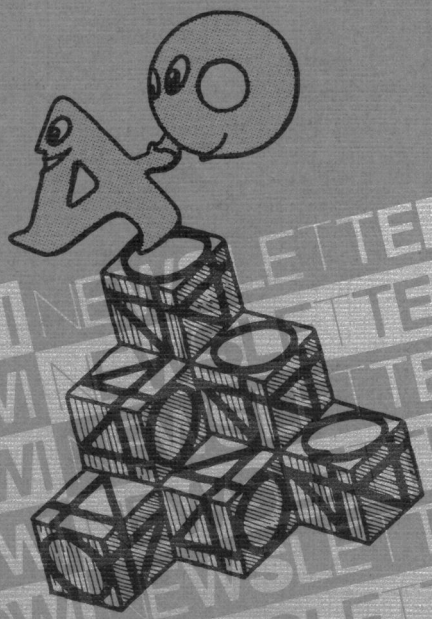
CWI NEWSLETTER

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

Quarterly, Issue no. 10
March 1986



CWI NEWSLETTER

Number 10, March 1986

Editors

Arjeh M. Cohen Richard D. Gill Jo C. Ebergen

Technical editor Editorial secretary

Wim A.M. Aspers Wilma E.G. van Eijk

The CWI Newsletter is published quarterly by the Centre for Mathematics and Computer Science (Centrum voor Wiskunde en Informatica), Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. The Newsletter will report on activities being conducted at the Centre and will also contain articles of general interest in the fields of Mathematics and Computer Science, including book reviews and mathematical entertainment. The editors encourage persons outside and in the Centre to contribute to the Newsletter. Normal referee procedures will apply to all articles submitted.

The Newsletter is available free of charge to all interested persons. The Newsletter is available to libraries on an exchange basis.

Material may be reproduced from the CWI Newsletter for non-commercial use with proper credit to the author, the CWI Newsletter, and CWI.

All correspondence should be addressed to: *The CWI Newsletter, Centre for Mathematics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands.*

ISSN 0168-826x

Contents

Centre for Mathematics and Computer Science

Centrum voor Wiskunde en Informatica

Bibliotheek
CWI Centrum voor Wiskunde en Informatica
Amsterdam

NEWSLETTER CWI NEWSLETTER CWI

The Generator Paradigm in Smalltalk

Tim Budd[1]

Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

The method of *generators* is a powerful programming technique for many problems. Unfortunately, it is not widely known or employed. This paper describes the generator paradigm, illustrating the use of the technique by several programs in the language Smalltalk.

Introduction

In computer science, as in many other areas of human endeavor, people frequently attempt to solve problems by relating them to a paradigm, that is an example or model. Students, for example, when presented with a novel problem, will often try to formulate it in a manner similar to problems they have already been shown or solved for themselves. This is only natural, and is one reason why it is important for students in computer science to be exposed to a wide variety (not necessarily a wide number) of computer languages. Just as with natural languages, the computer language in which a problem is solved colors in a forceful way the type of solution employed. The wider the variety of languages a student is exposed to, the larger the number of paradigms they will have seen, and the broader will be their outlook on a new problem.

This paper describes one such programming paradigm, that of *generators*. The concept of generators occurs in many computer languages, but unfortunately is not a natural technique in the languages most commonly encountered by the student (Pascal, C, Fortran). Thus, while it is a valuable problem solving method for many classes of problems, it is not widely known or employed.

The generator paradigm is most useful in situations where there may not be a single, unambiguous, correct answer to a given question. This occurs frequently in such fields as pattern matching, data base systems, artificial intelligence as well as others. To illustrate, consider the pattern matching question

1 This paper was written while the author was on leave from the University of Arizona. The author's present address is: Department of Computer Science, Oregon State University, Corvallis, Oregon, 97331 USA.

"At what location does the letter 'i' occur in the word 'Mississippi'?" A single answer, such as '2', is correct but unsatisfactory. For example, this question might be a result of a larger problem, such as "at what location does the string 'ip' occur in the word 'Mississippi'?" Thus it is not sufficient to consider only the first answer to a question, but one must produce *all* answers to the question. One possibility is to return a *set* of answers. This works if the set of solutions is small, but becomes unwieldy if the set of answers is large or infinite. The generator paradigm is an attractive alternative.

For the purposes of this paper we will say a *generator* is any object (procedure, module, abstract datatype, class instance, or whatever) that can respond with a *sequence* of answers to a given query, one at a time. A generator for our example problem when asked for the position of 'i' in the word 'Mississippi' would first respond '2'. When asked again it would respond '5', then '8', then '11', and finally 'no more answers'.

Generators can be written in any language that supports co-routines, such as Simula [1], or SL5 [7]. They are used, although not always in ways that are obvious to the programmer, in Prolog [4]. They appear as a fundamental programming technique in the language Icon [6]. They can even be written, with the aid of a little run-time support, in C [2]. In this paper we will discuss the use of generators in Smalltalk [5], and in particular the examples will use the dialect of Smalltalk called Little Smalltalk [3].

SMALLTALK AND OBJECT ORIENTED PROGRAMMING

There is not sufficient space to present more than an elementary introduction to the Smalltalk language, however a few concepts are central to the discussion and must be advanced. A more complete description of the language can be found in [3,5].

The traditional model describing the behavior of a computer executing a program can be characterized as the process-state, or 'pigeon-hole' model. In this view the computer is a data manager, following some pattern of instructions, wandering through memory pulling values out of various slots (memory addresses), transforming them in some manner, and pushing the results back into other slots. By examining the values in the slots one can determine the state of the machine, or the results produced by the computation. While this may be a more or less accurate picture of what is physically taking place in a computer, it does little to help us understand how to solve problems using the computer and is certainly not the way most people (pigeons and postmen excepted) go about solving problems.

Let us examine a real world situation and then ask how we might make the solution of problems on a computer more closely model the methods people use in everyday life. Suppose I wish to send flowers to my grandmother for her birthday. She, however, lives in a city many miles away. It is a task easy enough to do; I merely go to a local florist, describe the nature and number of flowers I desire, and I can be assured that they will be automatically delivered. If I stopped to investigate how this gets accomplished I would probably discover that my florist sends a message describing my order to another florist in

my grandmother's city, who then takes care of the actual delivery. I might inquire further to find out how the florist in my grandmother's city obtains the flowers, finding perhaps that they are obtained in bulk in the morning from a flower wholesaler. If I persist, I might even be able to follow the chain all the way back to the farmer who grows the flowers, and discover what requests were made by each member of the chain in order to solicit the desired outcome from the next.

The important point, however, is that I do not *need* to, indeed most of the time do not *want* to, know how my simple directive 'send flowers to my grandmother' is going to be carried out. In real life we call this process 'delegation of authority'. In computer science it is called 'abstraction' or 'information hiding'. At the heart, these terms amount to the same thing. There is a resource (a florist, a file server) that I wish to use. In order to communicate, I must know the commands the resource will respond to (send flowers to my grandmother, return a copy of the file named 'chapter1'). The steps the resource must take in order to respond to my request are in all likelihood much more complex than I realize, but in any case there is no reason for me to know the details of how my directive is implemented, as long as the response (the delivery of the flowers, receiving a copy of my file) is well defined and predictable.

The *object-oriented* model of problem solving views the computer in much the same fashion as just described. Indeed many people who have no training in computer science and no idea how a computer works find the object-oriented model of problem solving quite natural. Surprisingly, however, many people who have a traditional background in computer programming initially think there is something strange about the object-oriented view. The notion that '7' is an object, and '+' a request for an addition, may at first seem strange. But soon, the uniformity, power, and flexibility the object-message metaphor brings to problem solving makes this interpretation seem natural.

As we have been suggesting, the Smalltalk universe is inhabited by *objects*. If we invert the metaphor, using it to describe my flower example, I am an object and the flower shop (or the florist in it) is another object. Actions are instigated by sending requests (or *messages*) between objects. I transmitted the request 'send flowers to my grandmother' to the florist-object. The reaction of the *receiver* for the message is to execute some sequence of actions, or *method*, to satisfy my request. It may be the case that the receiver can immediately satisfy my request. On the other hand it will often be the case that in order to meet my needs, the receiver is required to transmit other messages to yet more objects (the message my florist sends to the florist in my grandmothers city, or a command to a disk drive). In addition, there is an explicit *response* (a receipt, for example, or a result code) returned directly back to me. DAN INGALLS describes the Smalltalk philosophy [8]:

> 'Instead of a bit-grinding processor raping and plundering data structures, we have a universe of well-behaved objects that courteously ask each other to carry out their various desires.'

Such anthropomorphic viewpoints are common among Smalltalk programmers. In subsequent sections we will see how the Smalltalk language embodies this object-oriented view of programming. By describing the solution of several problems in Smalltalk, we hope to show how the object-oriented model aids in the creation of software systems, and assists in the solution of problems using the computer.

SMALLTALK SYNTAX

In this section we present a brief overview of Smalltalk syntax, just enough to make the examples presented later in the paper understandable. Once more, the reader interested in further information should consult the references.

An *object* can be a literal object, such as a number (2, for example), or a named object, such as an identifier (x, for example). An assignment arrow is used to associate a name with an object. The statement

$$x \leftarrow 2$$

makes the identifier x temporarily represent the same object as the literal object 2. This assignment may be later overwritten by other assignments to the same identifier.

Action is initiated by sending messages to objects. A message can simply be a command, with no arguments. For example, the following statement:

x squared

illustrates the message *squared* being sent to the object x. In response, the object x will return a new object. The particular nature of the response is always defined by the category (in Smalltalk terms, the *Class*) of the recipient for the message. If x is a number, the response to the message *squared* will be the object representing the value of the number multiplied by itself. Thus, the following example will make the identifier y represent the object 4.

$y \leftarrow x$ *squared*

Messages can also take arguments. The arithmetic operations, for example, are interpreted as messages to the left side, having the right side as argument. Thus the expression:

$x + 3$

shows the message '+' being passed to the object x, accompanied by argument 3.

A third form of message is permitted to take an arbitrary number of arguments. This form of message is written as a sequence of *keywords*, that is names followed by colons, separating the receiver and arguments. For example:

x between : 2 and : 4

shows the message *between:and:* being passed to the object x, accompanied by two arguments. Messages can be composed, with messages having no

arguments taking precedence over binary (arithmetic style) messages, and binary messages taking precedence over keyword messages. Parenthesis can be used to provide alternative groupings.

Some messages have side effects, in addition to returning a value. The message *print*, for example, will display a value on an output device. Thus:

$$(x \textit{ squared } + 3) \textit{ print}$$

will cause the value 7 to be displayed.

A novel feature of Smalltalk is the ability to easily encapsulate actions for performance at a later time. This is accomplished using a *block*, which is written as a pair of square braces surrounding a list of Smalltalk statements. Because a block is an object, it can be assigned to an identifier or used as an argument in an expression, like other objects. For example:

$$z \leftarrow [x \textit{ print. } x \leftarrow x+1]$$

assigns a block to the identifier z. Note that a period is used to separate the statements within the block. These statements are not immediately executed; instead, they are executed when the message *value* is passed to the block. If at some later time the statement:

$$z \textit{ value}$$

is executed, the value 2 (the current binding of the object x) will be displayed and x will be updated. The block continues to exist, and the actions can be repeated merely by sending the *value* message to the block again, as often as necessary. Note that the binding for the identifier x derives from the surrounding context of the definition of the block, and not from the context in which the message *value* is used.

Blocks are used to implement a number of control structures in Smalltalk. For example the conditional execution statement is constructed as a message passed to an object of type boolean using a block as an argument.

$$(x < 3) \textit{ ifTrue: } [x \leftarrow x+1]$$

If the boolean (the recipient of the *ifTrue:* message) is true, the block is executed; otherwise not. Similarly the while loop is constructed using a block for both the recipient and the argument.

$$[x < 10] \textit{ whileTrue: } [x \textit{ print. } x \leftarrow x+1]$$

Blocks can also be written so as to take parameter values, and thus in many ways act like statically scoped in-line procedures. For example, the following block:

$$w \leftarrow [:a \mid a \textit{ squared print}]$$

defines w to be a block taking one argument. The keyword message *value:* is used to invoke such a block. For example:

$$w \textit{ value: } 6$$

will result in the value 36 being displayed.

The sequence of actions to be executed in response to a message is described by a *method*, which corresponds in some ways to a procedure in a conventional language. For example, the following method describes the message *squared* :

squared

↑ *self* * *self*

Within a method, the identifier *self* refers to the receiver of the message. The up arrow (↑) precedes the value to be returned in response to the message. A method for a message that takes arguments must provide identifier names for the arguments:

between: *lower* **and:** *upper*

↑ (*self* >= *lower*) & (*self* <= *upper*)

One final bit of syntax is useful in situations that might otherwise require the introduction of temporary variables. A *cascade* is formed from an expression, and one or more continuations of messages (messages without a receiver) separated by semicolons. An example might be:

(*x* + 2) ; *squared*

The result of a cascade is the result of the expression to the left of the first semicolon. This value is also used as the receiver for messages to the right of the semicolon, and whatever response they produce is discarded. In almost all situations where a cascade is used the expression to the left is creating a new object, and the message on the right is initializing it in some fashion. The cascade is used for the side effect whatever message on the right side may have, not for the response it will return.

GENERATORS

A *generator* is any object that represents a collection of other objects and that responds to the following two messages:

first	The response should be an element of the collection, or the special value **nil** if there are no elements in the collection.
next	The response should be another element in the collection, or **nil** if there are no more elements in the collection.

We do not require that the collections be in any specific order, only that all elements will eventually be produced if a sufficient number of *next* messages are received and no element will be produced more than once.

A simple generator is one used to produce values in arithmetic sequence. The message *to:*, used in conjunction with numbers, produces such a generator.

For example:

```
        x ← 3 to: 9
        x first
3
        x next
4
```

The following method describes a useful message for dealing with generators, the message *do:*.

```
do: aBlock              | item |

        item ← self first.
        [ item notNil ] whileTrue:
                [ aBlock value: item .  item ← self next ].
        ↑ nil
```

We will explain several features of this method. There is a temporary variable named *item* used in the method; this variable is declared by placing it in a list surrounded by vertical bars following the pattern describing the names of the method and of the arguments. This particular method appears as part of the description of all generators. The special identifier *self* refers to the recipient of the do: message. Since this must be able to respond to *first* and *next*, it must be a generator. The body of the method is a simple loop. Before entering the loop the temporary variable *item* is assigned the result of passing the message *first* to the recipient. While *item* is not **nil** the value is used as an argument in a message *value:*, passed to the variable *aBlock*. The variable *item* is then updated by requesting the next value from the sequence.

The argument used with this message must be a one argument block. The block is executed on each element of the collection. For example:

```
        (0 to: 5) do: [ :x | x squared print ]
0
1
4
9
16
25
```

Subsequent sections will illustrate the utility of the generator concept.

A Simple example

An example will illustrate how generators can be written in Smalltalk. Consider the problem of producing prime numbers. By definition, a prime number is a value having only two distinct divisors, itself and 1. A generator for prime numbers will produce the first prime value (namely 2) when offered the message *first*, and successive prime numbers in response to each *next* message. If a number n divides a number m, then the prime factors of n must also divide m. Thus to tell if a number m is prime we need not test all values less than m, only those values that are prime. Therefore a simple generator for primes can be constructed by merely retaining the previously generated primes in a **List**, a data structure that will maintain elements in their order of insertion. Each time a value is requested, an object representing the last prime produced is incremented and tested until a value having no factors is found. The new value is then appended to the list and returned. Keeping the primes in order allows the loop to terminate as soon as a value larger than $\sqrt{n}$ is encountered, where n is the value to be tested.

The methods for each type of object in the Smalltalk language are gathered together to form what is known as a *class*. For example all integers are elements of class **Integer**, all arrays of class **Array** and so on. The response of an object when presented with any given message is determined by the methods associated with the class of that object. The following is a class description for a prime number generator. Each instance of this class will, in response to the messages *first* and *next*, return a stream of prime numbers. The variables *prevPrimes* and *lastPrime*, listed following the class name, are local variables for the class. Each instance of the class maintains its own copies of these variables, and they can be used only in the methods for the class.

New instances of a class are created using the message *new*. In this example, a new instance of the class **List** is created and stored in the identifier *prevprimes*. A List is a simple data structure that maintains elements in the order that they are inserted (using the message *add:*). Like most data structures in Smalltalk, a List is also a generator, and thus responds to the message *do:*

```
Class Primes
| prevPrimes lastPrime |
[
        first
                prevPrimes ← List new.
                prevPrimes add: (lastPrime ← 2).
                ↑ lastPrime
|
        next
                [ lastPrime ← lastPrime + 1.
                self testNumber: lastPrime ] whileFalse.
                prevPrimes add: lastPrime.
                ↑ lastPrime
|
        testNumber: n
                prevPrimes do: [:x|
                        (x squared > n) ifTrue: [ ↑ true ].
                        (n \\ x = 0) ifTrue: [ ↑ false ] ]
]
```

The message *testNumber:* is used to determine whether a proposed number is prime. It accomplishes this by performing a modular division (\ \ ) of the number with previous primes. If the remainder after division is zero, a previous prime divides the number and it cannot be prime. If no number less than the square root of the proposed number divides the number, then it must be prime.

An obvious problem with this prime number generator is that it requires an ever increasing amount of storage to maintain the list of previous primes. If one were constructing a long list of prime values, the size of this storage could easily become a problem. A recursive version is possible which trades longer computation time for reduced storage. This is analogous to a recursive procedure in programming languages such as Pascal. The following program does not maintain the list of previous primes, but instead regenerates the list each time a new number is to be tested. The expression '*Primes new*' produces a new instance of the prime generator each time the message *testNumber:* is received.

```
Class Primes
| lastPrime |
[
        first
                ↑ ( lastPrime ← 2 )
|
        next
                [ lastPrime ← lastPrime + 1.
                self testNumber: lastPrime ] whileFalse.
                ↑ lastPrime
|
        testNumber: n
                (Primes new) do: [:x |
                        (x squared > n) ifTrue: [ ↑ true ].
                        (n \\ x = 0) ifTrue: [ ↑ false ] ]
]
```

FILTERS

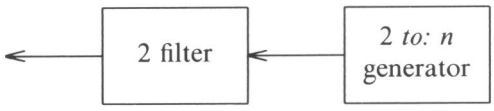
An entirely different program, solving the same task as the prime number generators described in the last section, will illustrate another programming technique that is frequently useful in conjunction with generators, which is the notion of *filters*. A filter is a generator that filters, or modifies, the values of another underlying generator. The class **FactorFilter** (below) exemplifies some of the essential features of a filter. Instances of **FactorFilter** are initialized by giving them a generator and a specific nonnegative value, using the message *remove:from:*. In response to *next* (the message *first* is in this case replaced by the initialization protocol *remove:from:*), values from the underlying generator are requested, and returned, with the exception of values for which the given number is a factor, which are repressed. Thus the sequence returned by an instance of **FactorFilter** is exactly the same as that given by the underlying generator, with the exception that values for which the given number is a factor are filtered out. (The symbol ~= is the Smalltalk message representing 'not equals').

```
Class FactorFilter
| myFactor generator |
[
        remove: factorValue from: generatorValue
                myFactor ← factorValue.
                generator ← generatorValue
|
        next            | possible |
                [ (possible ← generator next) notNil ]
                        whileTrue:
                                [ (possible \\ myFactor ~= 0)
                                        ifTrue: [ ↑ possible ] ].
                ↑ nil
]
```

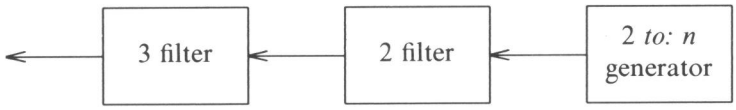
Using **FactorFilter**, a simple generator for prime numbers can be constructed. An instance of **Interval** (the generator that merely returns numbers in arithmetic progression) is first constructed generating all numbers from 2 to some fixed limit. As each value is removed, a filter is inserted in front of the generator to insure that all subsequent multiples of the value will be eliminated. A new value is then requested from the updated generator.

```
Class Primes
| primeGenerator lastFactor |
[
        first
                primeGenerator ← 2 to: 100 .
                lastFactor ← primeGenerator first .
                ↑ lastFactor
|
        next
                primeGenerator ← FactorFilter new;
                        remove: lastFactor from: primeGenerator
                lastFactor ← primeGenerator next .
                ↑ lastFactor
]
```

Pictorially, the underlying generator constructed by the first occurrence of the message *next* can be viewed as follows:



When asked for the next prime, the generator is modified by adding a filter, this time for the last prime value returned, the number 3.



The program continues, each time a new prime is requested a filter is constructed to remove all factors of the previous prime. In this fashion, all the primes are eventually generated.

Of course, like the first two programs in the last section, the storage required for the chain of filters is proportional to the number of primes generated so far. Despite this, actual timings on running programs show that the filter program is the fastest of the three prime number generating programs described here. However, we should note that these programs do not represent the fastest algorithms known for producing prime numbers, but are merely intended as instructive examples of classes and generators in Smalltalk.

Goal directed evaluation

A useful programming technique when used in conjunction with generators is *goal directed evaluation*. Using this technique, a generator is repeatedly queried for values until some condition is satisfied. In a certain sense the notion of filters we have just described represents a simple form of goal directed evaluation. The goal of instances of **FactorFilter**, for example, is to find a value from the underlying generator for which the given number is not a factor. In the more general case of goal directed evaluation the condition frequently involves the outcome of several generators acting together. An example will illustrate this method.

Consider the problem of placing eight queens on a chess board in such a way that no queen can attack any other queen (illustrated below). In this section we will describe how such a problem can be formulated and solved using generators, filters, and goal directed evaluation.

We first observe that in any solution, no two queens can occupy the same column, and that no column can be empty. We can therefore assign a specific column to each queen at the start, and reduce the problem to finding a correct row assignment for each of the eight queens.

In general terms, our approach will be to place queens from left to right (the order in which we assign numbers to columns). An *acceptable solution for columns 1 through n* is one in which no queen in columns 1 through *n* can attack any other queen in those columns. Once we have found an acceptable solution for columns 1 through 8 we are finished. Before that, however, we can formulate the problem of finding an acceptable solution for columns 1

	1	2	3	4	5	6	7	8
1	Q							
2					Q			
3								Q
4						Q		
5			Q					
6							Q	
7		Q						
8				Q				

A solution to the eight queens problem

through n recursively, as follows:

1. If $n > 1$, find an acceptable solution for columns 1 through $n-1$. If there is none, return nil, there is no acceptable solution. Otherwise place the queen for column n in row 1. Go to step 2.
2. Test to see if any queen in columns 1 through $n-1$ can attack the queen in column n. If not, then an acceptable solution has been found. If some other queen can attack, then go to step 3.
3. If the queen for column n is in row 8, then go to step 4, otherwise advance the queen by one row and go back to step 2.
4. Find the next acceptable solution for columns 1 through $n-1$. If there is none, return nil, otherwise place the queen for column n in row 1 and go to step 2.

Of course, all positions are acceptable in column 1. Responding to *first* corresponds to starting in step 1, whereas responding to *next* corresponds to starting in step 3. We represent each queen by a separate object, an instance of class **Queen**. Each queen maintains its own position in a pair of variables, and also a pointer to the immediate neighbor on the left. A skeleton for the class **Queen** can be given as follows:

```
Class Queen
| row column neighbor |
[
        setColumn: aNumber neighbor: aQueen
                column ← aNumber.
                neighbor ← aQueen
|
        ...
]
```

Using this skeleton, our eight queens can be initialized as follows:

```
lastQueen ← nil
(1 to: 8) do: [:i | lastQueen ← Queen new ;
                    setColumn: i neighbor: lastQueen ]
```

Following the execution of this code the variable *lastQueen* points to the last (rightmost) queen.

We have already described our algorithm in terms of finding the first acceptable position and finding the next acceptable position. It is therefore easy to apply our generator paradigm (using the messages *first* and *next*) to this situation. Step 1, for example, corresponds to the following method

```
first
        (neighbor notNil)
                ifTrue: [ neighbor first ].
        row ← 1.
        ↑ self testPosition
```

Rather than falling directly into step 2, as we did in the informal description of the algorithm, an explicit message (*testPosition*) is used to perform steps 2, 3 and 4. Thus one can read *self testPosition* as being the equivalent of 'go to step 2' in the informal description. Before describing the method for this message, we describe the method used to find the *next* acceptable position, which is a combination of steps 3 and 4 in our description.

```
next
    (row = 8)
        ifTrue: [  ((neighbor isNil) or: [ neighbor next isNil ])
                        ifTrue: [ ↑ nil ].
                   row ← 0 ].
    row ← row + 1.
    ↑ self testPosition
```

A coding trick is used here; the zero assigned to the identifier row is immediately incremented, resulting in the queen being placed into row 1. Once more the '*self testPosition*' message can be interpreted as 'go to step 2'.

All that remains is to test a position to see if any queen to the left can attack. As we have already noted, any position is acceptable to the leftmost queen. Suppose a queen, call her Q, is not the leftmost queen. We pass a message to the neighbor of Q asking if she can attack the position of the queen Q. If the neighbor queen can attack, she will return true, otherwise she will pass the message on to her neighbor, and so on until the leftmost queen is reached. If the leftmost queen cannot attack, she will return false. Notice the recursive use of the message *next* to find the next acceptable position, in case an attack is possible. This corresponds to the directive 'go to step 3' found in step 2 of our informal description.

```
testPosition
        (neighbor isNil) ifTrue: [ ↑ self ].
        (neighbor checkRow: row column: column)
                ifTrue: [ ↑ self next ]
                ifFalse: [ ↑ self ]
```

We have reduced the problem to the much simplier one of each queen taking a pair of coordinates for a queen positioned to the right, and responding whether she or any queen to the left can attack that position. Since we know the queen to the right is in a different column from the queen to the left, she can only be attacked if she is in the same row or if the differences in the columns is equal to the differences in the rows (i.e., a diagonal).

```
checkRow: testRow column: testColumn   | columnDifference |
        columnDifference ← testColumn − column.
        (((row = testRow) or:
          [ row + columnDifference = testRow ] ) or:
          [ row − columnDifference = testRow ])
                ifTrue: [ ↑ true ].
        (neighbor notNil)
                ifTrue: [ ↑ neighbor checkRow: testRow
                                        column: testColumn ]
                ifFalse: [ ↑ false ]
```

A final method is useful for producing the answer in a visual form:

```
printBoard
        (neighbor notNil)
                ifTrue: [ neighbor printBoard ].
        ('column ', column, ' row ', row) print
```

Putting all the methods for class **Queen** together, we could type the following example script:

```
lastQueen ← nil.
(1 to: 8) do: [:i | lastQueen ← Queen new;
                        setColumn: i neighbor: lastQueen ]
```

```
        lastQueen first
        lastQueen printBoard
column 1 row 1
column 2 row 5
column 3 row 8
column 4 row 6
column 5 row 3
column 6 row 7
column 7 row 2
column 8 row 4

        lastQueen next
        lastQueen printBoard
column 1 row 1
column 2 row 6
column 3 row 8
column 4 row 3
column 5 row 7
column 6 row 4
column 7 row 2
column 8 row 5
```

SUMMARY

We have, unfortunately, been able to present only the briefest glimpse of two topics in the paper; namely generators and the language Smalltalk. Readers interested in the first topic would do well to read the contrasting presentation of generators using the language Icon [6]. Readers interested in further information on Smalltalk can find the definitive description in the book by GOLDBERG and ROBSON [5]. The material in this paper is condensed from a fuller exposition in Chapter 8 of [3].

REFERENCES

1. G.M. BIRTWISTLE, O.J. DHAL, B. MYHRHAUG, K. NYGAARD. (1973). *Simula Begin,* Studentlitteratur, Lund, Sweden.
2. T.A. BUDD. (1982). An implementation of generators in C. *J. Computer Lang. 7, 2,* 69-88.
3. T.A. BUDD. *A Little Smalltalk,* Addison-Wesley (to be published in 1986).
4. W.F. CLOCKSIN, C.S. MELLISH. (1981). *Programming in Prolog,* Springer-Verlag, New York.
5. A. GOLDBERG, D. ROBSON, (1984). *Smalltalk-80: The language and Its Implentation,* Addison-Wesley.
6. R.E. GRISWOLD, M.T. GRISWOLD, (1983). *The Icon Programming Language,* Prentice-Hall, Inc., Englewood Cliffs, NJ.

7. D.R. HANSON, R.E. GRISWOLD (1978). The SL5 procedure mechanism. *Comm. ACM 21,* 392-400.
8. D.H. INGALLS (1981). Design principles behind smalltalk. *BYTE 6, 8,* 286-302.

Mathematics as a Cultural and as a Productive Force

- The Mathematical Centre founded February 11, 1946 -

G. Alberts
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

As of so many institutions entering their forties, the founding of the Mathematical Centre can only be properly understood within the context of post-war optimism. Optimism however does not account for its existence up to this day. The foundation of the Centre envisaged putting mathematics to the service of society. It meant the birth of the idea of a new societal role for mathematics, and apparently the time was right for it.

Key ideas of post-war optimism in the Netherlands were *breakthrough* and *reconstruction*: breakthrough of a new pattern of social and political values, reconstruction in the sense of rebuilding and restructuring society along more rational lines. The drum was beaten for the idea of reconstruction to counter what was conceived as a situation of cultural and economic distress.

The fact that mathematicians claimed their part in overcoming what they felt was a cultural and economic lag, certainly meant a great turn. Indeed, what then might be the contribution that mathematics had to offer to this reconstruction of society? Both D. van Dantzig and J.G. van der Corput, two of the founding fathers of the Mathematical Centre, had their ideas on this point. Mathematics in their view is a cultural force, 'a primordial asset of civilization', and simultaneously a productive force. The latter idea, mathematics in direct service of economic welfare, is a memorable breakthrough. This breakthrough is primarily due (in the Netherlands) to van Dantzig. We will show that van Dantzig took radical consequences of this new view on mathematics. A topologist at the outset, he specialized in mathematical statistics from 1939 and made a major contribution to the development of mathematical modelling in the Netherlands. In fact van der Corput stated in 1946 that, where the Mathematical Centre stands on the two pillars of pure mathematics and application-orientedness, this combination was van Dantzig's idea in the first place [2].

At the time the combination of pure research with consultation was internationally unique. In the same period around 1950 a number of Computation Centres, Statistical and Econometric Institutes were founded in several countries. Also the systematic funding of pure research was an international trend, exemplified by the NSF in the US 1945.

Towards a science policy

In 1945 the stimulation of science in the Netherlands was taken up at great pace. Not only scientists but also the government launched initiatives. G. van der Leeuw, Minister of Education, Arts and Sciences, took a crucial role. Taking the first steps towards what was to become in 1950 ZWO, the Netherlands Organization for the Advancement of Pure Research, he found immediate support with the Prime-Minister Schermerhorn. Inviting the members to the committee studying the possibilities for a ZWO to be founded Schermerhorn writes: March 1946 'As you undoubtedly will know, the Government intends to stimulate and support on unprecedented scale fundamental scientific research in the Netherlands both in the field of science and in the humanities. The final goal of such research will be that the results benefit the welfare of Dutch society' [21, pp. 6,7].

Reconstruction indeed, and, more than that, restructuring: new goals and new directions were set for *pure* scientific research. Both scientists and policy makers held great expectations of pure science. We have to keep in mind here that TNO, the Organization for Applied Scientific Research, had already been founded in 1930.

The new development on the side of pure science was that groups of scientists ascribed direct societal interest to their abstract activities and that they tried to organize this interest. A new feature on the other side is that policy makers recognized this importance of pure science and drew the consequences of a governmental role. From 1945 onwards the first elements of a science policy can be discerned.

In 1949 on the occasion of the parliamental debate on the definitive establishment of ZWO the Minister of Education, Arts and Sciences, Th. Rutten, quotes the founders (1945) of the US National Science Foundation, stating that: 'Today no nation is stronger than its scientific resources' [21 p. 26].

Groningen reconstruction ideals

Van der Leeuw, a Groningen professor of theology, stressed in 1945 the equal importance of the humanities and the natural sciences [21 p. 3]. According to his views the sciences had to play a role in economic and in spiritual welfare in order to counter distress. Van der Leeuw and van der Corput - cofounder of the Mathematical Centre - were among the authors of a manifesto *The Renewal of the University* [20] published immediately after the liberation in 1945. This manifesto also calls for a fundamental restructuring of the universities, not just a rebuilding.

'A new spirit should create a new academic order' [20, p.4], because 'University stands amidst the "crisis of certainties" ... and ... science as such is not in high esteem' [20, p.6]. The authors held that, particularly in those days, the scientist bears the social responsibility to offer spiritual (and economic) guidance. To this end they proposed a truly academic and social training of students, an active exchange between science and society, between science and industry in particular, and cooperation on a national scale. Science should step outside its specialized and isolated institutes: 'There is great need of a point from which the whole Dutch higher education system can be surveyed and controlled. ... What is needed is a Universitas Neerlandica' [20, p.8].

In minister van der Leeuw's innovative plans science was to put itself at the service of society by taking a leading role:
- firstly by the above mentioned spiritual guidance;
- secondly through closer contacts with society, namely with industry;
- thirdly by cooperative efforts to make science in the Netherlands meet high, i.e. international, standards.

In conformity with this view institutes were created where scientific activity could be built up at greater pace and to a higher level than the reconstruction of the separate universities would allow. The Mathematical Centre was the first such institute to be founded (February 11, 1946). Its regulations perfectly reflect the new science policy:

> 'Article 2: The foundation resides in Amsterdam and pursues the object of stimulating the systematic study of pure and applied mathematics in the Netherlands, in order to increase on the one hand the contribution of these fields of science to the rise of the level of welfare and civilization in the Netherlands, and in order to increase on the other hand the contribution of the Netherlands to the international civilization' [18, p.1].

The founding

October 26, 1945, the Minister of Education, Arts and Sciences, G. van der Leeuw, set up the committee for the Coordination of the Higher Education of Mathematics in the Netherlands. The committee was chaired by J.G. van der Corput, the aforementioned professor of mathematics from Groningen, with one of his pupils J.F. Koksma, mathematics professor at the Free University of Amsterdam, as a secretary. Further members were D. van Dantzig, Delft University of Technology mathematics professor; J.A. Schouten who had given up his Delft professorship in 1941 for disappointment in the lack of resistance against the Nazi occupier; the Leiden professor of physics H.A. Kramers and the astronomer, professor M.G.J. Minnaert, from Utrecht.

It was a strong committee in that its members were influential in the Dutch mathematics scene and agreed in new reconstruction-like ideas about university. Moreover it was extremely powerful by its assignment firstly to coordinate in filling up the many vacant chairs of mathematics, and secondly and more importantly, 'to study the possibility and desirability of scientific mathematical activity, and likewise to design the means of establishing closer contacts between pure mathematics and its applications to other fields' [14].

Apparently the minister knew what he was asking for. The response was accordingly quick, in November 1945 the first blueprint was written and on 11 February 1946 the foundation of the Mathematical Centre was a fact, the six members of the Committee themselves acting as its founders. The Centre was supported by the city of Amsterdam and by the Government. Later on it received smaller contributions from various industries.

The founders: two ideas

A changing view of the societal role of mathematics can be found in so many words in the writings of Van der Corput and Van Dantzig.

To Van der Corput is attributed the ambition to take up the role of prewar Göttingen, centre of mathematical Europe. This ambition is written down on paper only once, in a draft version of a letter to F.A. Vening Meinesz who was at the moment visiting the USA to study their ways of funding pure research. Later on it reads: 'to increase the contribution of the Netherlands to international culture' [18, 14]. In such an ambition mathematics is viewed as merely an asset of civilisation. The contribution to civilization is then made simply by cherishing this asset: by studying mathematics up to a high level.

Van der Corput wanted in fact more than that. Others should be brought into contact with this asset and learn to benefit from it. According to his view mathematicians have a duty towards society to disseminate their knowledge. In 1940 and 1941 in Groningen Van der Corput organized summer schools in mathematics for teachers and others.

The summer school was to become the oldest tradition at the Mathematical Centre, being held annually from 1946 onwards. In those first years the Centre organized courses of continuing education all over the country. Van der

Corput propagated these disseminating activities with a true spirit of mission [5].

> 'The only explanation of why someone chooses to study mathematics is that he gets caught by this science. No one should become a mathematician in search of personal success, but only to contribute to the expansion of a science which is of great importance to mankind. In doing the latter he will be a happy man, because he will enjoy what he does. But he will do this not only for joy, but also for sense of duty, because society sustaining him has a right to demand that he spends his talents to its interest. Hardy may say that it is not all that bad if a few university dons spend their lives on unimportant things, I think it is bad for society.' [3].

On the relation of mathematics to its fields of application Van der Corput speaks in terms of Cinderella who, from handmaiden, came to be the queen of science. Sometimes she must come down to the kitchen to the aid of her sisters - and now is such a time, Van der Corput states in 1947 - but she must not dwell there because another, a royal task is awaiting her [1,4].

In this view mathematics remains an autonomous force of culture, guiding the other sciences, guiding culture while it can, but not itself affected by this work.

A slightly different view was held by J.A. Schouten. Mathematics and the field of application - theoretical physics in his case - intermittently help and inspire each other. Contact with work outside mathematics is thus an exchange and does not depend on sense of duty, it is simply a vital neccessity for progress within mathematics [17]. In this view mathematics is no longer purely autonomous.

Schouten's view preludes the other idea (due to Van Dantzig) behind the Mathematical Centre: mathematics as a productive force. There must be no misunderstanding about it that Van Dantzig discerned the cultural role of mathematics as well. He was even very explicit about it, calling the mathematical way of thinking a general pattern of clear thought. Mathematical thought would (through so-called *signific analysis*) help clarify vague concepts in our language and mind [8]. Thus mathematics is seen not just as a cultural asset but as a *cultural force.*

More crucial with respect to the history of the Mathematical Centre, however, is Van Dantzig's recognition of mathematics as a *productive force.* He diagnosed the need of mathematicians with a special training in a variety of fields, such as in government, in industries, in insurance companies. As early as 1940 he proposed a curriculum offering such training at the Delft University of Technology. Only in 1958 could professor R. Timman start the program of educating mathematical engineers. Van Dantzig had further plans for a

mathematical service department. 'Equip a team with calculating devices and let them calculate, let them compute in service of others', he is remembered to have exclaimed in 1940 [10].

At the Mathematical Centre these projects were realized. Van Dantzig is considered to be the spiritual father of the institute. His was the idea of combining pure and applied mathematics within one institution. 'Time will come, when many positions in commerce will be claimed by mathematicians, which are today still occupied by lawyers and economists', Van Dantzig predicted in 1947 [16].

The Mathematical Centre sets out to offer mathematicians another professional perspective besides that of becoming a teacher. Orientating courses and seminars in applicable mathematics were set up. Certainly some students and some of the few industrial mathematicians in those days benefitted profesionally from these courses. All of the mathematicians who got jobs at the Centre in these early years however became academic professors of mathematics soon afterwards (cf. [13, appendix]).

The Mathematical Centre did develop active service branches: the Computing Department, the Consultation Division of the Statistical Department and, to a lesser extent, the Department of Applied Mathematics.[1]

New fields were opened for mathematics to prove its usefulness. Besides the traditional application to the physical sciences new opportunities arose in medical and biological science, in social science, and in the fields of organization and policy making. The exploitation of mathematical thought took new directions, which placed mathematics in a new societal role. With its changing role mathematics itself changed accordingly. New branches of mathematical theory were developed and, perhaps most important, mathematicians adopted a new commitment.

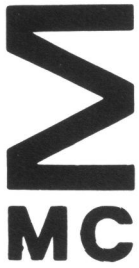
Van Dantzig was one of those mathematicians seeking adequate forms of mathematics to meet the expanding use of mathematical thought, and he drew the consequences radically. Being a topologist at the outset he cooperated with J.A. Schouten in geometrical theories for mathematical physics during the thirties. But then starting in 1939 he specialized in probability and mathematical statistics [6]. Not only did he change subject, but he also committed himself to (statistical) consultation, after and during the war (when he was expelled from Delft and had to go in hiding for some time). The consulting statistician bears, according to Van Dantzig, a dual responsibility: first the responsibility of doing mathematics right and second the commitment to deliver an amenable result within due time [9].

At this point the difference in view with Van der Corput is at its greatest. In

1. Surprisingly, operations research only started to play a role at the Mathematical Centre from about 1954; Dutch industry was earlier.

Van der Corput's view the contact with field of application is a rather one-directional affair. Mathematics is not affected. With Van Dantzig this contact is not only an exchange but so much of a mutual influence that the expanding role of mathematics gives rise to a change of commitment in doing mathematics; in particular a commitment to extra-mathematical goals.

What combines the two ideas, mathematics as a productive and as a cultural force, is service. Mathematics was to be put at the service of society. The idea of mathematics for its own sake was definitely left behind.



REALIZATION

What was realized of these ideas of mathematics as a productive and cultural force can only be briefly indicated here. The board of directors consisted of Van der Corput, Van Dantzig, Koksma and Van der Waerden. As a result of 'coordinating higher education in mathematics in the Netherlands' Van der Corput and Van Dantzig had meanwhile acquired chairs at the University of Amsterdam. B.L. Van der Waerden worked at the time with Shell Research in The Hague. With A. van Wijngaarden, the head of the Computing Department, the members of the board remained the only workers at the Centre for the first year of its existence.

The board, in particular the first three Van der Corput, Van Dantzig and Koksma, met with high frequency: at least once every week. Their first task was to prove the existence of the Mathematical Centre as an institution. Apart from their scientific work they fulfilled this task with discernable personal accents. Koksma was the man of the internal organization of the institute. He was the ideal secretary of the board. If service is the common part of the ideas behind the Mathematical Centre, Koksma was service in person. Van Dantzig could be named the 'philosopher' within the board of directors. As we saw, he was the main inventor of the Centre as an institute of pure and serviceable research. His publications [6, ..., 9] show a developing thought on the serviceability of mathematics. Van der Corput was the director-in-chief and the one who stepped most to the foreground in the public and political scene. Basically, however, they worked in close cooperation. As far as proving the existence of their institute is concerned, they spent fine hours in meetings with industry and with centres of applied scientific research. Fine hours, because in

general first reactions to their proposals were sceptical. Industries favoured the initiatives from the part of the mathematicians, but stated to do their own mathematical research or 'to manage without'. All the same contributions were acquired. Shell and Philips had representatives in the Board of Trustees, and a seminar on applicable mathematics was set up. One extreme was a visit of the complete Board of Directors to the meteorological institute, the KNMI, headed by the aforementioned Vening Meinesz, which by misinterpretation wound up as a working visit solving mathematical problems instead of an introductory talk. The other extreme was a rather 'existential' debate with TNO, the Netherlands Organization for Applied Scientific Research, in which financial support asked from TNO was at stake. The Mathematical Centre had to prove its right of existence next to TNO's own statistical department. After an exchange of formal arguments TNO's president had to give in, inundated by Van Dantzig and Van der Corput under a stream of examples of statistical and numerical problems occurring in applied science [16].

Starting slowly in 1947, a stream of computational, statistical (and later operations research) consultation projects built up during the late forties, stabilizing in the fifties. As far as computation is concerned we have to keep in mind that the work was done by hand, by a team of lady calculators handling electromechanical devices. Among the work were calculations of function tables, which no one would think of doing anymore today, but which were at the limits of computational capacity at that time. The Computing Department did build computers but the first machine actually working in service was not completed until 1954. The Computing Department has its own history which is not touched upon here any further.

The Statistical Department handled a growing number of statistical consultations. From 1950 a series of ready made Memoranda, explaining statistical tests, were published and added as appendices to reports. J. Hemelrijk, head of the statistical consultation division from 1948, built up an individual routine and style of consultation.

Mathematical modelling

One part of the developments after 1945 was, as we saw, a change of opinions among mathematicians and a change of expectations in society concerning science in general and concerning mathematics in particular. The broad support for a more application oriented view can be judged from the members of the Board of Advisors of the Mathematical Centre. All professors of mathematics and some from related fields had been invited to participate in the Board of Advisors, actually serving as a recommending committee, and practically all accepted.

A change of opinions, however, is not sufficient to make mathematics in fact serviceable. In order to be engaged in the pursuit of practical purposes, mathematics must first be taken to an adequate serviceable form. Mathematical modelling is such a form. In looking back we can now see that at that time

mathematical modelling was just becoming generally accepted as a common procedure. Mathematical modelling extends the activity of mathematics in a form amenable to practical purposes. Thus mathematics can be engaged directly for economic welfare.

It was again Van Dantzig and later on Timman who propagated this way of making mathematics useful [7,19]. They both describe the procedure of mathematical modelling extensively and offer their views on its scope and limitations. The expression 'mathematical modelling' was apparently not commonly known and accepted in those days. Though 'mathematical model' does occur as an expression with both authors, they do not present it as a keyword. Van Dantzig, following Mannoury, speaks of 'switching on' and 'switching off' the formalism [7,15]. The 'general pattern of clear thinking' which mathematics has to offer, according to Van Dantzig, has a concrete version through mathematical modelling.

Van Dantzig's example was of course mathematical statistics, which he calls a matter of 'testing probability-theoretical models'. Timman's example was applied scientific research: "Technology raises questions and demands an answer to them. Applied mathematics can proceed no other way than attacking such problems by 'third degree' methods in order to acquire the desired results. Consequently applied mathematics must be aware that it should take utmost care in drawing conclusions from results aquired this way" [19, p.15]. Timman worked with the Centre for a short period only, heading the Applied Mathematics Department in 1951-1952. He did however maintain close contacts from 1947 onwards.

Reflecting the context

The Mathematical Centre was founded in 1946 in the societal context of *reconstruction.* The Minister of Education, Arts and Sciences, Van der Leeuw, and the Prime Minister Schermerhorn staked out the policy frames for a new societal role of science. Their ideals of science guiding civilization and producing welfare were reflected for the particular case of mathematics in the background ideas of the Mathematical Centre: mathematics as a cultural force and as a productive force.

The general acceptance of such tendency towards a new societal role of mathematics certainly did mean a breakthrough in the usual sense for the Dutch mathematical community. This change in view was not merely an academic matter. The abstract science of mathematics proved to have in fact something to contribute to reconstruction. In the concrete form of mathematical modelling mathematics was made serviceable to industry and commerce, to society in general. Developing a tradition of mathematical modelling was one of the characteristics of the Mathematical Centre. Thus the postwar context was not only reflected in views concerning the particular case of mathematics but also given concrete form.

Being the reflection of a general cultural and political context the

Mathematical Centre was not unique. Others than the founders of the Centre, and of course others abroad, held similar views on mathematics. Outside the academic scene mathematical modelling was practiced and further developed, particularly in the fields of econometrics, statistics and operations research. Industry and commerce had not been waiting and had some right to be sceptical in the talks with the Centre's Board of Directors.

What is special about the Mathematical Centre is the fact that academic mathematicians were engaged in developing here practice-oriented mathematics, although here as well no uniqueness can be claimed. A singularly beautiful example of academic-commercial cooperation is given by Freudenthal (Utrecht State University) and Sittig (advisor for Applied Statistics, Rotterdam) conducting a large scale statistical enquiry in order to develop a sizing system for clothing [11]. At the Mathematical Centre however the pursuit of serviceable mathematics was systematic and institutionalized. Still in 1959 Hemelrijk claimed worldwide uniqueness for the Mathematical Centre as an institution engaging mathematicians in the combination of pure and application oriented mathematics [12].

In the Netherlands the Mathematical Centre acquired a central position in this field. Today a majority of Dutch academic computer scientists and statisticians count Van Wijngaarden and Van Dantzig respectively as their scientific father, grandfather, or great grandfather... . Indeed at the crossroads of application oriented mathematics and pure scientific research the Centre played an initiating and forerunning role in those early years. Part of this role was taken over when the Universities of Technology established a program for the education of Mathematical Engineers, for the first time at Delft in 1958, initiated by Timman.

When in 1952 Timman was called away from the Mathematical Centre to take a chair of mathematics in Delft he consolidated the attained changes and preluded further development [19, p.16]: 'It is my impression that a large influx of mathematicians with an insight in technology or of engineers with a sound knowledge of mathematical methods will be welcomed with joy in many places. Also in our country the road has been opened for mathematics to a new social function and I hope that the opportunity will be found to make it hold this function with honour'.

References

1. J.G. van der Corput (1940). Wiskunde. *Wegen der Wetenschap,* Wolters, Groningen, 29-44.
2. J.G. van der Corput (1946). *Het Mathematisch Centrum,* P. Noordhoff, Groningen.
3. J.G. van der Corput (1946). Het Mathematisch Centrum en het Middelbaar Onderwijs, *Archivalia CWI.*
4. J.G. van der Corput (1947). *Betekenis der Wiskunde Heden ten Dage voor Andere Wetenschappen,* Diligentia, 's Gravenhage.
5. J.G. van der Corput. Wordings of prof.dr. J. Korevaar remembering van der Corput in an interview with the author.
6. D. van Dantzig (1941). Mathematische en empirische grondslagen der waarschijnlijkheidsrekening. *Ned. Tijdschrift voor Natuurkunde 8,* 70-93.
7. D. van Dantzig (1947). General procedures of empirical science. *Synthese V-1,* 1-15.
8. D. van Dantzig (1948). Over de maatschappelijke functie van zuivere en toegepaste wetenschap. *De Functie der Wetenschap,* H.P. Leopolds, 's Gravenhage.
9. D. van Dantzig (1954). De verantwoordelijkheden van de statisticus. *Statistica Neerlandica 7,* 199-208.
10. D. van Dantzig. Van Dantzig as quoted by prof.dr. N.G. de Bruijn in an interview with the author.
11. H. Freudenthal, J. Sittig (1951). *De Juiste Maat,* Bijenkorf, Amsterdam.
12. J. Hemelrijk (1959). In Memoriam prof.dr. D. van Dantzig. *Statistica Neerlandica 13,* 416-432.
13. J.F. Koksma (1960). *Het Mathematisch Centrum 1946-1960,* Mathematical Centre.
14. LETTER 1945. Letter to F.A. Vening Meinesz by the committee for the coordination of the higher education of mathematics in the Netherlands. *Archivalia, CWI.*
15. G. Mannoury (1947). *Handboek der Analytische Significa I,* F.G. Kroonder, Bussum.
16. Minutes of the Board of Directors, *Archivalia, CWI.*
17. J.A. Schouten (1949). *Over de Wisselwerking tussen Wiskunde en Physica in de Laatste 40 Jaren,* Noordhoff, Groningen.
18. Stichtingsakte van de Stichting Mathematisch Centrum (1946). *Archivalia, CWI.*
19. R. Timman (1952). *De Betekenis van de Wiskunde voor het Toegepast Wetenschappelijk Onderzoek,* Waltman, Delft.

20. J.H. BROUWER, J.G. VAN DER CORPUT, M.N.J. DIRKSEN, G. VAN DER LEEUW, C.W. VAN DER POT, M.J. SIRKS. *De Vernieuwing van de Universiteit,* J.B. Wolters, Groningen.
21. *ZWO Voorbereiding en Werkzaamheden in de Oprichtingsperiode 1945-1949* (1950), ZWO 's Gravenhage.

Van Dantzig's car at the Mathematical Centre, 2e Boerhaavestraat 49, Amsterdam.

The MC Fortieth Anniversary

P.C. Baayen
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

A few months ago our Centre for Mathematics and Computer Science, CWI, entered its fifth decade. It is an occasion for reflection, for looking back and for a look forward.

It is an exciting time for research in computer science and in mathematics. New results and new products, as well as (sometimes unexpected) combinations of insights won in separate branches, lead to very powerful techniques indeed. More than ever, mathematics provides the language and the foundations for developments in science, technology and social organization, together with computer science which in addition furnishes vital and by now absolutely essential supporting tools, sc. computers, systems for complex information processing and, increasingly, for knowledge handling.

In early 1946, when CWI was founded, times were exciting, too. The world had just passed out of the dark shadows of a terrible and devastating world war into the bright and hopeful area of rebirth, of rebuilding society and civilisation. The founding fathers of the CWI - at that time named Mathematical Centre, as informatics and computer science had not yet emerged as a separate discipline - were convinced that mathematics and mathematicians could contribute to the restructuring of society. Their aim, as put down in the charter of the new centre, was 'to promote the systematic pursuit of pure and applied mathematics in the Netherlands, in order to increase, on the one hand the contributions of these disciplines to a higher level of prosperity and civilization in the Netherlands, on the other hand the contribution of the Netherlands to international culture'.

Elsewhere in this Newsletter G. Alberts devotes a study to these early days of the Mathematical Centre/Centre for Mathematics and Computer Science. He encapsulates the two main aims from the charter in the key words: mathematics as as *Productive* and as a *Cultural* force. Alberts succeeds well in

conveying the ideas and the idealism of those early days.

In the fourty years which passed since 1946, the CWI developed into an institute with a personnel of over 200, of which some 120 are directly involved with research. Several scores of university professors in the Netherlands and abroad worked at the CWI as researchers. In many hundreds of cases the CWI provided mathematical and computational support to technical and scientific projects, originating from industry, from government organizations or from other research institutes. The CWI was instrumental in introducing mathematical statistics and operations research as scientific disciplines into the Dutch university curriculum, and it was the cradle of computer science in the Netherlands. Through the years, several international conferences were organized at or by the CWI, researchers from all over the world visit us (and sometimes stay for a considerable time, e.g. a full sabbatical leave), and researchers of the CWI are regularly invited to visit colleagues or contribute to conferences abroad. Indeed, the researchers at the CWI have convincingly shown mathematics to be both a productive and a cultural force.

And what about the future? What will the next decade bring us? Well, as I already mentioned before, it is again an exciting time for mathematics and computer science. A recombination and unification is going on of mathematical subdisciplines. The support provided by the products of computer science have postered new developments in computational mathematics that are of great importance both for theory and for applications. The traditional superstition of a division between 'pure' and 'applied' mathematics is breaking down, and new applications of mathematical methods are made in fields as disparate as off-shore technology and the study of epidemics. The development of computer science itself generates interesting mathematical problems, e.g. with regard to concurrency and distributed systems (parallel algorithms and mathematical performance analysis are two examples that come to mind).

The CWI is well set up to contribute to the flourishing of computer science and mathematics. Both fields are well represented in our present research program. Good cooperation between the research groups provides an excellent basis for mutual support and interaction. Access to practical applications through cooperation with industrial groups provides vital stimuli.

Recently the government has increased its support of the information sciences, and in this framework computer science at the CWI has expanded considerably. The aim is to let the CWI become a 'centre of exellence' in this field. At the same time however the Board of Trustees of the CWI is cautious to maintain at the institute a balance between research in mathematics and computer science, in the conviction that the flourishing on one needs the support of the other. Only the future will learn if we will succeed in realising our ideals. We are convinced we will.

The Tetrahexes Puzzle

Herman J.J. te Riele & D.T. Winter
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

During the celebration of the 40-th anniversary of the Mathematical Centre on February 11, 1986, the participants received a small puzzle consisting of the following seven pieces, sawn from perspex:

These pieces represent the seven different ways to join four congruent hexagons.[1]

In the June 1967 issue of *Scientific American*, MARTIN GARDNER devoted a part of his famous column to this puzzle. He proposed to call the pieces 'polyhexes' after DAVID KLARNER, who was one of the first to investigate them (pieces consisting of four hexagons are called tetrahexes, pieces consisting of five hexagons are called pentahexes, and so on). The number of pentahexes is 22; computer programs have counted 82 hexahexes, 333 heptahexes and 1448 octahexes. No formula is known to determine the number of n-hexes for a given value of n. Together with the puzzle, 15 figures were provided as exercise material (the exercise being to form each figure with the seven tetrahexes).

1. A limited number of copies of this puzzle is available to visitors of the CWI who sign the guest book.

Below, we give 49 figures including 8 out of these 15. All of them (except the first one, see below) can be composed, in at least one way, by the seven different tetrahexes. The readers are invited to find solutions and, if possible, to find them all!

The first figure, the triangle, is to many the most appealing one which consists of 28 hexagons. Unfortunately, it has no solution with the seven different pieces, as will be shown below. Figures 2-35 represent all the solvable convex figures (by convex we mean that the curve which connects the midpoints of the outer hexagons is convex) with at most one convex hole and at least one symmetry (i.e. mirror or rotation symmetry). Although the hexagon has a 6-fold rotational symmetry, we found only one solvable pattern with 3-fold rotational symmetry (Figure 36). Figures 37-49 are solvable 'rectangular-like' figures with at most one hole and at least one symmetry. There may be others. For the triangle, DAVID KLARNER showed that it has no solution. As far as we know, Klarner's proof has not been published. Here, we will give a proof which we suspect to be similar to Klarner's proof.

PROOF THAT THE TRIANGLE CAN NOT BE COMPOSED OF THE SEVEN TETRAHEXES.

The positions of the triangle are numbered 1 - 28 as follows.

```
                1
              2   3
            4   5   6
          7   8   9  10
        11  12  13  14  15
      16  17  18  19  20  21
    22  23  24  25  26  27  28
```

As Klarner did, we start by observing that the propeller can only be placed in a limited number of positions, viz., in positions 4 8 9 12, 5 8 7 13 and 8 13 14 18 (apart from mirror-image forms). When the propeller is in position 4 8 9 12, the bee should be in position 1 2 3 5. Next, there are three possible locations for the arch; and so on. The complete proof is given in a tree below.

Some abbreviations are being used: the different pieces are identified by their first two characters. An 'x' after a piece indicates that this can not be placed, or it splits the figure in parts with numbers of free hexagons which are not a multiple of four. A 'd' after a piece means that it should be placed somewhere in the figure, but has been placed somewhere else earlier.

```
pr 4 8 9 12
  be 1 2 3 5
    ar 6 10 13 14
      wa 15 20 19 25
        pi 21 28 27 26
          ba 7 11 16 22
            wo x
      wa 18 19 26 27
        pi 20 15 21 28
          ba 7 11 16 22
            wo x
          ba 22 23 24 25
            wo x
    ar 7 11 17 18
      pi 16 22 23 24
        wa 6 10 14 20
          ba x
        wa 25 19 20 15
          ba x
    ar 18 13 14 20
      ba 6 10 15 21
        wa x

pr 8 13 14 18
  ar 5 4 7 12
    pi 2 1 3 6
      wa 19 20 27 18
        wo 9 10 15 21
          be x
  ar 7 4 5 9
    pi 2 1 3 6
      wa x
```

```
pr 7 8 5 13
  pi 3 1 2 4
    ar 9 14 19 18
      ba 6 10 15 21
        pi d
      wo 6 10 15 20
        pi d
    ar 12 18 24 23
      pi d
    ar 12 18 19 14
      pi d
    ar 12 17 24 25
      pi d
    ar 16 11 12 18
      pi d
    ar 11 12 18 24
      be 17 16 23 22
        wa 6 9 14 19
          wo x
        wa 14 20 21 28
          ba x
        wa 25 19 20 15
          ba x
```

The reader is invited to find a shorter proof (than this 54-lines one).

Of course, our method also finds solutions, if they exist. For example, for figure 38, one rather quickly finds two solutions as follows (the complete tree is very long and yields 21 more solutions).

The positions are numbered as follows:

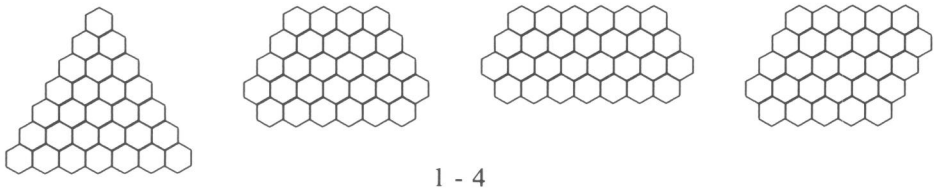
```
 1   2   3   4
 5   6   7   8
 9  10  11  12
13  14  15  16
17  18  19  20
21  22  23  24
25  26  27  28
```

The solution tree then reads:

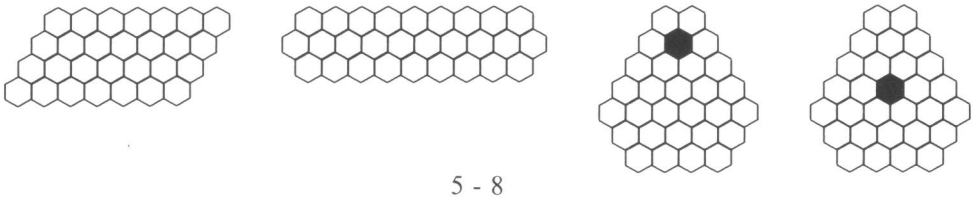
```
pr 3 7 6 11
  wa 4 8 12 16
    ar 2 1 5 9
      ba 17 18 19 20
        pi 10 15 14 13
          be 21 22 25 26
            wo x
          be 23 24 27 28
            wo x
      ba 21 22 23 24
        ba d
      ba 25 26 27 28
        be 10 14 15 18
          wo x
        be 18 17 22 21
          be d
        be 20 19 24 23
          wo 21 22 18 15
            pi 13 17 14 10  first solution
          wo 10 15 18 22
            pi 13 14 17 21  second solution
        be 13 14 17 18
          pi x
```

Of course, this may be programmed very efficiently in a language which allows for recursiveness. Some manual exercises with the above trees will certainly help to improve the performance of such programs.

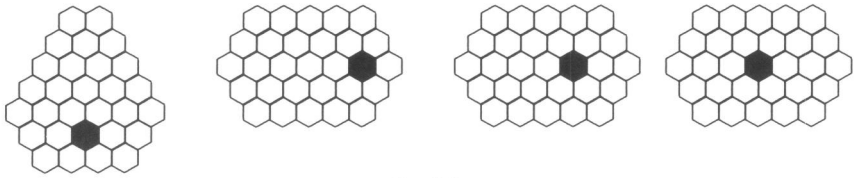
FIGURES 1-49



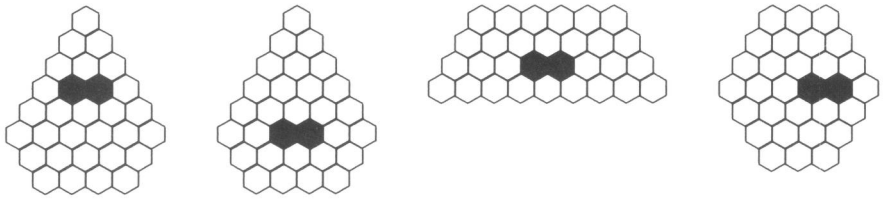
1 - 4



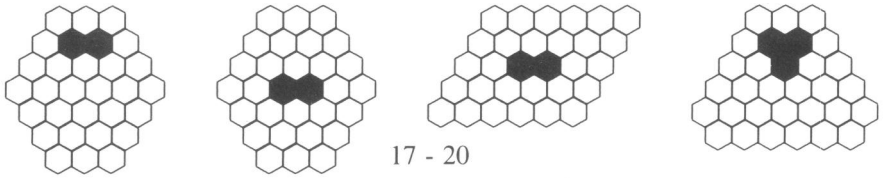
5 - 8



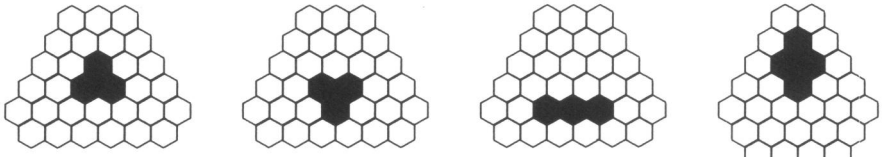
9 - 12



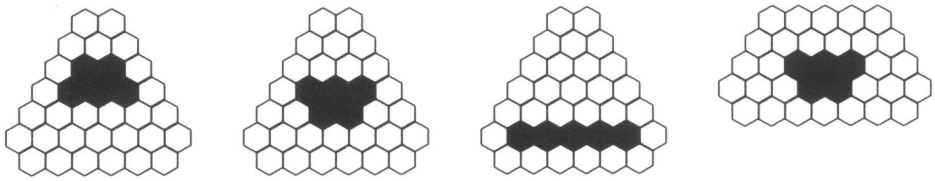
13 - 16



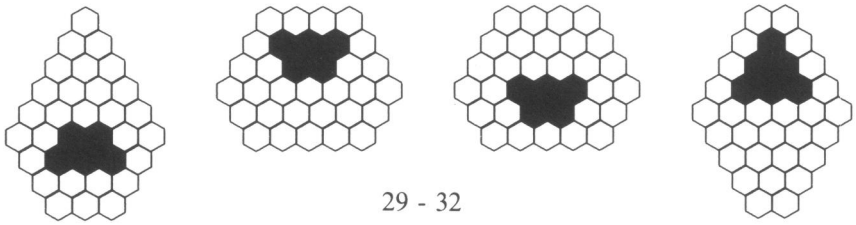
17 - 20



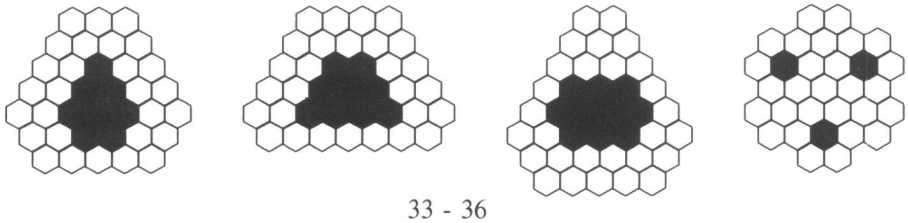
21 - 24



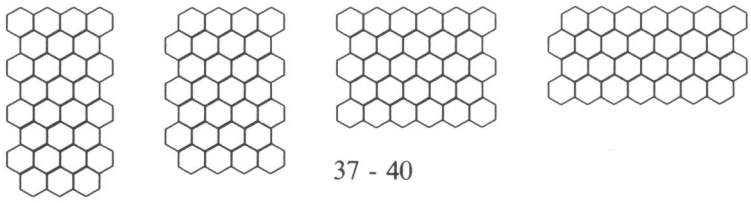
25 - 28



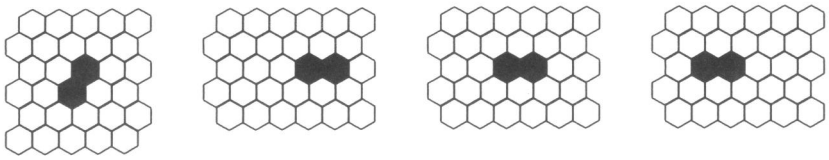
29 - 32



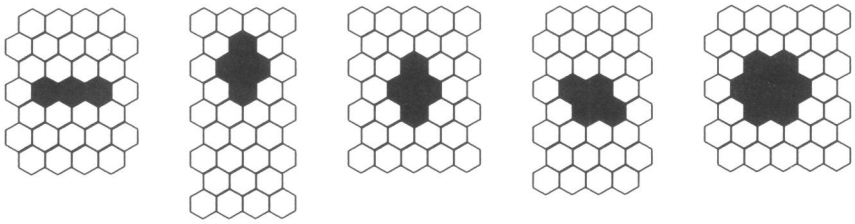
33 - 36



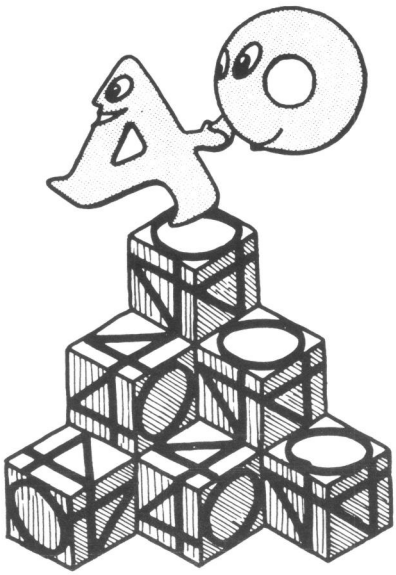
37 - 40



41 - 44



45 - 49



New Factorization Records on Supercomputers

Herman J.J. te Riele, Walter M. Lioen & Dik T. Winter
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Two new factorization records on supercomputers have been recently settled at the Centre for Mathematics and Computer Science (CWI) at Amsterdam. The numbers factorized into their prime factors are the 75-digit composite number:

$$(2^{271} - 2^{136} + 1) \;/\; 10474693 =$$

362232590504406868214617272006414447138055859511851874054762224427178428941 =

104167755499168696693743867494211841 *
347739652033970953169994378027632511310 1

(the product of a 36- and a 40-digit prime number)

and the 72-digit composite number:

$$10^{72} - 10^{36} + 1 =$$

999999999999999999999999999999999999900000000000000000000000000000000000001 =

172629000899150450017746330268869 7 *
579276943498154282123686999881829009033

(which is the product of a 34- and a 39-digit prime number).

The factorized numbers were taken from a list of 20 composite numbers whose factorizations are unknown, and 'more wanted', from the book (sometimes called the factor bible): *Factorizations of* $b^n \pm 1$, by JOHN BRILLHART, D.H. LEHMER, J.L. SELFRIDGE, BRYANT TUCKERMAN and S.S. WAGSTAFF, JR. (AMS Contemporary Mathematics Series, vol. 22, 1983).

Interest in factorization and primality testing has increased dramatically since the discovery, in 1978, by RIVEST, SHAMIR and ADLEMAN, that the difficulty of breaking certain cryptographic codes depends on the difficulty of factorization [3].

The method used is the multiple polynomial version of Peter Montgomery of the quadratic sieve method of Carl Pomerance as described in a recent paper by POMERANCE, J.W. SMITH and R. TULER [2]. The computer used is the 1-pipe CDC CYBER 205 of SARA at Amsterdam (SARA is the Academic Computer Centre Amsterdam). The total time used was about 4.3 hours CPU-time for the 72-digit number and 12.2 hours for the 75-digit number. Control Data Benelux has kindly provided part of the computer time for these (and other) factorizations. The method was implemented on the CYBER 205 by a team consisting of Herman J.J. te Riele, Walter M. Lioen and Dik T. Winter from the Department of Numerical Mathematics of the CWI. Advisory help was provided by J. Schlichting of Control Data.

The previous record for supercomputers was held by J.A. Davis and D.B. Holdridge from Sandia Labs (USA) who (in 1984) factorized the number $(10^{71} - 1) / 9$ (consisting of 71 1's) on a CRAY X/MP-24 of the Los Alamos Lab (USA) in 9.5 hours CPU-time, using a variant of the quadratic sieve method found by DAVIS [1]. This CRAY X/MP is about twice as fast as the CYBER 205 and has four million words of central memory (the CYBER 205 has one million words). In the heart of the quadratic sieve algorithm, the data to be handled are stored in non-contiguous memory locations. This is a handicap on the CYBER 205. All this illustrates the power of the Montgomery-variant of Pomerance's quadratic sieve.

It should be emphasized that larger difficult numbers have been factorized already by Robert Silverman, who did not use supercomputers, but VAX- and SUN-computers. His record is: a 81-digit composite number using a total of 1260 hours on 8 SUN-3/75 computers running in parallel. He also used the MP-QS method.

A few more details of our algorithm for the initiate:

	c72	c75
multiplier used:	none	5
factor base bound:	130000	160000
# primes in the factor base:	6071	7322
length of sieving interval:	$6(2^{16}-1)$	$6(2^{16}-1)$
# of completely factorized w's:	2672	3376
# of incompletely factorized w's:	24747	26062
# of large prime equalities in the incompletely factorized w's:	3401	3947
bound on the large primes allowed in incomplete w's:	30x130000	20x160000
Gauss elimination time (on a 6073x6072, resp. 7323x7323 linear system):	21 sec.	37 sec.
# of dependencies found:	65	509

References

1. J.A. Davis, D.B. Holdridge, G.J. Simmons (1985). Status report on factoring (at the Sandia National Laboratories). T. Beth, N. Cot, I. Ingemarsson (eds.). *Advances in Cryptology, Proceedings of EUROCRYPT 84*, 183-215 Springer, Berlin etc.
2. C. Pomerance, J.W. Smith, R. Tuler (1986). *A Pipe-Line Architecture for Factoring Large Integers with the Quadratic Sieve Algorithm.* Preprint.
3. R. Rivest, A. Shamir, L. Adleman (1978). A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM 21*, 120-126.

Abstracts

of Recent CWI Publications

When ordering any of the publications listed below please use the order form at the back of this issue.

CWI Tract 20. B.F. Schriever. *Order Dependence.*
AMS 62H17, 62H20, 62H10, 62H99; 115 pp.

Abstract: This monograph considers some aspects of the statistical analysis of ordered contingency tables. An often used method for analysing bivariate tables is Correspondence Analysis (CA). Basic properties of this technique and asymptotic properties of tests of independence based on statistics produced by CA are derived. Furthermore, it is shown that CA possesses an ordering property, which is quite relevant for the practice of CA, when the bivariate contingency table shows a specific form of ordinal dependence. This form is related to Lehmann's notion of positive regression dependence. Tests of independence sensitive to this form of positive dependence are discussed. Also, a partial ordering for positive dependent distributions is introduced such that these and other familiar tests become more powerful under 'increasing' positive dependence. Finally, ordering properties of a multivariate generalization of CA are given.

CWI Tract 21. D.P. van der Vecht. *Inequalities for Stopped Brownian Motion.*
AMS 60-02, 60G40, 60J65, 60G44, 60E10; 88 pp.

Abstract: The Blackwell-Dubins bound is a stochastic upperbound for the maximum of a uniformly integrable martingale. In this work it will be derived directly for standardly stopped Brownian motion. A similar upperbound is obtained for the maximum of the norm of standardly stopped d-dimensional Brownian motion. To show it is a least upperbound we define Azéma-Yor type stopping times. These are essentially the only ones for which the bound is attained. The definition of such a stopping time involves a characteristic of the embedded distribution. We prove continuity theorems for the involved characteristics.

CWI Syllabus 8. G.M. Tuynman. *Proceedings Seminar 1983-1985*

Mathematical Structures in Field Theories, Vol.1: Geometric Quantization.
AMS 58F06, 81D07; 158 pp.
Abstract: Geometric quantization attempts to give a mathematically rigorous procedure to derive the quantummechanical description of a physical system from the classical description (in the Hamilton form). This monograph gives explanations and (heuristic) motivations for the fundamental concepts of this theory, i.e., the prequantization line bundle (which also appears in classical mechanics!), polarizations (needed to reduce the 'size' of the Hilbert space) and the metalinear correction (which is needed, among other things, to recover the usual energy shift of $\frac{1}{2}\hbar$ of the harmonic oscillator). After each step several examples are given and for each new feature, the influence on the example is studied.

CWI Syllabus 9. J. van Leeuwen & J.K. Lenstra. *Parallel Computers and Computations.*
AMS 68A05, 68B20, 68Cxx, 68Exx, 65Fxx; CR C.1.2, D.1.3, D.4.7, F.1.0, F.2.1, F.2.2; 184 pp.
Abstract: In the Autumn of 1983 a series of eight lectures was organized at the University of Utrecht to focus attention at the new developments in 'parallel computers and computations'. Eight experts of different backgrounds were invited to survey or describe an aspect of this field of research. The lectures covered concrete supercomputer architectures and their programming, the new challenges for systems programming, the design of numeric and combinatorical parallel algorithms, and the complexity of parallel computations. This volume contains the full versions of the papers that were presented.

CS-R8601. J.W. de Bakker, J.-J.Ch. Meyer, E.-R. Olderog & J.I. Zucker. *Transition systems, metric spaces and ready sets in the semantics of uniform concurrency.*
AMS 68B10, 68C01, 68D25, 68F20; CR D.3.1, F.3.2, F.3.3; 104 pp.; **key words:** concurrency, denotational semantics, operational semantics, transition systems, metric spaces, ready sets, local nondeterminacy, global nondeterminacy, linear time, branching time, guarded recursion, streams, shuffle, merge, synchronization, parallel execution, interleaving, uniform concurrency, infinitary languages, processes, completion, merging lemma.
Abstract: Transition systems as proposed by Hennessy and Plotkin are defined for a series of three languages featuring concurrency. The first has shuffle and local nondeterminacy, the second synchronization merge and local nondeterminacy, and the third synchronization merge and global nondeterminacy. The languages are all uniform in the sense that the elementary actions are uninterpreted. Throughout, infinite behaviour is taken into account and modelled with infinitary languages in the sense of Nivat. A comparison with denotational semantics is provided. For the first two languages, a linear time model suffices; for the third language a branching time model with processes in the sense of De Bakker & Zucker is described. In the comparison an important role is played by an intermediate semantics in the style of Hoare & Olderog's specification oriented semantics. A variant on the notion of ready set is employed here. Precise statements are given relating the various semantics in terms of a number of abstraction operators.

CS-R8602. P.J.W. ten Hagen & C.G. Trienekens. *Pattern representation.*
AMS 69K31, 69K33, 69K36; CR I.3.1, I.3.3, I.3.6; 25 pp.; **key words:** computer graphics, raster displays, pattern representation, scanconversion.
Abstract: This paper introduces a new method of representing area oriented picture primitives. The representation aims at efficient conversion of these primitives to framebuffers for raster

displays. The representation and the conversion to raster can be the basis for area generators being comparable in speed to vector generators used in conventional vector displays. The representation is independent of the type and resolution of raster hardware. As a result interactive graphics on raster display can be of higher quality because of fast responses involving picture change without compromising picture quality. This paper concentrates on the treatment of the area's domain, e.g., definition and manipulation of domains. The generation of the texture in the domains will be dealt with in a subsequent paper.

CS-R8603. J.W. de Bakker, J.N. Kok, J.-J.Ch. Meyer, E.-R. Olderog & J.I. Zucker. *Contrasting themes in the semantics of imperative concurrency.*
AMS 68B10, 68C01; CR D.3.1, F.3.2, F.3.3; 58 pp.; **key words:** concurrency, imperative languages, denotational semantics, operational semantics, transition systems, metric spaces, ready sets, local nondeterminacy, global determinacy, linear time, branching time, guarded recursion, finite observations, streams, synchronization, communication, value passing, parallel execution, interleaving, shuffle, merge, processes, uniform concurrency, nonuniform concurrency, infinitary languages.

Abstract: A survey is given of work carried out by the authors in recent years concerning the semantics of imperative concurrency. Four sample languages are presented for which a number of operational and denotational semantic models are developed. All languages have parallel execution through interleaving, and the last three have a form of synchronization as well. Three languages are uniform, i.e., they have uninterpreted elementary actions; the fourth is nonuniform and has assignment, tests and value-passing communication. The operational models build on Hennessy-Plotkin transition systems; as denotational structures both metric spaces and cpo domains are employed. Two forms of nondeterminacy are distinguished, viz. the local and global variety. As associated model-theoretic distinction that of linear time versus branching time is investigated. In the former we use streams, i.e., finite or infinite sequences of actions; in the latter the (metrically based) notion of process is introduced. We furthermore study a model with only finite observations. Ready sets also appear, used as technical tool to compare various semantics. Altogether, ten models for the four languages are described, and precise statements on (the majority of) their interrelationships are made. The paper supplies no proofs; for these, references to technical papers by the authors are provided.

CS-R8604. T. Tomiyama & H. Yoshikawa. *Extended general design theory.*
AMS 03E30, 54A05, 54D35, 69K14, 69L60; CR H.2.1, I.2.4, J.6; 29 pp.; **key words:** design theory, axiomatic set theory, machine design, data model, knowledge representation, CAD.

Abstract: Computer Aided Design (CAD) systems are getting more and more popular in many industries. Because designing is an intellectual process, CAD systems of the next generation are supposed to have intelligence. This is why many researchers are recently working on knowledge based CAD systems. However, despite of these efforts, it is now becoming clear that ad hoc approaches are unsuccessful and that we need a guiding principle for implementing such CAD systems. In this paper, we propose a design theory to formalize design processes and design knowledge. General Design Theory is based on axiomatic set theory, and it clarifies what designing is, how to formalize a design process, and how to describe design knowledge. We begin with three axioms which define an ideal state. Then, we think about a real problem by adding a hypothesis about the physical aspect of our world, so that we can deduce theorems about real design processes and discuss theoretical problems of the data description method of future CAD systems.

CS-R8605. J.N. Kok. *Denotational semantics of nets with nondeterminism.*
AMS 68B10; CR D.3.1, F.3.2, F.3.3; 12 pp.; **key words:** dataflow programming, dataflow networks, denotational semantics, metric topology, multivalued functions, concurrency.

Abstract: We define a topological framework for streams of traces. With this approach Kahn's method generalizes to nets with bounded nondeterminism. We consider fixpoints of multivalued functions. We have a standard fixed point theorem, which can be used to model feed back loops. These fixed points can also be obtained by iteration. We give a general syntax of nets and see how we can analyze them in our streamframework. We show how to avoid the Brock-Ackerman and Keller anomalies. We are able to model the fair merge, which is a continuous function in our framework, and delay along lines. We prove a lemma that says that the order in which we connect nodes in our networks does not matter. If we have nets with nodes with unbounded nondeterminism, we can still use these fixpoints, but we do lose in our topological framework our iteration theorem.

CS-R8606. E. Kranakis. *Fixed point equations with parameters in the projective model.*
AMS 68B05; 23 pp.; **key words:** process algebra, process, fixed point equations, polynomial operator, metric space, continuous, contraction, dense, compact, guard, fixed point.

Abstract: Existence and uniqueness theorems are given for solving infinite and finite systems of fixed point equations with parameters in the projective model. The three main methods discussed depend on the topological properties of the projective model and they include: compactness argument, density argument, and Banach's contraction principle. As a converse to the uniqueness theorem it is also shown that in certain signatures guarded equations are the only ones that have unique fixed points.

CS-R8607. E. Kranakis. *Approximating the projective model.*
AMS 68B05; 15 pp.; **key words:** process algebra, projective model, polynomial operator, positive formula, metric space, approximation principle, ultrafilter, ultraproduct.

Abstract: An approximation principle for the projective model is given which makes it possible to prove assertions in this model by proving them in an infinite sequence of certain finite process algebras. Motivated from this principle a new model for process algebras is defined and its relationship to the projective model is studied.

CS-R8608. F.W. Vaandrager. *Verification of two communication protocols by means of process algebra.*
AMS 68B10, 68C01, 68D25, 68F20; CR F.1.1, F.1.2, F.3.2, F.4.3, C.2.2; 76 pp.; **key words:** process algebra, concurrency, communication protocol, verification, fairness, trace set, redundancy, local replacement.

Abstract: A positive acknowledgement with retransmission protocol, and a one bit sliding window protocol are verified in the framework of ACP_τ, the algebra of communicating processes with silent steps, augmented with some additional axioms. We present the Cluster Fair Abstraction Rule (CFAR), which is a generalization of Koomen's Fair Abstraction Rule (KFAR), and show that CFAR can be derived from $KFAR_1$. We introduce the notion of redundancy in a context, which makes it possible to use trace theoretic arguments in process algebra calculations. For the verification of the second protocol, we use the technique of local replacement. In this technique a concurrent system is simplified by repeated replacement of components, replacements which leave

the behaviour of the system invariant.

CS-R8609. J.A. Bergstra, J.W. Klop & E.-R. Olderog. *Failure semantics with fair abstraction.*
AMS 68B10, 68C01, 68D25, 68F20; CR F.1.1, F.1.2, F.3.2, F.4.3; 68 pp.; **key words:** process algebra, concurrency, failure semantics, bisimulation semantics.
Abstract: We consider countably branching processes subject to operations: alternative composition ($+$), sequential composition ($\cdot$) and abstraction (τ_I). Parallel operators are not yet considered. Axiom systems are given for such processes which contain δ (deadlock), ϵ (empty process), τ (silent move) and Δ (delay). The emphasis is on axiomatising divergence via the new constant Δ, both in the context of bisimulation semantics and failure semantics. A new process model is found which is intermediate between bisimulation semantics with fair abstraction and failure semantics with catastrophic divergence.

OS-R8601. R.K. Boel & J.H. van Schuppen. *Overload control for switches of communication systems - A two-phase model for call request processing.*
AMS 93E20, 90B22, 60K25; 16 pp.; **key words:** overload control, stochastic control, queueing theory, communication systems.
Abstract: In modern telephone networks the switching and connecting operations are performed by computer controlled switches called Stored Program Control (SPC) exchanges. One of the problems with these switches is the severe performance degradation during periods in which the demand for service exceeds the design capacity. The problem of overload control is to decide whether to admit or not to admit a call request such as to maximize the number of successfully completed calls. In this paper a new and rather general model for the switch is proposed in which the delay in processing a call request is modelled by two phases. From this a simplified model is deduced consisting of a series connection of three random delays. The problem of overload control is then formulated as a stochastic control problem. The solution is a bang-bang control, meaning that a customer is either admitted or not admitted to the switch without randomization.

OS-R8602. J.W. Polderman. *On the necessity of identifying the true parameter in adaptive LQ control.*
AMS 93C40; 5 pp.; **key words:** adaptive LQ control, closed loop identification, certainty equivalence.
Abstract: In adaptive control problems one may drop the requirement of identifying the true system in order to simplify the problem of control. It will be shown that in the adaptive LQ control problem this does not at all lead to an easier problem.

OS-R8603. O.J. Boxma. *Models of two queues: a few new views.*
AMS 60K25, 68M20; 24 pp.; **key words:** survey, two parallel queues, two queues in series.
Abstract: This paper presents a review of results for models of two queues, with an emphasis on mathematical analysis techniques. Two classes of models are investigated: (i) two parallel queues attended by a single server, and (ii) two queues in series.

OS-N8601. W.P. Groenendijk. *The $M/G/1$ queue with randomly alternating services.*
AMS 60K25, 68M20; 84 pp.; **key words:** $M/G/1$ queue, randomly alternating, boundary value problem.
Abstract: In this report the $M/G/1$ queue with randomly alternating services is analysed with

respect to its queue-length process. The queue-length distribution is obtained and explicit expressions are derived for the first moments of various interesting performance measures.

NM-R8524. M. Louter-Nool. *BLAS on the Cyber 205.*
AMS 65V05, 65Fxx; 24 pp.; **key words:** vectorization, basic linear algebra subprograms, stride problems, operations on sequentially and nonsequentially stored real and complex vectors.
Abstract: The subject of this paper is to examine the efficiency of a set of linear algebra subprograms, the so-called BLAS, as implemented on a 1-pipe Cyber 205. Beside this implementation, we have developed an alternative version of the subprograms. The performance of both implementations is compared. Special attention is paid to operations on nonsequentially stored vector elements, leading to so-called stride problems. Several routines, appropriate to deal with those stride problems, are treated. In this paper, we shall only consider the single precision real and complex BLAS subprograms, with positive strides.

NM-R8525. J.M. Sanz-Serna, J.G. Verwer & W.H. Hundsdorfer. *Convergence and order reduction of Runge-Kutta schemes applied to evolutionary problems in partial differential equations*
AMS 65X02, 65M10, 65M20; CR G.1.7; 12 pp.; **key words:** numerical analysis, initial boundary value problems in partial differential equations, method of lines, Runge-Kutta schemes, convergence analysis, order reduction.
Abstract: We address the question of convergence of fully discrete Runge-Kutta approximations. We prove, that under certain conditions, the order in time of the fully discrete scheme equals the conventional order of the Runge-Kutta formula being used. However, these conditions, which are necessary for the result to hold, are not natural. As a result, in many problems the order in time will be strictly smaller than the conventional one, a phenomenon called order reduction. This phenomenon is extensively discussed, both analytically and numerically. As distinct from earlier contributions we here treat explicit Runge-Kutta schemes. Although our results are valid for both parabolic and hyperbolic problems, the examples we present are therefore taken from the hyperbolic field, as it is in this area that explicit discretizations are most appealing.

NM-R8601. B. Koren. *Euler flow solutions for a transonic windtunnel section.*
AMS 35B30, 65N50, 76G15, 76H05; 15 pp.; **key words:** steady Euler equations, transonic flows, grid generation and adaptation, boundary conditions.
Abstract: Two dimensional Euler flow computations have been performed for a windtunnel section, designed for research on transonic shock-wave boundary-layer interaction. For the discretization of the Euler equations, a finite volume Osher discretization has been applied. The solution method is a non-linear multigrid iteration with symmetric point Gauss-Seidel as a relaxation method. Initial finest grid solutions have been obtained by full multigrid. Some grid adaptation has been applied for obtaining a sharp shock. An indication is given of the mathematical quality of 4 different boundary conditions for the outlet flow. The solutions of two transonic flows with shock are presented; a choked and a non-choked flow. Both flow solutions show a good shock capturing.

NM-R8602. P.W. Hemker & G.M. Johnson. *Multigrid approaches to the Euler equations.*
AMS 65N30; 13 pp.; **key words:** Euler equations, multigrid methods, Navier-Stokes equations.
Abstract: In this report we discuss different approaches to solve the Euler equations for compressible flow. The emphasis is on the multigrid acceleration of the solution process for finding

approximations to the steady state solution.

NM-R8603. W.H. Hundsdorfer & J.G. Verwer. *Linear stability of the hopscotch scheme.*
AMS 65M10; CR G.1.7; 9 pp.; **key words:** partial differential equations, convection-diffusion equation, hopscotch method, linear stability.
Abstract: This paper is devoted to the hopscotch scheme, which is a numerical integration technique for time-dependent partial differential equations. We examine its linear stability properties. A general theorem is presented which provides sufficient conditions for boundedness of the numerical solution during time stepping on a fixed space-time mesh. This theorem has applications in the field of parabolic problems. For the one-space dimensional convection-diffusion equation we present a detailed stability analysis of the odd-even scheme combined with central and one-sided finite differences. We compare stability based on the spectral condition with von Neumann stability.

NM-R8604. P.J. van der Houwen, B.P. Sommeijer, K. Strehmel & R. Weiner. *On the numerical integration of second-order initial value problems with a periodic forcing function.*
AMS 65L05; CR G.1.7; G.1.8; 21 pp.; **key words:** numerical analysis, ordinary differential equations, Runge-Kutta methods, predictor-corrector methods, periodic solutions.
Abstract: Runge-Kutta-Nyström type methods and special predictor-corrector methods are constructed for the accurate solution of second-order differential equations of which the solution is dominated by the forced oscillation originating from an external, periodic forcing term. For a family of second-order explicit and linearly implicit Runge-Kutta-Nyström methods it is shown that the forced oscillation is represented with zero phase lag. For a family of predictor-corrector methods of fourth-order, it is shown that both the phase lag order and the dissipation order of the forced oscillation can be made arbitrarily high. Numerical examples illustrate the effectiveness of our reduced phase lag methods.

MS-R8601. A.J. van Es & R. Helmers. *Elementary symmetric polynomials of increasing order.*
AMS 60F05; 10 pp.; **key words:** elementary symmetric polynomials of increasing order, normal and non-normal weak limits, Berry-Esseen bound, one-term Edgeworth expansion.
Abstract: The asymptotic behaviour of elementary symmetric polynomials $S_n^{(k)}$ of order k, based on n independent and identically distributed random variables $X_1,\ldots,X_n$, is investigated for the case that both k and n get large. If $k = o(n^{1/2})$, then the distribution function of a suitably normalised $S_n^{(k)}$ is shown to converge to a standard normal limit. The speed of this convergence to normality is of order $\mathcal{O}(kn^{-1/2})$, provided $k = \mathcal{O}(\log^{-1} n \log_2^{-1} n\, n^{1/2})$ and certain natural moment assumptions are imposed. This order bound is sharp, and cannot be inferred from one of the existing Berry-Esseen bounds for U-statistics. If $k \to \infty$ at the rate $n^{1/2}$ then a non-normal weak limit appears, provided the X_j's are positive and $S_n^{(k)}$ is standardised appropriately. On the other hand, if $k \to \infty$ at a rate faster than $n^{1/2}$ then it is shown that for positive X_j's there exists no linear norming which causes $S_n^{(k)}$ to converge weakly to a non-degenerate weak limit.

MS-N8601. S.G.A.J. Driessen. *The sieve method in multi-stage sampling.*
AMS 62E15; 70 pp.; **key words:** sieve sampling, two-stage sampling, Hoeffding inequalities.

Abstract: It is shown that the sieve method (a technique for probability proportional to size sampling) may, under certain conditions, be validly used for two- or three-stage sampling schemes, even though the statistical evaluation is based on true random sampling.

AM-R8601. O. Diekmann. *On the mathematical synthesis of physiological and behavioural mechanisms and population dynamics.*
AMS 92A15; 7 pp.; **key words:** physiologically structured population models, first order partial functional differential equations, dual semigroups.
Abstract: A concise description of a mathematical framework for the synthesis of physiological ecology and population dynamics is presented.

PM-R8601. B. Hoogenboom & T.H. Koornwinder. *Fonctions d'entrelacement sur les groupes de Lie compacts et polynômes orthogonaux de plusieurs variables.* (In French.)
AMS 17B20, 22E46, 33A65, 33A75, 43A75, 43A90; 16 pp.; **key words:** compact symmetric spaces, spherical functions, intertwining functions, orthogonal polynomials in several variables, root systems with two commuting involutions.
Abstract: This report, which is written in French, gives a survey of results by Vretare (1976) and Hoogenboom (1983), who showed that both spherical functions and intertwining functions on compact symmetric spaces can be written as orthogonal polynomials in several variables. The paper concludes with a sketchy discussion of orthogonal polynomials in several variables which can be associated with root systems. Some conjectures are posed for such classes of polynomials.

PM-R8603. J.C. van der Meer & R. Cushman. *Constrained normalization of Hamiltonian systems and perturbed Keplerian motion.*
AMS 58F05, 34C29, 70F15, 70F05, 70M05; 17 pp.; **key words:** Hamiltonian system, normal form for a Hamiltonian system, constrained Hamiltonian system, constrained normalization of Hamiltonian systems, Kepler system, perturbed Kepler system, lunar problem, main problem of artificial satellite theory.
Abstract: Consider a Hamiltonian system $(H,\mathbb{R}^{2n},\omega)$. Let M be a symplectic submanifold of $(\mathbb{R}^{2n},\omega)$. The system $(H,\mathbb{R}^{2n},\omega)$ constrained to M is $(H\,|\,M,M,\omega\,|\,M)$. In this paper we give an algorithm which normalizes the system on $\mathbb{R}^{2n}$ in such a way that restricted to M we have normalized the constrained system. This procedure is then applied to normalizing perturbed Kepler systems such as the lunar problem and the main problem of artificial satellite theory.

CWI Activities

Spring 1986

With each activity we mention its frequency and (between parentheses) a contact person at CWI. Sometimes some additional information is supplied, such as the location if the activity will not take place at CWI.

Symposium on Science of Industrial Organization. 11 June. Invited speakers: W.J. Deetman (Minister of Education and Science), F.J. Rauwenhoff (Philips BV), H.L. Beckers (Shell), P.C. Baayen (CWI), L.G.L.T. Meertens (CWI), H.J. van der Molen (ZWO). (H.M. Nieland)

Study group on Analysis on Lie groups. Jointly with University of Leiden. Biweekly. (T.H. Koornwinder)

Seminar on Lie groups. Jointly with Universities of Leiden, Utrecht, Groningen, Nijmegen, Delft and Amsterdam. 11, 12 April. Invited speakers: G. Heckman (Leiden, The Netherlands), L. Corwin (New Brunswick, USA), G. Olafsson (Göttingen, West Germany), M. Poel (Utrecht, The Netherlands), M. Flensted-Jensen (Copenhagen, Denmark), J.-Ph. Anker (Lausanne, Switzerland). (T.H. Koornwinder)

Seminar on Algebra and Geometry. Once a month. (A.M. Cohen)
Gordan's work on covariants of $SL_2,(\mathbb{C})$ (J. Brinkhuis). Mumford's construction of an algebraic surface resembling $\mathbb{P}^2$ (M. van der Put). The geometry on subgroups of order 3 in certain finite groups (F.G.J.M. Cuypers).

Cryptography working group. Monthly. (J.H. Evertse)

Colloquium 'STZ' on System Theory, Applied and Pure Mathematics. Twice a month. (J. de Vries)

Study group 'Biomathematics'. Lectures by visitors or members of the group. Jointly with University of Leiden. Topics for the next meetings are: *stochastic population dynamics, dynamics of structured populations.* Bimonthly. (J.

Grasman)

Study group on Nonlinear Analysis. Lectures by visitors or members of the group. Jointly with University of Leiden. The purpose is: to follow and investigate recent developments on qualitative analysis of nonlinear equations; to stimulate and support the research of the participants. Bimonthly. (O. Diekmann)

Workshop on Models for Physiologically Structured Populations. 31 August - 6 September at Texel. (O. Diekmann)

Progress meetings of the Applied Mathematics Department. New results and open problems on the research topics of the department: biomathematics, mathematical physics, asymptotic and applied analysis, image analysis. Weekly. (N.M. Temme)

Colloquium and symposium 'Theory and Practice of Image Processing'. The departments of Applied Mathematics and Mathematical Statistics are organizing a colloquium on image processing this autumn. Prior to that, on May 15th, a one day symposium will be held. The aim is to create a meeting place for applied scientists, and mathematicians interested in this topic, where in particular the latter can be confronted with the challenging new mathematical problems arising in this area. Invited speakers on May 15th:
J.J. Gerbrands (Technical University of Delft) *Introduction to digital picture processing,* B.M. ter Haar Romeny (Academic Hospital, Utrecht) *Picture processing in clinical practice: some examples with background and implementation,* A.C.M. Gieles (Philips, Eindhoven) *Industrial application of picture processing,* J.S. Ploem (Histo- and cytochemical Lab., Univ. of Leiden) *Automatic recognition of cells.* (R.D. Gill, H.J.A.M. Heijmans & J.B.T.M. Roerdink)

Study group on Statistical Image Analysis. Biweekly. (R.D. Gill)

Symposium on Semiparametric Models. Jointly with Dutch Statistical Society. 27 May. (R.D. Gill) Programme:
N.L. Hjort (Norwegian Computing Centre, Oslo) *Bootstrapping Cox's regression model,* A. van der Vaart (University of Leiden) *Estimating a real parameter in a class of semiparametric,* P. Green (University of Durham, U.K.) *Semiparametric regression by penalized likelihood, including model selection.*

Progress meetings of the Mathematical Statistics Department. New results in research and consultation. Monthly. (H.C.P. Berbee)

Colloquium 'Models for Discrete Variables'.
The emphasis will be on generalized linear models and variants thereof for discrete variables. After a short introduction to log-linear models, based on Fienberg's book 'The Analysis of Cross-Classified Categorical Data' and the standard package GLIM, we will turn to a number of modern developments in the data analysis of models for discrete variables, such as: Goodman's bilinear models, comparison with other models and models for continuous data, comparison of cross products in different tables, Lauritzen and Speed's graphical models, computation of asymptotic standard errors, also for more

complex sample setups, quasi-likelihood, formalisation of model-selection and testing with the same data. Biweekly. (A. Verbeek)

Study group on Combinatorial Optimization. Biweekly. (B.J. Lageweg)

System Theory Days. Irregular. (J.H. van Schuppen, J.M. Schumacher)

Study group on System Theory. Biweekly. (J.M. Schumacher)

Colloquium on Queueing Theory and Performance Evaluation. Irregular. (O.J. Boxma)

International Seminar on Teletraffic Analysis and Computer Performance Evaluation. 2-6 June. (O.J. Boxma)

Progress meetings on Numerical Mathematics. Weekly. (H.J.J. te Riele)

International Colloquium on Numerical Aspects of Vector and Parallel Processors. Monthly, every last Friday.

International Meeting: 30 May. Invited speakers:

J. Dongarra (Argonne National Lab., USA), O. Axelsson & V. Eijkhout (Catholic University Nijmegen, The Netherlands), D.P. O'Leary (University of Maryland, College Park, USA), F. Sullivan (National Bureau of Standards, Gaithersburg, USA). (H.J.J. te Riele)

Study group on Numerical Software for Vector Computers. Monthly. (H.J.J. te Riele)

Study group on Differential and Integral Equations. Lectures by visitors or group members. Irregular. (H.J.J. te Riele)

Study group on Graphics Standards. Monthly. (M. Bakker)

Study group on Dialogue Programming. (P.J.W. ten Hagen)

Concurrency Colloquium C^3/LPC. 15,16 May. Invited speakers:

P. Azema (LAAS, Toulouse, France), IJ.J. Aalbersberg (University of Leiden, The Netherlands), M. Diaz (LAAS, Toulouse, France), J. Vautherin (University of Paris-Sud, France), J.P. Banatre (IRISA, University of Rennes, France), K.R. Apt (LITP, University of Paris VII, France), L. Kott (IRISA/INRIA, Rennes, France), J.-J.Ch. Meyer (Free University, Amsterdam, The Netherlands), N. Halbwachs (IMAG, Grenoble, France), R. Koymans (Techn. University Eindhoven, The Netherlands). (J.W. de Bakker)

Post-academic Course on Computer Networks. 19,20,26,27 June. (S.J. Mullender)

Colloquium Knowledge Based Systems. Biweekly. (M.L. Kersten & P.J.F. Lucas)

Process Algebra Meeting. Weekly. (J.W. Klop)

Visitors to CWI from Abroad

P.P. Chen (Louisiana State University, USA) 24 January. N. Christopeit (University of Bonn, West Germany) 20 March. J.P. Coleman (University of Durham, England) 19-21 March. P.J. Courtois (Philips Research Laboratory, Brussels, Belgium) 7 January. P.M. van Dooren (Philips Research Laboratory, Brussels, Belgium) 7 January. A. Emerson (University of Texas at Austin, USA) 24 January. J. Grüger (University of Dortmund, West Germany) 17-21 February. Kai-Tai Fang (Academia Sinica, Beijing, People's Republic of China) 24-28 February. A.B. Kurzhanski (IIASA, Laxenburg, Austria) 6 February. I. Meijlijsson (University of Tel Aviv, Israel) 11 February. C.E. Molnar (Washington University, St. Louis, USA) 5 March. P. Nevai (Ohio State University, Columbus, USA) 31 January. D. Stanton (University of Minnesota, Minneapolis, USA) 27 January - 5 February. R. Syski (University of Maryland, USA) 17 March. J.V. Tucker (University of Leeds, England) 20-27 February. L. Winzgard (Royal Institute of Technology, Stockholm, Sweden) 14 January. Y. Yamaguchi (University of Tokyo, Japan) March-July. A. Yashin (IIASA, Laxenburg, Austria) 10-15 February.

Fundamental Science:

40 Years of Dutch Research in Mathematics and Computer Science

6 and 7 October 1986

This year the Foundation Mathematical Centre and its research institute, the Centre for Mathematics and Computer Science, celebrate their 40th anniversary. To mark this event the CWI is organizing a symposium.

PROGRAMME:

6 October

P.C. Baayen (Scientific Director of CWI)	*Opening address*
H.W. Lenstra, Jr. (University of Amsterdam)	*Codes from algebraic number fields*
P.M.B. Vitányi (CWI)	*Archirithmics or algotecture?*
A.O.H. Axelsson (Catholic University, Nijmegen)	*The numerical solution of partial differential equations*
A. Schrijver (Catholic University, Tilburg & CWI)	*Geometric methods in discrete optimization*

7 October

J. van Mill (Free University, Amsterdam & University of Amsterdam)	*Infinite-dimensional normed linear spaces and domain invariance*
J.W. Klop (CWI)	*Process algebra: a survey of recent results*
O. Diekmann (CWI & State University, Leiden)	*Dynamics in bio-mathematical perspective*
L.F.M. de Haan (Erasmus University, Rotterdam)	*Fighting the arch enemy with mathematics*

Further information is available from Prof. M. Hazewinkel, or Mrs. E. Both.

Order Form for CWI Publications

Sales Department
Centre for Mathematics and Computer Science
Kruislaan 413
1098 SJ Amsterdam
The Netherlands

☐ Please send the publications marked below on an exchange basis
☐ Please send the publications marked below with an invoice

	Publication code	Price per copy	Number of copies wanted
☐	CWI Tract 20 *)	Dfl. 17.60	.....
☐	CWI Tract 21 *)	13.90	.....
☐	CWI Syllabus 8 *)	23.90	.....
☐	CWI Syllabus 9 *)	28.80	.....
☐	CS-R8601	16.30	.....
☐	CS-R8602	3.90	.....
☐	CS-R8603	8.80	.....
☐	CS-R8604	5.10	.....
☐	CS-R8605	3.90	.....
☐	CS-R8606	3.90	.....
☐	CS-R8607	3.90	.....
☐	CS-R8608	11.40	.....
☐	CS-R8609	10.10	.....
☐	OS-R8601	3.90	.....

*) not available on exchange

	Publication code	Price per copy	Number of copies wanted
☐	OS-R8602	3.90	.....
☐	OS-R8603	3.90	.....
☐	OS-N8601	12.50	.....
☐	NM-R8524	3.90	.....
☐	NM-R8525	3.90	.....
☐	NM-R8601	3.90	.....
☐	NM-R8602	3.90	.....
☐	NM-R8603	3.90	.....
☐	NM-R8604	3.90	.....
☐	MS-R8601	3.90	.....
☐	MS-N8601	11.40	.....
☐	AM-R8601	3.90	.....
☐	PM-R8601	3.90	.....
☐	PM-R8603	3.90	.....

If you wish to order any of the above publications please tick the appropriate boxes and return the completed form to our Sales Department.
Don't forget to add your name and address!
Prices are given in Dutch guilders and are subject to change without notice. Foreign payments are subject to a surcharge per remittance to cover bank, postal and handling charges.

Name .....

Street .....

City .....

Country .....

Signature .....

Date .....

Contents