it continues with its processing, while the event occurrence propagates through the environment independently. Any receiver process that is interested in such an event occurrence will automatically receive it in its *event memory*. The observed event occurrences in the event memory of a process can be examined and reacted on by this process at its own leisure. The event memory of a process behaves as a set: there can be at most one copy of the occurrence of the same event generated by the same source in the event memory. If an event source repeatedly generates an event faster than an observer reacts on that event occurrence, the event memory of the observer induces an automatic sampling effect: the observer detects only one such event occurrence.

4.4. State transitions

The most important primitive actions in a simple state body are: create and activate processes, generate event occurrences, and connect streams between ports of various processes. Upon transition to a state, the actions specified in its body are performed atomically in some non-deterministic order. Then, the state becomes *preemptable*: if the conditions for transition to another state are satisfied, the current state is preempted, meaning that all streams that have been constructed are dismantled and a transition to a new state takes place. This event-driven state transition mechanism is the only control mechanism in the **MANIFOLD** language. More familiar control structures, such as the sequential flow of control represented by the connective ';' (as in Pascal and C), conditional (i.e., 'if') constructs, and loop constructs can be built out of this event mechanism, and are also available in the **MANIFOLD** language as convenience features.

4.5. Example: Fibonacci series

It is beyond the scope of this paper to present the details of the syntax and semantics of the **MANIFOLD** language. However, because **MANIFOLD** is not

very similar to any other well-known language, in this section we present a simple example to illustrate its features and the capabilities. The purpose of the program shown below is to print the Fibonacci series, defined as: $f(1) = 1$, $f(2) = 2$, $f(n) = f(n-1) + f(n-2)$, for $n > 2$.

The first line of this code defines a manifold named `PrintUnits` that takes no arguments, and states (through the keyword `import`) that the real definition of its body is contained in another source file. This defines the 'interface' to a process type definition whose actual 'implementation' is given elsewhere. Whether the actual implementation of this process is an atomic process (e.g., a C function) or it is itself another manifold is indeed irrelevant in this source file. We assume that `PrintUnits` waits to receive units through its standard input port and prints them. When `PrintUnits` detects that there are no incoming streams left connected to its input port and it has done printing the units it has received, it terminates.

In this program, we use two other imported manifolds: `variable` and `sum`. The manifold `variable` reads units from its `input` port, and produces a copy of the most recent received unit whenever a stream is connected to its `output` port. The parameter specifies an initial value.

In the specification of `sum` there is a new linguistic element. In addition to the default ports that all manifolds have, this manifold has two input ports named `x` and `y`. The interface declaration of `sum`, thus, contains the declarations for these ports.

```
manifold PrintUnits() import.
manifold variable(port in) import.
manifold sum(event)
  port in x.
  port in y.
  import.
event overflow.

auto process v0 is variable(0).
auto process v1 is variable(1).
auto process print is PrintUnits.
auto process sigma is sum(overflow).

manifold Main()
{
  begin: (v0 → sigma.x, v1 → sigma.y, v1 → v0, sigma → v1, sigma → print).
  overflow.sigma: halt.
}
```


An instance of `sum` reads a pair of units, one from each of its input ports `x` and `y`, verifies that they contain numeric values, adds them together, and produces the result in a unit on its `output` port. It then tries to obtain a new pair of input units to produce their sum, and continues to do so indefinitely, as long as its input ports are still connected to incoming streams. If a pair of

input values are so large that their addition causes an overflow. `sum` produces a special error unit on its `output` and generates the event `overflow`. Next, is the declaration of `overflow` as an event.

The following four lines define new instances of the manifolds `variables`, `PrintUnits`, and `sum`, and state (through the keyword `auto`) that these process instances are to be automatically activated upon creation, and deactivated upon departure from the scope wherein it is defined; in this case, this is the end of the application. Because the declaration of the process instance `print` appears outside of any blocks in this source file, it is a global process, known by every instance of every manifold whose body is defined in this source file.

The last lines of this code define a manifold named `Main` that takes no parameters. Every manifold definition (and therefore every process instance) always has at least three default ports: `input`, `output`, and `error`. The definition of these ports are not shown in this example, but the ports are defined for `Main` by default. The body of this manifold is a block (enclosed in a pair of braces) and contains only a single state. The name `Main` is indeed special in MANIFOLD: there must be a manifold with that name in every MANIFOLD application and an automatically created instance of this manifold, called `main`, is the first process that is started up in an application. Activation of a manifold instance automatically generates an (internal) occurrence of the special event `begin` in the event memory of that process instance; in this case, `main`. This makes the initial state transition possible: `main` enters its only state—the `begin` state.

Figure 1 shows the connections made among various processes in the `begin` state of `main`. In order to understand how our set of connections produces the Fibonacci series, we consider the sequence of units that flow through each stream. Let α be the sequence of units produced through the `output` port of the process `sigma`. Clearly, this is the sequence of units printed by `print`, and we want to show that it is indeed the Fibonacci series.

The sequence of units that show up at the `input` port of `v1` is, obviously,

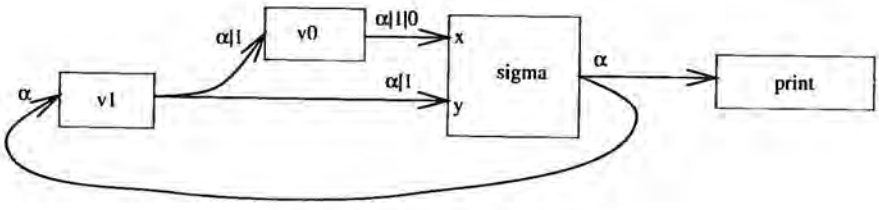


Figure 1. The Fibonacci series.

α. This means that the sequence of units produced through the `output` port of `v1` consists of '1' (the initial value of `v1`) followed by α. This same sequence shows up at the `input` port of `v0` and at the port `y` of `sigma`. It follows that the sequence of units produced through the `output` port of `v0` (which shows up at the `x` port of `sigma`) consists of '0' (the initial value of `v0`), followed by '1', followed by α.

Now, observe that the first pair of units that arrive at the ports `x` and `y` of `sigma` contain respectively, '0' and '1'. Thus, by the definition of `sum` (of which `sigma` is an instance), the first unit in α contains '0 + 1', i.e., '1'. Therefore, the second pair of units that arrive at the ports `x` and `y` of `sigma` contain respectively, '1' and '1' (the first unit in α). Hence, the second unit in α contains '1 + 1', i.e., '2'. The third pair of units that arrive at the ports `x` and `y` of `sigma` contain respectively, '1' (the first unit in α) and '2' (the second unit in α), which produces a '3' for the third unit in α.

This configuration of processes will continue to produce the Fibonacci numbers 1, 2, 3, 5, 8, 13, 21, 34, 55, etc., until `sigma` encounters an overflow. In reaction to the occurrence of the event `overflow` generated by `sigma`, `main` makes a transition to its corresponding state. The transition out of the `begin` state preempts (i.e., breaks up) its stream connections. Both `sigma` and `print` terminate as soon as they detect they have no incoming streams. The transition into the new state executes the action `halt`, which terminates the `main` process.

This simple example shows the power of the 'plumbing paradigm' that is the basis of IWIM and MANIFOLD. What is not demonstrated in this example is the dynamic capabilities of MANIFOLD: that processes can be dynamically created and deleted, and the topology of their interconnecting streams can dynamically change in reaction to the events of interest. Some such examples are presented elsewhere, e.g., in [2, 3].

Separation of computation from communication concerns which is enforced by MANIFOLD leads to separate modules for computation and coordination of communication. This enhances the re-usability of modules, especially, that of the communication modules which are the most complex and time consuming parts of a parallel/distributed application. As a concrete example of this notion of re-usable coordinator modules, it is worth mentioning that a MANIFOLD program written to implement a parallel/distributed bucket sort algorithm was later used, with no change, to implement a single-grid domain decomposition numerical algorithm. It turned out that, although the computations performed in the sort and the domain decomposition problems are very different, the coordination models they required were exactly the same. Furthermore, extension of the domain decomposition problem to the multi-grid case, required only a small change (the addition of a feed-back stream) to this coordinator module.

No modification to any source code is necessary when a MANIFOLD ap-

plication is to run on a single processor machine, a multiprocessor machine, or on a (homogeneous or heterogeneous) network of such machines.

5. Conclusion

In this paper, we illustrate the shortcomings of the direct use of communication models that are based on 'Targeted-Send/Receive' primitives in large concurrent applications. We argue that there is an urgent need for practical models and languages wherein various models of cooperation can be built out of simple primitives and structuring constructs.

We present the IWIM model as a suitable basis for control-oriented coordination languages. The significant characteristics of the IWIM model include compositionality, which it inherits from data-flow, anonymous communication, and separation of computation concerns from communication concerns. These characteristics lead to clear advantages in large concurrent applications.

MANIFOLD is a specific coordination language based on the IWIM model. **MANIFOLD** uses the concepts of modern programming languages to describe and manage connections among a set of independent processes. The unique blend of event driven and data driven styles of programming, together with the dynamic connection graph of streams seem to provide a promising paradigm for concurrent systems. The emphasis of **MANIFOLD** is on orchestration of the interactions among a set of autonomous *agents*, each providing a well-defined segregated piece of computation, into an integrated concurrent system for accomplishing a larger task.

The **MANIFOLD** system runs on multiple platforms. Presently, it runs on IBM RS6000, IBM SP1/2, HP, SUNOS, Solaris, and SGI IRIX. Linux and Cray Unicos ports are under way, and other ports are planned. Our present and future work involving **MANIFOLD** includes completion of a visual programming and debugging environment we are developing for **MANIFOLD**, and using **MANIFOLD** in industrial High Performance Computing applications.

References

1. D. Gelernter, N. Carriero (1992). Coordination languages and their significance. *Communications of the ACM 35*, 97-107.
2. F. Arbab, I. Herman, P. Spilling (1993). An overview of manifold and its implementation. *Concurrency: Practice and Experience 5*, 23-70.
3. F. Arbab (1995). *Coordination of Massively Concurrent Activities*. CWI report CS-R9564, Amsterdam.
4. F. Arbab (1995). *Manifold Version 2: Language Reference Manual*. Tech. Rep. preliminary version, CWI Amsterdam.

From Paper Plotters to Interactive Multimedia Systems

M. Bakker, P.J.W. ten Hagen

1. Introduction

CWI's earliest activity in computer graphics dates back to the 1960s, when it manufactured the X1 plotter and engineered a library of ALGOL 60 plot procedures. Today, CWI is one of the driving forces behind the production of two multimedia applications: the ESPRIT-funded *Multimedia Applications Development Environment* (MADE)[2] and the international computer graphics standard *Presentation Environment of Multimedia Objects* (PREMO)[3].

All that time CWI has continuously contributed to the research, development and promotion of computer graphics technology, on scientific level, organizational level, and last but not least on standardization level. This last item is being highlighted in this chapter of the CWI golden jubilee book, since it covers a highly interesting and even sometimes adventurous episode in the history of the interactive systems department.

2. History

2.1. The need for a graphics standard

In the 1950s and 1960s, when the computer was in its infancy, computer graphics consisted only of some primitive paper plotters and cathode ray tubes. There was little need for standardization: these plotters interpreted drawing instructions from a paper tape or from a deck of punched cards.

which had been generated by some computer program and it was generally accepted that other plotting devices could not interpret these drawing instructions. In that era CWI manufactured its X1 plotter and also wrote an Algol 60 library for this plotter. Numerous CWI publications have been illustrated with figures drawn by the X1 plotter (see also figure 1). That same plotter, by the way, was also used to draw music notes generated by some music program.

In the 1970s, the display devices became cheaper and widely available, and the need for standard plot software increased, especially when Fortran captured the market for simulation software and the porting of Fortran application programs to other configurations became commonplace. It was perceived as problematic that a Fortran program could not run elsewhere for the single reason that the visualization component was totally dependent on the graphics peripherals. There was a *de facto* programming language (Fortran; Algol, C and Pascal were much less wide used) but no *de facto* plotting library: existing libraries like CALCOMP, and GINO only had a modest market share and were hardly compatible with each other. In summary, the virtual monopoly of Fortran in the simulation software market and the absence of a dominating visualization library made standardization of visualization software very pressing. It was these factors that eventually led to the making of GKS (see section 3).

2.2. Participation of CWI

At the time of the first standardization efforts (second half of the 1970s), CWI's computer science department was already for more than ten years doing research in the area of computer graphics; these research efforts resulted in the early 1980s in the graphics language *ILP* (*Intermediate Language for Pictures*), which had an architecture very similar to the architecture of the later graphics standard *PHIGS*. In 1976 CWI joined the working group of experts who had to make *from scratch* the first computer graphics standard. From that moment till today, CWI has been active in the computer graphics standards arena.

CWI introduced an abstract level of functionality which could bring order and structure in the functional diversity typical among graphics programming libraries at the time. This structuring allowed for a consequent separation of geometrical and non-geometrical aspects of picture parts and put the binding between the two under application control. The latter formed the basis for elementary feedback mechanisms for interaction. For instance, a picture element could change colour when pointed at by rebinding it to a new colour. Moreover, these mechanisms would work the same across implementations on different platforms. This facility, that interactive applications could become portable, was hitherto unheard of. It had the additional effect that a new generation of graphics workstations was

(a) Then (b) Now

(a) A simple graph drawn in 1971 by the X1 plotter. The X1 plotter interpreted drawing instructions on a paper tape generated on the X8 computer by an Algol 60 program using the X1 plot library.
(b) An Escher-like fractal. Courtesy Noel Giffin, Tri-University Meson Facility, California. It is 1024 x 768 pixels in GIF89a format made available in 1993 on the internet. The fractal was coloured using level decomposition methods and is generated from a formula using a simple square root function. It was produced using the fractint formula system and a formula of Giffin.

Figure 1. Computer graphics then and now.

developed by the computer industry which closely followed this functional architecture, although its design was aimed at a software layer rather than the underlying hardware.

Even in today's graphics architectures these structuring principles have been maintained. The field has matured to the extent that now industry standards have taken over the role of the ISO standards; for instance, since 1994 the machine independent layer is based on OpenGL which originates from Silicon Graphics. The knowledge of standards making has been transferred to industry. Even for the methods of consensus building, industry closely follows the procedures pioneered by the graphics R&D community.

3. The first graphics standard: GKS

The first steps towards an international standard were set in 1976 at an International Federation for Information Processing (IFIP) workshop in Seillac, France, where some dozens of people from industry and from academia in the US, the UK, Japan, France, Germany, The Netherlands, Norway, etc. convened for the first time to convert their computer graphics expertise into an international graphics library.

Many, many other meetings would follow (see, e.g., [1] for a detailed report), all over the world, initially under responsibility of IFIP, later under

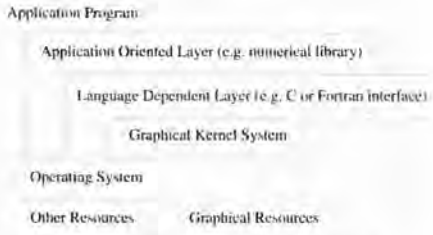


Figure 2. The GKS layer model.

responsibility of the International Organisation of Standards (ISO). In the period 1977-1981 several drafts of candidates for the first graphics standard were produced and revised—amongst them the American standard CORE—but in 1982 the *Graphical Kernel System* (GKS), as it was called then, got stable forms and was registered by ISO as a Draft International Standard, the next-to-last version. In 1985 GKS was published by ISO.

3.1. The strength of GKS

In accordance with the layer model in figure 2, GKS enables application programs to visualize (geometric) data in a *device-independent* way. This strongly facilitates the porting of application programs to other configurations, which was till then problematic. GKS addresses a wide range of graphical *workstations*, including laser printers, photo-typesetters, abstract metafiles, and interactive graphical workstations with window managers like SUNVIEW, NEWS, and X-WINDOWS.

This flexibility of GKS was realized by, inter alia, the following:

- Separation of geometry (e.g. vertices of a polygon) and attributes (e.g. filling style) in the definition of graphical output.
- Definition of three coordinates systems in the viewing process: a *World Coordinates* systems on application level, a *Device Coordinates* system for the graphics device, and a 'dimensionless' intermediary *Normalized Device Coordinates* system for the composition of the picture.
- Adoption of abstract *device-independent* concepts like *workstation*, *logical input device*, *viewport*, *linewidth scale factor*, *colour index*, etc.

3.2. The importance of GKS for CWI

From the very beginning CWI has been actively involved in the development of GKS. This involvement included the writing of technical contributions, chairing the GKS working group, attending and hosting meetings, and engineering a pilot implementation (in C) of GKS, thus illustrating the

feasibility of the GKS objectives. In a later stage, CWI also edited the ISO C binding of GKS.

When GKS emerged in 1982 as an international standard, there were only few implementations available—one of them was the CWI implementation. At the same time there was a widespread demand for GKS. As a consequence, the computer graphics project group of CWI has been kind of GKS agency for some years: in the period 1982-1987 both a C and a Fortran implementation of GKS were developed, documented and tested by CWI. In 1985 the marketing and vending of GKS was transferred to a professional software house. Today CWI is still receiving revenues from GKS.

3.3. The impact of GKS on computer graphics (standards)

The influence of GKS on computer graphics and its standards has been enormous. For the (European) software industry, GKS provided a *device-independent* interface between application software and graphics devices. This property of GKS made them free in purchasing visualization hardware of their own choice.

In the standards arena, both in industry and ISO, GKS also had much influence. Not only was GKS adopted all over the world as a national standard, but its terminology—workstation, polyline, to take some examples—and its methodology were widely adopted, both in later ISO standards—like PHIGS and Computer Graphics Metafile—and industrial standards—like X-WINDOWS, PostScript, and OpenGL.

4. Integration of graphics and multimedia standards

4.1. The new generation of multimedia standards

Graphics and text are media which can be generated by computer programs using basic system support. In contrast, sound, moving pictures and video images are usually captured from the outside world and mixed into presentation schemes. Hence the first generation of multimedia applications could only be off-line editing systems comparable to desk top publishing tools.

The second generation of multimedia systems tries to base itself on computer generation of all media, and at the same time then reap the benefit of merging the various media into integrated presentations automatically. In this more generalized multimedia system concept the use of externally captured source material becomes merely a special case.

It has been recognized that the size and complexity of modern information systems require multimedia presentations in order to be able to effectively communicate and at the same time require that these presentations are generated in real-time on demand.

Moreover, this type of interaction and output generation must be provided by powerful services capable of combining information from distributed

sources, who provide data but no built-in presentation functionality. This calls for another level of integrated, uniform functionality based on recognized standards. These standards go way beyond agreements about common exchange formats, which were sufficient for the first generation multimedia systems.

The second generation multimedia applications can only be successfully developed if the modern advanced technology is used. The design of the standard functionality assumes that such advanced techniques are available. Examples are distributed object systems, multi-threaded concurrent systems and efficient synchronization support. On top of this extendible and dynamically adaptable object classes must be supported.

Each of these features can be justified by some application programmer's need. For instance, the enormous variety of low level presentation functions must be reduced by object specialization to a workable subset for a given application, thereby making a better match between conceptual and concrete functionality; adaptive methods must be used to produce object instances which behave sufficiently efficient in a given situation. The major area which is addressed by this standard, is the area of virtual reality.

Virtual reality is not only a new gadget for the entertainment industry, it is the ultimate means to communicate computer-based information making full use of all human cognitive powers.

4.2. The first and second generation of graphics standards

GKS and its sister ISO standards *PHIGS, Computer Graphics Metafile*, and *Computer Graphics Interface* are so-called *first generation* computer graphics standards, which were completed in the 1980s. These standards only address text and graphics. In spite of important differences in their functionality, they share a common architectural approach, which has resulted in implementations that are large monolithic libraries of a set of functions with precisely defined semantics. They reflect an approach towards graphical software libraries predominant in the seventies and the eighties. However, these standards have little chance of providing appropriate responses to the rapid changes in today's technology, and in particular, they fail to fit into the software and hardware system architectures prevailing on today's systems.

When the revision process of GKS started in the late 1980s (the second edition was published in 1994), it soon became apparent that a second generation of graphics standards was needed. These new standards should also address more modern technologies which had emerged in the late 1980s, such as

- Programming environments supported by windowing technology and open distributed processing.
- Advanced rendering methods like *ray tracing* and *radiosity*.

- Modern insights in software engineering, such as the use of object-oriented specification methods.
- Modern presentation techniques, in particular multimedia and hypermedia technology using international standards like, e.g., HyTime and MHEG.

The ISO subcommittee responsible for the development and maintenance of graphics and image processing standards recognized the need to develop a new line of graphics standards, along radically different lines from previous methods. To this end, a new project was started at an ISO meeting at Chiemsee, Germany, in October 1992. Subsequent meetings resulted in a draft for a new standard called PREMO (Presentation Environment for Multimedia Objects)[3]. Publication of the final text is expected in 1997.

4.3. PREMO

General architecture. Underlying all of PREMO is a concise conceptual framework, comprising a description technique, an abstract object model used for the definition of data types and the operations upon them, and the notion of components which contain and organize the PREMO functionality needed to address specific problem areas.

Object Model. In PREMO, a strong emphasis is placed on the ability of objects (e.g., enhanced geometric data) to be active. This feature of PREMO stems from the need for synchronization in multimedia environments. Conceptually, different media (e.g., a video sequence and a corresponding sound track) may be considered as *parallel* activities that have to reach specific milestones at distinct and possibly user-definable synchronization points.

Events, Event Model. The PREMO framework includes the notion of non-objects, primarily for efficiency reasons. Non-objects have no requests defined on them, they cannot take part in subtyping and inheritance hierarchies. *Events* form a special category of PREMO non-object types, and are the basic building block for the PREMO event model. Events and their propagation (described by the event model) play a fundamental role in the synchronization mechanism.

Components. The object model, the event model, the concept of non-objects, etc., give a conceptual framework for all the basic notions in PREMO. *Components* allow for a structuring of PREMO in terms of the services provided.

A component in PREMO is a collection of object types and non-object data types, from which objects and non-objects can be instantiated. A component can offer *services* usable in a distributed environment, or it may

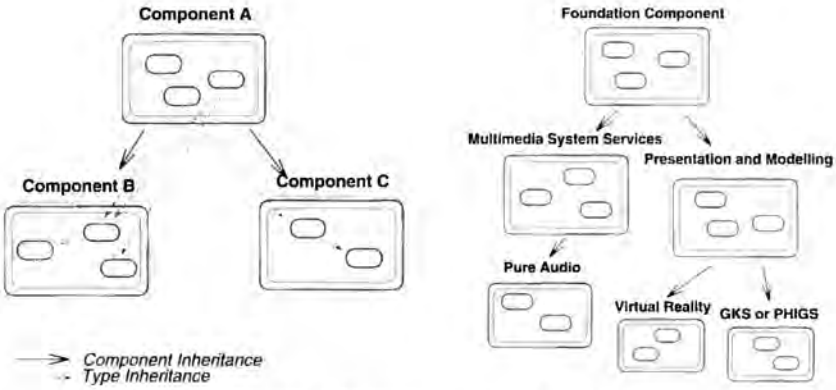


Figure 3. PREMIO component model

be used as a set of objects directly linked to an application.

Underlying all PREMIO components is a *Foundation Component* providing functionality which is necessary for all PREMIO components. It is mandatory that all other PREMIO components inherit from this foundation component (see also figure 3). The rules for components form the basis, in conjunction with the object model, for the properties of configuration, customization, extension, and interoperation. PREMIO will furthermore include the specification of some other components, namely:

- A component for multimedia system services.
- A modelling, presentation, and interaction component, which will provide the basis of components inherently related to modelling, geometry, traditional computer graphics, etc.

REFERENCES

1. G. ENDERLE, K. KANSY, G. PFAFF (1987). *Computer Graphics Programming—Graphical Kernel System*. Springer-Verlag, Berlin-Heidelberg.

2. I. HERMAN, G. REYNOLDS, J. DAVY (1994). MADE: A multimedia application development environment. *CWI Quarterly* 7, 27–46.

3. I. HERMAN, P. TEN HAGEN, G. REYNOLDS (1994). PREMIO: an ISO standard for a presentation environment for multimedia objects. K.R. APT, A. SCHRIJVER, N.M. TEMME (eds.). *From Universal Morphisms to Megabytes: a Baayen Space Odyssey*. CWI, Amsterdam, 347–362.

The Authors

A.M. Cohen
Eindhoven University of Technology
Faculty of Mathematics and Computer Science
P.O. Box 513, 5600 MB Eindhoven
E-mail: amc@win.tue.nl

H.P. Barendregt
Catholic University of Nijmegen
Faculty of Mathematics and Computer Science
Toernooiveld, 6525 ED Nijmegen
E-mail: henk@cs.kun.nl

R.D. Gill
Utrecht University
Faculty of Mathematics and Computer Science
P.O. Box 80010, 3508 TA Utrecht
E-mail: gill@math.ruu.nl

H.J. Sips
Delft University of Technology
Faculty of Technical Physics
Lorentzweg 1, 2628 CJ Delft
E-mail: henk@cp.tn.tudelft.nl

and:
University of Amsterdam
Faculty of Mathematics, Computer Science, Physics, and Astronomy
Kruislaan 403, 1098 SJ Amsterdam
E-mail: henk@fwi.uva.nl

National Activities Mathematics

C.R. Traas
University of Twente
Faculty of Applied Mathematics
P.O. Box 217, 7500 AE Enschede
E-mail: C.R.Traas@math.utwente.nl

M.S. Keane
Centre for Mathematics and Computer Science (CWI)
P.O. Box 94079, 1090 GB Amsterdam
E-mail: Mike.Keane@cwi.nl

A. Hordijk
University of Leiden
Faculty of Mathematics and Natural Sciences
P.O. Box 9512, 2300 RA Leiden
E-mail: hordijk@wi.LeidenUniv.nl

R.F. Curtain
University of Groningen
Faculty of Mathematics and Natural Sciences
P.O. Box 800, 9700 AV Groningen
E-mail: R.F.Curtain@math.rug.nl

J.H. van Lint
Eindhoven University of Technology
Rector
P.O. Box 513, 5600 MB Eindhoven
E-mail: wsdwjhvl@urc.tue.nl

G. van Dijk
University of Leiden
Faculty of Mathematics and Natural Sciences
P.O. Box 9512, 2300 RA Leiden
E-mail: dijk@wi.LeidenUniv.nl

J.A. Sparenberg
Meerkoetlaan 7, 9765 TC Paterswolde

P. Sijtsma
National Aerospace Laboratory NLR
Department of Aero-acoustics
P.O. Box 153, 8300 AD Emmeloord
E-mail: sijtsma@nlr.nl

H.P. Urbach
Philips Research Laboratories
P.O. Box 218, 5600 MD Eindhoven
E-mail: urbach@natlab.research.philips.com

T. de Jong
Catholic University of Nijmegen
Faculty of Mathematics and Computer Science
Toernooiveld, 6525 ED Nijmegen
E-mail: tdejong@sci.kun.nl

J.H.M. Steenbrink
Catholic University of Nijmegen
Faculty of Mathematics and Computer Science
Toernooiveld, 6525 ED Nijmegen
E-mail: steenbri@sci.kun.nl

G.B.M. van der Geer
University of Amsterdam
Faculty of Mathematics, Computer Science, Physics, and Astronomy
Plantage Muidergracht 24, 1018 TV Amsterdam
E-mail: geer@fwi.uva.nl

F. Oort
Utrecht University
Faculty of Mathematics and Computer Science
P.O. Box 80010, 3508 TA Utrecht
E-mail: oort@math.ruu.nl

C.A.M. Peters
Université de Grenoble, Institut Fourier
F-38402 Saint-Martin d'Hères, FRANCE
E-mail: peters@fourier.grenet.fr

I. Moerdijk
Utrecht University
Faculty of Mathematics and Computer Science
P.O. Box 80010, 3508 TA Utrecht
E-mail: moerdijk@math.ruu.nl

B. van Dalen
Johann Wolfgang Goethe-Universität
Institut für Geschichte der Naturwissenschaften
Postfach 111932, D-60054 Frankfurt am Main, GERMANY
E-mail: dalen@em.uni-frankfurt.d400.de

H.W. Broer
University of Groningen
Faculty of Mathematics and Natural Sciences
P.O. Box 800, 9700 AV Groningen
E-mail: H.W.Broer@math.rug.nl

F. Takens
University of Groningen
Faculty of Mathematics and Natural Sciences
P.O. Box 800, 9700 AV Groningen
E-mail: F.Takens@math.rug.nl

Centre for Mathematics and Computer Science

[Unless stated otherwise, the address of the following authors is:
CWI, P.O. Box 94079, 1090 GB Amsterdam]

O. Diekmann
Utrecht University
Faculty of Mathematics and Computer Science
P.O. Box 80010, 3508 TA Utrecht
E-mail: diekmann@math.ruu.nl

A. Schrijver. E-mail: Lex.Schrijver@cwi.nl

O.J. Boxma. E-mail: Onno.Boxma@cwi.nl

J.M. van den Hof. E-mail: Jacqueline.van.den.Hof@cwi.nl

J.H. van Schuppen. E-mail: J.H.van.Schuppen@cwi.nl

R. Helmers. E-mail: R.Helmers@cwi.nl

H.J.A.M. Heijmans. E-mail: Henk.Heijmans@cwi.nl

J.G. VERWER, E-mail: Jan.Verwer@cwi.nl

P.W. HEMKER, E-mail: P.W.Hemker@cwi.nl

H.J.J. TE RIELE, E-mail: Herman.te.Riele@cwi.nl

J.W. DE BAKKER, E-mail: jaco@cwi.nl

F.W. VAANDRAGER
Catholic University of Nijmegen
Faculty of Mathematics and Computer Science
Toernooiveld, 6525 ED Nijmegen
E-mail: Frits.Vaandrager@cs.kun.nl

J. HEERING, E-mail: Jan.Heering@cwi.nl

P. KLINT, E-mail: Paul.Klint@cwi.nl

I. BETHKE, E-mail: Inge.Bethke@cwi.nl

J.W. KLOP, E-mail: J.W.Klop@cwi.nl

P.M.B. VITÁNYI, E-mail: Paul.Vitanyi@cwi.nl

R. HIRSCHFELD, E-mail: R.Hirschfeld@cwi.nl

L.G.L.T. MEERTENS, E-mail: Lambert.Meertens@cwi.nl

S. PEMBERTON, E-mail: Steven.Pemberton@cwi.nl

K.R. APT, E-mail: K.R.Apt@cwi.nl

D.J.N. VAN EIJCK, E-mail: D.J.N.van.Eijck@cwi.nl

R. VAN LIERE, E-mail: Robert.van.Liere@cwi.nl

J.J. VAN WIJK
Netherlands Energy Research Foundation ECN
P.O. Box 1, 1755 ZG Petten
E-mail: vanwijk@ecn.nl

A.A.M. KUIJK, E-mail: Fons.Kuijk@cwi.nl

F. ARBAB, E-mail: Farhad.Arbab@cwi.nl

M. BAKKER, E-mail: Miente.Bakker@cwi.nl

P.J.W. TEN HAGEN, E-mail: Paul.ten.Hagen@cwi.nl