Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

MATHEMATICAL CENTRE TRACTS 70

W.P. DE ROEVER Jr.

RECURSIVE PROGRAM SCHEMES: SEMANTICS AND PROOF THEORY

MATHEMATISCH CENTRUM AMSTERDAM 1976

AMS(MOS) subject classification scheme (1970): primary 68A05, 02J10,
secondary 68J20, 68K10, 02J99.

ACM-Computing Reviews-categories: 5.24, 5.21.

ISBN 90 6196 127 0

CONTENTS

Contents v

Acknowledgements vii

Abstract ix

0. SURVEY

0.1. *Objectives* 1

0.2. *Structure of the paper* 3

0.3. *Related work* 5

1. A FRAMEWORK FOR PROGRAM CORRECTNESS

1.1. *Introduction* 9

1.2. *A framework for program correctness* 12

1.3. *The formulation of specific correctness properties of programs* 14

2. THE PROGRAM SCHEME LANGUAGE PL

2.1. *Definition of* PL 18

2.2. *The union theorem* 24

3. THE CORRECTNESS LANGUAGE ML

3.1. *Definition of* ML 32

3.2. *Validity of Scott's induction rule and the translation theorem* 37

3.3. *Rebuttal of Manna and Vuillemin on call-by-value* 43

4. AXIOMATIZATION OF ML

4.1. *Axiomatization of typed binary relations* 44

4.2. *Axiomatization of Boolean relation constants* 47

4.3. *Axiomatization of binary relations over cartesian products* 49

4.4. *Axiomatization of the "μ_i" operators* 53

5. APPLICATIONS

5.1. *An equivalence due to Morris* 58

5.2. *An equivalence involving nested while statements* 60

5.3. *Wright's regularization of linear procedures* 61

5.4. *Axiomatization of the natural numbers* 62

5.5. *The primitive recursion theorem* 65

6. AXIOMATIC LIST PROCESSING
6.1. *Lists, linear lists and ordered linear lists* 68
6.2. *Properties of head and tail* 75
6.3. *Correctness of the TOWERS OF HANOI*
6.3.a. *Informal part* 77
6.3.b. *An axiomatic correctness proof for the TOWERS OF HANOI* 80
7. ASSESSMENT 86

APPENDIX 1: SOME TOOLS FOR REASONING ABOUT COMPUTATION MODELS 89

APPENDIX 2: PROOFS OF MONOTONICITY, CONTINUITY AND SUBSTITUTIVITY 99

APPENDIX 3: PROOF OF TARSKI'S "UNPROVABLE ASSERTION" 106

REFERENCES 108

ACKNOWLEDGEMENTS

I wish to express my gratitude to the Mathematical Centre for providing the opportunity and the atmosphere which made the research described in this monograph possible.
I am deeply indebted to J.W. de Bakker for his continuous help, advice and criticism. Furthermore, I am grateful to David Park for his interest in my research, and for critically reading a preliminary edition.
The original incentive which led to this work arose out of the lectures of E.W. Dijkstra, C.A.R. Hoare and N. Wirth at the International Summer School on Program Structures and Fundamental Concepts of Programming, organized by F.L. Bauer, H.J. Helms and M. Paul in 1971.
I thank (in alphabetical order) P.C. Baayen, Peter van Emde Boas, Joost Engelfriet, Henk Goeman, Michael Gordon, Peter Hitchcock, Giles Kahn, Erik Krabbe, Robin Milner, Maurice Nivat, Reind van de Riet, Paul Vitányi and A. van Wijngaarden for their various suggestions, Th. Gunsing, and Tobias Baanders for the editing, Astrid Schuyt-Fasen for the Sisyphean labor of typing my arduous manuscript, and D. Zwarst, J. Suiker and J. Schipper for the printing.
The results of my investigations in the field of semantics of programming languages such as reported in this monograph would not have been possible without the pioneer work of J.W. de Bakker, Robert Milner, David Park and Dana Scott.

ABSTRACT

The language *PL* for first-order recursive program schemes with call-by-value as parameter mechanism is developed, using models for sequential and independent parallel computation. The language *MU* for binary relations over cartesian products which has least fixed point operators is formally defined, and the validity of the monotonicity, continuity, and substitutivity properties and Scott's induction rule is proved. After specifying an injection between *PL* and *MU*, it is proved that this injection induces a translation; hence the body replacement characterization of the semantics of recursive program schemes results in the same input-output behaviour as the least fixed point characterization. Then *MU* is axiomatized using a many-sorted generalization of Tarski's axioms for binary relations, Scott's induction rule and fixed point axiom, and new axioms to characterize projection functions, whence, by the translation result, a calculus for first-order recursive program schemes is obtained. Next we define an operator composing relations with predicates, the so-called "$\circ$"-operator, relate the properties of this operator axiomatically to the structure of the relations and predicates composed, and demonstrate the relevance of this operator to correctness proofs of programs in general and proofs involving the call-by-value parameter mechanism in particular. Axiomatic proofs are given of numerous properties of recursive program schems, some of which involve different modular decompositions of a program. Our calculus is then applied to the axiomatic characterization of the natural numbers, lists, linear lists and ordered linear lists, and used to prove many properties relating the head, tail and append list-manipulation functions to each other. Finally both an informal and an axiomatic correctness proof is given of the well-known recursive solution of the Towers of Hanoi problem.

Keywords: semantics of programming languages, recursion, call-by-value, least fixed point operators, axiomatization of polyadic binary relation algebras, Scott's induction rule, axiomatic program correctness, axiomatic list processing, predicate transformers.

0. SURVEY

0.1. *Objectives*

The objectives of the present investigation are to provide a self-contained description of:

1. A conceptually attractive framework for studying the foundations of program correctness.
2. An expedient axiomatization of the properties of first-order recursive programs with call-by-value as parameter mechanism.

Ad 1.

In reasoning about programs and their properties one is always confronted with the following two aspects:

1.1 A program serves to describe a class of computations on a possibly idealized computer. In consequence, most programmers *conceptualize* its execution. Whether this conceptualization figures on the very concrete level of bit manipulation or on the very abstract level of an ALGOL 68 machine, it always uses some model of computation as vehicle for the process of understanding a program. (However, the level on which this conceptualization takes place does matter when considering the ease with which one reasons about the outcome of a program: the less the amount of detail necessary to understand the operation of a program, the better the insight as to whether a program serves its purpose.)

1.2 If we abstract from this variety in understanding a program, we arrive at the relational structure which embodies the mathematical essence of that program: *its properties*.

This leads one to consider two notions of meaning:

operational and *mathematical* semantics.

How do these notions relate?

First, one has to choose a language, whose operational semantics is defined by some *interpreter*. Then, one decides which properties of the computations defined by this interpreter to investigate. Finally, one gives an *independent* mathematical characterization of these properties.

Our choice has been the following one:

a. To introduce an idealized interpreter for a language for *first-order recursive program schemes with call-by-value as parameter mechanism* (first-order recursive programs manipulate neither labels nor procedures as values).

b. To consider the *input-output behaviour* of programs as a property subject to investigation.

c. To use Scott's *least fixed point characterization* for the input-output behaviour of recursive procedures in the setting of binary relations and projection functions.

However, other choices are very well possible, e.g., BEKIC [1], BLIKLE [3], KAHN [32] and MILNER [48] incorporate also the intermediate stages of a computation into their mathematical semantics. *) This does not necessarily imply that then all properties of a computation have been taken into account (whence equivalence becomes equality). For instance, the two sequences $(A_1(A_2A_3))$ and $((A_1A_2)A_3)$ may be considered equivalent, as their execution amounts to executing the same elementary statements in the same order: first A_1, then A_2 and finally A_3, although these elementary statements are differently grouped together (cf. corollary 2.1).

Ad 2.

Once the appropriate mathematical semantics has been defined, a proper framework for *proving* properties of programs is obtained. As the proofs of these properties may be quite cumbersome and lengthy, one might wish to investigate the possibilities of computer-assisted proofs. cf. KING [34], MILNER [47] and WEYRAUCH and MILNER [63]. One then has to *calculate* the

*) A possible approach in this direction is suggested in appendix 1.

correctness of a program, whence a formal system is needed. Our system is an extension of the one given in DE BAKKER and DE ROEVER [11] in that we consider binary relations over *cartesian products* of domains, i.e., our domains are *structured*.
Other formal systems are considered in MILNER [47], which axiomatizes higher order recursive functionals with call-by-name as parameter mechanism, and SCOTT [57], which contains an axiomatization of the universal λ-calculus model called "logical space".

0.2. *Structure of the paper*

Chapter 1

Expression of properties of programs as properties of relations. Introduction to the correctness operator "$\circ$" between relational terms and predicates: ξ satisfies $X \circ p$ iff X terminates for input ξ with output η and output η satisfies p.

Chapter 2

Formal definition of PL, a language for first-order recursive program schemes with call-by-value as parameter mechanism, which allows for mutually dependent recursive declarations. Rigorous investigation of the input-output behaviour o of the program schemes of PL, consisting of proofs for (1) o is a *homomorphism* with respect to the algebraic structure of PL, (2) the main theorem, the *union* theorem, using *monotonicity*, *substitutivity* and transformation of a computation into a normal form, (3) the *modularity* property, using the least fixed point property; the modularity property relates to the modular design of program schemes and is applied to yield a two-line proof for the tree traversal result of section 4.5 of DE BAKKER and DE ROEVER [11].
This chapter is a generalization of chapter 3 of DE BAKKER and MEERTENS [12].

Chapter 3

Formal definition of MU, a language for binary relations over cartesian products, which has "simultaneous" least fixed point operators. Rigorous investigation of the mathematical semantics of MU, consisting of proofs

for (1) the monotonicity, substitutivity and *continuity* properties, (2) the union theorem (3) *validity of Scott's induction rule* (4) the *translation* theorem, which relates the input-output behaviour o of the recursive program schemes defined in chapter 2 to the mathematical interpretation of certain terms of MU, by stating that the body replacement characterization of the semantics of recursive program schemes results in the same input-output behaviour as the least fixed point characterization. Rebuttal of MANNA and VUILLEMIN [43] on the subject of call-by-value.

Chapter 4

Axiomatization of MU in four successive stages: (1) a many-sorted version of Tarski's axioms for binary relations; derivation of, amongst others, the fundamental lemma $\vdash R;S \cap T = R;(\breve{R};T \cap S) \cap T$, (2) axiomatization of boolean relation constants; derivation of the properties of the "$\circ$" operator, (3) axiomatization of projection functions; derivation of another characterization of the converse of a relation, involving the application of the conversion operator to projection functions, but not to the relation itself, (4) axiomatization of the least fixed point operators μ_i, resulting in a *calculus for first-order recursive program schemes with call-by-value as parameter mechanism*; derivation of the monotonicity, fixed point, least fixed point, *generalized iteration* and modularity properties; statement of a result on *functionality* of terms.

Chapter 5

Application of the calculus for recursive program schemes developed in chapter 4 to the formal derivation of (1) an equivalence due to MORRIS [50], (2) a property involving nested while statements, contained in section 5.1 of DE BAKKER and DE ROEVER [11], using *modular decomposition* and *simultaneous* μ-terms, (3) the *regularization* of *linear* procedures following WRIGHT [65]. An applied calculus for the natural numbers N featuring an improved axiom system for N and a deriviation of the characterizing property of the *equality* relation between natural numbers. Axiomatic proof of the primitive recursion theorem using structural induction.

Chapter 6

Formal list manipulation, applied calculi for *lists*, *linear lists* and

ordered linear lists. Linear lists as a special case of ordered linear lists. Proofs for (1) a characterization of termination of and associativity of the *concatenation* function with ordered linear lists as arguments, (2) many properties relating the *head*, *tail* and concatenation functions with ordered linear lists as arguments to each other, (3) both *informal* and *formal* versions of correctness of the *Towers of Hanoi* program.

Chapter 7

Assessment consisting of (1) a listing of the four main (technical) accomplishments of this paper, (2) some open problems, the main one being proof or disproof of Park's conjecture of the completeness of our axiomatization of polyadic binary relations, and (3) a brief discussion of the vast discrepancy between intuitive insight in the correctness of a program, and understanding of the artificial reasoning involved in the axiomatic correctness proof of such a program.

0.3. *Related work*

We discuss the *relational* approach to the correctness of recursive procedures, confining ourselves to those methods which are based upon the least fixed point characterization of the semantics of these procedures. Within the context of recursive function theory, this characterization was stated and proved originally by KLEENE [35], where it appears as the first recursion theorem.
The recursion induction rule for recursive procedures over arbitrary domains was formulated by McCARTHY [45]; for more references to the pre-1969 state of affairs in this branch of programming theory see DE BAKKER [8].
About 1969 the least fixed point characterization was formulated again independently by BEKIC [1], MORRIS [49], PARK [51], and SCOTT and DE BAKKER [59]. MORRIS stated the result within the λ-calculus, using Curry's paradoxical combinator Y. PARK formulated his theory (initially) within the second-order predicate calculus, and discovered the fixed point induction rule. SCOTT and DE BAKKER, using a relational framework, discovered that powerful induction rule which now carries Scott's name, and formulated the μ-calculus, a formal system based upon this rule.
BURSTALL formulated in [5] the rule of structural induction, whose main attraction (according to me) is its informal flavour (see for instance section

6.3.a of the present paper for an informal correctness proof of the Towers of Hanoi and section 3.4.1 of DE ROEVER [16], in which an informal correctness proof for a version of Floyd's iterative tree marking algorithm, cf. exercise 2.3.5.7 of KNUTH [36], is given).
MANNA began his impressive series of publications by expressing program correctness within the first-order predicate calculus [39,40].
In 1970 MILNER generalized the μ-calculus to a system dealing with polyadic functions [46], and PARK formulated his theory within the context of polyadic relations [52].
In 1971 DE BAKKER devoted his monograph [9] to an investigation of the μ-calculus, proving a completeness result for those recursive procedures which correspond to flow diagrams; MORRIS formulated the truncation induction rule [50].
DE BAKKER and DE ROEVER [11] crossbred the μ-calculus with Tarski's algebra of relations [61] to yield an axiomatic framework for proving equivalence, (partial) correctness and termination of first-order recursive program schemes with *one* variable. The present paper amplifies on the latter in that (1) the restriction to one variable is removed by considering arbitrary subdivisions of a state, and (2) the distinction on the one hand and the connection on the other between *operational* and *mathematical* semantics is clarified (MORRIS [49] also studies this topic). Subdivisions of a state are incorporated within the relational framework by considering relations over *cartesian products* of domains; these were introduced in MILNER [46] and PARK [52].

The connection between *induction rules* and *termination proofs* is described by HITCHCOCK and PARK in [28] and elaborated in Hitchcock's dissertation [27], which also contains a correctness proof of a *translation* of recursive programs into flowcharts with stacks and clarifies the notion of *representation* of (recursive) data structures.
Greatest fixed points, introduced by PARK in [51], are applied in MAZURKIEWICZ [44] to obtain a mathematical characterization of *divergent* computations and may lead to the axiomatization of Hitchcock and Park's results within an extension of our framework.
In DE ROEVER [17] relational calculi are developed for first-order recursive procedures each parameter of which may be either called-by-value or called-by-name; in DE ROEVER [66] it is proved that the input-output behaviour of first-order recursive procedures whose parameters are called-by-name

can be expressed within the ordinary framework of polyadic relations such as developed in this thesis, i.e., without any need for special points such as Scott's undefined element [55] or De Roever's basepoint [16].

In a different setting BLIKLE and MAZURKIEWICZ [4] also use an algebra of relations to investigate programs.

The equivalence between the method of *inductive assertions* and the least fixed point characterization is the subject of DE BAKKER and MEERTENS [12]. In general, the number of inductive assertions required to characterize a system of mutually dependent recursive procedures turns out to be infinite; however, in the regular case this number is finite, as is proved in FOKKINGA [22]. The *completeness* of the method of inductive assertions for general recursive procedures, as opposed to the merely regular ones, is treated in DE BAKKER and MEERTENS [13].

The relation between the least fixed point characterization and various *rules of computation* is studied by MANNA, CADIOU, NESS and VUILLEMIN in a number of papers: MANNA and CADIOU [41], MANNA, NESS and VUILLEMIN [42], MANNA and VUILLEMIN [43], CADIOU [7] and VUILLEMIN [62]. In section 3.3 we demonstrate that MANNA and VUILLEMIN are mistaken in their conclusion that call-by-value does not lead to the computation of least fixed points; DE ROEVER [15,16,17] and DE BAKKER [14] explain the reason why. In [62] VUILLEMIN, furthermore, compares the power of various induction rules.

The *distinction* between operational and mathematical semantics and the *need* for a mathematical semantics has been convincingly argued in SCOTT [55,56] and SCOTT and STRACHEY [60].

ROSEN [53] studies conditions under which *normal forms* for computations exist; implicitly, normal forms are used in appendix 1 to derive the "difficult" half of the union theorem.

The works of DIJKSTRA [18,19], HOARE [29,30] and WIRTH [64] relate to the present paper in that we provide a possible axiomatic basis for some techniques of structured programming; e.g., our correctness operator "$\circ$" is independently described in DIJKSTRA [20]*).

*) Some confusion is caused by the fact that the intended meaning of Dijkstra's wp-operator is not captured by his axioms, as can be shown by a counterexample. However, in the functional case such confusion does not exist; then our "$\circ$" operator and his wp-operator are the same.

Recently, BURSTALL and THATCHER [6], and GOGUEN and THATCHER [24] unified operational and mathematical semantics within the framework of category theory.

Scott's discovery of λ-calculus models [54] gave a powerful impetus to the field of formal semantics of programming languages; for a discussion of the literature related to and based upon this discovery, see SCOTT [58].

1. A FRAMEWORK FOR PROGRAM CORRECTNESS

1.1. *Introduction*

This report is devoted to a calculus for *recursive* programs written in a simple *first-order* programming language, i.e., a language in which neither procedures nor labels occur as values.
In order to express and prove properties of these programs such as equivalence, correctness and termination, one needs a more *comprehensive* language.
We shall abstract in that language from the usual meaning of programs (characterized by sequences of computations) by considering only the input-output relationships established by their execution.
Thus we are interested only in the binary relation *described* by a program, its *input-output behaviour*:

> the collection of all pairs of an initial state of the memory, for which this program terminates, and its corresponding final state of the memory.

EXAMPLE 1.1. Let D be a domain of initial states, intermediate states and final states.

a. The *undefined* statement L: goto L describes the *empty* relation Ω over D.

b. The *dummy* statement describes the *identity* relation E over D.

c. Define the composition $R_1;R_2$ of relations R_1 and R_2 by

$$R_1;R_2 = \{\langle x,y\rangle \mid \exists z\ \langle x,z\rangle \in R_1 \text{ and } \langle z,y\rangle \in R_2\}.$$

d. In order to express the input-output behaviour of the *conditional* if p then S_1 else S_2 one first has to transliterate p: Let D_1 be p^{-1} (true) and D_2 be p^{-1} (false) then the predicate p is uniquely determined by the

pair $\langle p,p' \rangle$ of disjoint subsets of the identity relation defined by: $\langle x,x \rangle \in p$ iff $x \in D_1$, and $\langle x,x \rangle \in p'$ iff $x \in D_2$. This way of looking at predicates is attributed to KARP [33]. If R_i is the input-output behaviour of S_i, $i = 1,2$, the relation described by the conditional above is $p;R_1 \cup p';R_2$.

e. Let $\pi_i : D^n \to D$ be the *projection function* of D^n on its i-th component, $i = 1,\ldots,n$, let the *converse* $\breve{R}$ of a relation R be defined by $\breve{R} = \{\langle x,y \rangle \mid \langle y,x \rangle \in R\}$ and let $R_1,\ldots,R_n$ be arbitrary relations over D. Consider

$$R_1;\breve{\pi}_1 \cap \ldots \cap R_n;\breve{\pi}_n \qquad (*).$$

This relation consists exactly of those pairs $\langle x,\langle y_1,\ldots,y_n \rangle \rangle$ such that $\langle x,y_i \rangle \in R_i$ for $i = 1,\ldots,n$. *Thus (*) terminates in x iff all its components R_i terminate in x.* Observe the analogy with the following: The evaluation of a list of parameters called-by-value terminates iff the evaluation of all its constituent actual parameters terminates. This suggests the possibility of describing the call-by-value parameter mechanism relationally, an idea which will be worked out in chapters 2 and 3.

Note that the input-output behaviour of recursive procedures has not been expressed above; this will be done by extending the language for binary relations with least fixed point operators, introduced by SCOTT and DE BAKKER in [59].

Once the input-output behaviour of a program has been described in relational terms, its correctness properties should be proved within a relational framework, e.g., properties of conditionals such as listed in McCARTHY [45] are proved as properties of $p;R_1 \cup p';R_2$.

Suitably rich programming- and relational languages, called *PL* and *MU*, and a precise formulation of the connections between the two by means of a *translation* will be specified in the next section and will justify that the axiomatization of *MU* results in a calculus for recursive programs.

The problem which correctness properties of programs can be formulated within *MU* will be discussed in section 1.3 and is closely related to the expressiveness of this language itself.

EXAMPLE 1.2. With D as above, let the *universal* relation U be defined by $U = D \times D$.

a. $R_1 \subseteq R_2$ and $R_2 \subseteq R_1$ together express *equality* of R_1 and R_2, and will be abbreviated by $R_1 = R_2$. If programs S_1 and S_2 have input-output behaviour R_1 and R_2, respectively, then S_1 and S_2 are called *equivalent* iff $R_1 = R_2$.

b. $E \subseteq R;\breve{R}$ and $E \subseteq R;U$ both express *totality* of R.

c. $R;R \subseteq R$ expresses *transitivity* of R.

d. $\breve{R};R \subseteq E$ expresses that R describes the graph of a function, i.e., *functionality* of R.

e. $R;\breve{R} \cap E = \{\langle x,y\rangle \mid \langle x,y\rangle \in E \text{ and } \langle x,y\rangle \in R;\breve{R}\}$

$= \{\langle x,y\rangle \mid x = y \text{ and } \exists z[\langle x,z\rangle \in R \text{ and } \langle z,y\rangle \in \breve{R}]\}$

$= \{\langle x,x\rangle \mid \exists z[\langle x,z\rangle \in R]\}$.

Hence $R;\breve{R} \cap E$ determines that subset of E which consists of all pairs $\langle x,x\rangle$ such that there exists some z with $\langle x,z\rangle \in R$: this indicates a correspondence with a predicate expressing the *domain of convergence* of R. Note that $R;\breve{R} \cap E = R;U \cap E$.

f. Let $p \subseteq E$. Then $p;U \cap U;p \subseteq p$ expresses that p contains at most *one* pair $\langle a,a\rangle$ only. This can be understood by deriving a contradiction from the assumption that both $\langle a,a\rangle \in p$ and $\langle b,b\rangle \in p$ for different a and b: for that implies that both $\langle a,b\rangle \in p;U$ and $\langle a,b\rangle \in U;p$, whence $\langle a,b\rangle \in p;U \cap U;p$ and therefore $\langle a,b\rangle \in p$ for different a and b, contradicting $p \subseteq E$. This requirement therefore states the correspondence of p with the characteristic function of an atom. *)

The axiomatization of MU proceeds in several stages.
First a sublanguage for binary relations over cartesian products is axiomatized by adding the following two axioms to typed versions of Tarski's axioms for binary relations (see [61]):

$C_1 : \pi_1;\breve{\pi}_1 \cap \ldots \cap \pi_n;\breve{\pi}_n = E$

$C_2 : R_1;S_1 \cap \ldots \cap R_n;S_n = (R_1;\breve{\pi}_1 \cap \ldots \cap R_n;\breve{\pi}_n) \;;\; (\pi_1;S_1 \cap \ldots \cap \pi_n;S_n)$

*) This observation is due to Peter VAN EMDE BOAS.

with π_i denoting the projection function of an n-fold cartesian product on its i-th component, i = 1,...,n, and E the identity relation over this product.
In the resulting formal system one can derive properties such as
$R = (R;\breve{R} \cap E);R$, obtained from example 1.2.e, and $R_1;\breve{\pi}_1 \cap R_2;\breve{\pi}_2 =$
$= (R_1;\breve{R}_1 \cap E);(R_2;\breve{R}_2 \cap E);(R_1;\breve{\pi}_1 \cap R_2;\breve{\pi}_2)$, obtained by combining examples 1.1.d and 1.2.e.
Secondly we axiomatize the least fixed point operators by (1) *Scott's induction rule* and (2) an axiom stating essentially the fixed point property of terms containing these operators. Both of these were formulated for the first time in [59].

The addition of further axioms to the system for *MU* yields various *applied* calculi, used, e.g., for the characterization of a number of special domains such as: finite domains with a fixed number of elements (axiomatized below), finite domains ([27]), natural numbers (chapter 5) and various kinds of lists (chapter 6).

EXAMPLE 1.3. Following example 1.2.f an *atom* a is characterized by

$$a \subseteq E \quad \text{and} \quad a;U \cap U;a \subseteq a.$$

Now D contains precisely n elements iff $E \subseteq D \times D$ is the disjoint union of n atoms $a_1,\ldots,a_n$, i.e., iff

(1) $\quad a_i;U \cap U;a_i \subseteq a_i, \quad i = 1,\ldots,n,$

(2) $\quad a_1 \cup a_2 \cup \ldots \cup a_n = E,$

(3) $\quad a_i \cap a_j = \Omega, \quad 1 \leq i < j \leq n,$

(4) $\quad U \subseteq U;a_i;U, \quad i = 1,\ldots,n.$

1.2. *A framework for program correctness*

In the previous section we discussed program correctness as follows: Starting with a scheme T, one considers its input-output behaviour and realizes that this is a relation, whence its properties should be expressed and deduced within a relational framework.
The present section presents an outline of the formalization of this point of view as contained in chapters 2 and 3.
In section 2.1 we define *PL*, a language for first-order recursive program

schemata.
First-order recursive program schemata are abstractions of certain classes of programs. The statements contained in these programs operate upon a state whose components are isolated by *projection functions*; a new state is obtained by (1) execution of elementary statements, the dummy statement or projection functions (2) calls of previously declared and possibly *recursive* procedures (3) execution of conditional statements (4) the parallel and independent execution of statements $S_1,\ldots,S_n$ in the *call-by-value product* $[S_1,\ldots,S_n]$, a new construct which unifies properties of the *assignment* statement and the call-by-value *parameter mechanism* and allows for the expression of both of these concepts, and (5) composition of statements by the ";" operator.
The definition of the *operational* semantics of these schemata involves an abstraction from the actual processes taking place within a computer by describing a *model* for the computations evoked by execution of a program. This leads to the characterization of the input-output behaviour or *operational interpretation* $o(T)$ of a program scheme T.

In section 3.1 we define MU, a language for binary relations over cartesian products which has least fixed point operators in order to characterize the input-output behaviour of recursive programs.
As the binary relations considered are subsets of the cartesian product of one domain or cartesian product of domains and another domain or cartesian product of domains, terms denoting these relations have to be *typed* for the definition of operations.
Elementary terms are individual relation constants, boolean relation constants, logical relation constants (for the empty, identity, and universal relations Ω, E, U and projection functions π_i) and relation variables.
Compound terms are constructed by means of the operators ";" (relational or Peirce product), "$\cup$" (union), "$\cap$" (intersection), "$\smile$" (converse) and "-" (complementation) and the least fixed point operators "μ_i", which bind for $i = 1,\ldots,n$, n different relation variables in n-tuples of terms provided *none of these variables occurs in any complemented subterm*, i.e., these terms are *syntactically continuous* in these variables.
Terms of MU are elementary or compound terms.
The well-formed formulae of MU are called *assertions* and are of the form $\Phi \vdash \Psi$, where Φ and Ψ are sets of inclusions between terms.
A *mathematical* interpretation *m* of MU is defined by:

(1) providing arbitrary (type-consistent) interpretations for the individual relation constants and relation variables, interpreting pairs $\langle p,p'\rangle$ of boolean relation constants as pairs $\langle m(p),m(p')\rangle$ of disjoint subsets of identity relations (cf. KARP [33]) and interpreting the logical relation constants as empty, identity and universal relations and projection functions,

(2) interpreting ";", "$\cup$", "$\cap$", "$\smile$", "$-$" as usual,

(3) interpreting each μ-term $\mu_i X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]$ as the i-th component of the least fixed point of the functional $\langle\sigma_1,\ldots,\sigma_n\rangle$ acting on n-tuples of relations.

An assertion $\Phi \vdash \Psi$ is *valid* provided for all m the following holds:
If the inclusions contained in Φ are satisfied by m, then the inclusions contained in Ψ are satisfied by m.

The precise correspondence between the operational semantics of *PL* and the mathematical semantics of *MU* is specified by the *translation* theorem of chapter 3:
After defining an injection tr between schemes and terms we prove that tr induces a *meaning preserving* mapping, i.e., a *translation*, provided the interpretation of the elementary statement constants and predicate symbols specified by o "agrees" with the interpretation of the individual relation constants and boolean relation constants specified by m. If these requirements are fulfilled the resulting correspondence between *PL* and *MU* is illustrated by

$$\begin{array}{ccc} T & \longmapsto & tr(T) \\ \vdots & & \vdots \\ \downarrow & & \downarrow \\ o(T) & = & m(tr(T)). \end{array}$$

Thus we conclude that, in order to prove properties of T, it suffices to prove properties of tr(T), whence axiomatization of *MU* leads to a calculus for first-order recursive program schemata.*)

1.3. *The formulation of specific correctness properties of programs*

*) By an abuse of language we suppress any mention of interpretations o and m satisfying $o(T) = m(tr(T))$.

Globally, in order to formulate the correctness of a program one has to state certain criteria which have to be satisfied in a specific environment. If these criteria depend on input-output behaviour only, one might hope to express them in the present formalism.
Sometimes this condition is not satisfied. Then these criteria concern intrinsic properties of the computation processes involved. As these are the very features we abstracted from, one cannot expect to formulate them in MU. For instance, when trying to formulate the correctness criteria for the TOWERS OF HANOI program discussed in chapter 6, it turns out that the requirement of moving one disc at a time cannot be expressed in our language. Accordingly we restrict ourselves to criteria which can be formulated in terms of input-output behaviour only.
These may be subdivided as follows:

(a) Equivalence of or inclusions between programs.

(b) Termination provided some input condition is satisfied.

(c) Correctness in the sense of HOARE [29]:
Given (partial) predicates p and q and a relation tr(T) describing (the input-output behaviour of) a program T *), this criterion is expressed by

$$\forall x,y[p(x) \wedge x\ tr(T)\ y \rightarrow q(y)]$$

and amounts to

if x satisfies p and T terminates for x with output y, then y satisfies q.*)

These criteria can all be formulated as inclusion between terms:
For (a) this is evident. As to (b): Let p be represented by $\langle p,p' \rangle$ satisfying $p \subseteq E$, $p' \subseteq E$ and $p \cap p' = \Omega$, and tr(T) describe program T, then

$$p \subseteq tr(T);\widetilde{tr(T)}$$

*) This corresponds with p{T}q in Hoare's notation and with {p}T{q} in Dijkstra's notation (cf. [19]).

or, equivalently,

$$p \subseteq tr(T);U$$

both express (b) (note that $p \subseteq R;\breve{R}$ is equivalent to $p \subseteq R;U$).
As to (c): Let p and q be represented by $\langle p,p' \rangle$ and $\langle q,q' \rangle$, then (c) is expressed by

$$p;tr(T) \subseteq tr(T);q.$$

It will be clear that the underlying supposition for the expression of these criteria is that we are able to express all the predicates involved indeed. This was not the case in the formalism described by SCOTT and DE BAKKER in [59] in which predicates were only expressible by primitive symbols, no operations on these symbols or other ways of constructing them being available.

Our main device for the construction of new predicates is the "$\circ$" operator defined by

$$\forall x[(X \circ p)(x) \longleftrightarrow \exists y[xXy \text{ and } p(y)]]. \quad *)$$

Accordingly, if X = tr(T) then $(tr(T) \circ p)(x)$ is <u>true</u> iff T produces for input x <u>some</u> output y which satisfies p.

In the present formalism $X \circ p$ can be expressed by

$$X \circ p = X;p;U \cap E.$$

In example 1.2 we showed that $X;\breve{X} \cap E = X;U \cap E = X \circ E$ describes the domain of convergence of X. Thus $X \circ E$ is the least predicate p satisfying X = p;X. In chapter 4 we obtain the following characterization of $X \circ p$:

$$X \circ p = \cap\{q \mid X;p \subseteq q;X\}.$$

---

*) Let X denote the function f, then $(X \circ p)(x) = p(f(x))$.

Therefore $X \circ p$ is the least predicate q, sometimes called the *weakest precondition*, satisfying $X;p \subseteq q;X$.
This observation raises the following question:
When does

$$X;p = X \circ p\ ;X \qquad \ldots (*)$$

hold?
We shall prove that (*) holds if $\breve{X};X \subseteq E$, i.e., X denotes the graph of a function.
Therefore the translation theorem implies that

> *one is allowed to retract predicates occurring in between statements on input conditions provided these statements describe functions, i.e., are deterministic.*

2. THE PROGRAM SCHEME LANGUAGE *PL*

2.1. *Definition of PL*

PL is a language for first-order recursive program schemes using call-by-value as parameter mechanism.

A *statement* scheme of *PL* is constructed from *basic symbols* using the sequencing, conditional, call-by-value product operations and recursion, and contains a *type* indication in the form of a superscript $\langle\eta,\xi\rangle$ in order to distinguish between input domain D_η and output domain D_ξ. The call-by-value product $[S_1,\ldots,S_n]$ expresses the independent parallel execution of statements $S_1,\ldots,S_n$, yielding for input x an output $\langle y_1,\ldots,y_n\rangle$ composed of the individual outputs of S_i, i = 1,...,n, and is used to describe the assignment statement and the call-by-value parameter mechanism as follows:

Assignment statement. An assignment statement $x_i := f(x_{i1},\ldots,x_{im})$ occurring in an environment $x_1,\ldots,x_n$ of variables is expressed by $[\pi_1,\ldots,\pi_{i-1},[\pi_{i1},\ldots,\pi_{im}];S,\pi_{i+1},\ldots,\pi_n]$, where S denotes f.

Call-by-value parameter mechanism. A procedure call $proc(f_1(x_1,\ldots,x_m),\ldots,f_n(x_1,\ldots,x_m))$ with parameters which are called-by-value is expressed by $[S_1,\ldots,S_n];P$, were S_k denotes f_k, for k = 1,...,n, and P denotes proc.

A *declaration* scheme of *PL* is a possibly empty collection of pairs $P_j \Leftarrow S_j$ which are indexed by some index set J; for each $j \in J$ such a pair contains a procedure symbol P_j and a statement scheme S_j of the same type as P_j.

A *program* scheme of *PL* is a pair consisting of a declaration and a statement scheme.

The well-formed formulae of *PL* are called *assertions*.

DEFINITION 2.1 (Syntax of *PL*) *)

Types. Let *G* be the collection $\{\alpha,\alpha_1,\ldots,\beta,\beta_1\ \ldots\}$ of possibly subscripted

*) Sections 2.1 and 2.2 follow closely section 3 of DE BAKKER and MEERTENS [12] which deals, however, with schemes operating upon one variable.

greek letters. A domain type is (1) an element of G, (2) any string $(\xi_1 \times \ldots \times \xi_n)$, where $\xi_1,\ldots,\xi_n$ are domain types. A type is a pair $<\eta,\xi>$ of domain types.

Basic symbols. The class of basic symbols is the union of the classes of relation and procedure symbols.

Relation symbols. The class of relation symbols R is the union of the classes of elementary statement symbols, predicate symbols, constant symbols and variable symbols.

a. The class of *elementary statement symbols* A contains for all types $<\eta,\xi>$ elements denoted by $A^{\eta,\xi}, A_1^{\eta,\xi},\ldots$.

b. The class of *predicate symbols* B contains for all η elements denoted by $p^{\eta,\eta}, p_1^{\eta,\eta},\ldots, q^{\eta,\eta}, q_1^{\eta,\eta},\ldots$.

c. The class of *constant symbols* C contains the symbols $\Omega^{\eta,\xi}$ for all types $<\eta,\xi>$, $E^{\eta,\eta}$ for all η and $\pi_1^{(\eta_1\times\ldots\times\eta_n),\eta_1},\ldots,\pi_n^{(\eta_1\times\ldots\times\eta_n),\eta_n}$ for all types $\eta_1,\ldots,\eta_n$.

d. The class of *variable symbols* X, introduced for purposes of substitution, contains for all types $<\eta,\xi>$ elements denoted by $X^{\eta,\xi}, X_1^{\eta,\xi},\ldots, Y^{\eta,\xi},\ldots, Z^{\eta,\xi},\ldots$.

Procedure symbols. The class of procedure symbols P contains for all types $<\eta,\xi>$ the symbols $P^{\eta,\xi}, P_1^{\eta,\xi},\ldots$.

Schemes.

a. *Statement schemes.* The class of statement schemes SS (arbitrary elements $S^{\eta,\xi}, S_1^{\eta,\xi},\ldots, V^{\eta,\xi},\ldots, W^{\eta,\xi},\ldots$) is the smallest collection satisfying:

1. $A \cup C \cup X \cup P \subseteq SS$. *)

2. If $S_1^{\eta,\theta}, S_2^{\theta,\xi} \in SS$ then $(S_1;S_2)^{\eta,\xi} \in SS$. **)

3. If $p^{\eta,\eta} \in B$ and $S_1^{\eta,\xi}, S_2^{\eta,\xi} \in SS$ then $(p \rightarrow S_1,S_2)^{\eta,\xi} \in SS$.

4. If $S_1^{\eta,\xi_1},\ldots,S_n^{\eta,\xi_n} \in SS$ then $[S_1,\ldots,S_n]^{\eta,(\xi_1\times\ldots\times\xi_n)} \in SS$.

---

*) Hence, a predicate symbol is *no* statement scheme.

**) These parentheses will be often deleted, using the following conventions: (1) the outer pair of parentheses is suppressed, (2) right preferent parenthesis deletion. E.g., $A_1;A_2$ stands for $(A_1;A_2)$ and $A_1;A_2;A_3$ stands for $A_1;(A_2;A_3)$ which stands in its turn for $(A_1;(A_2;A_3))$.

b. *Declaration schemes.* The class of declaration schemes $\mathcal{DS}$ (arbitrary elements $D, D_1, \ldots$) contains all sets $\{P_j^{\eta,\xi} \Leftarrow S_j^{\eta,\xi}\}_{j\in J}$ with J any index set, and, for each $j \in J$, $P_j \in \mathcal{P}$ and $S_j \in \mathcal{SS}$, such that no S_j contains any $X \in \mathcal{X}$.

c. *Program schemes.* The class of program schemes $\mathcal{PS}$ (arbitrary elements $T, T_1, \ldots$) contains all pairs $\langle D,S\rangle$ with $D \in \mathcal{DS}$ and $S \in \mathcal{SS}$. If $D = \emptyset$, $\langle D,S\rangle$ will be written as S.

Assertions. An *atomic formula* is of the form $T_1 \subseteq T_2$ with $T_1, T_2 \in \mathcal{PS}$. A *formula* is a set of atomic formulae $\{T_{1,1} \subseteq T_{2,1}\}_{1\in L}$ with L any index set. An assertion is of the form $\Phi \vdash \Psi$ with Ψ and Φ formulae.

Remarks. 1. $T_1 = T_2$ will be used as abbreviation for $T_1 \subseteq T_2$, $T_2 \subseteq T_1$.
2. Brackets around domain types, and type indications in general, will be omitted provided this causes no confusion.

DEFINITION 2.2. (Substitution)

Substitution operator. Let $S \in \mathcal{SS}$ and J be any nonempty index set such that, for $j \in J$, $\{R_j\}_{j\in J} \subseteq \mathcal{X} \cup \mathcal{P}$ denote a set of pairwise distinct variable or procedure symbols, and $\{V_j\}_{j\in J}$ denote a set of statement schemes such that R_j and V_j are of the same type, then $S[V_j/R_j]_{j\in J}$ is defined as follows:

a. If $S = R_j$ for some $j \in J$, then $S[V_j/R_j]_{j\in J} = V_j$.

b. If $S = R$ and, for all $j \in J$, $R \neq R_j$, then $S[V_j/R_j]_{j\in J} = R$.

c. If $S = S_1;S_2$, $(p \to S_1,S_2)$ or $[S_1,\ldots,S_n]$, then $S[V_j/R_j]_{j\in J} = S_1[V_j/R_j]_{j\in J};S_2[V_j/R_j]_{j\in J}$, $(p \to S_1[V_j/R_j]_{j\in J}, S_2[V_j/R_j]_{j\in J})$ or $[S_1[V_j/R_j]_{j\in J},\ldots,S_n[V_j/R_j]_{j\in J}]$, respectively.

$\tilde{S}$. $\tilde{S}$ is defined as $S[X_j/P_j]_{j\in J}$, where $\{P_j\}_{j\in J}$ contains all procedure symbols occurring in S.

Closed. If no $X \in \mathcal{X}$ occurs in $S \in \mathcal{SS}$, S is called closed.

Remarks. 1. From now on the substitution operator is used in the following forms: taking for J the index set of some declaration scheme, we (a) restrict ourselves to $R_j \in \mathcal{X}$, for $j \in J$, and (b) reserve the "~" operator for substitution with $R_j \in \mathcal{P}$ and $V_j = X_j$, for $j \in J$. Hence, explicit substitution in S is performed as in (a). This explains our notion of *closed* statement scheme.

2. The substitution operator can be generalized to formulae by writing $\{V_{1,1} \leq V_{2,1}\}_{1\in L}[V_j/X_j]_{j\in J}$ for $\{V_{1,1}[V_j/X_j]_{j\in J} \leq V_{2,1}[V_j/X_j]_{j\in J}\}_{1\in L}$, restricting ourselves as above.

3. If $J = \{1,\ldots,n\}$, $S[V_j/X_j]_{j\in J}$ is written as $S[V_j/X_j]_{j=1,\ldots,n}$ or $S(V_1,\ldots,V_n)$. If $J = \{1\}$ we also use $S[V/X]$.

4. $S[V_j/X_j]_{j\in J}$ is defined according to the complexity of S. Therefore properties such as the chain rule, $S[V_j/X_j]_{j\in J}[W_j/X_j]_{j\in J} = S[V_j[W_j/X_j]_{j\in J}/X_j]_{j\in J}$ can be proved by induction on the complexity of S.

An *interpretation* of the schemes of *PL* is determined by an initial interpretation o_0 which extends to an *operational* interpretation o of program schemes using models for sequential and independent parallel (to characterize the call-by-value product) computation.

DEFINITION 2.3. (Initial interpretation). An initial interpretation is a function o_0, such that

a. For each $\eta \in G$, $o_0(\eta)$ is a set denoted by D_η, and for each compound domain type $(\eta_1 \times \ldots \times \eta_n)$, $o_0(\eta_1 \times \ldots \times \eta_n)$ is the cartesian product of $o_0(\eta_1),\ldots,o_0(\eta_n)$.

b. For $A^{\eta,\xi} \in A$ and $X^{\eta,\xi} \in X$, $o_0(A^{\eta,\xi})$ and $o_0(X^{\eta,\xi})$ are subsets of $o_0(\eta) \times o_0(\xi)$.

c. For $p^{\eta,\eta} \in B$, $o_0(p^{\eta,\eta})$ is a partial predicate with arguments in $o_0(\eta)$.

d. For each projection function symbol $\pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i}$, $o_0(\pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i})$ is the projection function of $o_0(\eta_1) \times \ldots \times o_0(\eta_n)$ on its i-th constituent coordinate.

e. For all constants $\Omega^{\eta,\xi}$ and $E^{\eta,\eta}$, $o_0(\Omega^{\eta,\xi})$ and $o_0(E^{\eta,\eta})$ are the empty subset of $o_0(\eta) \times o_0(\xi)$ and the identity relation over $o_0(\eta)$, respectively.

The main problem in defining the semantics of a program scheme *operationally* is the fact that the resulting computation cannot be represented serially in any natural fashion: factors $S_1,\ldots,S_n$ of a product $[S_1,\ldots,S_n]$ first all have to be executed independent of another, before the computation can continue. Therefore the computations involved are described as a parallel and sequentially structured hierarchy of actions, a *computation model*.

At the first level of such a hierarchy any execution of a factor of a product is delegated to the second level; assuming this results in an output, this output becomes available as a component of the input for the still-to-be-executed part of the original scheme, if present. When all these components have been computed, the remaining computation at the first level, if present, is initiated on the resulting vector. The same holds, mutatis mutandis, for the relative dependency between computations on any n-th and n+1-st level of this hierarchy, if present.
Provided one has a finite computation, this delegating will end on a certain level. On that level the execution (of a factor of a product on a previous level) does not anymore involve the computation of any product on a state, whence this computation can be characterized by a sequence of, in our model, *atomic* actions of the following forms: (1) computation of a by-some-initial-interpretation-interpreted relation symbol (2) replacing a procedure symbol by its body, without changing the current state and (3) making a choice between two possible continuations of a computation, depending on whether a by-some-initial-interpretation-interpreted predicate symbol is <u>true</u> or <u>false</u> on the current state.

The extension of an initial interpretation o_0 to an *operational* interpretation o is defined in

DEFINITION 2.4. (Computation model) *)

Relative to an initial interpretation o_0 and a declaration scheme D, a computation model for xSy is pair $\langle x_1 S_1 x_2 \cdots x_n S_n x_{n+1}, CM\rangle$ with $S_i \in SS$ for $i = 1,\ldots,n$, $S_1 = S$, $x_1 = x$ and $x_{n+1} = y$, consisting of a computation sequence and a set of computation models relative to o_0 and D, called *associated* computation models, satisfying the following conditions:

a. If $S_i = R$ or $S_i = R;V$ with $R \in A \cup C \cup X$, $\langle x_i, x_{i+1}\rangle \in o_0(R)$ and $i = n$ or $S_{i+1} = V$, respectively.

b. If $S_i = P_j$ or $S_i = P_j;V$ and $P_j \Longleftarrow S_j \in D$, then $x_{i+1} = x_i$ and $S_{i+1} = S_j$ or $S_{i+1} = S_j;V$, respectively.

c. If $S_i = (V_1;V_2);V_3$ then CM contains an associated computation model for $x_i\ V_1;V_2\ x_{i+1}$ and $S_{i+1} = V_3$.

*) As described in appendix 1, this definition implies that the set of computation models can be structured as an *algebra*. This superposition of structure allows for simple proofs about certain transformations, by induction arguments on the complexity of these models, in case these transformations are *morphisms* w.r.t. this structure.

d. If $S_i = (p \rightarrow V_1, V_2)$ or $S_i = (p \rightarrow V_1, V_2);V_3$ and $o_0(p)(x_i)$ is either *true* or *false*, then $x_{i+1} = x_i$ and, if $o_0(p)(x_i) =$ *true* then $S_{i+1} = V_1$ or $S_{i+1} = V_1;V_3$, and, if $o_0(p)(x_i) =$ *false* then $S_{i+1} = V_2$ or $S_{i+1} = V_2;V_3$, respectively.

e. If $S_i = [V_1,\ldots,V_k]$ or $S_i = [V_1,\ldots,V_k];V$, $x_{i+1} = \langle y_1,\ldots,y_k\rangle$ such that CM contains associated computation models for $x_i V_l y_l$, for $l = 1,\ldots,k$, and $i = n$ or $S_{i+1} = V$, respectively.

Remark. A computation model represents the entire computation of program $\langle D,S\rangle$ on input x $(= x_1)$ resulting in output y $(= x_{n+1}$, for some $n)$. At each step of its constituent computation sequence, S_i is the statement which remains to be executed on the current state x_i. Clause a describes the execution of elementary statements, clause b reflects the *copy rule* for procedures, clause c describes preference in execution order, clause d describes the conditional and clause e describes the independent execution of statements, terminating iff all its constituent statements have terminated. The meaning of ";" is expressed by clause c and the second part of clauses a, b, d and e, and expresses continuation of a computation with appointed successor.

Suppose one defines a computation model as a set of computation sequences such that each "delegated" computation sequence occurs in this set. This leads to undesirable results, as demonstrated by the program scheme $T = \langle P \Leftarrow [P,P];\pi_1,P\rangle$. Clearly, T defines Ω. However the set $\{xPx[P,P];\pi_1\langle x,x\rangle\pi_1 x\}$ is a computation model for xTx in the sense of this definition (P. VAN EMDE BOAS).

DEFINITION 2.5.

Operational interpretation. Let $T = \langle D,S^{\eta,\xi}\rangle$ be a program scheme and o_0 be an initial interpretation. Then the operational interpretation of this scheme is the relation $o(T)$ defined as follows: for each $\langle x,y\rangle \in o_0(\eta) \times o_0(\xi)$, $\langle x,y\rangle \in o(T)$ iff there exists a computation model w.r.t. o_0 and D for xSy.

Validity.

a. $T_1 \subseteq T_2$ *satisfies* o iff $o(T_1) \subseteq o(T_2)$ holds. If $T_1 \subseteq T_2$ satisfies all o, it is called *valid.*

b. Φ satisfies o (is valid) iff all its inclusions satisfy o (are valid).

c. An assertion $\Psi \vdash \Psi$ such that, for all σ, if Φ satisfies σ, then Ψ satisfies σ, is called valid.

Remark. In case it is clear from the context that the same declaration scheme D is used with varying statement schemes S, $\sigma(\langle D,S\rangle)$ will be abreviated to $\sigma(S)$.

2.2. *The union theorem*

First we mention properties of the operational interpretation σ such as $\sigma(S_1;S_2) = \sigma(S_1);\sigma(S_2)$, $\sigma(p \to S_1,S_2) = m(p);\sigma(S_1) \cup m(p');\sigma(S_2)$, $\sigma([S_1,\ldots,S_n]) = \sigma(S_1);\widetilde{\sigma(\pi_1)} \cap \ldots \cap \sigma(S_n);\widetilde{\sigma(\pi_n)}$, the fixed point property $\sigma(P_j) = \sigma(S_j)$ and the monotonicity property. Then the union theorem is proved as a culmination of these results. Finally we establish the least fixed point property, which is a generalization of McCarthy's induction rule (cf. [45]), and prove a lemma legitimating the *modular* design of program schemes.

LEMMA 2.1.

a. *If* $S \in A \cup C \cup X$ *then* $\sigma_0(S) = \sigma(S)$.

b. $\sigma(S_1;S_2) = \sigma(S_1);\sigma(S_2)$.

c. $\sigma(p \to S_1,S_2) = m(p);\sigma(S_1) \cup m(p');\sigma(S_2)$, *with* $m(p)$ *and* $m(p')$ *defined as follows:* $\langle x,x\rangle \in m(p)$ *iff* $\sigma_0(p)(x) = \underline{\text{true}}$ *and* $\langle x,x\rangle \in m(p')$ *iff* $\sigma_0(p)(x) = \underline{\text{false}}$.

d. $\sigma([S_1,\ldots,S_n]) = \sigma(S_1);\widetilde{\sigma(\pi_1)} \cap \ldots \cap \sigma(S_n);\widetilde{\sigma(\pi_n)}$.

e. (*Fixed point property, fpp*) $\sigma(P_j) = \sigma(S_j)$, *for each* $j \in J$.

Proof. By induction on the complexity of the statement schemes concerned. □

COROLLARY 2.1. $\sigma((S_1;S_2);S_3) = \sigma(S_1;(S_2;S_3))$.

Remarks. 1. From the definitions and parts a, b, c and d of lemma 2.1 the validity of standard properties of program schemes, such as the validity of $\Omega \subseteq S$ and $E;S = S$ easily follows. These and similar properties will be used without explicit mentioning.

2. As execution of $[S_1,\ldots,S_n]$ corresponds to computation of a list of a actual parameters which are called-by-value, part d of lemma 2.1 implies the relational description of the call-by-value parameter mechanism.

LEMMA 2.2. (Monotonicity).

$$\{V_{1,j} \subseteq V_{2,j}\}_{j\in J} \vdash S[V_{1,j}/X_j]_{j\in J} \subseteq S[V_{2,j}/X_j]_{j\in J}.$$

Proof. By induction on the complexity of S.

a. $S = X_j$, then $o(S[V_{1,j}/X_j]_{j\in J}) = o(V_{1,j}) \subseteq o(V_{2,j}) = o(S[V_{2,j}/X_j]_{j\in J}$.

b. $S \in (R \cup P) - \{X_j\}_{j\in J}$, then $o(S[V_{1,j}/X_j]_{j\in J}) = o(S[V_{2,j}/X_j]_{j\in J}) = o(S)$.

c. $S = S_1;S_2$, then $o((S_1;S_2)[V_{1,j}/X_j]_{j\in J}) =$
$= o(S_1[V_{1,j}/X_j]_{j\in J};S_2[V_{1,j}/X_j]_{j\in J}) =$ (lemma 2.1)
$o(S_1[V_{1,j}/X_j]_{j\in J});o(S_2[V_{1,j}/X_j]_{j\in J}) \subseteq$ (induction hypothesis)
$o(S_1[V_{2,j}/X_j]_{j\in J});o(S_2[V_{2,j}/X_j]_{j\in J}) =$ (lemma 2.1)
$o(S_1[V_{2,j}/X_j]_{j\in J};S_2[V_{2,j}/X_j]_{j\in J}) = o((S_1;S_2)[V_{2,j}/X_j]_{j\in J})$.

d. $S = (p \rightarrow S_1,S_2)$ or $S = [S_1,\ldots,S_n]$, similar to c. □

COROLLARY 2.2. (Substitutivity rule).

$$\{V_{1,j} = V_{2,j}\}_{j\in J} \vdash S[V_{1,j}/X_j]_{j\in J} = S[V_{2,j}/X_j]_{j\in J}.$$

Next we state a technical result concerning substitution.

LEMMA 2.3.

a. *For closed* S, $\tilde{S}[P_j/X_j]_{j\in J} = S$.

b. *For arbitrary* S, $\{V_j \subseteq P_j\}_{j\in J} \vdash \widetilde{S[P_j/X_j]}_{j\in J}[V_j/X_j]_{j\in J} \subseteq S[V_j/X_j]_{j\in J}$.

c. *For arbitrary* S, $\widetilde{S[V_j/X_j]}_{j\in J} = \tilde{S}[\tilde{V}_j/X_j]_{j\in J}$.

Proof. Follows from the definitions, properties of substitution and monotonicity, by induction on the complexity of S. □

Informally, if a recursive procedure $P^{\eta,\xi}$ terminates for a given argument, this happens after a finite number of "inner calls" of this procedure. We may think of these calls as being nested (where a call on a deeper level is invoked by a call on a previous level). By the recursion depth of the original call we mean the depth of this nesting. At the innermost level, calls of $P^{\eta,\xi}$ are not executed again, whence they may be replaced by $\Omega^{\eta,\xi}$ without affecting the computation.

This process of replacement can be generalized to calls of *simultaneously* declared recursive procedures: Let $S^{\theta,\zeta}$ be a statement scheme. Then $S^{(n)}$

is obtained from S by *uniformly* replacing calls of $P_j^{\eta,\xi}$ at level n by $\Omega^{\eta,\xi}$ for $j \in J$ with $S^{(0)}$ defined as $\Omega^{\theta,\zeta}$. We may think of $o(S^{(n)})$ as restricting $o(S)$ to those arguments which during execution of S cause execution of calls of P_j with recursion depth less than n.
Thus we conclude that

$$x\ o(S)\ y \textit{ iff } \exists n[x\ o(S^{(n)})\ y].$$

THEOREM 2.1. (Union theorem). *Let* S *be a closed statement scheme. Then, for all operational interpretations* o,

$$o(S) = \bigcup_{n=0}^{\infty} o(S^{(n)}).$$

In order to prove the union theorem we need some auxiliary definitions characterizing (1) which occurrences of procedure symbols are *executed* in a computation model, (2) the relation between occurrences of the same procedure symbol in proceeding computations, (3) statement schemes obtained by successive uniform replacement of procedure calls by their bodies and (4) $S^{(n)}$.

DEFINITION 2.6.

Executable occurrence. A procedure symbol P_j occurs *executable* in a computation model CM if it occurs in some computation sequence $x_1\ S_1\ x_2\ \ldots\ x_n\ S_n\ x_{n+1}$ contained in CM, such that for some i, $1 \le i \le n$, $S_i = P_j$ or $S_i = P_j;S$.

To identify. Let CM be a computation model with constituent sequence $x_1\ S_1\ x_2\ \ldots\ x_n\ S_n\ x_{n+1}$. Consider an occurrence of P_j in some S, with S occurring in S_i. $1 \le i \le n$. This occurrence *directly identifies* the corresponding occurrence of P_j in S occurring in S_{i+1} or S_1' below, in each of the following cases:

(a) $S_i = R;S$ and $S_{i+1} = S$ with $R \in A \cup C \cup X$,
(b) $S_i = P_k;S$ and $S_{i+1} = S_k;S$, $k \in J$,
(c1) $S_i = (S);V_3$ and S occurs as first statement S_1' of the associated computation model for $x_i S x_{i+1}$, *)

*) Hence, for some V_1 and V_2, $S = V_1;V_2$.

(c2) $S_i = (V_1;V_2);S$ and $S_{i+1} = S$,

(d1) $S_i = (p \to S,V)$ or $S_i = (p \to V,S)$, and $S_{i+1} = S$,

(d2) $S_i = (p \to S,V_1);V_2$ or $S_i = (p \to V_1,S);V_2$, and $S_{i+1} = S;V_2$,

(d3) $S_i = (p \to V_1,V_2);S$ and $S_{i+1} = V_1;S$ or $S_{i+1} = V_2;S$,

(e1) $S_i = [V_1,\ldots,V_m]$ or $S_i = [V_1,\ldots,V_m];V$, and $S = V_k$ for some k, $1 \le k \le m$, CM contains an associated computation model CM' for $x_i S x_{i+1,k}$, and S occurs as first statement S'_1 of the constituent computation sequence of CM',

(e2) $S_i = [V_1,\ldots,V_m];S$ and $S_{i+1} = S$.

The relationship *to identify* is defined as the reflexive and transitive closure of the relationship to identify directly, defined above. *)

$S^{[n]}$. $S^{[0]} = S$, $S^{[k+1]} = \tilde{S}[S_j^{[k]}/X_j]_{j\in J}$ for k = 0,1,2,... .

$S^{(n)}$. $S^{(0)} = \Omega$, $S^{(k+1)} = \tilde{S}[S_j^{(k)}/X_j]_{j\in J}$ for k = 0,1,2,... .

The connections between $P^{(n+1)}$, $S^{(n)}$ and $S^{[n]}$ are established in

LEMMA 2.4. *Let* n *be a natural number. Then* $P_j^{(n+1)} = S_j^{(n)}$, $S^{(n+1)} = \widetilde{S^{[n]}}[\Omega_j/X_j]_{j\in J}$ *and* $S^{[k+1]} = S^{[k][1]}$.

Proof. We prove the second result only. Use induction on n.

1. k = 0. $S^{(1)} = \tilde{S}[\Omega_j/X_j]_{j\in J} = \widetilde{S^{[0]}}[\Omega_j/X_j]_{j\in J}$.

2. Assume the result for n = k. We have

$\widetilde{S^{[k+1]}}[\Omega_j/X_j]_{j\in J} = \widetilde{\tilde{S}[S_j^{[k]}/X_j]_{j\in J}}[\Omega_j/X_j]_{j\in J} =$ (lemma 2.3)

$\tilde{S}[\widetilde{S_j^{[k]}/X_j}]_{j\in J}[\Omega_j/X_j]_{j\in J} =$ (chain rule) $\tilde{S}[\widetilde{S_j^{[k]}[\Omega_j/X_j]_{j\in J}/X_j}]_{j\in J} =$

$=$ (induction hypothesis) $\tilde{S}[S_j^{(k+1)}/X_j]_{j\in J} = S^{(k+2)}$. □

In order to prove $o(S) \subseteq \bigcup_{n=0}^{\infty} o(S^{(n)})$ we shall transform a computation model for xSy for some n into a computation model for $xS^{(n)}y$.
Let S be closed and CM be a computation model for xSy with constituent sequence $x_1 S_1 x_2 \cdots x_n S_{n+1} x_{n+1}$. If *no* occurrences of P_j in S are executed to compute y, all occurrences of P_j identified by occurrences of P_j in S_1

*) Hence, if $S_i = P_j$ or $S_i = P_j;V$, the only or first occurrence, respectively, of P_j in S_i identifies *no* occurrence in S_{i+1}.

may be replaced by arbitrary statements of appropriate type for all $j \in J$ *without affecting the computation of* y:

LEMMA 2.5. *Let* CM *and* S *be as stated above. If* CM *contains no executable occurrences of* P_j, *the following holds: If statement schemes* V_j *are of the same type as* P_j *for all* $j \in J$, *there exists a computation model for* $x\tilde{S}[V_j/X_j]_{j\in J}y$.

Observe as a corollary that by choosing Ω for V_j one obtains a computation model for $xS^{(1)}y$. If P_j is executed in CM, there exists at least one occurrence of P_j identifying an earliest executable occurrence of P_j with respect to a certain order. CM can then be transformed into a computation model in which all occurrences of P_j in CM identified by such an occurrence are replaced by S_j, except the executable one, which is deleted together with the $x_i\ S_i$ part in which it is contained. The resulting model still computes the same output as CM, but contains at least one executable occurrence of some P_j *less* than CM, as at least one application of the copy-rule has been dealt with:

LEMMA 2.6. (VAN EMDE BOAS). *Let* CM *and* S *be as stated above. If for some* $j \in J$ *an occurrence of* P_j *in* S_1 *identifies an executable occurrence of* P_j, *there exists a computation model for* $xS^{[1]}y$ *which contains at least one executable occurrence of* P_j *less than* CM.

As $S^{[k][1]} = S^{[k+1]}$ by lemma 2.4, repeated application of lemma 2.6 leads finally to a computation model for $xS^{[n]}y$ in which *all* executable occurrences of P_j have been removed for all $j \in J$. Therefore lemma 2.5 applies, yielding a computation model for $x\widetilde{S^{[n]}}[\Omega_j/P_j]_{j\in J}y$ and hence, by lemma 2.4, for $xS^{(n+1)}y$:

LEMMA 2.7. *Let* CM *and* S *be as stated above. Then there exists for some* n *a computation model for* $xS^{(n)}y$.

The proofs of these three lemmas are contained in appendix 1.

Next we prove $\bigcup_{n=0}^{\infty} o(S^{(n)}) \subseteq o(S)$:
First we show that for each $j \in J$ and each k, $P_j^{(k)} \subseteq P_j$. Use induction on k.

1. $k = 0$. Clear.

2. Assume the result for k. $P_j^{(k+1)} =$ (lemma 2.4) $S_j^{(k)} = \tilde{S}_j[S_j^{(k-1)}/X_j]_{j\in J} =$
$= \tilde{S}_j[P_j^{(k)}/X_j]_{j\in J} \subseteq$ (induction hypothesis and lemma 2.2) $\tilde{S}_j[P_j/X_j]_{j\in J} =$
$= S_j =$ (lemma 2.1) P_j.

Next we show that $S^{(k)} \subseteq S : S^{(k)} = \tilde{S}[S_j^{(k-1)}/X_j]_{j\in J} = \tilde{S}[P_j^{(k)}/X_j]_{j\in J} \subseteq$
$\subseteq$ (lemma 2.2) $\tilde{S}[P_j/X_j]_{j\in J} =$ (lemma 2.3) S.
Thus $\bigcup_{n=0}^{\infty} S^{(n)} \subseteq S$ follows. □

Remark. In the sequel we abbreviate "For all o, $o(S) = \bigcup_{n=0}^{\infty} o(S^{(n)})$" to $S = \bigcup_{n=0}^{\infty} S^{(n)}$.

As a corollary to theorem 2.1 we immediately obtain the least fixed point property (called lfpp) of procedures:

COROLLARY 2.3. $\{\tilde{S}_j[V_j/X_j]_{j\in J} \subseteq V_j\}_{j\in J} \vdash \{P_j \subseteq V_j\}_{j\in J}$.

Proof. Use $P_j = \bigcup_{k=0}^{\infty} P_j^{(k)}$ and induction on k.
1. $P_j^{(0)} \subseteq V_j$ is clear.

2. Assume the result for k, then $P_j^{(k+1)} = S_j^{(k)} = \tilde{S}_j[P_j^{(k)}/X_j]_{j\in J} \subseteq$
$\subseteq$ (induction hypothesis) $\tilde{S}_j[V_j/X_j]_{j\in J} \subseteq V_j$. □

Remark. Combination of the fixed point and least fixed point properties yields, for all $i \in J$,

$$o(p_i) = \cap\{o(V_i) \mid o(\tilde{S}_k[V_j/X_j]_{j\in J}) \subseteq o(V_k), \text{ for all } k \in J\}.$$

This formula may be misunderstood on account of notational difficulties; however, by standard mathematical practice, it is an abbreviated linearized form of the much more unwieldy formula below:

$$o(P_i) = \bigcap_{\substack{\text{For all } \langle V_k\rangle_{k\in J} \text{ s.t. for each} \\ k \in J,\ o(\tilde{S}_k[V_j/X_j]_{j\in J}) \subseteq o(V_k)}} o(V_i).$$

This characterization of $o(P_i)$ is the key to the definition of the mathematical interpretation of μ-terms in the next section.

The following lemma legitimates the modular approach to programming and is a simple consequence of fpp (lemma 2.1.e), the substitutivity rule (corollary 2.2) and lfpp (corollary 2.3).

LEMMA 2.8. (Modularity lemma). *Let* J *and* K *be disjoint index sets, let* S_j *for all* $j \in J$ *be a closed statement scheme of which the procedure symbols are indexed by* K, *and let* S *and, for all* $\langle j,k \rangle \in J \times K$, $S_{j,k}$ *be closed statement schemes the procedure symbols of which are indexed by* J, *then*

$\langle \{P_j \Leftarrow \tilde{S}_j[S_{j,\ell}/X_\ell]_{\ell \in K}\}_{j \in J}, S \rangle =$

$= \langle \{P_{j,k} \Leftarrow \tilde{S}_{j,k}[\tilde{S}_j[P_{j,\ell}/X_\ell]_{\ell \in K}/X_j]_{j \in J}\}_{\langle j,k \rangle \in J \times K}, \tilde{S}[\tilde{S}_j[P_{j,\ell}/X_\ell]_{\ell \in K}/X_j]_{j \in J} \rangle$

is valid.

PROOF. The case $J = \{0\}$ and $K = \{1,2\}$ is considered to be representative. Then one has to prove $\langle P_0 \Leftarrow S_0(S_1(P_0),S_2(P_0)),P_0 \rangle =$
$= \langle P_1 \Leftarrow S_1(S_0(P_1,P_2)),P_2 \Leftarrow S_2(S_0(P_1,P_2)),S_0(P_1,P_2) \rangle$.
Consider the following declaration scheme:
$\{P_0 \Leftarrow S_0(S_1(P_0),S_2(P_0)),P_1 \Leftarrow S_1(S_0(P_1,P_2)),P_2 \Leftarrow S_2(S_0(P_1,P_2)),$
$P_3 \Leftarrow S_0(P_1,P_2),P_4 \Leftarrow S_1(P_0),P_5 \Leftarrow S_2(P_0)\}$.
With respect to this declaration scheme one proves $P_0 = P_3$ by applying lfpp on $\{P_0 \subseteq P_3, P_1 \subseteq P_4, P_2 \subseteq P_5, P_3 \subseteq P_0, P_4 \subseteq P_1, P_5 \subseteq P_2\}$.
E.g., $S_0(S_1(P_3),S_2(P_3)) \subseteq P_3$ is derived by $S_0(S_1(P_3),S_2(P_3)) =$
$=$ (fpp and substitution rule) $S_0(S_1(S_0(P_1,P_2)),S_2(S_0(P_1,P_2))) =$ (similarly) $S_0(P_1,P_2) =$ (fpp) P_3.
As $P_3 =$ (fpp) $S_0(P_1,P_2)$, the desired result is obtained by deleting declarations for uncalled procedures. □

Let us introduce the following convention. Calls of recursive procedures P, with P declared by $P \Leftarrow (p \rightarrow S;P,E)$, are written as $p * S$. Hence declarations of such P are omitted.

Next we demonstrate how to apply this lemma to obtain a simple proof for a tree-traversal result in DE BAKKER and DE ROEVER [11], section 4.5, and mention that the equivalences between certain procedures which do not have the form of while statements and *nested* while statements, contained in the same paper, section 5.1, can be proved as simple application of modularity, too. We quote, mutatis mutandis:

"The following problem, which at first sight appeared to be a problem of tree searching, was suggested to us ... by J.D. ALANEN.
Suppose one wishes to perform a certain action A in all nodes of all trees of a forest (in the sense of KNUTH [36], pp. 305-307). Let, for x any node, s(x) be interpreted as "has x a son?", and b(x) as "has x a brother?". Let S(x) be: "Visit the first son of x", B(x) be: "Visit the first brother of x", and F(x): "Visit the father of x". The problem posed to us can then be formulated as:

$\langle P \Leftarrow A;(s \rightarrow S;P;F,E);(b \rightarrow B;P,E),P\rangle =$

$= \langle P \Leftarrow A;(s \rightarrow S;P;\ b * (B;P);F,E),P;\ b * (B;P)\rangle$."

This equivalence can be obtained from lemma 2.8 by taking $P_1;P_2$ for S_0, $A;(s \rightarrow S;P_0;F,E)$ for S_1 and $(b \rightarrow B;P_0,E)$ for S_2.

3. THE CORRECTNESS LANGUAGE *MU*

3.1. *Definition of MU*

MU is a formal language for binary relations over cartesian products which has least fixed point operators in order to characterize the input-output behaviour of recursive program schemes. Its semantics will be described using elementary model-theoretic concepts. This involves a *mathematical*, as opposed to operational, characterization of its semantics, and results in a rigorous definition of its interpretations m, which will be axiomatized in the next chapter.

DEFINITION 3.1. (Syntax of *MU*)

Basic symbols. The class of basic symbols is the union of the classes of symbols for individual relation constants, boolean relation constants, logical relation constants and relation variables.

a. The class of *individual relation constant* symbols A contains for all types $\langle\eta,\xi\rangle$ elements denoted by $A^{\eta,\xi}, A_1^{\eta,\xi}, \ldots, A_i^{\eta,\xi}, \ldots$,

b. The class of *Boolean relation constant* symbols B contains for all η elements denoted by $p^{\eta,\eta}, p_1^{\eta,\eta}, \ldots$ and $p_1'^{\eta,\eta}, \ldots, q'^{\eta,\eta}, \ldots$.

c. The class of *logical relation constant* symbols C contains for all types concerned the symbols $\Omega^{\eta,\xi}, U^{\eta,\xi}, E^{\eta,\eta}, \pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i}$, $i = 1,\ldots,n$.

d. The class of *relation variable* symbols X contains for all types $\langle\eta,\xi\rangle$ elements denoted by $X^{\eta,\xi}, X_1^{\eta,\xi}, \ldots, Z^{\eta,\xi}, \ldots$.

Terms. The class of terms T, with arbitrary elements $\sigma^{\eta,\xi}, \sigma_1^{\eta,\xi}, \ldots, \tau^{\eta,\xi}, \ldots$ is the smallest collection satisfying:

a. $A \cup B \cup C \cup X \subseteq T$

b. If $\sigma^{\eta,\xi} \in T$, then $\overset{\smile}{\sigma}^{\xi,\eta}$ and $\bar{\sigma}^{\eta,\xi} \in T$.

c. If $\sigma^{\eta,\xi}, \tau^{\xi,\theta} \in T$ then $(\sigma;\tau)^{\eta,\theta} \in T$, and if $\sigma^{\eta,\xi}, \tau^{\eta,\xi} \in T$ then $(\sigma \cup \tau)^{\eta,\xi}, (\sigma \cap \tau)^{\eta,\xi} \in T$. *)

d. If $\sigma_1^{\eta_1,\xi_1}, \ldots, \sigma_n^{\eta_n,\xi_n} \in T$ and $X_1^{\eta_1,\xi_1}, \ldots, X_n^{\eta_n,\xi_n}$ denote pairwise distinct relation variables then $\mu_i X_1 \ldots X_n[\sigma_1, \ldots, \sigma_n]^{\eta_i,\xi_i} \in T$, for $i = 1, \ldots, n$.

Free variables. An occurrence of a relation variable X is free in σ iff this occurrence is not contained in a subterm of σ of the form $\mu_i \ldots X \ldots [\ldots]$.

Syntactically continuous. A term σ is syntactically continuous in X if no free occurrence of X in σ lies within any subterm $\bar{\tau}$.

Well-formed terms. A term σ is well-formed if, for all terms $\mu_i X_1 \ldots X_n[\sigma_1, \ldots, \sigma_n]$ occurring as subterms of σ, each σ_j is syntactically continuous in each X_k, $j,k = 1, \ldots, n$.

Assertions. An *atomic formula* is of the form $\sigma_1 \subseteq \sigma_2$ with $\sigma_1, \sigma_2 \in T$. A *formula* is a set of atomic formulae $\{\sigma_{1,1} \subseteq \sigma_{2,1}\}_{1 \in L}$ with L any index set. An assertion is of the form $\Phi \vdash \Psi$ with Φ and Ψ formulae.

Remarks. 1. $\sigma_1 = \sigma_2$ is an abbreviation for $\sigma_1 \subseteq \sigma_2$, $\sigma_2 \subseteq \sigma_1$ and $\mu_1 X_1[\sigma_1]$ is written as $\mu X[\sigma]$.

2. For *empty* Φ, $\Phi \vdash \Psi$ is written as $\vdash \Psi$.

DEFINITION 3.2. (Substitution)

Let $\sigma \in T$ and J be any index set, $\{X_j\}_{j \in J}$ denote a set of pairwise disjoint relation variables, and $\{\tau_j\}_{j \in J}$ denote a set of terms, such that, for $j \in J$, X_j and τ_j are of the same type, then $\sigma[\tau_j/X_j]_{j \in J}$ is defined as follows:

---

*) In accordance with the convention, that ";" binds stronger than "$\cap$" and "$\cap$" binds stronger than "$\cup$", the parentheses around $\sigma;\tau$, $\sigma \cap \tau$ and $\sigma \cup \tau$ will be often deleted. If the reader so wishes, he may stipulate any convention for parenthesis insertion in case the same binary operators occur adjacently. However, by associativity of these operators, the need for this is limited.

a. If $\sigma = X_j$ for some $j \in J$ then $\sigma[\tau_j/X_j]_{j\in J} = \tau_j$.

b. If $J = \emptyset$ or $\sigma \in A \cup B \cup C \cup (X - \{X_j\}_{j\in J})$ then $\sigma[\tau_j/X_j]_{j\in J} = \sigma$.

c. If $\sigma = \breve{\sigma}_1$ or $\bar{\sigma}_1$ then $\sigma[\tau_j/X_j]_{j\in J} = \breve{\sigma_1[\tau_j/X_j]_{j\in J}}$ or $\overline{\sigma_1[\tau_j/X_j]_{j\in J}}$, respectively.

d. If $\sigma = \sigma_1;\sigma_2$, $\sigma_1 \cup \sigma_2$ or $\sigma_1 \cap \sigma_2$ then $\sigma[\tau_j/X_j]_{j\in J} =$
$= \sigma_1[\tau_j/X_j]_{j\in J};\sigma_2[\tau_j/X_j]_{j\in J}$, $\sigma_1[\tau_j/X_j]_{j\in J} \cup \sigma_2[\tau_j/X_j]_{j\in J}$ or
$\sigma_1[\tau_j/X_j]_{j\in J} \cap \sigma_2[\tau_j/X_j]_{j\in J}$, respectively.

e. If $\sigma = \mu_i Z_1 \ldots Z_n[\sigma_1,\ldots,\sigma_n]$ then, for $i = 1,\ldots,n$,
$\sigma[\tau_j/X_j]_{j\in J} = \mu_i Y_1 \ldots Y_n[\sigma_1[Y_\ell/Z_\ell]_{\ell\in\{1,\ldots,n\}}[\tau_j/X_j]_{j\in J^*},\ldots$
$\ldots,\sigma_n[Y_\ell/Z_\ell]_{\ell\in\{1,\ldots,n\}}[\tau_j/X_j]_{j\in J^*}]$,
where $J^* = J - \{j \mid j\in J \text{ and } (\exists i[1\le i\le n \text{ and } Z_i = X_j])\}$, whence
$\{X_j\}_{j\in J^*} = \{X_j\}_{j\in J} - \{X_j \mid j\in J \text{ and } (\exists i[1\le i\le n \text{ and } Z_i = X_j])\}$,
and $Y_1,\ldots,Y_n$ are pairwise distinct relation variables such that, for $i = 1,\ldots,n$,

1. Y_i and Z_i are of the same type,
2. for $j \in J$, $Y_i \neq X_j$,
3. Y_i does not occur in any σ_k $(k = 1,\ldots,n)$, nor in any τ_j $(j \in J^*)$,
4. (to make the definition definite) the choice of Y_i is determined in advance, e.g., if for $i = 1,\ldots,k$ Y_i has been chosen, and $k < n$, then Y_{k+1} is taken to be the first variable in some fixed alphabetical arrangement of the variables such that it fulfills (1) to (3) above.

Remarks. 1. Thus $\sigma[\tau_j/X_j]_{j\in J}$ is obtained from σ by simultaneous substitution of τ_j for X_j, replacing bound variables whenever necessary in order to prevent binding of free occurrences of X_k in any substituted τ_j, and omitting substitution for bound variables (cf. HINDLEY, LERCHER and SELDIN [26], definition 1.4), for $j \in J$.

2. Definition 3.2 is extended to formulae by writing
$\{\sigma_{1,1} \subseteq \sigma_{2,1}\}_{1\in L}[\tau_j/X_j]_{j\in J}$ for $\{\sigma_{1,1}[\tau_j/X_j]_{j\in J} \subseteq \sigma_{2,1}[\tau_j/X_j]_{j\in J}\}_{1\in L}$.

3. Properties involving the substitution operator such as the chain rule can be proved by induction on the complexity of σ.

4. If $J = \{1,\ldots,n\}$, $\sigma[\tau_j/X_j]_{j\in J}$ is written as $\sigma[\tau_j/X_j]_{j=1,\ldots,n}$ or $\sigma(\tau_1,\ldots,\tau_n)$. If $J = \{1\}$ we also use $\sigma[\tau/X]$.

Compared with the everyday relational language the μ-terms $\mu_i X_1 \ldots X_n[\tau_1,\ldots,\tau_n]$ represent the only new feature of *MU* and its predecessors (cf. SCOTT and DE BAKKER [59], DE BAKKER [9] and DE BAKKER and DE ROEVER [11]). In order to explain their interpretation we first describe the concept of *continuity*.

A term τ induces upon interpretation of its constants a functional of tuples of relations to relations by selecting a fixed component of these tuples as interpretation for each free variable occurring in τ. Therefore interpretations of variables, called *variable valuations* v, have to be separated from interpretations of constants, called *initial interpretations* ι. Thus a pair $\langle\tau,\iota\rangle$ determines a functional; this functional is called *model function* and denoted by $\phi_\iota\langle\tau\rangle$.

Continuity of $\phi_\iota\langle\tau\rangle$ in $X_1,\ldots,X_n$ can now be defined as follows: Let τ be a term, $X_1,\ldots,X_n$ be variables, ι be an initial interpretation and v and, for each $j \in N$, v_j, be variable valuations satisfying, for $i = 1,\ldots,n$, $v(X_i) = \bigcup_{j=0}^{\infty} v_j(X_i)$, $v_j(X_i) \subseteq v_{j+1}(X_i)$ and $v(X) = v_j(X)$ for X different from X_i, for all j. Then $\phi_\iota\langle\tau\rangle$ is *continuous* in $X_1,\ldots,X_n$ iff $\phi_\iota\langle\tau\rangle(v) = \bigcup_{j=0}^{\infty} \phi_\iota\langle\tau\rangle(v_j)$ for all v and $\langle v_j\rangle_{j=0}^{\infty}$ considered above and all ι.

This concept derives its importance from the fact that only if $\phi_\iota\langle\tau_1\rangle,\ldots,\phi_\iota\langle\tau_n\rangle$ are continuous in $X_1,\ldots,X_n$, is Scott's induction rule for establishing properties of $\phi_\iota\langle\mu_i X_1 \ldots X_n[\tau_1,\ldots,\tau_n]\rangle(v)$ valid.

A syntactically sufficient, although not necessary condition for continuity of $\phi_\iota\langle\tau\rangle$ in $X_1,\ldots,X_n$, is the following one: free occurrences of $X_1,\ldots,X_n$ are not contained in complemented subterms of τ, i.e., τ is syntactically continuous in $X_1,\ldots,X_n$.

We therefore define the interpretation of $\mu_i X_1 \ldots X_n[\tau_1,\ldots,\tau_n]$ only if $\tau_1,\ldots,\tau_n$ are syntactically continuous in $X_1,\ldots,X_n$, and refer to HITCHCOCK and PARK [28] for more general considerations.

DEFINITION 3.3. (Semantics of *MU*)

Assignment of types. An *initial* assignment of types is a function t_0: $G \to \mathcal{D}$, where G is the collection of possibly subscripted greek letters and $\mathcal{D}$ is a class of non-empty domains. An assignment of types, relative to a given initial assignment of types t_0, is a function t defined by (1) for $\eta \in G$, $t(\eta) = t_0(\eta)$, and (2) for any compound (domain type, cf. definition 2.1) $(\eta_1 \times \ldots \times \eta_n)$, $t(\eta) = t(\eta_1) \times \ldots \times t(\eta_n)$. For $\eta \in G$, $t(\eta)$ will be referred

to as D_η, and for $\eta = (\eta_1 \times \ldots \times \eta_n)$ with $\eta_i \in G$, $i = 1,\ldots,n$, $t(\eta)$ will be referred to as $D_{\eta_1} \times \ldots \times D_{\eta_n}$.

Initial interpretation. Relative to a given assignment of types t, an initial interpretation is a function $\iota: A \cup B \cup C \rightharpoonup \bigcup_{D_1,D_2 \in \mathcal{D}} 2^{D_1 \times D_2}$ satisfying for all types involved.

a. $\iota(A^{\eta,\xi}) \subseteq t(\eta) \times t(\xi)$.

b. For $p^{\eta,\eta}, p'^{\eta,\eta} \in B$, $\iota(p^{\eta,\eta})$ and $\iota(p'^{\eta,\eta})$ are *disjoint* subsets of the identity relation over $t(\eta)$.

c. $\iota(\Omega^{\eta,\xi})$ is the empty subset of $t(\eta) \times t(\xi)$, $\iota(E^{\eta,\eta})$ is the identity relation over $t(\eta)$, $\iota(U^{\eta,\xi})$ is $t(\eta) \times t(\xi)$ itself and $\iota(\pi_i^{\eta_1 \times \ldots \times \eta_n, \eta_i})$ is the projection function of $t(\eta_1) \times \ldots \times t(\eta_n)$ on its i-th constituent component.

Variable valuation. Relative to a given assignment of types t, the class of variable valuations V contains the functions $v : X \to \bigcup_{D_1,D_2 \in \mathcal{D}} 2^{D_1 \times D_2}$, satisfying $v(X^{\eta,\xi}) \subseteq t(\eta) \times t(\xi)$ for all $X^{\eta,\xi} \in X$.

Model function. Relative to a given assignment of types t and an initial interpretation ι, the model function $\phi_\iota\langle\sigma^{\eta,\xi}\rangle : V \to 2^{D_\eta \times D_\xi}$ is defined as follows for *well-formed* terms $\sigma^{\eta,\xi}$:

a. $\phi_\iota\langle R\rangle(v) = \iota(R)$, $R \in A \cup B \cup C$.

b. $\phi_\iota\langle X\rangle(v) = v(X)$, $X \in X$.

c. $\phi_\iota\langle\sigma_1;\sigma_2\rangle(v) = \phi_\iota\langle\sigma_1\rangle(v);\phi_\iota\langle\sigma_2\rangle(v)$, $\phi_\iota\langle\sigma_1 \cup \sigma_2\rangle(v) = \phi_\iota\langle\sigma_1\rangle(v) \cup \phi_\iota\langle\sigma_2\rangle(v)$,
$\phi_\iota\langle\sigma_1 \cap \sigma_2\rangle(v) = \phi_\iota\langle\sigma_1\rangle(v) \cap \phi_\iota\langle\sigma_2\rangle(v)$, $\phi_\iota\langle\breve{\sigma}\rangle(v) = \widetilde{\phi_\iota\langle\sigma\rangle(v)}$,
$\phi_\iota\langle\bar{\sigma}\rangle(v) = \overline{\phi_\iota\langle\sigma\rangle(v)}$.

d. $\phi_\iota\langle\mu_i X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v) =$
$(\cap\{\langle v'(X_k)\rangle_{k=1}^n \mid \phi_\iota\langle\sigma_k\rangle(v') \subseteq v'(X_k),\ k=1,\ldots,n,\ \text{and } v'(X)=v(X) \text{ for } X \in X - \{X_1,\ldots,X_n\}\})_i$ *).

*) Cf. remark on page 29.

Interpretation of terms. An interpretation of terms is a triple $\langle t_0, \iota, v\rangle$ where each term σ is interpreted as $\phi_\iota\langle\sigma\rangle(v)$. This triple will often be referred to as m. Then $\phi_\iota\langle\sigma\rangle(v)$ is abbreviated by $m(\sigma)$.*)

Satisfaction. An atomic formula $\sigma_1 \subseteq \sigma_2$ satisfies an interpretation of terms m iff $m(\sigma_1) \subseteq m(\sigma_2)$. A formula $\{\sigma_{1,l} \subseteq \sigma_{2,l}\}_{l\in L}$ satisfies an interpretation of terms m iff $\sigma_{1,l} \subseteq \sigma_{2,l}$ satisfies m for all $l \in L$.

Validity. An assertion $\Phi \vdash \Psi$ is valid iff for every interpretation of terms m such that Φ satisfies m, Ψ satisfies m.

Remark. The definition of μ-terms can be straightforwardly generalized to the case where the μ-operators bind an infinite number of variables in an infinite sequence of terms.
The results of the next section are formulated and proved in such a way that they still apply if this generalization is effected.

3.2. *Validity of Scott's induction rule and the translation theorem.*

First the *union* theorem for $\mathcal{MU}$ is proved. This theorem is then applied to proving (1) validity of Scott's induction rule and (2) the translation theorem.
The reader who has followed the technical development of the previous chapter will observe a certain analogy between the results contained therein and the results of the present section. Notably, *monotonicity* is used in both chapters in proving *union* theorems. The *substitutivity* property, however, plays a more important role in this section and the *continuity* property is only defined in section 3.1. We state these properties in the following lemmas and refer to appendix 2 for proofs.

LEMMA 3.1. (Monotonicity).**) *Let* J *be any index set,* $X_j \in \mathcal{X}$ *for all* $j \in J$, $\sigma \in \mathcal{T}$ *be syntactically continuous in* X_j, $j \in J$, *and variable valuations* v_1 *and* v_2 *satisfy* (1) $v_1(X_j) \subseteq v_2(X_j)$ *for* $j \in J$ *and* (2) $v_1(X) = v_2(X)$ *for* $X \in \mathcal{X} - \{X_j\}_{j\in J}$. *Then the following holds:*

$$\phi\langle\sigma\rangle(v_1) \subseteq \phi\langle\sigma\rangle(v_2).$$

*) In the sequel m is often called the *mathematical* interpretation, as opposed to o, the operational interpretation.

**) Reference to a given initial interpretation is tacitly assumed. Accordingly, $\phi_\iota\langle\sigma\rangle$ will be written as $\phi\langle\sigma\rangle$.

LEMMA 3.2. (Continuity). *Let* J *be any index set,* $X_j \in X$ *for all* $j \in J$, $\sigma \in T$ *be syntactically continuous in* X_j, $j \in J$, *and* v *and* v_i ($i \in N$) *be variable valuations which satisfy, for* $i \in N$ *and* $j \in J$, (1) $v(X_j) = \bigcup_{i=0}^{\infty} v_i(X_j)$, (2) $v_i(X_j) \subseteq v_{i+1}(X_j)$ *and* (3) $v(X) = v_i(X)$ *for* $X \in X - \{X_j\}_{j\in J}$. *Then the following holds:*

$$\phi\langle\sigma\rangle(v) = \bigcup_{i=0}^{\infty} \phi\langle\sigma\rangle(v_i).$$

LEMMA 3.3. (Substitutivity). *Let* J *be any index set,* $\sigma \in T$, $X_j \in X$ *and* $\tau_j \in T$ *for* $j \in J$, *and variable valuations* v_1 *and* v_2 *satisfy* (1) $v_1(X_j) = \phi\langle\tau_j\rangle(v_2)$ *for* $j \in J$ *and* (2) $v_1(X) = v_2(X)$ *for* $X \in X - \{X_j\}_{j\in J}$. *Then the following holds:*

$$\phi\langle\sigma\rangle(v_1) = \phi\langle\sigma[\tau_j/X_j]_{j\in J}\rangle(v_2).$$

COROLLARY 3.1. (Change of bound variables). If $Y_1,\ldots,Y_n$ do not occur free in $\sigma_1,\ldots,\sigma_n$,

$$\phi\langle\mu_i X_1\ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v) =$$

$$= \phi\langle\mu_i Y_1\ldots Y_n[\sigma_1[Y_1/X_1]_{1=1,\ldots,n},\ldots,\sigma_n[Y_1/X_1]_{1=1,\ldots,n}]\rangle(v).$$

Proof. Follows by definition 3.2 from lemma 3.3. □

The union theorem for MU states that least fixed points $\langle\phi\langle\mu_1 X_1\ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v),\ldots,\phi\langle\mu_n X_1\ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v)\rangle$ of continuous functionals $\lambda v\langle\phi\langle\sigma_1\rangle(v),\ldots,\phi\langle\sigma_n\rangle(v)\rangle$ can be obtained as unions of sequences of finite approximations $\langle\phi\langle\sigma_1^i\rangle(v),\ldots,\phi\langle\sigma_n^i\rangle(v)\rangle$, $i = 0,1,\ldots$, with σ_k^i similarly defined as $S_k^{(i)}$, $k = 1,\ldots,n$, cf. definition 2.6.

DEFINITION 3.4. σ_k^i. Let $X_1^{\eta_1,\xi_1},\ldots,X_n^{\eta_n,\xi_n} \in X$ be the free variables in $\sigma_1^{\eta_1,\xi_1},\ldots,\sigma_n^{\eta_n,\xi_n} \in T$, then σ_k^i is defined by (1) $\sigma_k^0 = \Omega^{\eta_k,\xi_k}$ and (2) $\sigma_k^{i+1} = \sigma_k[\sigma_1^i/X_1]_{1=1,\ldots,n}$, for $k = 1,\ldots,n$.

THEOREM 3.1. (Union theorem for MU). *Let* $\sigma_1,\ldots,\sigma_n \in T$ *be syntactically continuous in* $X_1,\ldots,X_n \in X$. *Then the following holds for all variable valuations* v:

$$\phi\langle\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v) = \bigcup_{i=0}^{\infty} \phi\langle\sigma_k^i\rangle(v), \qquad k = 1,\ldots,n.$$

Proof. The proof splits into three parts. In the first part we prove $\phi\langle\sigma_k^i\rangle(v) \subseteq \phi\langle\sigma_k^{i+1}\rangle(v)$ for $i \in N$, in the second part $\phi\langle\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v) \subseteq \bigcup_{i=0}^{\infty} \phi\langle\sigma_k^i\rangle(v)$, and in the third part $\phi\langle\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v) \supseteq \bigcup_{i=0}^{\infty} \phi\langle\sigma_k^i\rangle(v)$ (the reverse inclusion).

Part 1. By induction on i. Obviously, $\phi\langle\sigma_k^0\rangle(v) \subset \phi\langle\sigma_k^1\rangle(v)$.
Assume by hypothesis $\phi\langle\sigma_k^{i-1}\rangle(v) \subseteq \phi\langle\sigma_k^i\rangle(v)$ and prove $\phi\langle\sigma_k^i\rangle(v) \subseteq \phi\langle\sigma_k^{i+1}\rangle(v)$, $k = 1,\ldots,n$. Define variable valuation v_1 by $v_1(X_k) = \phi\langle\sigma_k^i\rangle(v)$ for $k = 1,\ldots,n$ and $v_1(X) = v(X)$, otherwise.
Then $\phi\langle\sigma_k^{i+1}\rangle(v) = \phi\langle\sigma_k[\sigma_1^i/X_1]_{1=1,\ldots,n}\rangle(v) =$ (substitutivity) $\phi\langle\sigma_k\rangle(v_1)$.
Similarly, $\phi\langle\sigma_k^i\rangle(v) = \phi\langle\sigma_k\rangle(v_2)$ with v_2 defined by $v_2(X_k) = \phi\langle\sigma_k^{i-1}\rangle(v)$ for $k = 1,\ldots,n$ and $v_2(X) = v(X)$, otherwise.
As $\sigma_1,\ldots,\sigma_n$ are syntactically continuous, $\phi\langle\sigma_k^i\rangle(v) = \phi\langle\sigma_k\rangle(v_2) \subseteq$
$\subseteq$ (monotonicity and hypothesis) $\phi\langle\sigma_k\rangle(v_1) = \phi\langle\sigma_k^{i+1}\rangle(v)$, for $k = 1,\ldots,n$.

Part 2. $\subseteq$: Define variable valuations v' and, for $i \in N$, v_i, as follows: $v'(X_k) = \bigcup_{i=0}^{\infty} \phi\langle\sigma_k^i\rangle(v)$ for $k = 1,\ldots,n$, and $v'(X) = v(X)$, otherwise, and similarly $v_i(X_k) = \phi\langle\sigma_k^i\rangle(v)$ for $k = 1,\ldots,n$, and $v_i(X) = v(X)$, otherwise.
Then $v'(X_k) = \bigcup_{i=0}^{\infty} v_i(X_k)$ for $k = 1,\ldots,n$ and $v'(X) = v_i(X)$, otherwise. In part 1 we proved $\phi\langle\sigma_k^i\rangle(v) \subseteq \phi\langle\sigma_k^{i+1}\rangle(v)$, whence $v_i(X_k) \subseteq v_{i+1}(X_k)$. As σ_k is syntactically continuous in $X_1,\ldots,X_n$, the assumptions for continuity are fulfilled, whence $\phi\langle\sigma_k\rangle(v') = \bigcup_{i=0}^{\infty} \phi\langle\sigma_k\rangle(v_i) =$ (substitutivity) $\bigcup_{i=0}^{\infty} \phi\langle\sigma_k^{i+1}\rangle(v) = \bigcup_{i=0}^{\infty} \phi\langle\sigma_k^i\rangle(v) = v'(X_k)$. Thus v' satisfies $\phi\langle\sigma_k\rangle(v') \subseteq v'(X_k)$ for $k = 1,\ldots,n$ and $v'(X) = v(X)$, otherwise, whence

$$\left(\cap\{\langle v''(X_1)\rangle_{1=1}^{n} \mid \phi\langle\sigma_1\rangle(v'') \subseteq v''(X_1),\ 1=1,\ldots,n, \text{ and } v''(X)=v(X) \text{ for } X \in X - \{X_1,\ldots,X_n\}\}\right)_k \subseteq$$

$$\subseteq v'(X_k) = \bigcup_{i=0}^{\infty} \phi\langle\sigma_k^i\rangle(v).$$

Part 3. $\supseteq$: Let v' satisfy $\phi\langle\sigma_k\rangle(v') \subseteq v'(X_k)$ for $k = 1,\ldots,n$ and $v'(X) = v(X)$, otherwise.
Then we prove $\phi\langle\sigma_k^i\rangle(v') \subseteq v'(X_k)$ for $i \in N$ by induction on i. Obviously, $\phi\langle\sigma_k^0\rangle(v') \subseteq v'(X_k)$.

Assume by hypothesis $\phi<\sigma_k^i>(v') \subseteq v'(X_k)$ and prove $\phi<\sigma_k^{i+1}>(v') \subseteq v'(X_k)$, $k=1,\ldots,n$.
Define variable valuation v" by $v''(X_k) = \phi<\sigma_k^i>(v')$ for $k = 1,\ldots,n$ and $v''(X) = v'(X)$, otherwise.
Then $\phi<\sigma_k^{i+1}>(v') = \phi<\sigma_k[\sigma_1^i/X_1]_{1=1,\ldots,n}>(v') =$ (substitutivity) $\phi<\sigma_k>(v'') \subseteq$ $\subseteq$ (monotonicity, as $v''(X_k) = \phi<\sigma_k^i>(v') \subseteq v'(X_k)$ by hypothesis and $v''(X) =$ $= v'(X)$, otherwise) $\phi<\sigma_k>(v') \subseteq v'(X_k)$. Thus $\bigcup_{i=0}^{\infty} \phi<\sigma_k^i>(v) =$ ($X_1,\ldots,X_n$ not occurring in σ_k^i) $\bigcup_{i=0}^{\infty} \phi<\sigma_k^i>(v') \subseteq v'(X_k)$. As this holds for all v' considered above,

$$\bigcup_{i=0}^{\infty} \phi<\sigma_k^i>(v) \subseteq$$

$$(\cap\{<v'(X_1)>_{1=1}^n \mid \phi<\sigma_1>(v') \subseteq v'(X_1),\ 1=1,\ldots,n, \text{ and } v'(X)=v(X) \text{ for } X \in X - \{X_1,\ldots,X_n\}\})_k. \quad \square$$

Scott's induction rule is the main innovation of SCOTT and DE BAKKER [59], represents a general formulation for inductive arguments which does not assume any knowledge of the integers, and unifies methods for proof by induction such as recursion induction (McCARTHY [45]), structural induction (BURSTALL [5]) and computational induction (MANNA and VUILLEMIN [43]). Its formulation is given by

$$I: \quad \frac{\Phi \vdash \Psi[\Omega^{\eta_k,\xi_k}/X_k^{\eta_k,\xi_k}]_{k=1,\ldots,n} \qquad \Phi,\Psi \vdash \Psi[\sigma^{\eta_k,\xi_k}/X_k^{\eta_k,\xi_k}]_{k=1,\ldots,n}}{\Phi \vdash \Psi[\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]/X_k^{\eta_k,\xi_k}]_{k=1,\ldots,n}},$$

where Φ has no free occurrences of X_i, and Ψ only contains occurrences of X_i which are *not* contained in complemented subterms.

THEOREM 3.2. (Validity of Scott's induction rule, I). *If Φ and Ψ are formulae such that Φ does not contain any free occurrence of X_k, $k = 1,\ldots,n$, and all terms contained in Ψ are syntactically continuous in X_k, $k = 1,\ldots,n$, then I is valid.*

Proof. Let v be any variable valuation satisfying Φ, let v' be defined by $v'(X_k) = \phi<\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]>(v)$ for k = 1,...,n and v'(X) = v(X), otherwise, and let $\tau_{1,1} \subseteq \tau_{2,1}$ be any atomic formula contained in $\Psi = \{\tau_{1,1} \subseteq \tau_{2,1}\}_{1\in L}$.
We prove $\phi<\tau_{1,1}[\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]/X_k]_{k=1,\ldots,n}>(v) \subseteq$
$\subseteq \phi<\tau_{2,1}[\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]/X_k]_{k=1,\ldots,n}>(v)$.
By substitutivity, $\phi<\tau_{j,1}[\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]/X_k]_{k=1,\ldots,n}>(v) = \phi<\tau_{j,1}>(v')$, j = 1,2.

By the union theorem for *MU*, $v'(X_k) = \phi<\mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]>(v) =$
$= \bigcup_{i=0}^{\infty} \phi<\sigma_k^i>(v)$.
Let variable valuations v_i be defined by $v_i(X_k) = \phi<\sigma_k^i>(v)$ for k = 1,...,n, and $v_i(X) = v(X)$, otherwise, $i \in N$.
Then $\phi<\tau_{j,1}>(v') = \bigcup_{i=0}^{\infty} \phi<\tau_{j,1}>(v_i)$, j = 1,2, by continuity.
Therefore we must prove $\bigcup_{i=0}^{\infty} \phi<\tau_{1,1}>(v_i) \subseteq \bigcup_{i=0}^{\infty} \phi<\tau_{2,1}>(v_i)$ in order to obtain the desired result.
It is sufficient to prove $\phi<\tau_{1,1}>(v_i) \subseteq \phi<\tau_{2,1}>(v_i)$ by induction on i.
For i = 0, $\sigma_k^i = \Omega^{\eta_k,\xi_k}$, whence $\phi<\tau_{1,1}>(v_0) \subseteq \phi<\tau_{2,1}>(v_0)$ follows by substitutivity from validity of $\Phi \vdash \Psi[\Omega^{\eta_k,\xi_k}/X_k]_{k=1,\ldots,n}$, as (1) v and v_0 differ only in their assignments of relations to $X_1,\ldots,X_n$, (2) Φ satisfies v and $X_1,\ldots,X_n$ do not occur free within Φ, whence (3) Φ satisfies v_0.
Assume as hypothesis $\phi<\tau_{1,1}>(v_i) \subseteq \phi<\tau_{2,1}>(v_i)$, $1 \in L$, and prove $\phi<\tau_{1,1}>(v_{i+1}) \subseteq \phi<\tau_{2,1}>(v_{i+1})$, $1 \in L$.

Validity of $\Phi,\Psi \vdash \Psi[\sigma_k/X_k]_{k=1,\ldots,n}$ implies in particular that if Φ and Ψ satisfy v_i, $\Psi[\sigma_k/X_k]_{k=1,\ldots,n}$ satisfies v_i. Now Φ satisfies v_i by an argument similar to the one above for i = 0. By hypothesis, Ψ satisfies v_i.
Therefore we conclude that $\Psi[\sigma_k/X_k]_{k=1,\ldots,n}$ satisfies v_i and in particular $\phi<\tau_{1,1}[\sigma_k/X_k]_{k=1,\ldots,n}>(v_i) \subseteq \phi<\tau_{2,1}[\sigma_k/X_k]_{k=1,\ldots,n}>(v_i)$. By definitions of v_{i+1} and σ_k^i, $\phi<\sigma_k>(v_i) = \phi<X_k>(v_{i+1})$ follows by substitutivity, whence $\phi<\tau_{j,1}[\sigma_k/X_k]_{k=1,\ldots,n}>(v_i) = \phi<\tau_{j,1}>(v_{i+1})$, j = 1,2, by substitutivity, too.
Thus we conclude $\phi<\tau_{1,1}>(v_{i+1}) \subseteq \phi<\tau_{2,1}>(v_{i+1})$ for $1 \in L$. □

Finally we define the mapping tr : *PL* → *MU* (compare section 1.2) and prove the translation theorem.

DEFINITION 3.5. (tr). The mapping tr of program schemes of *PL* into terms of *MU* is defined as follows: consider a program scheme $T = \langle \{P_k \Leftarrow S_k\}_{k=1,\ldots,n}, S\rangle$, with all procedure symbols contained in S amongst those denoted by $P_1,\ldots,P_n$, then tr(T) is inductively defined by

a. $tr(R) = R$, for $R \in A \cup C \cup X$.

b. $tr(P_i) = \mu_i X_1 \ldots X_n[tr(\tilde{S}_1),\ldots,tr(\tilde{S}_n)]$, $i = 1,\ldots,n$.

c. $tr(S_1;S_2) = tr(S_1);tr(S_2)$, $tr(p \rightarrow S_1,S_2) = p;tr(S_1) \cup p';tr(S_2)$ and $tr([S_1,\ldots,S_n]^{\eta,\xi_1\times\ldots\times\xi_n}) = tr(S_1);\breve{\pi}_1 \cap \ldots \cap tr(S_n);\breve{\pi}_n$, with π_i of type $\langle \xi_1\times\ldots\times\xi_n, \xi_i\rangle$, $i = 1,\ldots,n$.

THEOREM 3.3. (Translation theorem). *Let o be an operational interpretation of PL, m be a mathematical interpretation of MU, and o and m satisfy* (1) *if* $R \in A \cup C \cup X$ *then* $o(R) = m(R)$ *and* (2) *if* $p \in B$ *then* $o(p)(x) = \underline{\text{true}}$ *iff* $\langle x,x\rangle \in m(p)$ *and* $o(p)(x) = \underline{\text{false}}$ *iff* $\langle x,x\rangle \in m(p')$. *Then* $o(T) = m(tr(T))$ *for all* $T \in PS$, *i.e.*, tr *is meaning preserving relative to o and m.*

Proof. By induction on the values under a certain measure of the complexities of the program schemes concerned and relative to some declaration scheme $D = \{P_j \Leftarrow S_j\}_{j=1,\ldots,n}$. Let $N \cup N \times \{0\}$ be well-ordered by $\prec$, with $\prec$ defined by:
$x \prec y$ iff (1) $x \in N$ and $y \in N$ and $x \leq y$, or (2) $x \in N$ and $y \in N \times \{0\}$, or (3) $x = \langle u,0\rangle$ and $y = \langle v,0\rangle$ and $u \leq v$.

Then this measure of complexity is the function $c : PS \rightarrow N \cup N \times \{0\}$, defined by

a. If $S \in A \cup C \cup X$ then $c(S) = 1$.

b. If $S \in P$, then $c(P) = \langle 0,0\rangle$.

c. If $S = S_1;S_2$, $S = (p \rightarrow S_1,S_2)$, let x or $\langle x,0\rangle$ be the maximum of $c(S_1)$ and $c(S_2)$ under the well-order $\prec$. Then $c(S_1;S_2)$ and $c(p \rightarrow S_1,S_2)$ are defined as $x+1$ or $\langle x+1,0\rangle$.

d. If $S = [S_1,\ldots,S_n]$ let x or $\langle x,0\rangle$ be the maximum of $c(S_1),\ldots,c(S_n)$ under the well-order $\prec$. Then $c(S_1,\ldots,S_n)$ is defined as $x+1$ or $\langle x+1,0\rangle$.

Thus $c(S_i) \overset{\prec}{\neq} c(S_1;S_2)$ and $c(S_i) \overset{\prec}{\neq} c(p \rightarrow S_1,S_2)$ for $i = 1,2$, $c(S_i) \overset{\prec}{\neq} c([S_1,\ldots,S_n])$, $i = 1,\ldots,n$, and $c(S_j^{(k)}) \overset{\prec}{\neq} c(P_j)$ for $k \in N$ and $j = 1,\ldots,n$.

Hence c provides the basis for the inductive proof of the translation theorem below:

a. If $S \in A \cup C \cup X$ then $o(S) = m(tr(S))$ is obvious.

b. If $S = S_1;S_2$ then $o(S_1;S_2) =$ (lemma 2.1) $o(S_1);o(S_2) =$ (induction hypothesis) $m(tr(S_1));m(tr(S_2)) = m(tr(S_1);tr(S_2)) = m(tr(S_1;S_2))$.

c. If $S = (p \rightarrow S_1,S_2)$ then $o(p \rightarrow S_1,S_2) =$ (lemma 2.1) $m(p);o(S_1) \cup$
$\cup\ m(p');o(S_2) =$ (induction hypothesis) $m(p);m(tr(S_1)) \cup$
$\cup\ m(p');m(tr(S_2)) = m(p;tr(S_1) \cup p';tr(S_2)) = m(tr(p \rightarrow S_1,S_2))$.

d. If $S = [S_1,\ldots,S_n]$ then $o(S) =$ (lemma 2.1) $o(S_1);\widetilde{o(\pi_1)} \cap \ldots \cap$
$\cap\ o(S_n);\widetilde{o(\pi_n)} =$ (induction hypothesis) $m(tr(S_1));\widetilde{m(\pi_1)} \cap \ldots \cap$
$\cap\ m(tr(S_n));\widetilde{m(\pi_n)} = m(tr(S_1);\breve{\pi}_1 \cap \ldots \cap tr(S_n);\breve{\pi}_n) = m(tr([S_1,\ldots,S_n]))$.

e. If $S = P_j$ then $o(P_j) =$ (union theorem for PL) $\bigcup_{i=0}^{\infty} o(P_j^{(i)}) =$ (lemma 2.4)
$\bigcup_{i=0}^{\infty} o(S_j^{(i)}) =$ (induction hypothesis) $\bigcup_{i=0}^{\infty} m(tr(S_j^{(i)}))$.
$tr(S_j^{(i)}) = tr(\tilde{S}_j)^i$ is easily proved by induction on i. Hence
$\bigcup_{i=0}^{\infty} m(tr(S_j^{(i)})) = \bigcup_{i=0}^{\infty} m(tr(\tilde{S}_j)^i) =$ (union theorem for MU)
$m(\mu_j X_1 \ldots X_n[tr(\tilde{S}_1),\ldots,tr(\tilde{S}_n)] = m(tr(P_j))$, $j = 1,\ldots,n$. □

COROLLARY 3.3. *The body replacement characterization of the semantics of the considered recursive program schemes results in the same input-output behaviour as the least fixed point characterization.*

3.3. *Rebuttal of Manna and Vuillemin on call-by-value*

In [43] MANNA and VUILLEMIN *discard* call-by-value as a computation rule, because, in their opinion, it does not lead to computation of the *least* fixed point. Clearly, our translation theorem *invalidates* their conclusion. As it happens, they work with a formal system in which least fixed points coincide with recursive solutions computed with *call-by-name* as rule of computation; this has been demonstrated in DE ROEVER [15]. Quite correctly they observe that *within such a system* call-by-value does not necessarily lead to computation of least fixed points. We may point out that observations like this one hardly justify discarding call-by-value as rule of computation in general.
For more remarks on the topic of parameter mechanisms (or rules of computation) and least fixed point operators we refer to DE ROEVER [17] and DE BAKKER [14].

4. AXIOMATIZATION OF MU

The axiomatization of MU proceeds in four successive stages:

1. In section 4.1 we develop the axiomatization of typed binary relations.
2. This axiomatization is extended in section 4.2 to boolean constants.
3. The axiomatization of projection functions in section 4.3 then results in the axiomatization of binary relations over cartesian products.
4. The additional axiomatization of μ-terms in section 4.4 completes the axiomatization of MU.

4.1. *Axiomatization of typed binary relations*

Consider the following sublanguage of MU, called MU_0:

> The elementary terms of MU_0 are restricted to the individual relation constants, relation variables and logical constants $\Omega^{\eta,\xi}$, $E^{\eta,\eta}$ and $U^{\eta,\xi}$ of MU, i.e., boolean constants and projection functions are excluded.
>
> The *compound* terms of MU_0 are those terms of MU which are constructed using these basic terms and the ";", "$\cup$", "$\cap$", "$\breve{}$" and "$-$" operators, i.e., the "μ_i" operators are excluded.
>
> The *assertions* of MU_0 are those assertions of MU whose atomic formulae are inclusions between terms of MU_0.

MU_0 is axiomatized by the following axioms and rules:

1. The typed versions of the axioms and rules of boolean algebra (including axioms for Ω and U).

2. The typed versions of Tarski's axioms for binary relations (cf. [61]):

$$T_1 : \vdash (X^{\eta,\theta};Y^{\theta,\zeta});Z^{\zeta,\xi} = X^{\eta,\theta};(Y^{\theta,\zeta};Z^{\zeta,\xi})$$

$$T_2 : \vdash \breve{X}^{\eta,\xi} = X^{\eta,\xi}$$

T_3 : $\vdash (X^{\eta,\theta};Y^{\theta,\xi})^{\smile} = \breve{Y}^{\theta,\xi};\breve{X}^{\eta,\theta}$

T_4 : $\vdash X^{\eta,\xi};E^{\xi,\xi} = X^{\eta,\xi}$

T_5 : $(X^{\eta,\theta};Y^{\theta,\xi}) \cap Z^{\eta,\xi} = \Omega^{\eta,\xi} \vdash (Y^{\theta,\xi};\breve{Z}^{\eta,\xi}) \cap \breve{X}^{\eta,\theta} = \Omega^{\theta,\eta}$

3. U : $\vdash U^{\eta,\xi} \subseteq U^{\eta,\theta};U^{\theta,\xi}$

4. The substitution rule:

if $\Phi \vdash \Psi$ *then* $\Phi[\tau_j/X_j]_{j\in J} \vdash \Psi[\tau_j/X_j]_{j\in J}$,

for all suitable Φ,Ψ,J, $\{\tau_j\}_{j\in J}$, $\{X_j\}_{j\in J}$ as defined in 3.1 and 3.2.

In the sequel we omit parentheses in our formulae, based on the associativity of binary operators and on the convention that ";" has priority over "$\cap$", which has in turn priority over "$\cup$".

LEMMA 4.1.

a. $X^{\eta,\xi} \subseteq Y^{\eta,\xi} \vdash \breve{X}^{\eta,\xi} \subseteq \breve{Y}^{\eta,\xi},X^{\eta,\xi};Z^{\xi,\theta} \subseteq Y^{\eta,\xi};Z^{\xi,\theta},Z^{\theta,\eta};X^{\eta,\xi} \subseteq Z^{\theta,\eta};Y^{\eta,\xi}$

b. $\vdash \Omega^{\eta,\xi};X^{\xi,\theta} = \Omega^{\eta,\theta},X^{\eta,\xi};\Omega^{\xi,\theta} = \Omega^{\eta,\theta}$

c. $\vdash E^{\eta,\eta};X^{\eta,\xi} = X^{\eta,\xi}$

d. $\vdash U^{\eta,\xi};U^{\xi,\theta} = U^{\eta,\theta}$

e. $\vdash \breve{\Omega}^{\eta,\xi} = \Omega^{\xi,\eta}$, $\breve{E}^{\eta,\eta} = E^{\eta,\eta}$, $\breve{U}^{\eta,\xi} = U^{\xi,\eta}$

f. $\vdash X^{\eta,\xi};(Y^{\xi,\theta} \cup Z^{\xi,\theta}) = X^{\eta,\xi};Y^{\xi,\theta} \cup X^{\eta,\xi};Z^{\xi,\theta},(X^{\xi,\theta} \cup Y^{\xi,\theta});Z^{\theta,\eta} =$
$= X^{\xi,\theta};Z^{\theta,\eta} \cup Y^{\xi,\theta};Z^{\theta,\eta}$

g. $\vdash (X^{\eta,\xi} \cup Y^{\eta,\xi})^{\smile} = \breve{X}^{\eta,\xi} \cup \breve{Y}^{\eta,\xi},(X^{\eta,\xi} \cap Y^{\eta,\xi})^{\smile} = \breve{X}^{\eta,\xi} \cap \breve{Y}^{\eta,\xi},\breve{\breve{X}}^{\eta,\xi} = \breve{\breve{X}}^{\eta,\xi}$.

Proof. Except for the proof of lemma 4.1.d which is obtained using U and a law of boolean algebra, the proofs for the typed case are similar to the proofs for the untyped case as contained in TARSKI [61]. □

Lemma 4.1.a expresses monotonicity of "$\smile$" and ";". Together with the obvious monotonicity of "$\cup$" and "$\cap$", this will be used in lemma 4.9 to establish monotonicity of syntactically continuous terms in general.

Remarks. 1. Henceforward the laws of boolean algebra are used without explicit reference.

2. *Type indications are omitted provided no confusion arises.*

LEMMA 4.2. $\vdash X;Y \cap Z = X;(\breve{X};Z \cap Y) \cap Z$ [*)].

Proof. $X;Y \cap Z = X;(U \cap Y) \cap Z = X;((\overline{\breve{X};Z} \cup \breve{X};Z) \cap Y) \cap Z =$
$= \{X;(\overline{\breve{X};Z} \cap Y) \cap Z\} \cup \{X;(\breve{X};Z \cap Y) \cap Z\}$. Also $\breve{Z};X \cap \overline{\breve{Z};X} = \Omega$, whence by T_5, $X;(\overline{\breve{Z};X})\breve{} \cap \breve{Z} = \Omega$, thus by T_2, T_3 and lemma 4.1, $(X;\overline{\breve{X};Z}) \cap Z = \Omega$.
Therefore, $X;(\overline{\breve{X};Z} \cap Y) \cap Z = \Omega$, whence $X;Y \cap Z = X;(\breve{X};Z \cap Y) \cap Z$ follows. □

The first applications of lemma 4.2 follow in the proof of lemma 4.3, in which a number of useful properties of relations and functions are formally derived. Remember that $X \circ E$ has been defined as $X;U \cap E$ (section 1.3). *By convention the "$\circ$" operator has a higher priority than the ";" operator.*

LEMMA 4.3.

a. $\breve{X};X \subseteq E \vdash X;(Y \cap Z) = X;Y \cap X;Z$

b. $X \subseteq E \vdash X = \breve{X}$

c. $\vdash X = X \circ E;X,\ X = X;\breve{X} \circ E,\ X \circ E = X;\breve{X} \cap E,\ X;U = X \circ E;U$

d. $X \subseteq Y,\ \breve{Y};Y \subseteq E \vdash X \circ E;\ Y = X$

e. $\vdash \bigcap_{i=1}^{n} X_i;Y_i = X_1 \circ E;\ \ldots;\ X_n \circ E;\ (\bigcap_{i=1}^{n} X_i;Y_i);\ \breve{Y}_1 \circ E;\ \ldots;\ \breve{Y}_n \circ E.$

Proof. a. $\subseteq$. Clear.

$\supseteq$. $X;Y \cap X;Z =$ (lemma 4.2) $X;(\breve{X};X;Z \cap Y) \cap X;Z \subseteq$ (assumption) $X;(Y \cap Z)$.

b. $X = X \cap E =$ (lemma 4.2) $X;(\breve{X};E \cap E) \cap E \subseteq X;\breve{X} \subseteq \breve{X}$. Thus $X \subseteq \breve{X}$, whence $\breve{X} \subseteq \breve{\breve{X}} = X$.

c. $X = X \circ E;X$: $\breve{X} = \breve{X} \cap U =$ (lemma 4.2) $\breve{X};(X;U \cap E) \cap U = \breve{X};(X;U \cap E)$.
Thus, by T_3, $X = (X;U \cap E)\breve{};X =$ (part b) $X \circ E;X$.
$X \circ E = X;\breve{X} \cap E$: Direct from lemma 4.2.
$X;U = X \circ E;U$: $X;U =$ (from above) $(X;U;U \cap E);X;U \subseteq$ (lemma 4.1) $X \circ E;U \subseteq$
$\subseteq X;U;U = X;U$.

d. $\supseteq$. $X \subseteq Y$ implies $\breve{Y};X \subseteq \breve{Y};Y \subseteq$ (assumption) E, whence $X;\breve{X};Y \subseteq$ (part b and T_3) X and $(X;\breve{X} \cap E);Y \subseteq X;\breve{X};Y \subseteq X$.
$\subseteq$. Immediate from part c.

e. We prove $X;Y \cap Z = X \circ E;(X;Y \cap Z)$ only. $\supseteq$. Obvious.
$\subseteq$. $X;Y \cap Z =$ (part c) $X \circ E;X;Y \cap Z =$ (part b and lemma 4.2)
$X \circ E;(X \circ E;Z \cap X;Y) \cap Z \subseteq X \circ E;(X;Y \cap Z)$. □

*) This assertion can also be proved without making use of the "$\overline{\ }$" operation, by using projection functions; an example of that style of proof is given in appendix 3.

The given axiomatization of MU_0 is *incomplete*. This can be understood as follows:
Consider the assertion

$$\vdash X_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2 \subseteq$$
$$\subseteq X_1;(\breve{X}_1;Y_1 \cap X_2;\breve{Y}_2 \cap (\breve{X}_1;Z_1 \cap X_2;\breve{Z}_2);(\breve{Z}_1;Y_1 \cap Z_2;\breve{Y}_2));Y_2.$$

This assertion holds in every proper relation algebra, and is therefore valid in MU_0. However, in LYNDON [37] a finite relation algebra (i.e., a finite algebra satisfying the untyped versions of the axioms and rules for MU_0) is exhibited which is not isomorphic to any proper relation algebra and in which the assertion stated above does not hold.*) Therefore this assertion does not follow from our axiomatization of MU_0,**) whence the result. We emphasize that this observation does not contradict the result of HITCHCOCK and PARK [28] that every valid assertion of MU_0 can be effectively translated into a valid sentence of first-order predicate calculus, thus implying, by the existence of a semi-decision procedure for first-order predicate calculus, that there exists such a procedure for MU_0.
We refer to LYNDON [38] for a very complicated *complete* axiomatization of proper relation algebras.

4.2. *Axiomatization of boolean relation constants*

Partial predicates are represented within MU by pairs $\langle p^{\eta,\eta}, p'^{\eta,\eta}\rangle$ whose interpretation is restricted to pairs of disjoint subsets of the identity relation corresponding to inverse images of <u>true</u> and <u>false</u>. MU_0 is extended to MU_1 by adding the boolean relation constants of MU to the basic terms of MU_0. MU_1 is axiomatized by adding the following two axioms to those of MU_0:

$$P_1 : \vdash p^{\eta,\eta} \subseteq E^{\eta,\eta},\ p'^{\eta,\eta} \subseteq E^{\eta,\eta}$$
$$P_2 : \vdash p^{\eta,\eta} \cap p'^{\eta,\eta} = \Omega^{\eta,\eta}.$$

---

*) Properly speaking, LYNDON constructs a relation algebra in which
if $X_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2 \neq \Omega$ *then*
$(\breve{X}_1;Y_1 \cap X_2;\breve{Y}_2 \cap (\breve{X}_1;Z_1 \cap X_2;\breve{Z}_2);(\breve{Z}_1;Y_1 \cap Z_2;\breve{Y}_2)) \neq \Omega$
does not hold.

**) However, as demonstrated in appendix 3, this assertion can be proved using our axiomatization of MU_2.

The translation theorem implies $o(p \rightarrow S_1,S_2) = m(p;tr(S_1) \cup p';tr(S_2))$, provided $o(S_i) = m(tr(S_i))$, i = 1,2, and $o(p)$ is represented by $\langle m(p),m(p')\rangle$. Thus axiomatization of MU_1 leads to *a theory* of conditionals. This will be demonstrated by deriving the usual axioms for conditionals, cf. McCARTHY [45], as a corollary from

LEMMA 4.4. $\vdash \breve{p} = p$, $p;q = p \cap q$.

Proof. $\breve{p} = p$: Follows from lemma 4.3.b, and axiom P_1.
$p;q = p \cap q$: $\subseteq$. Since $\vdash p \subseteq E, q \subseteq E$, monotonicity implies
$\vdash p;q \subseteq q, p;q \subseteq p$. Thus $\vdash p;q \subseteq p \cap q$.
$\supseteq$. $\vdash p \cap q =$ (lemma 4.2) $p;(\breve{p};q \cap E) \cap q \subseteq p;(\breve{p};q \cap E) \subseteq p;\breve{p};q \subseteq p;q$. □

COROLLARY 4.1. *Using the notation* $(p \rightarrow X,Y) = p;X \cup p';Y$, *we have*
$\vdash (p \rightarrow (p \rightarrow X,Y),Z) = (p \rightarrow X,Z),(p \rightarrow X,(p \rightarrow Y,Z)) =$
$= (p \rightarrow X,Z),(p \rightarrow (q \rightarrow X_1,X_2),(q \rightarrow Y_1,Y_2)) = (q \rightarrow (p \rightarrow X_1,Y_1),(p \rightarrow X_2,Y_2))$.

Proof. Immediate from lemma 4.4, using P_1 and P_2. □

COROLLARY 4.2. $\vdash p;X \cap Y = p;(X \cap Y)$.

Proof. $p;X \cap Y =$ (lemma 4.2) $p;(\breve{p};Y \cap X) \cap Y =$ (lemmas 4.3.a and 4.4)
$p;Y \cap p;X =$ (lemma 4.3.a) $p;(X \cap Y)$. □

In section 1.3 we already mentioned the "$\circ$" operator, defined by $X \circ p \underset{DEF}{=} X;p;U \cap E$. The basic properties of this operator are collected in *)

LEMMA 4.5.

a. $\vdash (X;Y) \circ p = X \circ (Y \circ p)$

b. $\vdash (X \cup Y) \circ p = X \circ p \cup Y \circ p$

c. $\vdash (X \cap Y) \circ p = X;p;\breve{Y} \cap E$

d. $\vdash X;p \subseteq X \circ p\ ;X$ **)

e. $\breve{X};X \subseteq E \vdash X;p = X \circ p\ ;X$

f. $X;p \subseteq q;X \vdash X \circ p \subseteq q$

---

*) Some connections between μ-*terms* and the "$\circ$" operator are collected in section 5.3.

**) Henk Goeman has observed that one can prove $X;p = X \circ p;X$ *iff* $\breve{X};X \circ p;X \subseteq p;U;p$.

Proof. a. By definition, $(X;Y)\circ p = X;Y;p;U \cap E$ and $X\circ(Y\circ p) = X;(Y;p;U \cap E);U \cap E$. Since by lemma 4.3.c $\vdash Y;p;U = (Y;p;U \cap E);U$, the result follows.

b. Immediate from the definitions and lemma 4.1.

c. $X;p;\breve{Y} \cap E$ = (lemmas 4.2 and 4.4) $X;p;(p;\breve{X} \cap \breve{Y}) \cap E$ = (corollary 4.2 and lemma 4.4) $X;p;(\breve{X} \cap \breve{Y}) \cap E$ = (lemma 4.3.b) $(X \cap Y);p;\breve{X} \cap E$ = = monotonicity and lemma 4.3.c) $(X \cap Y);p;U \cap E$.

d. Applying lemma 4.3.c we obtain $\vdash X;p = (X;p;U \cap E);X;p \subseteq (X;p;U \cap E);X = X\circ p\ ;X$.

e. $\subseteq$. By part d above.
$\supseteq$. $X\circ p\ ;X$ = (lemmas 4.2 and 4.4) $X\circ p\ ;X;(\breve{X};\ X\circ p\ ;U \cap E) \subseteq$ (lemma 4.3.c) $X;(\breve{X};X;p;U \cap E) \subseteq$ (assumption) $X;(p;U \cap E) =$ (corollary 4.2) $X;p$.

f. Assume $X;p \subseteq q;X$. Then $\vdash X\circ p = X;p;U \cap E \subseteq q;X;U \cap E \subseteq$ (corollary 4.2) q. □

Observe that from parts d and f of lemma 4.5, we obtain that the following equality holds in all interpretations (compare section 1.3):

$$X\circ p = \cap\{q \mid X;p \subseteq q;X\}.$$

4.3. *Axiomatization of binary relations over cartesian products*

The language MU_2 for binary relations over cartesian products is obtained from MU_1 by adding, for i = 1,...,n, projection function symbols $\pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i}$ to the basic terms of MU_1, for all types concerned. MU_2 is axiomatized by adding the following two axiom schemes to the axioms and rules of MU_1:

$$C_1 : \vdash \pi_1;\breve{\pi}_1 \cap \ldots \cap \pi_n;\breve{\pi}_n = E$$

$$C_2 : \vdash X_1;Y_1 \cap \ldots \cap X_n;Y_n = (X_1;\breve{\pi}_1 \cap \ldots \cap X_n;\breve{\pi}_n);(\pi_1;Y_1 \cap \ldots \cap \pi_n;Y_n), \quad *)$$

where π_i is of type $\langle\eta_1\times\ldots\times\eta_n,\eta_i\rangle$, E stands for $E^{\eta_1\times\ldots\times\eta_n,\eta_1\times\ldots\times\eta_n}$ and X_i and Y_i are of types $\langle\theta,\eta_i\rangle$ and $\langle\eta_i,\xi\rangle$, respectively.

An assignment $x_i := f(x_1,\ldots,x_n)$ is expressed by a statement scheme V of the form $[\pi_1,\ldots,\pi_{i-1},S,\pi_{i+1},\ldots,\pi_n]$. Hence Hoare's *axiom* for the assignment (cf. [29])

*) Note added in print: By a conjecture of PARK our axiomatization of MU_2 is *complete*; this conjecture is supported by the fact that the assertion mentioned at the end of section 4.1, which is not provable using Tarski's axiomatization of binary relation algebras only, can be proved by also using C_1 and C_2, as demonstrated in appendix 3.

$$\vdash \{p(x_1,\ldots,x_{i-1},f(x_1,\ldots,x_n),x_{i+1},\ldots,x_n)\}x_i := f(x_1,\ldots,x_n)\{p(x_1,\ldots,x_n)\}$$

corresponds with the assertion $\vdash$ $tr(V)\circ p\,;tr(V) \subseteq tr(V);p$, as $q_1\{V\}q_2$ is expressed by $q_1;tr(V) \subseteq tr(V);q_2$, and $(tr(V)\circ p)(x_1,\ldots,x_n) =$ $= p(x_1,\ldots,x_{i-1},f(x_1,\ldots,x_n),x_{i+1},\ldots,x_n)$ (compare section 1.3). As functionality of f implies $tr(V);tr(V) \subseteq E$ by lemma 4.11 below, this assertion follows from (the more general) lemma 4.5.e. Thus the axiomatization of MU_2 leads to a theory of assignments.

The following lemma establishes some necessary relationships between projection functions and the E and U constants.

LEMMA 4.6. *For* i=1,...,n:

a. $\vdash \pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i}\circ E^{\eta_i,\eta_i} = E^{\eta_1\times\ldots\times\eta_n,\eta_1\times\ldots\times\eta_n}$

b. $\vdash \pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i};U^{\eta_i,\xi} = U^{\eta_1\times\ldots\times\eta_n,\xi}$

c. $\vdash \breve{\pi}_i^{\eta_i,\eta_1\times\ldots\times\eta_n};\pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i} = E^{\eta_i,\eta_i}$

d. $\vdash \breve{\pi}_i^{\eta_i,\eta_1\times\ldots\times\eta_n};\pi_j^{\eta_1\times\ldots\times\eta_n,\eta_j} = U^{\eta_i,\eta_j}$, *for* $i \neq j$, $j = 1,\ldots,n$.

Proof. a. Let E_n denote $E^{\eta_1\times\ldots\times\eta_n,\eta_1\times\ldots\times\eta_n}$, then $E_n = (C_1)\pi_i;\breve{\pi}_i \cap E_n =$ $= $ (lemma 4.3.c) $\pi_i\circ E^{\eta_i,\eta_i} \subseteq E_n$.

b. $\pi_i;U^{\eta_i,\xi} =$ (lemma 4.3.c) $\pi_i\circ E^{\eta_i,\eta_i};U^{\eta_1\times\ldots\times\eta_n,\xi} =$ (part a above) $U^{\eta_1\times\ldots\times\eta_n,\xi}$.

c. Consider, e.g., n = 2 and i = 1:

$E^{\eta_1,\eta_1} =$ (lemma 4.1.d) $E^{\eta_1,\eta_1};E^{\eta_1,\eta_1} \cap U^{\eta_1,\eta_1};U^{\eta_1,\eta_1}$

$\ldots = (C_2)\ (E^{\eta_1,\eta_1};\breve{\pi}_1 \cap U^{\eta_1,\eta_2};\breve{\pi}_2);(\pi_1;E^{\eta_1,\eta_1} \cap \pi_2;U^{\eta_2,\eta_1}) =$

$=$ (lemma 4.1 and part b above) $\breve{\pi}_1;\pi_1$.

d. Consider, e.g., n = 2, i = 1 and j = 2:

$U^{\eta_1,\eta_2} = E^{\eta_1,\eta_1};U^{\eta_1,\eta_2} \cap U^{\eta_1,\eta_2};E^{\eta_2,\eta_2}$

$\ldots = (C_2)\ (E^{\eta_1,\eta_1};\breve{\pi}_1 \cap U^{\eta_1,\eta_2};\breve{\pi}_2);(\pi_1;U^{\eta_1,\eta_2} \cap \pi_2;E^{\eta_2,\eta_2}) =$

$=$ (part b above) $\breve{\pi}_1;\pi_2$. □

Already in example 1.1 we signalled the analogy between $\bigcap_{i=1}^{n} X_i;\breve{\pi}_i$ and a list of parameters called-by-value. From this point of view properties such as $(\bigcap_{i=1}^{n} X_i;\breve{\pi}_i)\circ E^{\eta_1\times\ldots\times\eta_n,\eta_1\times\ldots\times\eta_n} = \bigcap_{i=1}^{n} X_i\circ E^{\eta_i,\eta_i}$ - the computation of such a list terminates iff the computations of its individual members terminate - and $(\bigcap_{i=1}^{n} X_i;\breve{\pi}_i);\pi_j = (\bigcap_{i=1}^{n} X_i\circ E^{\eta_i,\eta_i});X_j$ - the request for the value of a parameter contained in such a list amounts to computation of the individual value of this parameter plus termination of the computations of the other parameters - are intuitively evident. These and similar properties follow from the following lemma and its corollary.

LEMMA 4.7. For $k,l \leq n$,

$$\vdash X_{i_1}\circ E\ ;\ldots;\ X_{i_k}\circ E;(\bigcap_{\substack{i_j=s_t,\\ j=1,\ldots,k\\ t=1,\ldots,l}} X_{i_j};Y_{s_t});\ \breve{Y}_{s_1}\circ E\ ;\ldots;\ \breve{Y}_{s_l}\circ E =$$

$$= (\bigcap_{j=1}^{k} X_{i_j};\breve{\pi}_{i_j});(\bigcap_{t=1}^{l} \pi_{s_t};Y_{s_t}),\ \textit{with}\ \pi_i\ \textit{of type}\ \langle\eta_1\times\ldots\times\eta_n,\eta_i\rangle,\ \textit{and}\ X_{i_j}\ \textit{and}\ Y_{s_t}\ \textit{of types}\ \langle\theta,\eta_{i_j}\rangle\ \textit{and}\ \langle\eta_{s_t},\xi\rangle,\ \textit{respectively}.$$

Proof. The case of $n = 3$, $k = l = 2$, $i_1 = 1$, $i_2 = 2$, $s_1 = 2$, $s_2 = 3$ is representative. Hence we prove

$$X_1\circ E\ ;X_2\circ E\ ;X_2;Y_2;\ \breve{Y}_2\circ E\ ;\breve{Y}_3\circ E = (X_1;\breve{\pi}_1 \cap X_2;\breve{\pi}_2);(\pi_2;Y_2 \cap \pi_3;Y_3).$$

By lemma 4.6, $X_1;\breve{\pi}_1 \cap X_2;\breve{\pi}_2 = X_1;\breve{\pi}_1 \cap X_2;\breve{\pi}_2 \cap U^{\theta,\eta_3};\breve{\pi}_3$ and $\pi_2;Y_2 \cap \pi_3;Y_3 = \pi_1;U^{\eta_1,\xi} \cap \pi_2;Y_2 \cap \pi_3;Y_3$, whence

$(X_1;\breve{\pi}_1 \cap X_2;\breve{\pi}_2);(\pi_2;Y_2 \cap \pi_3;Y_3) = (C_2)\ X_1;U^{\eta_1,\xi} \cap X_2;Y_2 \cap U^{\theta,\eta_3};Y_3$

= (lemma 4.3.c) $X_1\circ E\ ;U^{\theta,\xi} \cap X_2;Y_2 \cap U^{\theta,\xi};\ \breve{Y}_3\circ E$

= (lemma 4.3.e)

$X_1\circ E\ ;X_2\circ E\ ;(X_1\circ E\ ;U^{\theta,\xi} \cap X_2;Y_2 \cap U^{\theta,\xi};\ \breve{Y}_3\circ E);\ \breve{Y}_2\circ E;\ \breve{Y}_3\circ E.$

By corollary 4.2, $X_1\circ E\ ;U^{\theta,\xi} \cap X_2;Y_2 \cap U^{\theta,\xi};\ \breve{Y}_3\circ E = X_1\circ E\ ;X_2;Y_2;\ \breve{Y}_3\circ E$, whence the result follows by lemma 4.4. □

COROLLARY 4.3. $\vdash (\bigcap_{i=1}^{n} X_i;\breve{\pi}_i)\circ(\bigcap_{i=1}^{n} \pi_i;p_i;\breve{\pi}_i) = X_1\circ p_1\ ;\ldots;\ X_n\circ p_n$, *with* X_i *of type* $\langle\theta,\eta_i\rangle$ *and* p_i *of type* $\langle\eta_i,\eta_i\rangle$.

Proof. $(\bigcap_{i=1}^{n} X_i;\breve{\pi}_i)\circ(\bigcap_{i=1}^{n} \pi_i;p_i;\breve{\pi}_i) = (C_2)\ (\bigcap_{i=1}^{n} X_i;p_i;\breve{\pi}_i);U^{\eta_1\times\ldots\times\eta_n,\theta} \cap E^{\theta,\theta}$

$= $ (lemma 4.6.b) $(\bigcap_{i=1}^{n} X_i;p_i;\breve{\pi}_i);\pi_1;U_1^{\eta_1,\theta} \cap E^{\theta,\theta} =$

$= $ (lemma 4.7) $(X_1;p_1)\circ E\ ;\ldots;\ (X_n;p_n)\circ E\ ;X_1;p_1;U^{\eta_1,\theta} \cap E^{\theta,\theta}$

$= $ (corollary 4.2 and lemma 4.5.a) $X_1\circ p_1\ ;\ldots;\ X_n\circ p_n$. □

One of the consequences of lemma 4.7 is

$$\vdash (\bigcap_{i=1}^{n-1} X_i;\breve{\pi}_i);(\bigcap_{i=1}^{n-1} \pi_i;Y_i) = \bigcap_{i=1}^{n-1} X_i;Y_i,$$

with π_i, X_i and Y_i of types $\langle\eta_1\times\ldots\times\eta_n,\eta_i\rangle$, $\langle\theta,\eta_i\rangle$ and $\langle\eta_i,\xi\rangle$, respectively.

Assume $\eta_1 = \eta_2 = \ldots = \eta_n$ for simplicity, then, apart from the intended interpretation of π_i as special subset of $D^n \times D$,

"axiom C_2 for n-1, in which $\pi_1,\ldots,\pi_{n-1}$ are interpreted as subsets of $D^{n-1} \times D$ "follows from" axiom C_2 for n, n > 2".

This line of thought may be pursued as follows:
Change the definition of type in that only compounds $(\eta_1\times\eta_2)$ are considered, and introduce projection function symbols $\pi_1^{(\eta\times\xi),\eta}$ and $\pi_2^{(\eta\times\xi),\xi}$ only. For n > 2 define $(\eta_1\times\ldots\times\eta_n)$ as $(\ldots((\eta_1\times\eta_2)\times\eta_3)\times\ldots\times\eta_n)$ and $\pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i}$ as, e.g., for n = 3 and i = 1,2,3, $\pi_1^{((\eta_1\times\eta_2)\times\eta_3),(\eta_1\times\eta_2)};\pi_1^{(\eta_1\times\eta_2),\eta_1}$, $\pi_1^{((\eta_1\times\eta_2)\times\eta_3),(\eta_1\times\eta_2)};\pi_2^{(\eta_1\times\eta_2),\eta_2}$ and $\pi_2^{((\eta_1\times\eta_2)\times\eta_3),\eta_3}$. Then it is a simple exercise to *deduce* C_1 and C_2 for n = 3 from *axioms* C_1 and C_2 for n = 2. This indicates that our original approach may be conceived of as a "sugared" version of the more fundamental set-up suggested above. These considerations are related to the work of HOTZ on X-categories (cf. HOTZ [31]).

Arbitrary applications of the "$\breve{}$" operator can be restricted to pro-

jection functions, as demonstrated below; this result will be used in section 5.3 to prove Wright's result on the regularization of linear procedures.

LEMMA 4.8. $\vdash \breve{X} = \breve{\pi}_2;(E \cap \pi_1;X;\breve{\pi}_2);\pi_1$.

Proof. We prove $X = \breve{\pi}_1;(E \cap \pi_1;X;\breve{\pi}_2);\pi_2$. The result then follows by lemma 4.3.b.

$$\pi_1;X;\breve{\pi}_2 \cap E = (C_1)\ \pi_1;X;\breve{\pi}_2 \cap \pi_1;\breve{\pi}_1 \cap \pi_2;\breve{\pi}_2 =$$

$$= (\text{lemmas 4.6.c and 4.3.a})\ \pi_1;(X;\breve{\pi}_2 \cap \breve{\pi}_1) \cap \pi_2;\breve{\pi}_2.$$

Hence,

$$\breve{\pi}_1;(\pi_1;X;\breve{\pi}_2 \cap E) = (C_1)\ \breve{\pi}_1;(\pi_1;X;\breve{\pi}_2 \cap \pi_1;\breve{\pi}_1 \cap \pi_2;\breve{\pi}_2)$$

$$= (\text{lemma 4.3.a})\ \breve{\pi}_1;(\pi_1;(X;\breve{\pi}_2 \cap \breve{\pi}_1) \cap \pi_2;\breve{\pi}_2)$$

$$= ((X;\breve{\pi}_2 \cap \breve{\pi}_1);\breve{\pi}_1 \cap \pi_2;\breve{\pi}_2);\pi_1$$

$$= (\text{lemma 4.6})\ \pi_2 \circ E;(X;\breve{\pi}_2 \cap \breve{\pi}_1)$$

$$= (X;\breve{\pi}_2 \cap \breve{\pi}_1).$$

Therefore, $\breve{\pi}_1;(\pi_1;X;\breve{\pi}_2 \cap E);\pi_2 = X$ follows. □

4.4. *Axiomatization of the "μ_i" operators*

MU is obtained from MU_2 by introducing the "μ_i" operators, and is axiomatized by adding Scott's induction rule, formulated in section 3.2 and referred to as *I*, and the following axiom scheme to the axioms and rules of MU_2:

$$M : \vdash \{\sigma_j[\mu_i X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]/X_i]_{i=1,\ldots,n} \subseteq$$

$$\subseteq \mu_j X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]\}_{j=1,\ldots,n}.$$

The axiomatization of *MU* is motivated by the need to provide a convenient axiomatization of *PL*. Thus one expects axiomatic proofs of (the translations of) properties of *PL* such as the fixed point (lemma 2.1.e) and least fixed point (corollary 2.3) properties, monotonicity (lemma 2.2) and modularity (lemma 2.8), as the union theorem is embodied in Scott's in-

duction rule and substitution is by lemma 3.3 a valid rule of inference. These proofs are provided by the following lemmas:

LEMMA 4.9.

a. *If* $\tau_1(X_1,\ldots,X_n,Y),\ldots,\tau_n(X_1,\ldots,X_n,Y)$ *are monotonic in* $X_1,\ldots,X_n$ *and* Y, *i.e.*, $A_1 \subseteq B_1,\ldots,A_{n+1} \subseteq B_{n+1} \vdash \tau_i(A_1,\ldots,A_{n+1}) \subseteq \tau_i(B_1,\ldots,B_{n+1})$, $i=1,\ldots,n$, *then* $Y_1 \subseteq Y_2 \vdash \{\mu_j X_1\ldots X_n[\tau_1(X_1,\ldots,X_n,Y_1),\ldots,\tau_n(X_1,\ldots,X_n,Y_1)] \subseteq \subseteq \mu_j X_1\ldots X_n[\tau_1(X_1,\ldots,X_n,Y_2),\ldots,\tau_n(X_1,\ldots,X_n,Y_2)]\}_{j=1,\ldots,n}$.

b. (Monotonicity). *If* $\tau(X_1,\ldots,X_n)$ *is syntactically continuous in* $X_1,\ldots,X_n$ *then* τ *is monotonic in* $X_1,\ldots,X_n$, *i.e.*, $X_1 \subseteq Y_1,\ldots,X_n \subseteq Y_n \vdash \tau(X_1,\ldots,X_n) \subseteq \tau(Y_1,\ldots,Y_n)$.

c. (Fixed point property). $\vdash \{\tau_j[\mu_i X_1\ldots X_n[\tau_1,\ldots,\tau_n]/X_i]_{i=1,\ldots,n} = = \mu_j X_1\ldots X_n[\tau_1,\ldots,\tau_n]\}_{j=1,\ldots,n}$.

d. (Least fixed point property, PARK [51]). $\{\tau_j(Y_1,\ldots,Y_n) \subseteq Y_j\}_{j=1,\ldots,n} \vdash \{\mu_j X_1\ldots X_n[\tau_1,\ldots,\tau_n] \subseteq Y_j\}_{j=1,\ldots,n}$.

Proof. a. Use I, taking $\{Y_1 \subseteq Y_2\}$ for Φ and $\{X_j \subseteq \mu_j X_1\ldots X_n[\tau_1(X_1,\ldots,X_n,Y_2),\ldots,\tau_n(X_1,\ldots,X_n,Y_2)]\}_{j=1,\ldots,n}$ for Ψ, and $\tau_j(X_1,\ldots,X_n,Y_1)$ for σ_j, $j = 1,\ldots,n$.

1. $\vdash \{\Omega_j \subseteq \mu_j X_1\ldots X_n[\tau_1(X_1,\ldots,X_n,Y_2),\ldots,\tau_n(X_1,\ldots,X_n,Y_2)]\}_{j=1,\ldots,n}$. Obvious.

2. $\Phi,\Psi \vdash \{\tau_j(X_1,\ldots,X_n,Y_1) \subseteq \mu_j X_1\ldots X_n[\tau_1(X_1,\ldots,X_n,Y_2),\ldots \ldots,\tau_n(X_1,\ldots,X_n,Y_2)]\}_{j=1,\ldots,n}$. By monotonicity of τ_j in $X_1,\ldots,X_n$ and Y, and M.

b. Follows by induction on the complexity of τ, using lemma 4.1.a. and part a above.

c. $\supseteq$. Use I, with Φ empty and taking $\{X_j \subseteq \mu_j X_1\ldots X_n[\tau_1,\ldots,\tau_n]\}_{j=1,\ldots,n}$ for Ψ, proving the induction step with part b above.

d. $\subseteq$. M.

d. Use I, taking $\{\tau_j(Y_1,\ldots,Y_n) \subseteq Y_j\}_{j=1,\ldots,n}$ for Φ and $\{X_j \subseteq Y_j\}_{j=1,\ldots,n}$ for Ψ, proving the induction step with part b above. □

Modularity is but one of the many consequences of the generalized iteration lemma below. For m = 1 this lemma asserts that *simultaneous* minimalization by μ_i-terms is equivalent to *successive singular* minimalization by μ-terms.

LEMMA 4.10. (Generalized iteration). *Let* k *be a natural number s.t.* $k \geq 2$, *let* $K \equiv \{1,\ldots,k\}$ *be subdivided into two nonempty and disjoint subsets* $I \equiv \{p_1,\ldots,p_m\}$ *and* $J \equiv \{q_1,\ldots,q_n\}$, *whence* $k = m+n$. *Then the following holds:*

For $p_i \in I$,

$$\vdash \mu_{p_i} Z_1 \ldots Z_{m+n}[\rho_1,\ldots,\rho_{m+n}] =$$

$$= \mu_i X_1 \ldots X_m[\sigma_1[\mu_j Y_1 \ldots Y_n[\sigma_{m+1},\ldots,\sigma_{m+n}]/Y_j]_{j=1}^n,\ldots$$

$$\ldots,\sigma_m[\mu_j Y_1 \ldots Y_n[\sigma_{m+1},\ldots,\sigma_{m+n}]/Y_j]_{j=1}^n],$$

where $\sigma_s \equiv \rho_{p_s}[X_i/Z_{p_i}]_{p_i \in I}[Y_j/Z_{q_j}]_{q_j \in J}$, $s = 1,\ldots,m$, *and*

$\sigma_{m+s} \equiv \rho_{q_s}[X_i/Z_{p_i}]_{p_i \in I}[Y_j/Z_{q_j}]_{q_j \in J}$, $s = 1,\ldots,n$.

Proof. Generalized iteration is a generalization of the iteration property (cf. BEKIC [1], SCOTT and DE BAKKER [59]). For ease of notation, we establish this property just for the case $I \equiv \{1,\ldots,m\}$ and $J \equiv \{m+1,\ldots,m+n\}$; the general version should be clear.
We use the following notation:

$$\mu_i \equiv \mu_i Z_1 \ldots Z_{m+n}[\rho_1,\ldots,\rho_{m+n}],\ i = 1,\ldots,m+n,$$

$$\hat{\mu}_i X \equiv \mu_i X_1 \ldots X_m[\sigma_1[\mu_j Y_1 \ldots Y_n[\sigma_{m+1},\ldots,\sigma_{m+n}]/Y_j]_{j=1}^n,\ldots$$

$$\ldots,\sigma_m[\mu_j Y_1 \ldots Y_n[\sigma_{m+1},\ldots,\sigma_{m+n}]/Y_j]_{j=1}^n],\ i = 1,\ldots,m,$$

$$\hat{\mu}_j Y(X_1,\ldots,X_m) \equiv \mu_j Y_1 \ldots Y_n[\sigma_{m+1},\ldots,\sigma_{m+n}],\ j = 1,\ldots,n.$$

$\trianglerighteq$: $\hat{\mu}_i X \trianglelefteq \mu_i$, $i = 1,\ldots,m$.
First we notice that $\hat{\mu}_j Y(\mu_1,\ldots,\mu_m) \trianglelefteq \mu_{m+j}$, $j = 1,\ldots,n$, follows by the least fixed point property (lfpp, lemma 4.9.d) from
$\sigma_{m+j}(\mu_1,\ldots,\mu_m,\mu_{m+1},\ldots,\mu_{m+n})$ = (fixed point property, fpp, lemma 4.9.c) μ_{m+j}, $j = 1,\ldots,n$.
Then the result follows also by least fixed point property from
$\sigma_i(\mu_1,\ldots,\mu_m,\hat{\mu}_1 Y(\mu_1,\ldots,\mu_m),\ldots,\hat{\mu}_n Y(\mu_1,\ldots,\mu_m)) \trianglelefteq$ (monotonicity, lemma 4.9.b) $\sigma_i(\mu_1,\ldots,\mu_m,\mu_{m+1},\ldots,\mu_{m+n})$ = (fpp) μ_i, $i = 1,\ldots,m$.

$\subseteq$: $\mu_i \subseteq \hat{\mu}_i X$, $i = 1,\ldots,m$, and $\mu_{m+j} \subseteq \hat{\mu}_j Y(\hat{\mu}_1 X,\ldots,\hat{\mu}_m X)$, $j = 1,\ldots,n$.
Follows by the least fixed point property from

$\sigma_i(\hat{\mu}_1 X,\ldots,\hat{\mu}_m X,\hat{\mu}_1 Y(\hat{\mu}_1 X,\ldots,\hat{\mu}_m X),\ldots,\hat{\mu}_n Y(\hat{\mu}_1 X,\ldots,\hat{\mu}_m X)) =$ (fpp) $\hat{\mu}_i X$,
$i = 1,\ldots,m$, and

$\sigma_{m+j}(\hat{\mu}_1 X,\ldots,\hat{\mu}_m X,\hat{\mu}_1 Y(\hat{\mu}_1 X,\ldots,\hat{\mu}_m X),\ldots,\hat{\mu}_n Y(\hat{\mu}_1 X,\ldots,\hat{\mu}_m X)) =$ (fpp)
$\hat{\mu}_j Y(\hat{\mu}_1 X,\ldots,\hat{\mu}_m X)$, $j = 1,\ldots,n$. □

COROLLARY 4.4. (Modularity). *For* $i = 1,\ldots,n$,

$$\vdash \mu_i X_1\ldots X_n[\sigma_1(\tau_{11}(X_1,\ldots,X_n),\ldots,\tau_{1m}(X_1,\ldots,X_n)),\ldots$$
$$\ldots,\sigma_n(\tau_{n1}(X_1,\ldots,X_n),\ldots,\tau_{nm}(X_1,\ldots,X_n))] =$$
$$= \sigma_i(\mu_{i1}X_{11}\ldots X_{nm}[\tau_{11}(\sigma_1(X_{11},\ldots,X_{1m}),\ldots,\sigma_n(X_{n1},\ldots,X_{nm})),\ldots$$
$$\ldots,\tau_{nm}(\sigma_1(X_{11},\ldots,X_{1m}),\ldots,\sigma_n(X_{n1},\ldots,X_{nm}))],\ldots,\mu_{im}\ \ldots).$$

Proof. We use the following notation:

$$\mu_i \equiv \mu_i X_1\ldots X_n[\sigma_1(\tau_{11}(X_1,\ldots,X_n),\ldots,\tau_{1m}(X_1,\ldots,X_n)),\ldots$$
$$\ldots,\sigma_n(\tau_{n1}(X_1,\ldots,X_n),\ldots,\tau_{nm}(X_1,\ldots,X_n))],\ i = 1,\ldots,n,$$
$$\hat{\mu}_i \equiv \mu_i X_1\ldots X_n X_{11}\ldots X_{nm}[\sigma_1,\ldots,\sigma_n,\tau_{11},\ldots,\tau_{nm}],\ i = 1,\ldots,n,$$
$$\mu_{ij} \equiv \mu_{(i-1)*n+j}X_{11}\ldots X_{nm}[\tau_{11}(\sigma_1(X_{11},\ldots,X_{1m}),\ldots,\sigma_n(X_{n1},\ldots,X_{nm})),\ldots$$
$$\ldots,\tau_{nm}(\ldots)],\ i = 1,\ldots,n,\text{ and } j = 1,\ldots,m,$$
$$\hat{\mu}_{ij} \equiv \mu_{i*n+j}X_1\ldots X_n X_{11}\ldots X_{nm}[\sigma_1(X_{11},\ldots,X_{1m}),\ldots,\sigma_n(X_{n1},\ldots,X_{nm}),$$
$$\tau_{11}(X_1,\ldots,X_n),\ldots,\tau_{nm}(X_1,\ldots,X_n)],$$
$$i = 1,\ldots,n,\text{ and } j = 1,\ldots,m.$$

We have to prove: $\vdash \mu_i = \sigma_i(\mu_{i1},\ldots,\mu_{im})$, $i = 1,\ldots,n$.
This result is a straightforward consequence of parts a, b and c below.

a. $\mu_i = \hat{\mu}_i$.
$\hat{\mu}_i$ = (generalized iteration, lemma 4.10)
$\mu_i X_1\ldots X_n[\sigma_1[\mu_{1j}Y_{11}\ldots Y_{nm}[\tau_{11}(X_1,\ldots,X_n),\ldots,\tau_{nm}(X_1,\ldots,X_n)]/Y_{1j}]_{j=1}^m,\ldots$
$\ldots,\sigma_n[\mu_{nj}Y_{11}\ldots Y_{nm}[\tau_{11}(X_1,\ldots,X_n),\ldots,\tau_{nm}(X_1,\ldots,X_n)]/Y_{nj}]_{j=1}^m] =$
$=$ (fpp, lemma 4.9.c) μ_i.

b. $\hat{\mu}_i =$ (fpp) $\sigma_i(\hat{\mu}_{i1},\ldots,\hat{\mu}_{im})$.

c. $\hat{\mu}_{ij} = \mu_{ij}$.
$\hat{\mu}_{ij}$ = (generalized iteration)
$\mu_{ij}Y_{11}\ldots Y_{nm}[\tau_{11}[\mu_i X_1\ldots X_n[\sigma_1(Y_{11},\ldots,Y_{1m}),\ldots,\sigma_n(Y_{n1},\ldots,Y_{nm})]/X_i]_{i=1}^n,\ldots$
$\ldots,\tau_{nm}[\mu_i X_1\ldots X_n[\sigma_1(Y_{11},\ldots,Y_{1m}),\ldots,\sigma_n(Y_{n1},\ldots,Y_{nm})]/X_i]_{i=1}^n] =$
$=$ (fpp) μ_{ij}. □

Modularity itself has some interesting applications, too, e.g., corollary 4.5 below and the tree-traversal result of DE BAKKER and DE ROEVER [11]. The proof of this result, using modularity in *MU*, is a straightforward transformation of the proof given at the end of section 2.2, which uses modularity in *PL*.

COROLLARY 4.5. $\vdash \{\mu_i X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]^\smile = \mu_i X_1 \ldots X_n[\sigma_1(\breve{X}_1,\ldots,\breve{X}_n)^\smile,\ldots,\sigma_n(\breve{X}_1,\ldots,\breve{X}_n)^\smile]\}_{i=1,\ldots,n}$.

Proof. Let $\tau(X)$ be $\breve{X}$ and $\tau_i(X_1,\ldots,X_n)$ be $\sigma_i(\breve{X}_1,\ldots,\breve{X}_n)$, $i = 1,\ldots,n$. Then corollary 4.5 can be formulated as the following consequence of modularity:

$$\vdash \tau(\mu_i X_1 \ldots X_n[\tau_1(\tau(X_1),\ldots,\tau(X_n)),\ldots,\tau_n(\tau(X_1),\ldots,\tau(X_n))]) =$$

$$= \mu_i X_1 \ldots X_n[\tau(\tau_1(X_1,\ldots,X_n)),\ldots,\tau(\tau_n(X_1,\ldots,X_n))]. \quad \square$$

The last lemma of this chapter states some sufficient conditions for provability of $\Phi \vdash \breve{\sigma};\sigma \subseteq E$, i.e., *functionality* of σ, and is frequently applied in combination with lemma 4.5.e ($\breve{X};X \subseteq E \vdash X;p = X \circ p\ ;X$).

LEMMA 4.11. (Functionality). *The assertion* $\Phi \vdash \breve{\sigma};\sigma \subseteq E$ *is provable if one of the following assertions is provable:*

a. *If* $\sigma = \bigcup_{i=1}^{n} \sigma_i$, *then* $\Phi \vdash \{\sigma_i \circ E\ ;\sigma_j = \sigma_j \circ E\ ;\sigma_i\}_{1\le i<j\le n} \cup \cup \{\breve{\sigma}_i;\sigma_i \subseteq E\}_{i=1,\ldots,n}$.

b. *If* $\sigma = \sigma_1;\breve{\pi}_1 \cap \ldots \cap \sigma_n;\breve{\pi}_n$, *then* $\Phi \vdash \{\breve{\sigma}_i;\sigma_i \subseteq E\}_{i=1,\ldots,n}$.

c. *If* $\sigma = \sigma_1;\sigma_2$, *then* $\Phi \vdash \breve{\sigma}_1;\sigma_1 \subseteq E$, $\breve{\sigma}_2;\sigma_2 \subseteq E$.

d. *If* $\sigma = \sigma_1 \cap \sigma_2$, *then* $\Phi \vdash \breve{\sigma}_1;\sigma_1 \subseteq E$ *or* $\Phi \vdash \breve{\sigma}_2;\sigma_2 \subseteq E$ *or* $\Phi \vdash \breve{\sigma}_1;\sigma_2 \subseteq E$ *or* $\Phi \vdash \breve{\sigma}_2;\sigma_1 \subseteq E$.

e. *If* $\sigma = \mu_i X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]$, *then* $\Phi,\{\breve{X}_i;X_i \subseteq E\}_{i=1,\ldots,n} \vdash \{\breve{\sigma}_i;\sigma_i \subseteq E\}_{i=1,\ldots,n}$.

Proof. Straightforward. $\square$

In the following chapters we shall often use the following notations:

1. $[\sigma_1,\ldots,\sigma_n]$ *for* $\sigma_1;\breve{\pi}_1 \cap \ldots \cap \sigma_n;\breve{\pi}_n$.
2. $[\sigma_1|\ldots|\sigma_n]$ *for* $\pi_1;\sigma_1;\breve{\pi}_1 \cap \ldots \cap \pi_n;\sigma_n;\breve{\pi}_n$.

5. APPLICATIONS

5.1. *An equivalence due to Morris*

In [50] MORRIS proves equivalence of the following two recursive program schemes:

$$f(x,y) \Leftarrow \underline{\text{if}}\ p(x)\ \underline{\text{then}}\ y\ \underline{\text{else}}\ h(f(k(x),y))$$

and

$$g(x,y) \Leftarrow \underline{\text{if}}\ p(x)\ \underline{\text{then}}\ y\ \underline{\text{else}}\ g(k(x),h(y)).$$

We present a proof in our framework.
The following equivalence is stated without proof:

LEMMA 5.1. $\vdash [A_1|\ldots|A_{i-1}|A_i|A_{i+1}|\ldots|A_n];\pi_i =$

$$= [A_1|\ldots|A_{i-1}|E|A_{i+1}|\ldots|A_n];\pi_i;A_i.$$

THEOREM 5.1. (MORRIS)
Let $F \equiv \mu X[[p|E];\pi_2 \cup [p'|E];[K|E];X;H]$ *and* $G \equiv \mu X[[p|E];\pi_2 \cup [p'|E];[K|H];X]$.
Then

$$\vdash F = G,\ [E|H];G = G;H.$$

Proof. Let Φ be empty, $\Psi(X,Y) \equiv \{X = Y,\ [E|H];Y = Y;H\}$,
$\sigma(X) \equiv [p|E];\pi_2 \cup [p'|E];[K|E];X;H$ and $\tau(Y) \equiv [p|E];\pi_2 \cup [p'|E];[K|H];Y$.
Hence, we must prove

$$\vdash \Psi(\mu X[\sigma(X)],\ \mu Y[\tau(Y)]) \qquad \ldots \qquad (5.1.1)$$

We *intend* to use Scott's induction rule. Unfortunately, this rule (as formu-

lated in section 3.1) does *not* apply to (5.1.1), as, *in case of a simultaneous induction argument*, it only yields results about *components of one simultaneous* μ*-term*.
However, the observation that

$$\vdash \mu_1 XY[\sigma(X),\tau(Y)] = \mu X[\sigma(X)]$$

and

$$\vdash \mu_2 XY[\sigma(X),\tau(Y)] = \mu Y[\tau(Y)]$$

are straightforward applications of the iteration lemma (lemma 4.10), gives us the equivalent assertion

$$\vdash \Psi(\mu_1 XY[\sigma(X),\tau(Y)],\ \mu_2 XY[\sigma(X),\tau(Y)])$$

to which Scott's induction rule *does* apply.
Henceforth, such transitions will be tacitly assumed.

Thus, we have to prove:

1. $\vdash \Psi(\Omega,\Omega)$. Obvious.

2. $X = Y,\ [E|H];Y = Y;H \vdash \sigma(X) = \tau(Y),\ [E|H];\tau(Y) = \tau(Y);H.$

a. $\sigma(X) = \tau(Y)$: $[p|E];\pi_2 \cup [p'|E];[K|E];X;H$ = (hyp.)
$[p|E];\pi_2 \cup [p'|E];[K|E];Y;H$ = (hyp.)
$[p|E];\pi_2 \cup [p'|E];[K|E];[E|H];Y$ = (C_2)
$[p|E];\pi_2 \cup [p'|E];[K|H];Y.$

b. $[E|H];\tau(Y) = \tau(Y);H$: $[E|H];([p|E];\pi_2 \cup [p'|E];[K|H];Y)$ =
$= [E|H];[p|E];\pi_2 \cup [E|H];[p'|E];[K|H];Y$ = (C_2)
$= [p|H];\pi_2 \cup [p';K|H;H];Y$ =
$=$ (lemma 5.1) $[p|E];\pi_2;H \cup [p';K|H];[E|H];Y$ =
$=$ (hyp.) $[p|E];\pi_2;H \cup [p'|E];[K|H];Y;H$ =
$= ([p|E];\pi_2 \cup [p'|E];[K|H];Y);H.$ □

Remark. Bruno COURCELLE pointed out to me, that, although a formal derivation of F = G does not seem to be feasible using the least fixed point property instead of Scott's induction rule, no one has proved as yet the impossibility of such a proof.

5.2. *An equivalence involving nested while statements*

A proof of the following equivalence appeared, in a slightly different formulation, in [11]:

$$\vdash \mu X[A_1;X \cup A_2;X \cup E] = A_1{*}E\ ;(A_2;\ A_1{*}E){*}E, \quad \ldots \qquad (5.2.1)$$

where A*E stands for $\mu X[A;X \cup E]$ and "*" has priority over ";".
The present author feels, however, that the proof contained therein obscures some of the issues involved; these are: *modular decomposition* and *the use of simultaneous recursion* (compare *modularity*: lemma 2.8 and corollary 4.4). This can be understood as follows:

1. The modular decomposition of $A_1;X \cup A_2;X \cup E$ as $\sigma_1(X,\sigma_2(X))$, with $\sigma_1(X,Y) \equiv A_1;X \cup Y$ and $\sigma_2(X) \equiv A_2;X \cup E$, leads to $\mu_1 XY[A_1;X \cup Y,\ A_2;X \cup E] =$ (iteration) $\mu X[A_1;X \cup \mu Y[A_2;X \cup E]] =$ = (fpp) $\mu X[A_1;X \cup A_2;X \cup E]$.
2. $A_1{*}E\ ;(A_2;\ A_1{*}E){*}E = \mu_1 XY[A_1;X \cup E,\ A_2;X;Y \cup E];\mu_2 XY[A_1;X \cup E,\ A_2;X;Y \cup E]$, which is also a consequence of iteration (lemma 4.10).

These observations suggest that (5.2.1) is a consequence of the following equivalence:

THEOREM 5.2. $\vdash \mu_1 = \hat{\mu}_1;\hat{\mu}_2,\ \mu_2 = \hat{\mu}_2,$

with $\mu_i \equiv \mu_i XY[A_1;X \cup Y,\ A_2;X \cup E]$ and $\hat{\mu}_i \equiv \mu_i XY[A_1;X \cup E,\ A_2;X;Y \cup E]$, $i = 1,2$.

Proof. $\subseteq$: Follows by the least fixed point property (lemma 4.9.c) from:

a. $\sigma_1(\hat{\mu}_1;\hat{\mu}_2,\hat{\mu}_2) = A_1;\hat{\mu}_1;\hat{\mu}_2 \cup \hat{\mu}_2 = (A_1;\hat{\mu}_1 \cup E);\hat{\mu}_2 =$ (fpp) $\hat{\mu}_1;\hat{\mu}_2$,

b. $\sigma_2(\hat{\mu}_1;\hat{\mu}_2) = A_2;\hat{\mu}_1;\hat{\mu}_2 \cup E =$ (fpp) $\hat{\mu}_2$.

$\supseteq$: We prove $\vdash \hat{\mu}_1;\mu_2 \subseteq \mu_1,\ \hat{\mu}_2 \subseteq \mu_2,$
with $\hat{\mu}_1;\hat{\mu}_2 \subseteq \hat{\mu}_1;\mu_2 \subseteq \mu_1$ as obvious consequence.

Let $\tau_1(X) \equiv A_1;X \cup E$ and $\tau_2(X,Y) \equiv A_2;X;Y \cup E$. Then we must prove, using Scott's induction rule:

1. $\vdash \Omega \subseteq \mu_2,\ \Omega;\mu_2 \subseteq \mu_1$. Obvious.

2. $X;\mu_2 \subseteq \mu_1$, $Y \subseteq \mu_2 \vdash \tau_1(X);\mu_2 \subseteq \mu_1$, $\tau_2(X,Y) \subseteq \mu_2$.

   a. $\tau_1(X);\mu_2 = (A_1;X \cup E);\mu_2 \subseteq$ (hyp.) $A_1;\mu_1 \cup \mu_2 =$ (fpp) μ_1.

   b. $\tau_2(X,Y) = A_2;X;Y \cup E \subseteq$ (hyp.) $A_2;X;\mu_2 \cup E \subseteq$ (hyp.) $A_2;\mu_1 \cup E =$ $=$ (fpp) μ_2. □

5.3. *Wright's regularization of linear procedures*

In [65] WRIGHT obtains the following results:

a. The class of recursively enumerable subsets of N^2 is the smallest class of sets with the successor relation S as member and closed under the operations "$\breve{}$", ";" and "$\mu X[Q \cup P;X;R]$", where Q, P and R are subsets of N^2 which are contained in this class.

b. In the proof of part a the main auxiliary result can be generalized to a setting in which N is replaced by any abstract domain $\mathcal{D}$. This generalization is:

$$\vdash \mu X[Q \cup P;X;R] = \breve{\pi}_1;\mu Y[E \cup [P|\breve{R}];Y]\circ(E \cap \pi_1;Q;\breve{\pi}_2);\pi_2 \quad \ldots \quad (5.3.1)$$

In the present calculus (5.3.1) can be proved axiomatically.

The following two auxiliary lemmas are needed:

LEMMA 5.2. $\vdash [A|B]\circ p = E \cap \pi_1;A;\breve{\pi}_1;p;\pi_2;\breve{B};\breve{\pi}_2$.

Proof. Straighforward from lemma 4.5.c. □

LEMMA 5.3. $\vdash \mu X[A;X \cup B]\circ p = \mu X[A\circ X \cup B\circ p]$.

Proof. Amounts to a straightforward application of Scott's induction rule. □

Now Wright's result (5.3.1) follows from theorem 5.3 below by two applications of lemma 5.3.

THEOREM 5.3. (Wright)

$$\vdash \underbrace{\mu X[Q \cup P;X;R]}_{L} = \breve{\pi}_1;\underbrace{\mu X[(E \cap \pi_1;Q;\breve{\pi}_2) \cup [P|\breve{R}];X]\circ E}_{R};\pi_2.$$

Proof. $\subseteq$: Follows by the least fixed point property from:

$\breve{\pi}_1;\ R\circ E\ ;\pi_2$ = (fpp) $\breve{\pi}_1;\{(E \cap \pi_1;Q;\breve{\pi}_2) \cup [P|\breve{R}];R\}\circ E\ ;\pi_2$ = (lemma 4.5.a)
$\breve{\pi}_1;(E \cap \pi_1;Q;\breve{\pi}_2);\pi_2 \cup \breve{\pi}_1;\ [P|\breve{R}]\circ(R\circ E)\ ;\pi_2$ = (lemma 4.8)
$Q \cup \breve{\pi}_1;\ [P|\breve{R}]\circ(R\circ E)\ ;\pi_2$ = (lemma 5.2)
$Q \cup \breve{\pi}_1;(E \cap \pi_1;P;\breve{\pi}_1;\ R\circ E\ ;\pi_2;R;\breve{\pi}_2);\pi_2$ = (lemma 4.8)
$Q \cup P;\breve{\pi}_1;\ R\circ E\ ;\pi_2;R.$

$\supseteq$: One derives by similar techniques:

$$\breve{\pi}_1;((E \cap \pi_1;Q;\breve{\pi}_2) \cup [P|\breve{R}]\circ(E \cap \pi_1;L;\breve{\pi}_2))\ ;\pi_2 = L,$$

whence by lemmas 4.8 and 5.2

$$(E \cap \pi_1;Q;\breve{\pi}_2) \cup [P|\breve{R}]\circ(E \cap \pi_1;L;\breve{\pi}_2) \subseteq E \cap \pi_1;L;\breve{\pi}_2,$$

and by the least fixed point property

$$R\circ E \subseteq E \cap \pi_1;L;\breve{\pi}_2 \subseteq \pi_1;L;\breve{\pi}_2.$$

By lemma 4.6.c one therefore obtains

$$\breve{\pi}_1;\ R\circ E\ ;\pi_2 \subseteq L. \quad \square$$

The reader might notice that $\breve{\pi}_1;\mu X[(\pi_1;Q;\breve{\pi}_2 \cap E) \cup [P|\breve{R}];X]\circ E\ ;\pi_2$ does not correspond with any program scheme. Using work of GARLAND and LUCKHAM [23] this has been remedied in I. GUESSARIAN [25] by replacing this term by an equivalent one which does correspond with a program scheme.

5.4. *Axiomatization of the natural numbers*

In general, programs manipulate data *with a special structure*, such as natural numbers, lists and trees. Consequently, proofs about the input-output relationships of these programs often make use of the specific structural properties of these data. In order to axiomatize such proofs, we have to axiomatize relations over *special* domains. This is effected by adding certain axioms, *characterizing the structural properties of these data as properties of certain relation constants* (cf. example 1.3), to the general system of chapter 4. As the relational language MU is particularly suited to express induction arguments, the sequel is devoted to (1) the axiomatization of domains satisfying some induction rule and (2) the axiom-

atic derivation of properties of recursive programs manipulating data which belong to these domains.

To begin with, we discuss below an axiom system for the natural numbers N which improves on a similar system described in DE BAKKER and DE ROEVER [11]. In the next section an axiomatic proof of the primitive recursion theorem is presented involving a simple termination argument; the reader should consult HITCHCOCK and PARK [28] for a more elaborate theory of termination. Chapter 6 contains axiom systems for various types of trees and correctness proofs of programs, such as the TOWERS OF HANOI, which manipulate these structures.

In [11] the natural numbers N were axiomatized as follows:

Nonlogical constants are a boolean relation constant $p_0^{\eta,\eta}$ and an individual relation constant $S^{\eta,\eta}$. These satisfy:

$$N_1 : \vdash \breve{S};S \cap p_0 = \Omega.$$
$$N_2 : \vdash \breve{S};S \subseteq E,$$
$$N_3 : \vdash S;\breve{S} = E,$$
$$N_4^* : \vdash E \subseteq \mu X[p_0 \cup \breve{S};X;S].$$

Clearly, the *intended* interpretation of p_0 is $\{<0,0>\}$ and of S is $\{<n,n+1> \mid n \in N\}$. However, these axioms model also *any number of disjoint copies of* N:

> Let J be any nonempty index set, D_J be the disjoint union $\overset{\vee}{j\in J} N_j$ of $|J|$ copies of N, $m_J(p_0)$ be $\{<<0,j>,<0,j>> \mid j \in J\}$ and $m_J(S)$ be $\{<<n,j>,<n+1,j>> \mid n \in N, j \in J\}$.
> Then $<D_J,m_J(p_0),m_J(S)>$ satisfies N_1, N_2, N_3 and N_4^*.

Let $R^* \equiv \mu X[R;X \cup E]$. Note that

$$\vdash \mu X[R;X \cup E] = \mu X[X;R \cup E] \qquad \ldots \qquad (5.4.1)$$

is a consequence of Scott's induction rule.
Then we *exclude disjoint copies of N from being models* by replacing N_4^* by

$$N_4 : \vdash U \subseteq \breve{S}^*;p_0;S^*.$$

This can be understood as follows:

> Assume to the contrary that the underlying domain of some model for N_1, N_2, N_3 and N_4 contains two disjoint copies of N, say N_a and N_b. Certainly $\langle 0_a,0_b\rangle \in U$, whence N_4 implies $\langle 0_a,0_b\rangle \in \breve{S}^*;p_0;S^*$. By N_1 and N_2, $\langle 0_a,0_a\rangle \in \breve{S}^*$ and $\langle 0_b,0_b\rangle \in S^*$ are the only pairs contained in $\breve{S}^*$ and S^* with 0_a as first and 0_b as second element, respectively. Therefore, by definition of ";", $\langle 0_a,0_b\rangle \in p_0$, and this contradicts $p_0 \subseteq E$.

Henceforth, N *designates the (domain) type of the natural numbers, i.e., of any structure satisfying* N_1, N_2, N_3 *and* N_4.

As first consequence of these axioms *atomicity* of p_0 is derived. Following example 1.2.f this is expressed by

LEMMA 5.4. $\vdash p_0;U \cap U;p_0 \subseteq p_0$ *).

Proof. $p_0;U \cap U;p_0 =$ (lemma 4.3.e) $p_0;U;p_0 \subseteq (N_4)\ p_0;\breve{S}^*;p_0;S^*;p_0 =$
$=$ (fpp and (5.4.1)) $p_0;(\breve{S};\breve{S}^* \cup E);p_0;(S^*;S \cup E);p_0 =$
$= (N_1$ and $N_2)\ p_0;p_0;p_0 =$ (lemma 4.4) p_0. □

Secondly, N_4^* follows from

LEMMA 5.5. $\vdash E = \mu X[p_0 \cup \breve{S};X;S]$.

Proof. $\subseteq$: Derive $\vdash E \cap \breve{S}^*;p_0;S^* \subseteq \mu X[p_0 \cup \breve{S};X;S]$ by Scott's induction rule. Then the result follows from N_4.
We prove

$$E \cap X;p_0;S^* \subseteq \mu X[p_0 \cup \breve{S};X;S] \vdash E \cap (\breve{S};X \cup E);p_0;S^* \subseteq \mu X[p_0 \cup \breve{S};X;S].$$

As

$$E \cap (\breve{S};X \cup E);p_0;S^* = (E \cap \breve{S};X;p_0;S^*) \cup (E \cap p_0;S^*),$$

the proof of this splits into two parts:

a. $E \cap p_0;S^* =$ (lemma 4.3.e) $p_0 \cap p_0;S^* \subseteq p_0 \subseteq$ (fpp) $\mu X[p_0 \cup \breve{S};X;S]$.

b. $E \cap \breve{S};X;p_0;S^* = (N_1$ and N_2, (5.4.1) and fpp) $\breve{S};S \cap \breve{S};X;p_0;(S^*;S \cup E) =$

*) As the reader will have understood, the suppressed domain types of the assertions in this section and the following one are all equal to N, and the semantics must be interpreted accordingly. The same holds, mutatis mutandis, for the next chapter.

$= (N_1)\ \breve{S};S \cap \breve{S};X;p_0;S^*;S \subseteq$ (hyp., lemma 4.3.a) $\breve{S};\mu X[p_0 \cup \breve{S};X;S];S \subseteq$
$\subseteq$ (fpp) $\mu X[p_0 \cup \breve{S};X;S]$.

$\supseteq$: Straightforward from Scott's induction rule. □

Let *eq* stand for $\mu X[[p_0|p_0] \cup [\breve{S}|\breve{S}];X;[S,S]]$.
Clearly, $\langle\langle n,m\rangle,\langle n,m\rangle\rangle \in eq$ iff $n = m$. In relational formulation, this amounts to

LEMMA 5.6. $\vdash eq;\pi_1 = \pi_2$ ... (5.4.2)

Proof. First we prove $\vdash [p_0|p_0];\pi_1 = [p_0|p_0];\pi_2$ ... (5.4.3)

a. $[p_0|p_0];\pi_1 =$ (lemma 4.6.b) $(\pi_1;p_0;\breve{\pi}_1 \cap \pi_2;p_0;\breve{\pi}_2);(\pi_1 \cap \pi_2;U) =$
$=$ (C_2) $\pi_1;p_0 \cap \pi_2;p_0;U =$ (lemma 4.3.e) $\pi_1;p_0 \cap \pi_2;p_0;U;p_0 =$
$=$ (lemma 5.4 and monotonicity) $\pi_1;p_0 \cap \pi_2;p_0$.

b. $[p_0|p_0];\pi_2 = \pi_1;p_0 \cap \pi_2;p_0$ is similarly derived.

c. Combination of parts a and b then yields (5.4.3).

Next we prove (5.4.2).

$\subseteq$: Use Scott's induction rule on eq. By lemma 5.5 we have to prove parts d and e below:

d. $[p_0|p_0];\pi_1 = [p_0|p_0];\pi_2 \subseteq \pi_2$.

e. $X;\pi_1 \subseteq \pi_2 \vdash [\breve{S}|\breve{S}];X;[S|S];\pi_2$.
$[\breve{S}|\breve{S}];X;[S|S];\pi_2 = [\breve{S}|\breve{S}];X;\pi_1;S \subseteq [\breve{S}|\breve{S}];\pi_2;S = \pi_2;\breve{S};S \subseteq \pi_2$.

$\supseteq$: Similarly. □

5.5. *The primitive recursion theorem*

This is the following theorem:

THEOREM 5.4. *Let* G: $N^n \to N$ *and* H: $N^{n+2} \to N$ *be primitive recursive functions. Then there exists an unique total function* F: $N^{n+1} \to N$ *such that, for all* $x_1,\ldots,x_n,y \in N$:

$F(x_1,\ldots,x_n,y) =$ *if* $y = 0$ *then* $G(x_1,\ldots,x_n)$ *else*

$$H(x_1,\ldots,x_n,y-1,F(x_1,\ldots,x_n,y-1)) \quad \ldots \quad (5.5.1)$$

Proof. To simplify the notation we take n = 1.
The minimal solution of (5.5.1) is

$$\underbrace{\mu X[[E|p_0];\pi_1;G \cup [\pi_1,\pi_2;\breve{S},[E|\breve{S}];X];H]}_{\mu\tau}.$$

We prove below that $\mu\tau$ is *total*. By the least fixed point property, then certainly $\mu\tau \subseteq F$, if F is any solution of (5.5.1). If F is a function, then $\mu\tau \subseteq F$ implies by lemma 4.3.d that $\mu\tau = \mu\tau\circ E;F$, whence $\mu\tau = F$ follows from totality of $\mu\tau$. It remains to be demonstrated that such an F exists, i.e., $\mu\tau$ is functional; this follows from Scott's induction rule by repeated application of lemma 4.11. □

LEMMA 5.7. $G\circ E^{1,1} = E^{1,1}$, $H\circ E^{1,1} = E^{3,3} \vdash E^{2,2} \subseteq \mu\tau;U^{1,2}$,
with $\sigma^{j,k} \equiv \sigma\underbrace{N\times N\times\ldots\times N}_{j \text{ times}},\underbrace{N\times N\times\ldots\times N}_{k \text{ times}}$.

Proof. Assume $G\circ E^{1,1} = E^{1,1}$ and $H\circ E^{1,1} = E^{3,3}$ ... (5.5.2)
Then

$$\vdash E^{2,2} = [E^{1,1}|\mu X[p_0 \cup \breve{S};X;S]]$$

holds by lemma 5.5 and

$$\vdash [E^{1,1}|\mu X[p_0 \cup \breve{S};X;S]] \subseteq \mu\tau;U^{1,2}$$

follows from Scott's induction rule as proved below, whence the result.
We prove the induction step only:

$$[E^{1,1}|X] \subseteq \mu\tau;U^{1,2} \vdash [E^{1,1}|p_0 \cup \breve{S};X;S] \subseteq \mu\tau;U^{1,2}.$$

$\mu\tau;U^{1,2}$ = (fpp) $[E|p_0];\pi_1;G;U^{1,2} \cup [\pi_1,\pi_2;\breve{S},[E|\breve{S}];\mu\tau];H;U^{1,2}$

= (lemma 4.3.c by totality of π_1, G and H)
$[E|p_0];U^{2,2} \cup [\pi_1,\pi_2;\breve{S},[E|\breve{S}];\mu\tau];U^{3,2}$

= (lemma 4.6.b)
$[E|p_0];U^{2,2} \cup [\pi_1,\pi_2;\breve{S},[E|\breve{S}];\mu\tau];(\pi_1;U^{1,2} \cap \pi_2;U^{1,2} \cap \pi_3;U^{1,2})$

$$= [E|p_0];U^{2,2} \cup (\pi_2;\breve{S};U^{1,2} \cap [E|\breve{S}];\mu\tau;U^{1,2})$$

$$\supseteq [E|p_0];U^{2,2} \cup [E|\breve{S}];\mu\tau;U^{1,2};[E|S]$$

$$\supseteq \text{(hyp.)}\ [E|p_0 \cup \breve{S};X;S]. \quad \square$$

Remark. Since in the proof above the induction argument applies to the very structure of the underlying domain, we run here up against the axiomatic counterpart of Burstall's *structural induction* (cf. [5]).

6. AXIOMATIC LIST PROCESSING

6.1. *Lists, linear lists and ordered linear lists*

For our purpose it is sufficient to characterize a domain of *lists* as a collection of binary trees which is closed w.r.t. the following operations:

(1) *taking a binary tree* t *apart* by applying the *car* and *cdr* functions, resulting in its constituent subtrees car(t) and cdr(t), if possible; otherwise, t is an *atom* and satisfies the predicate *at*, whence at(t) = t,

(2) *constructing a new binary tree from two old ones* by application of the function *cons*,

where car, cdr and cons are related by $car = \breve{cons};\pi_1$ and $cdr = \breve{cons};\pi_2$. ... (6.1.1)

Thus we introduce one (applied) individual constant $cons^{\eta\times\eta,\eta}$ and one (applied) boolean constant $at^{\eta,\eta}$ and postulate these to satisfy the following axioms:

L_1 : $\vdash$ $cons;\breve{cons} = E^{\eta\times\eta,\eta\times\eta}$

L_2 : $\vdash$ $\breve{cons};cons \subseteq E^{\eta,\eta}$

L_3 : $\vdash$ $at \cap \breve{cons};cons = \Omega^{\eta,\eta}$

L_4 : $\vdash$ $E^{\eta,\eta} \subseteq \mu X[at \cup [\breve{cons};\pi_1;X,\breve{cons};\pi_2;X];cons]$.

Remarks. 1. L_1 implies that cons is total and also that cons is a function; hence $\breve{cons};\pi_1$ and $\breve{cons};\pi_2$ are also functions. L_2 yields that cons is a function, L_3 that an atom can never be taken apart, and L_4 that any list is either an atom or can be first taken apart and then fitted together again.

2. Satisfaction of these axioms establishes $<D_\eta,at,cons>$ as a structure of lists. This leads us to introduce a new type, L, reserved for lists, resulting in <L,L> and <L×L,L> as new types for at and cons. If there is no confusion between different domains of lists, L is also used to indicate a domain of lists.

3. An interesting application of this axiom system is the correctness proof of an iterative tree marking algorithm contained in section 3.4 of DE ROEVER [16] (this algorithm is essentially due to FLOYD, cf. exercise 2.3.5.7 of KNUTH [36]).

Linear lists are lists with the additional property that car(1) is always an atom.
Thus we obtain axioms for linear lists by replacing L_1 by

$$LL_1 : \vdash \text{cons};\breve{\text{cons}} = [\pi_1;at,\pi_2],$$

and postulating L_2, L_3 and L_4.
The reader may wonder why we didn't replace L_4 by

$$\vdash E^{\eta,\eta} \subseteq \mu X[at \cup [car,cdr;X];cons].$$

The reason for this is, that LL_1, L_2 and L_3 imply

$$\vdash \mu X[at \cup [car;X,cdr;X];cons] = \mu X[at \cup [car,cdr;X];cons], \quad \ldots (6.1.2)$$

cf. the proof of lemma 6.1.b.
LL is then introduced as type for linear lists.
With linear lists as domain and range some interesting properties can be proved, such as

(1) if conc stands for $\mu X[cons \cup [\pi_1;car,[\pi_1;cdr,\pi_2];X];cons]$, i.e.,
$conc(1_1,1_2) \Longleftarrow \underline{\text{if}}\ atom(1_1)\ \underline{\text{then}}\ cons(1_1,1_2)\ \underline{\text{else}}\ cons(car(1_1), conc(cdr(1_1),1_2))$, ... (6.1.3)
then conc is *associative*, i.e., $conc(conc(1_1,1_2),1_3) = conc(1_1,conc(1_2,1_3))$, cf. McCARTHY [45],

(2) if *first* and *last* stand for (at ∪ car) and $\mu X[at \cup cdr;X]$, ... (6.1.4)

respectively, then conc;first = π_1;first and conc;last = π_2;last,

(3) conc is a *total* function.

It is proved in lemma 6.3 that these properties of linear lists can be obtained as corollaries of the analoguous properties for *ordered* linear lists.

Ordered linear lists are linear lists with the additional property that some relation holds between the subsequent atoms of these lists. For convenience, we do not use a relation $\prec'$, holding, e.g., between 1_1 and 1_2: $1_1 \prec' 1_2$, but introduce the *characteristic predicate* $\prec$ of this relation: $\langle 1_1,1_2\rangle \prec \langle 1_1,1_2\rangle$ iff $1_1 \prec' 1_2$, i.e., $\prec = \pi_1;\prec';\breve{\pi}_2 \cap E$. ... (6.1.5)
In principle $\prec'$ need not be a partial order at all; many interesting properties can be proved *without* this requirement: theorems 6.1 and 6.3 establish (1) and a variant of (2) above for ordered linear lists and theorem 6.2 establishes conc$\circ$E = $\prec$, i.e., conc$(1_1,1_2)$ is defined iff $1_1 \prec' 1_2$.
In order to axiomatize ordered linear lists we introduce therefore a boolean constant $\prec^{\eta\times\eta,\eta\times\eta}$, replace LL_1 by $\vdash$ cons;$\breve{\text{cons}}$ = $[\pi_1;\text{at},\pi_2];\prec$, i.e., $\langle$car(1),cdr(1)$\rangle \prec \langle$car(1),cdr(1)$\rangle$, and stipulate that $\langle at_i,at_{i+1}\rangle \prec \prec \langle at_i,at_{i+1}\rangle$ holds for all subsequent atoms at_i and at_{i+1} which constitute an ordered linear list. This leads to the following axioms for ordered linear lists:

$$OLL_1 : \vdash \text{cons};\breve{\text{cons}} = [\pi_1;\text{at},\pi_2];\prec$$
$$OLL_2 : \vdash \breve{\text{cons}};\text{cons} \subseteq E^{\eta,\eta}$$
$$OLL_3 : \vdash \text{at} \cap \breve{\text{cons}};\text{cons} = \Omega^{\eta,\eta}$$
$$OLL_4 : \vdash E^{\eta,\eta} \subseteq \mu X[\text{at} \cup [\text{car},\text{cdr};X];\text{cons}] \ ^{*)}$$
$$OLL_5 : \vdash \prec = [\pi_1;\text{last},\pi_2;\text{first}]\circ\prec,$$

with last and first as defined in (6.1.4).

Remarks. OLL is introduced as type for ordered linear lists and (at $\cup$ [car,cdr;X];cons) will be referred to as τ_{OLL}. Then OLL_4 reads as $\vdash E^{\eta,\eta} \subseteq \mu X[\tau_{OLL}]$.

First some simple properties of at, car, cdr, cons and $\prec$ are collected in

*) We might have chosen L_4, alternatively, as follows from the related discussion above, cf. (6.1.2).

LEMMA 6.1. *Let* at' *denote* [car,cdr];cons (*or* $\breve{\text{cons}}$;cons, *which is equivalent*) *then the following properties hold for*

a. *Lists*: $\vdash$ E = μX[at ∪ [car;X,cdr;X];cons], at ∪ at' = E, cons;at' = cons, cons;at = Ω.

b. *Linear lists*: $\vdash$ E = μX[at ∪ [car,cdr;X];cons], cons;$\breve{\text{cons}}$ = π_1∘at, car;at = car, car;at' = Ω.

c. *Ordered linear lists*: $\vdash$ cons;$\breve{\text{cons}}$ = π_1∘at;∝.

Proof. a. E = μX[at ∪ [car;X,cdr;X];cons]: ⊆. Axiom L_4.

⊇. Use $\bar{I}$ with Φ empty, taking {X ⊆ E} for Ψ and (at ∪ [car;X,cdr;X];cons) for σ.

at ∪ at' = E : E = μX[at ∪ [car;X,cdr;X];cons] =
= (fpp) at ∪ [car,cdr];cons.

cons;at' = cons : cons;at' = cons;$\breve{\text{cons}}$;cons = (L_1) cons.

cons;at = Ω : cons;at = cons;$\breve{\text{cons}}$∘E ;at = (L_2) cons;($\breve{\text{cons}}$;cons ∩ at) =
= (L_3) Ω.

b. cons;$\breve{\text{cons}}$ = π_1∘at : Obvious from LL_1.

car;at = car : $\breve{\text{cons}}$;π_1;at = (lemma 4.5.e) $\breve{\text{cons}}$;cons∘E ;π_1∘at ;π_1 =
= (from above) $\breve{\text{cons}}$;cons∘E ;π_1 = $\breve{\text{cons}}$;π_1.

car;at' = Ω : $\breve{\text{cons}}$;π_1;at' = $\breve{\text{cons}}$;[π_1;at,π_2];π_1;at' =
= $\breve{\text{cons}}$;π_1;(at ∩ at') = (LL_3) Ω.

E = μX[at ∪ [car,cdr;X];cons]: Prove (6.1.2),

μX[at ∪ [car;X,cdr;X];cons] = $\underbrace{\mu X[at \cup [car,cdr;X];cons]}_{\equiv\ \mu X[\tau_{LL}]}$,

first, using lfpp (lemma 4.9.d) in both directions:

⊆: at ∪ [car;μX[τ_{LL}],cdr;μX[τ_{LL}]];cons = (LL_1 and lemma 4.3.c)
at ∪ [$\breve{\text{cons}}$;[π_1;at,π_2];π_1;μX[τ_{LL}], cdr;μX[τ_{LL}]];cons =
= at ∪ [$\breve{\text{cons}}$;π_1;at;μX[τ_{LL}],cdr;μX[τ_{LL}]];cons = (fpp and part a above)
at ∪ [$\breve{\text{cons}}$;π_1;at,cdr;μX[τ_{LL}]];cons = (from above)
at ∪ [car,cdr;μX[τ_{LL}]];cons = (fpp) μX[τ_{LL}].

⊇: Similarly.

The remainder of the proof is similar to that of part a above.

c. cons;$\breve{\text{cons}}$ = π_1∘at ;∝ : Obvious from OLL_1. □

In the proofs of this chapter the following property, lemma 4.5.e, is

often implicitly applied: $\breve{X};X \subseteq E \vdash X;p = X \circ p\ ;X$. Functionality of the terms involved is proved by repeated application of lemma 4.11 and may require in the induction steps $\breve{X};X \subseteq E$ as additional hypothesis and $\widetilde{\tau_{OLL}(X)};\tau_{OLL}(X) \subseteq E$ as additional conclusion.

Next we establish an auxiliary lemma.

LEMMA 6.2. $\vdash [[\pi_1;at,\pi_2];cons,\pi_3];conc =$

$= [\pi_1;at,\pi_2];\prec;[\pi_1,[\pi_2,\pi_3];conc];cons.$

Proof. $\vdash [[\pi_1;at,\pi_2];cons,\pi_3];conc =$
$= [[\pi_1;at,\pi_2];cons,\pi_3];[\pi_1;car,[\pi_1;cdr,\pi_2];conc];cons =$
$= [[\pi_1;at,\pi_2];cons;\widetilde{cons};\pi_1,[[\pi_1;at,\pi_2];cons;\widetilde{cons};\pi_2,\pi_3];conc];cons$, as may be proved using C_2 and (6.1.1),
$\ldots = (OLL_1)\ [[\pi_1;at,\pi_2];\prec;\pi_1,[[\pi_1;at,\pi_2];\prec;\pi_2,\pi_3];conc];cons$, whence by lemma 4.5.e and cor. 4.2 the result follows. □

The fundamental theorem of this section is

THEOREM 6.1. $\vdash$ conc;first = $\prec;\pi_1$;first,conc;last = $\prec;\pi_2$;last.

Proof. We derive $\vdash$ conc;first = $\prec;\pi_1$;first as an example; the proof of $\vdash$ conc;last = $\prec;\pi_2$;last uses similar techniques.

By lemma 6.1 it is sufficient to prove $\vdash [\pi_1;\mu X[\tau_{OLL}],\pi_2];conc;first =$
$= [\pi_1;\mu X[\tau_{OLL}],\pi_2];\prec;\pi_1;first$. Use I with Φ empty, taking $\{[\pi_1;X,\pi_2];conc;first = [\pi_1;X,\pi_2];\prec;\pi_1;first\}$ for Ψ and τ_{OLL} for σ. *)

$\vdash \Psi(\Omega)$. Obvious.

$\Psi(X) \vdash \Psi(\tau_{OLL}(X))$.

1. $[\pi_1;at,\pi_2];cons;first =$ (lemma 6.1) $[\pi_1;at,\pi_2];cons;car =$
   $= (OLL_1)\ [\pi_1;at,\pi_2];\prec;\pi_1 = [\pi_1;at,\pi_2];\prec;\pi_1;first.$

2. The nucleus of the proof:
   $[\pi_1;car,[\pi_1;cdr;X,\pi_2];conc]\circ\prec =$
   $= (OLL_5)\ [\pi_1;car,[\pi_1;cdr;X,\pi_2];conc;first]\circ\prec =$
   $= [\pi_1;car,[\pi_1;cdr,\pi_2];[\pi_1;X,\pi_2];conc;first]\circ\prec$ = (induction hypothesis)

*) This corresponds with structural induction on the first coordinate, cf. section 5.5.

$[\pi_1;car,[\pi_1;cdr,\pi_2];[\pi_1;X,\pi_2];\prec;\pi_1;first]\circ\prec$ =
= $[\pi_1;car,[\pi_1;cdr;X,\pi_2];\prec;\pi_1;first]\circ\prec$ = (lemma 4.5.e)
$[\pi_1;car,[\pi_1;cdr;X,\pi_2]\circ\prec;[\pi_1;cdr;X,\pi_2];\pi_1;first]\circ\prec$ = (cor. 4.2)
$[\pi_1;cdr;X,\pi_2]\circ\prec;[\pi_1;car,[\pi_1;cdr;X,\pi_2];\pi_1;first]\circ\prec$ =
= $[\pi_1;cdr;X,\pi_2]\circ\prec;([\pi_1;car,\pi_1;cdr;X];[\pi_1;last,\pi_2;first])\circ\prec$ = (lemma 4.5.a)
$[\pi_1;cdr;X,\pi_2]\circ\prec;([\pi_1;car,\pi_1;cdr;X]\circ([\pi_1;last,\pi_2;first]\circ\prec))$ = (OLL_5)
$[\pi_1;car,\pi_1;cdr;X]\circ\prec;[\pi_1;cdr;X,\pi_2]\circ\prec$.

3. $[[\pi_1;car,\pi_1;cdr;X];cons,\pi_2];conc;first$ = (lemmas 6.1 and 6.2)
   $[\pi_1;car,\pi_1;cdr;X]\circ\prec;[\pi_1;car,[\pi_1;cdr;X,\pi_2];conc];cons;first$ =
   = (using $cons;first = \prec;\pi_1;at$, lemma 4.5.e and part 2)
   $[\pi_1;car,\pi_1;cdr;X]\circ\prec;[\pi_1;cdr;X,\pi_2]\circ\prec;\pi_1;car$.

4. $[[\pi_1;car,\pi_1;cdr;X];cons,\pi_2];\prec;\pi_1;first$ = (lemma 4.5.e)
   $[[\pi_1;car,\pi_1;cdr;X];cons,\pi_2]\circ\prec;[\pi_1;car,\pi_1;cdr;X];cons;first$ =
   = (using $cons;first = \prec;\pi_1;at$, lemma 4.5.e and cor. 4.2)
   $[[\pi_1;car,\pi_1;cdr;X];cons,\pi_2]\circ\prec;\pi_1;car$.

5. $[[\pi_1;car,\pi_1;cdr;X];cons,\pi_2]\circ\prec$ = (OLL_5 and cor. 4.2)
   $[\pi_1;car,\pi_1;cdr;X]\circ\prec;[\pi_1;cdr;X,\pi_2]\circ\prec$.

6. The proof of the induction step follows from part 1 and
   $[\pi_1;[car,cdr;X];cons,\pi_2];conc;first$ =
   = $[[\pi_1;car,\pi_1;cdr;X];cons,\pi_2];conc;first$ = (part 3)
   $[\pi_1;car,\pi_1;cdr;X]\circ\prec;[\pi_1;cdr;X,\pi_2]\circ\prec;\pi_1;car$ = (parts 4 and 5)
   $[\pi_1;[car,cdr;X];cons,\pi_2];\prec;\pi_1;first$. □

We apply this theorem for the first time in

THEOREM 6.2. $\vdash$ $conc\circ E = \prec$.

Proof.

1. $conc\circ E$ = (fpp)
   $([\pi_1;at,\pi_2];cons \cup [\pi_1;car,[\pi_1;cdr,\pi_2];conc];cons)\circ E$.

2. $([\pi_1;at,\pi_2];cons)\circ E = [\pi_1;at,\pi_2]\circ\prec$.

3. $([\pi_1;car,[\pi_1;cdr,\pi_2];conc];cons)\circ E$ =
   = (OLL_5 and theorem 6.1) $[\pi_1;car,[\pi_1;cdr,\pi_2];\prec;\pi_1]\circ\prec$ =
   = $[\pi_1;car,\pi_1;cdr]\circ\prec;[\pi_1;cdr,\pi_2]\circ\prec$ =
   = $[\pi_1;[car,cdr];cons,\pi_2]\circ\prec$.

By combining parts 1, 2 and 3 one obtains the result from lemmas 4.5.b and 6.1. □

Next we prove the classical

THEOREM 6.3. (Associativity of conc).

$$\vdash [[\pi_1,\pi_2];\mathrm{conc},\pi_3];\mathrm{conc} = [\pi_1,[\pi_2,\pi_3];\mathrm{conc}];\mathrm{conc}.$$

Proof. By lemma 6.1 it is sufficient to prove $\vdash [[\pi_1;\mu X[\tau_{OLL}],\pi_2];\mathrm{conc},\pi_3];\mathrm{conc} = [\pi_1;\mu X[\tau_{OLL}],[\pi_2,\pi_3];\mathrm{conc}];\mathrm{conc}$. Use I with Φ empty, taking $\{[[\pi_1;X,\pi_2];\mathrm{conc},\pi_3];\mathrm{conc} = [\pi_1;X,[\pi_2;\pi_3];\mathrm{conc}];\mathrm{conc}\}$ for Ψ and τ_{OLL} for σ.

$\vdash \Psi(\Omega)$. Obvious.

$\Psi(X) \vdash \Psi(\tau_{OLL}(X))$. Follows from parts 1 and 2 below.

1. Lemma 6.2 and theorem 6.1 imply $[[\pi_1;\mathrm{at},\pi_2];\mathrm{cons},\pi_2];\mathrm{conc} = [\pi_1;\mathrm{at},[\pi_2,\pi_3];\mathrm{conc}];\mathrm{cons}$.

2. $[[[\pi_1;\mathrm{car},\pi_1;\mathrm{cdr};X];\mathrm{cons},\pi_2];\mathrm{conc},\pi_3];\mathrm{conc} =$
   $=$ (fpp, OLL_5, theorem 6.1) $[[\pi_1;\mathrm{car},[\pi_1;\mathrm{cdr};X,\pi_2];\mathrm{conc}];\mathrm{cons},\pi_3];\mathrm{conc} =$
   $=$ (similarly) $[\pi_1;\mathrm{car},[[\pi_1;\mathrm{cdr};X,\pi_2];\mathrm{conc},\pi_3];\mathrm{conc}];\mathrm{cons} =$
   $=$ (hypothesis) $[\pi_1;\mathrm{car},[\pi_1;\mathrm{cdr};X,[\pi_2,\pi_3];\mathrm{conc}];\mathrm{conc}];\mathrm{cons} =$
   $= [\pi_1;[\mathrm{car},\mathrm{cdr};X];\mathrm{cons},[\pi_2,\pi_3];\mathrm{conc}];\mathrm{conc}$. □

Finally we observe that, *although intuitively not obvious*, linear lists are a special case of ordered linear lists.
This follows from

(1) totality of last and first for linear lists, the proof of which is a matter of routine,

and

(2) the fact that substitution in $OLL_1,\ldots,OLL_5$ of $E^{\eta\times\eta,\eta\times\eta}$ for $\prec^{\eta\times\eta,\eta\times\eta}$ results in $LL_1,\ldots,LL_4$ *) and $\vdash E^{\eta\times\eta,\eta\times\eta} = [\pi_1;\mathrm{last},\pi_2;\mathrm{first}]\circ E^{\eta\times\eta,\eta\times\eta}$, which is proved by $[\pi_1;\mathrm{last},\pi_2;\mathrm{first}]\circ E^{\eta\times\eta,\eta\times\eta} =$ (corollary 4.3) $(\pi_1;\mathrm{last})\circ E^{\eta,\eta}\ ;(\pi_2;\mathrm{first})\circ E^{\eta,\eta} = \pi_1\circ(\mathrm{last}\circ E^{\eta,\eta})\ ;\pi_2\circ(\mathrm{first}\circ E^{\eta,\eta}) =$
$=$ (part 1 above) $\pi_1\circ E^{\eta,\eta}\ ;\pi_2\circ E^{\eta,\eta} =$ (lemma 4.6) $E^{\eta\times\eta,\eta\times\eta}$.

Hence we have, a fortiori,

*) By (6.1.2).

LEMMA 6.3. *Any property of ordered linear lists holds upon substitution of* $\propto$ *by* $E^{LL\times LL,LL\times LL}$ *for linear lists.*

6.2. *Properties of head and tail*

The head and tail functions *hd* and *tl*, both of type $\langle N^+\times OLL,OLL\rangle$, where N^+ is the type of the positive natural numbers and OLL the type of ordered linear lists, are defined by

(1) hd(n,l) is the ordered linear list of n elements which constitutes the initial part of l of length n, if extant, and

(2) tl(n,l) is the ordered linear list which constitutes the remainder of l, after hd(n,l) has been chopped off, if possible.

If both sides are defined, clearly properties such as
conc(hd(n,l),tl(n,l)) = l, tl(n+1,l) = cdr(tl(n,l)),
conc(hd(n,l),car(tl(n,l))) = hd(n+1,l), tl(n,conc(hd(n,l_1),l_2)) = l_2 and
hd(n,conc(hd(n,l_1),l_2)) = hd(n,l_1) are valid and therefore amenable to proof within our system.

First we observe that the axioms for N^+ are the axioms for N which are modified by "renaming" p_0 as p_1 (p_0' is renamed as p_1', too).
Next we introduce some notation:

hd denotes $\mu X[\pi_1\circ p_1\ ;\pi_2;car \cup [\pi_2;car,[\pi_1;\check{S},\pi_2;cdr];X];cons]$, ... (6.2.1)

tl denotes $\mu X[\pi_1\circ p_1\ ;\pi_2;cdr \cup [\pi_1;\check{S},\pi_2;cdr];X]$, ... (6.2.2)

$\pi_{i_1,\ldots,i_n}$ denotes $[\pi_{i_1},\ldots,\pi_{i_n}]$. ... (6.2.3)

Then the above mentioned properties are established in

THEOREM 6.4.

a. $\vdash$ [hd,tl];conc $= [hd,tl]\circ\propto\ ;\pi_2$, *of type* $\langle N^+\times OLL,OLL\rangle$.

b. $\vdash$ tl;cdr $= [\pi_1;S,\pi_2];tl$, *of type* $\langle N^+\times OLL,OLL\rangle$.

c. $\vdash$ [hd,tl;car];conc $= [\pi_1;S,\pi_2];hd$, *of type* $\langle N^+\times OLL,OLL\rangle$.

d. $\vdash [\pi_1,[\pi_{1,2};hd,\pi_3];conc];tl = [\pi_{1,2};hd,\pi_3]\circ\propto\ ;\pi_3$,
of type $\langle N^+\times OLL\times OLL,OLL\rangle$.

e. $\vdash [\pi_1,[\pi_{1,2};hd,\pi_3];conc];hd = [\pi_{1,2};hd,\pi_3]\circ\propto\ ;\pi_{1,2};hd$,
of type $\langle N^+\times OLL\times OLL,OLL\rangle$.

f. $\vdash$ tl$\circ$E $= [hd,tl]\circ\prec$, *of type* $\langle N^+\times OLL, N^+\times OLL\rangle$.

Proof. The techniques required for proving this theorem are illustrated by proving parts a and e.

a. First we prove $\vdash$ $[hd,tl];conc \subseteq \pi_2$. Then the result follows from $[hd,tl];conc = $ (lemma 4.3.d) $([hd,tl];conc)\circ E ;\pi_2 = $ (theorem 6.2) $[hd,tl]\circ\prec ;\pi_2$.
Apply I, with Φ empty and taking $\{[hd,tl];X \subseteq \pi_2\}$ for Ψ and $(cons \cup [\pi_1;car,[\pi_1;cdr,\pi_2];X];cons)$ for σ. Then $\Psi(X) \vdash \Psi(\sigma(X))$ follows from parts 1 and 2 below.

1. $[hd,tl];cons = (OLL_1)$ $[hd;at,tl];\prec;cons = $ (fpp and lemma 6.1) $\pi_1\circ p_1 ;[\pi_2;car,\pi_2;cdr];\prec;cons \subseteq (OLL_2)\ \pi_2$.
2. $[hd,tl];[\pi_1;car,[\pi_1;cdr,\pi_2];X];cons = [hd;car,[hd;cdr,tl];X];cons =$ $= $ (fpp and lemma 6.1) $[\pi_2;car,[[\pi_1;\breve{S},\pi_2;cdr];hd,[\pi_1;\breve{S},\pi_2;cdr];tl];X];cons \subseteq$ (hypothesis) $[\pi_2;car,[\pi_1;\breve{S},\pi_2;cdr];\pi_2];cons \subseteq (OLL_2)\ \pi_2$.

e. Apply I, with Φ empty, taking $\{[\pi_1,[\pi_{1,2};hd,\pi_3];conc];X =$ $= [\pi_{1,2};hd,\pi_3]\circ\prec ;\pi_{1,2};X\}$ for Ψ and $(\pi_1\circ p_1 ;car \cup$ $\cup [\pi_2;car,[\pi_1;\breve{S},\pi_2;cdr];X];cons)$ for σ. Then $\Psi(X) \vdash \Psi(\sigma(X))$ follows from part 1 and 4 below.

1. It follows from lemma 4.3.d that $[\pi_{1,2};hd,\pi_3];conc;car \subseteq$ (fpp) $\pi_2;car$ and $([\pi_{1,2};hd,\pi_3];conc;car)\circ E = [\pi_{1,2};hd,\pi_3]\circ(conc\circ at') =$ (fpp) $[\pi_{1,2};hd,\pi_3]\circ(conc\circ E) =$ (theorem 6.2) $[\pi_{1,2};hd,\pi_3]\circ\prec$ together imply $[\pi_{1,2};hd,\pi_3];conc;car = [\pi_{1,2};hd,\pi_3]\circ\prec ;\pi_2;car$.
2. $[\pi_{1,2};hd,\pi_3];conc;cdr =$ $= [\pi_{1,2};hd,\pi_3]\circ\prec ;(\pi_1\circ p_1 ;\pi_3 \cup \pi_1\circ p_1' ;[\pi_{1,2};hd;cdr,\pi_3];conc)$ is proved similarly.
3. $\pi_{1,2};hd;cdr =$ (fpp) $[\pi_1;\breve{S},\pi_2;cdr];hd$.
4. $[\pi_1,[\pi_{1,2};hd,\pi_3];conc];\pi_1\circ p_1' ;[\pi_2;car,[\pi_1;\breve{S},\pi_2;cdr];X];cons =$ $= $ (parts 1 and 2) $[\pi_{1,2};hd,\pi_3]\circ\prec ;\pi_1\circ p_1' ;$ $[\pi_2;car,[\pi_1;\breve{S},\pi_2;cdr,\pi_3];[\pi_1,[\pi_{1,2};hd,\pi_3];conc];X];cons =$ $= $ (hypothesis) $[\pi_{1,2};hd,\pi_3]\circ\prec ;\pi_1\circ p_1' ;$ $[\pi_2;car,[\pi_1;\breve{S},\pi_2;cdr,\pi_3];[\pi_{1,2};hd,\pi_3]\circ\prec ;\pi_{1,2};X];cons =$

$= [\pi_{1,2};hd,\pi_3]\circ\prec;\pi_1\circ p_1' ;$
$[\pi_2;car,[[\pi_1;\breve{S},\pi_2;cdr];hd,\pi_3]\circ\prec;[\pi_1;\breve{S},\pi_2;cdr,\pi_3];\pi_{1,2};X];cons =$
$=$ (part c)
$[\pi_{1,2};hd,\pi_3]\circ\prec;[\pi_{1,2};hd;cdr,\pi_3]\circ\prec;\pi_1\circ p_1' ;$
$[\pi_2;car,\pi_{1,2};[\pi_1;\breve{S},\pi_2;cdr];X];cons =$ (using OLL_5)
$[\pi_{1,2};hd,\pi_3]\circ\prec;\pi_{1,2};\ \pi_1\circ p_1' ;[\pi_2;car,[\pi_1;\breve{S},\pi_2;car];X];cons.$ □

Since $\prec = \pi_1;\prec';\breve{\pi}_2 \cap E$ (6.1.5), transitivity of the *relation* $\prec'$, i.e., the property $\prec';\prec' \subseteq \prec'$, implies $\pi_{1,2}\circ\prec;\pi_{2,3}\circ\prec \subseteq \pi_{1,3}\circ\prec$, *transitivity of the predicate* $\prec$ *in its two arguments* or *transitivity of* $\prec$, for short. This follows from $\pi_{1,2}\circ\prec;\pi_{2,3}\circ\prec = (\pi_1;\prec';\breve{\pi}_2 \cap E);(\pi_2;\prec';\breve{\pi}_3 \cap E) =$
$= \pi_1;\prec';\breve{\pi}_2 \cap \pi_2;\prec';\breve{\pi}_3 \cap E \subseteq \pi_1;\prec';\prec';\breve{\pi}_3 \cap E \subseteq$ (assumption)
$\pi_1;\prec';\breve{\pi}_3 \cap E = \pi_{1,3}\circ\prec.$ ... (6.2.1)

COROLLARY 6.1. *Let* $\prec$ *be transitive (in its two arguments), then*

a. $\vdash [[\pi_1;S,\pi_2];hd,\pi_3]\circ\prec =$
$= [\pi_{1,2};hd,\pi_{1,2};tl;car]\circ\prec;[\pi_{1,2};tl;car,\pi_3]\circ\prec;[\pi_{1,2};hd,\pi_3]\circ\prec.$

b. $\vdash ([\pi_1;S,\pi_2];tl)\circ E = [hd,tl;car]\circ\prec;[tl;car,tl;cdr]\circ\prec;[hd,tl;cdr]\circ\prec.$

Proof.

a. $[[\pi_1;S,\pi_2];hd,\pi_3]\circ\prec$ = (theorem 6.4.c) $[[\pi_{1,2};hd,\pi_{1,2};tl;car];conc,\pi_3]\circ\prec =$
$=$ (theorem 6.1) $[\pi_{1,2};hd,\pi_{1,2};tl;car]\circ\prec;[\pi_{1,2};tl;car,\pi_3]\circ\prec$, whence the result can be deduced from the assumption.

b. $([\pi_1;S,\pi_2];tl)\circ E = $ (theorem 6.4.f) $[[\pi_1;S,\pi_2];hd,[\pi_1;S,\pi_2];tl]\circ\prec =$
$=$(theorem 6.4.b and 6.4.c) $[[hd,tl;car];conc,tl;cdr]\circ\prec =$ (theorem 6.1 and transitivity of $\prec$) $[hd,tl;car]\circ\prec;[tl;car,tl;cdr]\circ\prec;[hd,tl;cdr]\circ\prec.$ □

6.3. *Correctness of the TOWERS OF HANOI*

6.3.a. *Informal part*

We present an informal argument for the correctness of a certain version of the TOWERS OF HANOI program. This version looks in ALGOL-like notation as follows:

```
procedure TVH(n,x,y,ℓ1,ℓ2,ℓ3); integer n,x,y; ordered linear list ℓ1,ℓ2,ℓ3;
          if n=1 then MOVE(n,x,y,ℓ1,ℓ2,ℓ3) else
```

```
        begin n:= n-1; y:= alt(x,y); TVH(n,x,y,ℓ1,ℓ2,ℓ3);
              y:= alt(x,y); MOVE(n,x,y,ℓ1,ℓ2,ℓ3); x:= alt(x,y);
              TVH(n,x,y,ℓ1,ℓ2,ℓ3); n:= n+1; x:= alt(x,y)
        end;

procedure MOVE(n,x,y,ℓ1,ℓ2,ℓ3); integer n,x,y; ordered linear list ℓ1,ℓ2,ℓ3;
        if x=1∧y=2 then begin ℓ2:= cons(car(ℓ1),ℓ2); ℓ1:= cdr(ℓ1) end else
        if x=1∧y=3 then begin ℓ3:= cons(car(ℓ1),ℓ3); ℓ1:= cdr(ℓ1) end else
        if x=2∧y=3 then begin ℓ3:= cons(car(ℓ2),ℓ3); ℓ2:= cdr(ℓ2) end else
        if x=2∧y=1 then begin ℓ1:= cons(car(ℓ2),ℓ1); ℓ2:= cdr(ℓ2) end else
        if x=3∧y=1 then begin ℓ1:= cons(car(ℓ3),ℓ1); ℓ3:= cdr(ℓ3) end else
        if x=3∧y=2 then begin ℓ2:= cons(car(ℓ3),ℓ2); ℓ3:= cdr(ℓ3) end else
                                                              undefined;

integer procedure alt(x,y); integer x,y; if x≥1 ∧ x≤3 ∧ y≥1 ∧ y≤3 then
                    alt:= 6-x-y else undefined
```

To which conditions does correctness of TVH amount?
First we have to assume the transitivity of the relation ordering the ordered linear lists considered above. We do not wish to elaborate this assumption in the present informal setting; for this the reader is referred to the next section.

Let us assume $x \neq y$, then execution of TVH(n,x,y,ℓ1,ℓ2,ℓ3), if defined,

1. Has to result in the removal of the top n discs of the pin "identified by" x, to the pin identified by y.
2. These discs are moved in correct order, i.e., never a larger disc is placed on a smaller disc.
3. The discs are moved *one at a time*.

As to (3): we cannot formalize this requirement, as the present formalism deals only with input-output relationships and not with intermediate stages: cf. section 1.3.

As to (2): this condition is implicit in our approach as all functions are only defined for ordered linear lists. Thus, the question whether or not the order is disturbed amounts to whether or not the execution is defined.

As to (1): let us declare R(n,x,y,ℓ1,ℓ2,ℓ3) by

```
procedure R(n,x,y,ℓ1,ℓ2,ℓ3); integer n,x,y; ordered linear list ℓ1,ℓ2,ℓ3;
    if x=1∧y=2 then begin ℓ2:= conc(hd(n,ℓ1),ℓ2); ℓ1:= tl(n,ℓ1) end else
    if x=1∧y=3 then begin ℓ3:= conc(hd(n,ℓ1),ℓ3); ℓ1:= tl(n,ℓ1) end else
    if x=2∧y=3 then begin ℓ3:= conc(hd(n,ℓ2),ℓ3); ℓ2:= tl(n,ℓ2) end else
    if x=2∧y=1 then begin ℓ1:= conc(hd(n,ℓ2),ℓ1); ℓ2:= tl(n,ℓ2) end else
    if x=3∧y=1 then begin ℓ1:= conc(hd(n,ℓ3),ℓ1); ℓ3:= tl(n,ℓ3) end else
    if x=3∧y=2 then begin ℓ2:= conc(hd(n,ℓ3),ℓ2); ℓ3:= tl(n,ℓ3) end else
                                                          undefined.
```

If we assume $x \neq y$, (1) amounts to

$$TVH(n,x,y,\ell1,\ell2,\ell3) = R(n,x,y,\ell1,\ell2,\ell3),$$

provided both sides are defined.

Proof. As TVH(1,x,y,ℓ1,ℓ2,ℓ3) = R(1,x,y,ℓ1,ℓ2,ℓ3) follows from the declarations, we concentrate on the case n > 1:

The induction hypothesis is TVH(n-1,x,y,ℓ1,ℓ2,ℓ3) = R(n-1,x,y,ℓ1,ℓ2,ℓ3), provided both sides are defined. Start with statevector $\xi_0 \equiv$ <n,1,2,ℓ1,ℓ2,ℓ3>.

1. Execution of *n:= n-1; y:= alt(x,y); TVH(n,x,y,l1,l2,l3)* with ξ_0 as input results in

   $$\xi_1 \equiv \text{<n-1,1,3,tl(n-1,ℓ1),ℓ2,conc(hd(n-1,ℓ1),ℓ3)>},$$

   by the induction hypothesis.

2. Execution of *y:= alt(x,y); MOVE(n,x,y,l1,l2,l3)* with ξ_1 as input results in

   $$\xi_2 \equiv \text{<n-1,1,2,cdr(tl(n-1,ℓ1)),cons(car(tl(n-1,ℓ1)),ℓ2),}$$
   $$\text{conc(hd(n-1,ℓ1),ℓ3)>}$$

3. Execution of *x:= alt(x,y); TVH(n,x,y,l1,l2,l3); n:= n+1; x:= alt(x,y)* with ξ_2 as input results in

   $$\xi_2 \equiv \text{<n,1,2,}\underbrace{\text{cdr(tl(n-1,ℓ1))}}_{\text{Expr 1}},$$
   $$\underbrace{\text{conc(hd(n-1,conc(hd(n-1,ℓ1),ℓ3)),cons(car(tl(n-1,ℓ1),ℓ2)))}}_{\text{Expr 2}},$$
   $$\underbrace{\text{tl(n-1,conc(hd(n-1,ℓ1),ℓ3))}}_{\text{Expr 3}}\text{>}.$$

We demonstrate that, provided ξ_3 is defined, ξ_3 equals
<n,1,2,tl(n,ℓ1),conc(hd(n,ℓ1),ℓ2),ℓ3>.

Expr 1: cdr(tl(n-1,ℓ1)) = tl(n,ℓ1) by theorem 6.4.b.

Expr 2: 1. hd(n-1,conc(hd(n-1,ℓ1),ℓ3)) = if hd(n-1,ℓ1) $\prec$ ℓ3 then hd(n-1,ℓ1)
else undefined,
by theorem 6.4.e.

2. conc(hd(n-1,ℓ1),cons(car(tl(n-1,ℓ1)),ℓ2)) =
= conc(conc(hd(n-1,ℓ1),car(tl(n-1,ℓ1))),ℓ2), by associativity of conc, theorem 6.3.

3. conc(hd(n-1,ℓ1),car(tl(n-1,ℓ1))) = hd(n,ℓ1), by theorem 6.4.c.

Thus Expr 2 = if hd(n-1,ℓ1) $\prec$ ℓ3 then conc(hd(n,ℓ1),ℓ2)
else undefined.

Expr 3: tl(n-1,conc(hd(n-1,ℓ1),ℓ3)) = if hd(n-1,ℓ1) $\prec$ ℓ3 then ℓ3
else undefined,
by theorem 6.4.d.

Thus ξ_3 = if hd(n-1,ℓ1) $\prec$ ℓ3 then <n,1,2,tl(n,ℓ1),conc(hd(n,ℓ1),ℓ2),ℓ3>
else undefined, whence the result. □

6.3.b. *An axiomatic correctness proof for the TOWERS OF HANOI*

First we introduce some auxiliary notions:

By example 1.3 it is possible to axiomatize a three-element set {a,b,c} of type 3. Furthermore we need the function *alt* of type <3,3> defined by: if $x \neq y$ then alt(x,y) $\in$ {a,b,c} - {x,y}, and alt(x,y) is undefined, otherwise. Then alt has the following properties: alt(x,y) = alt(y,x), alt(alt(x,y),x) = y and alt(alt(x,y),y) = x. The formal definition of alt, using the *predicates* a, b and c, and the subsequent derivation of these properties is a matter of routine.

$$\pi_{i-j} \stackrel{=}{\text{DEF}} \pi_{i,i+1,\ldots,j} \qquad \text{,for } i < j.$$

Secondly we define TVH, of type $<N^+ \times 3 \times 3 \times OLL \times OLL \times OLL, N^+ \times 3 \times 3 \times OLL \times OLL \times OLL>$, by

$$\text{TVH} \stackrel{=}{\text{DEF}} \mu X[\underbrace{\pi_1 \circ p_1 ; \text{MOVE}}_{\tau_0} \cup \underbrace{\pi_1 \circ p_1' ; [\pi_1 ; \breve{S}, \pi_{2,3} ; \text{alt}, \pi_{4-6}]}_{\tau_1} ; X;$$

$$\underbrace{[\pi_{1-2},\pi_{2,3};\text{alt},\pi_{4-6}];\text{MOVE};[\pi_1,\pi_{2,3};\text{alt},\pi_{3-6}]}_{\tau_2};\text{X};$$

$$\underbrace{[\pi_1;\text{S},\pi_{2,3};\text{alt},\pi_{3-6}]}_{\tau_3}] \qquad \ldots (6.3.1)$$

and

$$\begin{aligned}\text{MOVE} \overset{=}{\text{DEF}}\ & p_{a,b};[\pi_{1-3},\pi_4;\text{cdr},[\pi_4;\text{car},\pi_5];\text{cons},\pi_6] \cup \\ \cup\ & p_{a,c};[\pi_{1-3},\pi_4;\text{cdr},\pi_5,[\pi_4;\text{car},\pi_6];\text{cons}] \cup \\ \cup\ & p_{b,c};[\pi_{1-4},\pi_5;\text{cdr},[\pi_5;\text{car},\pi_6];\text{cons}] \cup \\ \cup\ & p_{b,a};[\pi_{1-3},[\pi_5;\text{car},\pi_4];\text{cons},\pi_5;\text{cdr},\pi_6] \cup \\ \cup\ & p_{c,a};[\pi_{1-3},[\pi_6;\text{car},\pi_4];\text{cons},\pi_5,\pi_6;\text{cdr}] \cup \\ \cup\ & p_{c,b};[\pi_{1-4},[\pi_6;\text{car},\pi_5];\text{cons},\pi_6;\text{cdr}].\end{aligned}$$

with

$$p_{x,y} \overset{=}{\text{DEF}} \pi_2 \circ x\ ;\pi_3 \circ y \qquad \text{for } x,y \in \{a,b,c\}. \qquad \ldots (6.3.2)$$

Thirdly we define p'_{eq}, O and R in order to express correctness of TVH:

$$p'_{eq} \overset{=}{\text{DEF}} \bigcup_{x \neq y} p_{x,y}, \qquad \text{cf. (6.3.2).}$$

$$\begin{aligned}O \overset{=}{\text{DEF}}\ & \underbrace{\pi_2 \circ a\ ;[\pi_{1,4};\text{hd},\pi_5] \circ\!\!\prec\ ;[\pi_{1,4};\text{hd},\pi_6] \circ\!\!\prec}_{O_a} \cup \\ \cup\ & \underbrace{\pi_2 \circ b\ ;[\pi_{1,5};\text{hd},\pi_4] \circ\!\!\prec\ ;[\pi_{1,5};\text{hd},\pi_6] \circ\!\!\prec}_{O_b} \cup \\ \cup\ & \underbrace{\pi_2 \circ c;;[\pi_{1,6};\text{hd},\pi_4] \circ\!\!\prec\ ;[\pi_{1,6};\text{hd},\pi_5] \circ\!\!\prec}_{O_c} \qquad \ldots (6.3.3)\end{aligned}$$

and

$$\begin{aligned}R \overset{=}{\text{DEF}}\ & p_{a,b};[\pi_{1-3},\pi_{1,4};\text{tl},[\pi_{1,4};\text{hd},\pi_5];\text{conc},\pi_6] \cup \\ \cup\ & p_{a,c};[\pi_{1-3},\pi_{1,4};\text{tl},\pi_5,[\pi_{1,4};\text{hd},\pi_6];\text{conc}] \cup \\ \cup\ & p_{b,c};[\pi_{1-4},\pi_{1,5};\text{tl},[\pi_{1,5};\text{hd},\pi_6];\text{conc}] \cup\end{aligned}$$

$$\cup\ p_{b,a};[\pi_{1-3},[\pi_{1,5};hd,\pi_4];conc,\pi_{1,5};tl,\pi_6]\ \cup$$
$$\cup\ p_{c,a};[\pi_{1-3},[\pi_{1,6};hd,\pi_4];conc,\pi_5,\pi_{1,6};tl]\ \cup$$
$$\cup\ p_{c,b};[\pi_{1-4},[\pi_{1,6};hd,\pi_5];conc,\pi_{1,6};tl].$$

Then the correctness of TVH is established by

THEOREM 6.5. (Correctness of TOWERS OF HANOI). *Let* $\prec$ *be transitive (in the sense indicated in (6.2.1)), then*

$$\vdash\ p'_{eq};0;TVH = p'_{eq};0;R.$$

Proof. The proof of this theorem proceeds by induction on N^+, i.e., we prove

$$\vdash\ p'_{eq};[\pi_1;\mu X[p_1\ \cup\ \breve{S};X;S],\pi_{2-6}];0;TVH =$$
$$= p'_{eq};[\pi_1;\mu X[p_1\ \cup\ \breve{S};X;S],\pi_{2-6}];0;R$$

by applying I as follows: let Φ be empty, Ψ be $\{p'_{eq};[\pi_1;X,\pi_{2-6}];0;TVH = p'_{eq};[\pi_1;X,\pi_{2-6}];0;R\}$ and σ be $(p_1\ \cup\ \breve{S};X;S)$. Then the result follows from $\mu X[p_1\ \cup\ \breve{S};X;S] = E^{N^+,N^+}$, cf. lemma 5.5.

We adopt the following strategy:

Using the notation introduced in (6.3.1) we associate in the proof of the induction step terms $P_0,\ldots,P_3$ and $Q_0,\ldots,Q_3$, which are defined below, with

$$p'_{eq};[\pi_1;(p_1\ \cup\ \breve{S};X;S),\pi_{2-6}];0;TVH = \text{(fpp)}$$

$$\underset{P_0}{p'_{eq};0};\underset{Q_0}{\tau_0}\ \cup\ \underset{P_1}{p'_{eq};[\pi_1;\breve{S};X;S,\pi_{2-6}];0};\underset{Q_1}{\tau_1};\underset{P_2}{TVH};\underset{Q_2}{\tau_2};\underset{P_3}{TVH};\underset{Q_3}{\tau_3}\ .$$

Then our correctness proof consists in proving, with Ψ as hypothesis,

$$P_0;\tau_0 = Q_0 \qquad \ldots\ (6.3.4)$$

and

$$P_1;\tau_1;TVH;\tau_2;TVH;\tau_3 =$$
$$= \text{(parts 1 and 2) } Q_1;TVH;\tau_2;TVH;\tau_3 =$$

$$= \text{(part 3) } P_2;\tau_2;\mathrm{TVH};\tau_3 =$$

$$= \text{(parts 4, 5 and 6) } Q_2;\mathrm{TVH};\tau_3 =$$

$$= \text{(part 7) } P_3;\tau_3 = \text{(part 8) } Q_3, \quad ^{*)} \qquad \ldots (6.3.5)$$

since $P_0 \equiv \pi_1 \circ p_1 ; p'_{eq}; 0$, $Q_0 \equiv \pi_1 \circ p_1 ; p'_{eq}; 0; R$, $P_1 \equiv p'_{eq};[\pi_1;\breve{S};X;S,\pi_{2-6}];0$ and $Q_3 \equiv p'_{eq};[\pi_1;\breve{S};X;S,\pi_{2-6}];0;R$, whence (6.3.4) and (6.3.5) together imply

$$p'_{eq};[\pi_1;(p_1 \cup \breve{S};X;S),\pi_{2-6}];0;\mathrm{TVH} = p'_{eq};[\pi_1;(p_1 \cup \breve{S};X;S),\pi_{2-6}];0;R.$$

Without loss of generality we prove

$$p'_{eq};[\pi_1;X,\pi_{2-6}];0;\mathrm{TVH} = p'_{eq};[\pi_1;X,\pi_{2-6}];0;R \vdash$$

$$\vdash [\pi_1;(p_1 \cup \breve{S};X;S),\pi_2;a,\pi_3;b,\pi_{4-6}];0_a;\mathrm{TVH} =$$

$$= [\pi_1;(p_1 \cup \breve{S};X;S),\pi_2;a,\pi_3;b,\pi_{4-6}];0_a;R.$$

Next terms P_i and Q_i are defined as below, $i = 0,\ldots,3$.

Let $0_a(X) \overset{=}{\mathrm{DEF}} [[\pi_1;X,\pi_4];\mathrm{hd},\pi_5]\circ\!\prec ;[[\pi_1;X,\pi_4];\mathrm{hd},\pi_6]\circ\!\prec$, whence $\pi_2 \circ a ; 0_a(E) = 0_a$ (see (6.3.3), and let $0_{a,b} \overset{=}{\mathrm{DEF}} [\pi_{1,4};\mathrm{hd},\pi_5]\circ\!\prec$ and $0_{a,c} \overset{=}{\mathrm{DEF}} [\pi_{1,4};\mathrm{hd},\pi_6]\circ\!\prec$, whence $0_a = \pi_2 \circ a ; 0_{a,b}; 0_{a,c}$. For 0_b and 0_c we introduce similar notations.

$P_0 \overset{=}{\mathrm{DEF}} [\pi_1;p_1,\pi_2;a,\pi_3;b,\pi_{4-6}];0_a.$

$Q_0 \overset{=}{\mathrm{DEF}} [\pi_1;p_1,\pi_2;a,\pi_3;b,\pi_{4-6}];0_a;R.$

$P_1 \overset{=}{\mathrm{DEF}} [\pi_1;\breve{S};X;S,\pi_2;a,\pi_3;b,\pi_{4-6}];0_a.$

$Q_1 \overset{=}{\mathrm{DEF}} 0_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];[\pi_1;X,\pi_2;a,\pi_3;c,\pi_{4-6}];0_a.$

$P_2 \overset{=}{\mathrm{DEF}} 0_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];$
$[\pi_1;X,\pi_2;a,\pi_3;c,[\pi_1;X,\pi_4];\mathrm{tl},\pi_5,[[\pi_1;X,\pi_4];\mathrm{hd},\pi_6];\mathrm{conc}].$

$Q_2 \overset{=}{\mathrm{DEF}} 0_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];[\pi_1,\pi_2;c,\pi_3;b,[\pi_1;X;S,\pi_4];\mathrm{tl},$
$[[\pi_1;X,\pi_4];\mathrm{tl};\mathrm{car},\pi_5];\mathrm{cons},[[\pi_1;X,\pi_4];\mathrm{hd},\pi_6];\mathrm{conc}];$
$[\pi_1;X,\pi_{2-6}];0_c.$

$P_3 \overset{=}{\mathrm{DEF}} 0_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];[\pi_1;X,\pi_2;c,\pi_3;b,[\pi_1;X;S,\pi_4];\mathrm{tl},$
$[[\pi_1;X,[[\pi_1;X,\pi_4];\mathrm{hd},\pi_6];\mathrm{conc}];\mathrm{hd},$

*) Parts 1 to 8 refer to the formal proof at the end of this section.

$[[\pi_1;X,\pi_4];t1;car,\pi_5];cons];conc,$

$[\pi_1;X,[[\pi_1;X,\pi_4];hd,\pi_6];conc];t1].$

$Q_3 \overset{=}{DEF} [\pi_1;\breve{S};X;S,\pi_2;a,\pi_3;b,\pi_{4-6}];O_a;R.$

Finally we prove the induction step as indicated in (6.3.4) and (6.3.5). Assume transitivity of $\prec$, i.e., $\pi_{1,2}\circ\prec;\pi_{2,3}\circ\prec \subseteq \pi_{1,3}\circ\prec$, and the induction hypothesis Ψ.

The proof of $P_0;TVH = Q_0$ is a matter of routine and therefore omitted.

1. $[\pi_1;\breve{S};X;S,\pi_2;a,\pi_3;b,\pi_{4-6}];\tau_1 = (S;\breve{S} = E^{N^+,N^+}$, cf. axiom N_3)
$[\pi_1;\breve{S},\pi_{2-6}];[\pi_1;X,\pi_2;a,\pi_3;c,\pi_{4-6}].$

2. $P_1;\tau_1 = [\pi_1;\breve{S};X;S,\pi_2;a,\pi_3;b,\pi_{4-6}];O_a;[\pi_1;\breve{S},\pi_2,\pi_{2-3};alt,\pi_{4-6}]$ = (lemma 4.5.e)
$= P_1;\tau_1;O_a(S)$ = (corollary 6.1.a, $\prec$ being transitive, and part 1)
$O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];[\pi_1;X,\pi_2;a,\pi_3;c,\pi_{4-6}];O_a = Q_1.$

3. $Q_1;TVH$ = (hypothesis)
$O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];$
$[\pi_1;X,\pi_2;a,\pi_3;c,[\pi_1;X,\pi_4];t1,\pi_5,[[\pi_1;X,\pi_4];hd,\pi_6];conc] = P_2.$

4. $P_2;\tau_2 = P_2;[\pi_{1-2},\pi_{2,3};alt,\pi_{4-6}];MOVE;[\pi_1,\pi_{2,3},alt,\pi_{4-6}] =$
(theorem 6.4) $O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];$
$\underbrace{[\pi_1;X,\pi_2;c,\pi_3;b,[\pi_1;X;S,\pi_4];t1,[[\pi_1;X,\pi_4];t1;car,\pi_5];cons,[[\pi_1;X,\pi_4];hd,\pi_6];conc]}_{Q_2'}.$

5. $Q_2';[\pi_{1,6};hd,\pi_4]\circ\prec =$
$= [[\pi_1;X,[[\pi_1;X,\pi_4];hd,\pi_6];conc];hd,[\pi_1;X;S,\pi_4];t1]\circ\prec;Q_2' =$
$=$ (theorem 6.4) $[[\pi_1;X,\pi_4];hd,\pi_6]\circ\prec;$
$[[\pi_1;X,\pi_4];hd,[\pi_1;X;S,\pi_4];t1]\circ\prec;Q_2'.$

6. (i) $Q_2' = ([\pi_1;X;S,\pi_4];t1)\circ E;Q_2' =$
$[[\pi_1;X,\pi_4];hd,[\pi_1;X;S,\pi_4];t1]\circ\prec;Q_2'.$

(ii) $O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}] = O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];[[\pi_1;X;S,\pi_4];hd,\pi_6]\circ\prec =$
$=$ (corollary 6.1) $\ldots;[[\pi_1;X,\pi_4];hd,\pi_6]\circ\prec.$

By combining parts 4, 5 and (i), (ii) above, we obtain
$P_2;\tau_2 = O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];Q_2';O_{c,b}$. $P_2;\tau_2 = O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];Q_2';O_{c,a}$ is proved similarly. Thus we have $P_2;\tau_2 = O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];Q_2';O_c = Q_2.$

7. $Q_2;TVH$ = (hypothesis) $Q_2;R = P_3.$

8. (i) $[[\pi_1;X,[[\pi_1;X,\pi_4];hd,\pi_6];conc];hd,[[\pi_1;X,\pi_4];tl;car,\pi_5];conc];conc =$

$= $ (theorem 6.4) $[[\pi_1;X,\pi_4];hd,\pi_6]\circ\prec;$

$[[\pi_1;X,\pi_4];hd,[[\pi_1;X,\pi_4];tl;car,\pi_5];conc];conc =$

$= $ (theorems 6.3 and 6.4) $[[\pi_1;X,\pi_4];hd,\pi_6]\circ\prec;$

$[[\pi_1;X;S,\pi_4];hd,\pi_5];conc$.

(ii) $[\pi_1;X,[[\pi_1;X,\pi_4];hd,\pi_6];conc];tl =$ (theorem 6.4)

$[[\pi_1;X,\pi_4];hd,\pi_6]\circ\prec;\pi_6$.

(iii) By part 6(ii), $O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}] = \ldots;[[\pi_1;X,\pi_4];hd,\pi_6]\circ\prec$.

By combining parts (i), (ii) and (iii) above, we obtain

$$P_3 = O_a(\breve{S};X;S);[\pi_1;\breve{S},\pi_{2-6}];$$
$$[\pi_1;X,\pi_2;c,\pi_3;b,[\pi_1;X;S,\pi_4];tl,[[\pi_1;X;S,\pi_4];hd,\pi_5];conc,\pi_6],$$

whence $P_3;\tau_3 = [\pi_1;\breve{S};X;S,\pi_2;a,\pi_3;b,\pi_{4-6}];O_a;R = Q_3$. □

7. ASSESSMENT

The present investigation shows that:

1. A conceptually attractive framework for a mathematical theory of correctness of programs comprises:

   1.1. The notion of execution of a program by introducing an idealized interpreter.

   1.2. An operational semantic function o which abstracts the relevant information from the computations defined by this interpreter.

   1.3. A mathematical language (with semantic function m) in which to express and derive properties of programs.

   1.4. A translation tr between programs and terms of this mathematical language, i.e., a mapping satisfying

   $$o(T) = m(tr(T))$$

   for every program T.

2. A theory of correctness of programs requires an operator describing the interaction between programs and predicates; in the present theory this is the "$\circ$" operator.

3. The "$\circ$" operator is crucial to an expedient axiomatization of the call-by-value parameter mechanism.

4. The axiomatization of correctness proofs of recursive programs can be applied to the axiomatization of recursive data structures; this leads to a unified theory of recursive programs and recursive data.

Our system of proof is based on the least fixed point characteriza-

tion, as opposed to Floyd's method of inductive assertions [21]; the least fixed point characterization derives from McCarthy's recursion induction [45]. We restricted ourselves to the axiomatization of first-order programs with a particular parameter mechanism, call-by-value. As demonstrated in LYNDON [37] the given axiomatization of MU_0 is incomplete; however, as noted by PARK (personal communication) our axiomatization of MU_2 may very well be complete. Consequently, the following problems remain open:

1. An axiomatization of call-by-value for higher-order programs.
2. The equivalence of the least fixed point characterization with a generalization of the method of inductive assertions is proved by DE BAKKER and MEERTENS in [12] in case of a simple language for recursive programs with one variable.
   Generalization of this result to more complicated programming languages.
3. Proof or disproof of Park's conjecture that our axiomatization of MU_2 is complete.

The diligent reader of these chapters should pause a moment, and ponder upon the vast discrepancy existing between

> *the combination of intuition, understanding, and plausibility of arguments used, by which a human being gets convinced of the truth of some statement,*

and

> *the linguistic obstacles which are posed by the axiomatic method, and the sheer size of the resulting machine-checkable proofs, which seems inversely proportional to any understanding by a human being.*

Even if one tries to meet halfway between these two seemingly contradictory extremes, as in the informal correctness proof of the Towers of Hanoi program contained in section 6.3.a, one still faces the problem that the human brain (a product of five billion years of evolution) has its own direct methods of grasping a problem, methods which lead to a process of understanding often orders of magnitude faster than the means by which this human brain understands the meticulous step-by-step derivations of artificial reasoning.
We may view this monograph as embodying an experiment about the extent to which a limited portion of the workings of the human intellect, in this

case in the field of semantics of programming languages, may be replaced by artificial reasoning.

While this experiment is motivated by the need to replace the frail and error-prone intuitive human reasoning in order to obtain machine-checkable proofs, the fact remains that artificial reasoning of the type and complexity as presented in this monograph is not particularly suited anymore for human understanding.

APPENDIX 1: SOME TOOLS FOR REASONING ABOUT COMPUTATION MODELS

Definition A.1.1 below imposes an algebraic structure upon the set of computation models relative to some initial interpretation σ_0 and some declaration scheme D, thus making this set into an *algebra*. Next we propose an alternative to our method of defining the operational interpretation of a program scheme, an alternative which captures the whole structure of the computations involved in executing a statement scheme. Then we prove that certain transformations essential to the proofs of lemma 2.5, 2.6 and 2.7 are *morphisms* with respect to the algebra of computation models. These lemmas then follow as simple corollaries of this fact.

DEFINITION A.1.1. Let σ_0 be some initial interpretation and let D be some declaration scheme. Then we define the following (partial) operations between computation models, where it is understood that all computation models involved are computation models relative to σ_0 and D:

a. Let $CM_1 = \langle x_1\ V_1\ x_2\ V_2 \cdots x_n\ V_n\ x_{n+1},\ CM_1\rangle$ be a computation model for $x_1\ V_1^{\eta,\theta}\ x_{n+1}$ with $V_1 \in A \cup C \cup X \cup P$, let $CM_2 = \langle y_1\ W_1\ y_2\ W_2 \cdots y_m\ W_m\ y_{m+1},\ CM_2\rangle$ be a computation model for $y_1\ W_1^{\theta,\zeta}\ y_{m+1}$, and let $x_{n+1} = y_1$, then the computation model $CM_1;CM_2$ is defined by

$$CM_1;CM_2 = \langle x_1\ V_1;W_1\ x_2\ V_2;W_1 \cdots x_n\ V_n;W_1\ x_{n+1}\ W_1\ y_2\ W_2 \cdots y_m\ W_m\ y_{m+1},\ CM_1 \cup CM_2\rangle.$$

b. Let $CM_1 = \langle x_1\ V_1\ x_2\ V_2 \cdots x_n\ V_n\ x_{n+1},\ CM_1\rangle$ be a computation model for $x_1\ V_1^{\eta,\theta}\ x_{n+1}$ with $V_1 = V';V''$ for some statement schemes V' and V'', let $CM_2 = \langle y_1\ W_1\ y_2\ W_2 \cdots y_m\ W_m\ y_{m+1},\ CM_2\rangle$ be a computation model for $y_1\ W_1^{\theta,\zeta}\ y_{m+1}$, and let $x_{n+1} = y_1$, then the computation model $(CM_1);CM_2$ is defined by

$$(CM_1);CM_2 = \langle x_1\ (V_1);W_1\ y_1\ W_1 \cdots y_m\ W_m\ y_{m+1},\ \{CM_1\} \cup CM_2\rangle.$$

c. Let $CM = \langle x_1\ V_1\ x_2\ V_2 \cdots x_n\ V_n\ x_{n+1},\ CM\rangle$ be a computation model for $x_1\ V_1^{\eta,\theta}\ x_{n+1}$, let $W^{\eta,\theta}$ be an arbitrary statement scheme, and let $p^{\eta,\eta}$ be a predicate symbol. If

(1) $\sigma_0(p) =$ <u>true</u>, then the computation model $(\sigma_0(p) \rightarrow CM,W)$ is defined by $(\sigma_0(p) \rightarrow CM,W) = \langle x_1(p \rightarrow V_1,W)\ x_1\ V_1 \cdots x_n\ V_n\ x_{n+1},\ CM\rangle$,

(2) $\sigma_0(p) =$ <u>false</u>, then the computation model $(\sigma_0(p) \rightarrow W,CM)$ is defined by $(\sigma_0(p) \rightarrow W,CM) = \langle x_1(p \rightarrow W,V_1)\ x_1\ V_1 \cdots x_n\ V_n\ x_{n+1},\ CM\rangle$.

d. Let for $j = 1,\ldots,n$, $CM_j = \langle x_{j,1}\ V_{j,1}\ x_{j,2}\ V_{j,2} \cdots x_{j,mj}\ V_{j,mj}\ x_{j,mj+1},\ CM_j\rangle$ be computation models for $x_{j,1}\ V_{j,1}^{\eta,\theta_j}\ x_{j,m_j+1}$, and let $x_{1,1} = \ldots = x_{n,1}$,

then the computation model $[CM_1,\ldots,CM_n]$ is defined by
$[CM_1,\ldots,CM_n] = \langle x_{1,1}[V_{1,1},\ldots,V_{n,1}]\langle x_{1,m1+1},\ldots,x_{n,mn+1}\rangle,\{CM_1,\ldots,CM_n\}\rangle$.

Remark. With definition A.1.1 in mind, one may conceive of the following notion of operational interpretation, which differs from the one defined in def. 2.5:

> The operational interpretation $\psi_D\langle S\rangle(\mathcal{O}_0)$ of a statement scheme S relative to the initial interpretation $\mathcal{O}_0$ and the declaration scheme D is the set
> $\{CM \mid \exists x,y[CM \text{ is, relative } \mathcal{O}_0 \text{ and } D, \text{ a computation model for } x\ S\ y]\}$.

This definition captures the whole structure of the computations involved in executing S and resembles the method of defining the semantics of *MU* as given in def. 3.3, in that both ψ_D and $\psi_D\langle S\rangle$ are conceived of as functions. Definition 2.5 of the operational interpretation $\mathcal{O}(S)$ of a statement scheme S relative to $\mathcal{O}_0$ and D can be recovered from $\psi_D\langle S\rangle(\mathcal{O}_0)$ by *forgetting* the *internal* structure of the computation models constituting $\psi_D\langle S\rangle(\mathcal{O}_0)$ and *preserving* the *external* input-output relationship of these models.
After defining the appropriate operations one can establish results such as:

$$\psi_D\langle S_1;S_2\rangle(\mathcal{O}_0) = \psi_D\langle S_1\rangle(\mathcal{O}_0);\psi_D\langle S_2\rangle(\mathcal{O}_0)$$
$$\psi_D\langle (S_1;S_2);S_3\rangle(\mathcal{O}_0) = (\psi_D\langle S_1;S_2\rangle(\mathcal{O}_0));\psi_D\langle S_3\rangle(\mathcal{O}_0)$$
$$\psi_D\langle (p \to S_1,S_2)\rangle(\mathcal{O}_0) = (\mathcal{O}_0(p) \to \psi_D\langle S_1\rangle(\mathcal{O}_0),S_2) \cup (\mathcal{O}_0(p) \to S_1,\psi_D\langle S_2\rangle(\mathcal{O}_0))$$
$$\psi_D\langle [S_1,\ldots,S_n]\rangle(\mathcal{O}_0) = [\psi_D\langle S_1\rangle(\mathcal{O}_0),\ldots,\psi_D\langle S_n\rangle(\mathcal{O}_0)],$$

from which the proofs of parts b, c and d of lemma 2.1 can be derived.

Let us now analyse how the notions "to identify" and "executable occurrence", defined in def. 2.6, relate to this way of structuring computation models:

a. $CM = CM_1;CM_2$:

$$CM_1 = \langle \underbrace{x_1\ V_1\ x_2\ V_2\ \ldots\ x_n\ V_n\ x_{n+1}}_{cs_1},\ \mathcal{CM}_1\rangle,$$

$$CM_2 = \langle \underbrace{y_1\ W_1\ y_2\ W_2\ \ldots\ y_m\ W_m\ y_{m+1}}_{cs_2},\ \mathcal{CM}_2\rangle,\ x_{n+1} = y_1 \text{ and}$$

$$CM = \langle \underbrace{x_1\ V_1;W_1\ x_2\ V_2;W_1\ \ldots\ x_n\ V_n;W_1\ x_{n+1}}_{cs_1^*}\ \underbrace{W_1\ y_2\ W_2\ \ldots\ y_m\ W_m\ y_{m+1}}_{cs_2^*},\ \mathcal{CM}_1 \cup \mathcal{CM}_2\rangle.$$

It follows from the definitions that

(1) Two occurrences of some procedure symbol, which are both contained in CM_i, identify each other w.r.t CM_i iff the corresponding occurences in CM, i.e., in cs_i^* or CM_i, identify each other w.r.t. CM, i = 1,2; an occurrence of some procedure symbol contained in W_1 identifies also the corresponding occurrences of this symbol in the n copies of W_1 contained in cs_1^*.

(2) An occurrence of some procedure symbol contained in CM_i is executable w.r.t. CM_i iff the corresponding occurrence in cs_i^* or CM_i is executable, i = 1,2; these are the only executable occurrences.

b. $CM = (CM_1);CM_2$:

$CM_1 = \langle x_1\ V_1\ x_2\ V_2 \cdots x_n\ V_n\ x_{n+1},\ CM_1\rangle$, $V_1 = V;W$ for some statement schemes V and W,

$CM_2 = \langle y_1\ W_1\ y_2\ W_2 \cdots y_m\ W_m\ y_{m+1},\ CM_2\rangle$, $x_{n+1} = y_1$ and

$CM = \langle x_1\ (V_1);W_1\ y_1\ W_1 \cdots y_m\ W_m\ y_{m+1},\ \{CM_1\} \cup CM_2\rangle$.

It follows from the definitions that

(1) Two occurrences of some procedure symbol, which are both contained in CM_1 (or CM_2) identify each other w.r.t. CM_1 (or CM_2) iff these occurrences (or, the corresponding occurrences contained in cs_2^* or CM_2) identify each other w.r.t. CM; an occurrence of some procedure symbol contained in V_1 or W_1 also identifies the corresponding occurrence of this symbol in $(V_1);W_1$.

(2) An occurrence of some procedure symbol contained in CM_1 (or CM_2) is executable w.r.t. CM_1 (or CM_2) iff this occurrence as contained in CM (or, its corresponding occurrence in cs_2^* or CM_2) is executable w.r.t. CM; these are the only executable occurrences.

c. $CM = (o_0(p) \to CM_1,V_2)$ (the case $CM = (o_0(p) \to V_1,CM_2)$ is similar):

$CM_1 = \langle y_1\ W_1\ y_2\ W_2 \cdots y_n\ W_n\ y_{n+1},\ CM_1\rangle$,

$CM = \langle x\ (p \to W_1,W_2)\ y_1\ W_1 \cdots y_n\ W_n\ y_{n+1},\ CM_1\rangle$ and $x = y_1$.

It follows from the definitions that

(1) Two occurrences of some procedure symbol which are both contained in CM_1 identify each other w.r.t. CM_1 iff the corresponding occurrences in cs_1^* or CM_1 identify each other w.r.t. CM; an occurrence of some procedure symbol in W_1 identifies also the corresponding occurrence of this symbol in $(p \to W_1, V_2)$.

(2) An occurrence of some procedure symbol contained in CM_1 is executable w.r.t. CM_1 iff its corresponding occurrence in cs_1^* or CM_1 is executable w.r.t. CM; these are the only executable occurrences.

d. $CM = [CM_1, \ldots, CM_n]$:

$CM_j = \langle x_{j,1}\ V_{j,1}\ x_{j,2}\ V_{j,2} \cdots x_{j,mj}\ V_{j,mj}\ x_{j,mj+1},\ CM_j \rangle$, $j = 1,\ldots,n$,

$CM = \langle x_1 [V_{1,1},\ldots,V_{n,1}] \langle x_{1,m1+1},\ldots,x_{n,mn+1} \rangle, \{CM_1,\ldots,CM_n\} \rangle$

and $x_1 = x_{j,1}$, $j = 1,\ldots,n$.

It follows from the definitions that

(1) Two occurrences of some procedure symbol both contained in CM_j identify each other w.r.t. CM_j iff they identify each other w.r.t. CM, $j = 1,\ldots,n$; an occurrence of some procedure symbol contained in $V_{j,1}$ as occurring in $[V_{1,1},\ldots,V_{n,1}]$ also identifies the corresponding occurrence of this symbol contained in CM_j, $j = 1,\ldots,n$.

(2) An occurrence of some procedure symbol contained in CM_j is executable w.r.t. CM_j iff it is executable w.r.t. CM, $j = 1,\ldots,n$; these are the only executable occurrences.

Next we define two transformations of computation models, t_1 and t_2, which are essential to the proofs of lemmas 2.5 and 2.6:

In the following definition $x_1\ V_1\ x_2\ V_2 \cdots x_n\ V_n\ x_{n+1}$ stands for the constituent computation sequence of any model CM.

Let CM contain no executable occurrences of any P_j, $j \in J$, and $W_j \in SS$ be for every $j \in J$ of the same type as P_j, then $t_1(CM)$ is obtained from CM by executing the following steps:

Step 1: Consider for every $j \in J$ *all occurrences of* P_j *in* CM *identified by occurrences of* P_j *in* V_1.

Step 2: Replace all considered occurrences by W_j, *for all* $j \in J$.

For arbitrary CM, $t_2(CM)$ is obtained from CM by executing the following steps:

Step 1: Consider for every $j \in J$ *all occurrences of* P_j *in* CM *identified by occurrences of* P_j *in* V_1.

Step 2: Mark all those considered occurrences which are executable.

Step 3: Replace all other considered occurrences of P_j *by* S_j *(with* $P_j \Longleftarrow S_j$*).*

Step 4: Replace every combination $\ldots\ x_k\ P_j^*\ x_{k+1}\ S_j\ x_{k+2}\ \ldots$ *by* $\ldots\ x_k\ S_j\ x_{k+2}\ \ldots$ *and every combination* $x_k\ P_j^*;S\ x_{k+1}\ S_j;S\ x_{k+2}\ \ldots\ \ldots$ *by* $\ldots\ x_k\ S_j;S\ x_{k+2}\ \ldots$, *where* P_j^* *denotes the marking of* P_j *performed in step 2.*

Transformations t_1 and t_2 are *morphisms* w.r.t. the operations defined above (in def. A.1.1), i.e.,

(1) $t_1(CM_1;CM_2) = t_1(CM_1);t_1(CM_2)$,

$t_1((CM_1);CM_2) = (t_1(CM_1));t_1(CM_2)$,

$t_1((o_0(p) \to CM,W)) = (o_0(p) \to t_1(CM),\tilde{W}[W_j/X_j]_{j\in J})$, *)

$t_1((o_0(p) \to W,CM)) = (o_0(p) \to \tilde{W}[W_j/X_j]_{j\in J},t_1(CM))$ *) and

$t_1([CM_1,\ldots,CM_n]) = [t_1(CM_1),\ldots,t_1(CM_n)]$,

(2) $t_2(CM_1;CM_2) = t_2(CM_1);t_2(CM_2)$,

$t_2((CM_1);CM_2) = (t_2(CM_1));t_2(CM_2)$,

$t_2((o_0(p) \to CM,W)) = (o_0(p) \to t_2(CM),W^{[1]})$, *)

$t_2((o_0(p) \to W,CM)) = (o_0(p) \to W^{[1]},t_2(CM))$ *) and

$t_2([CM_1,\ldots,CM_n]) = [t_2(CM_1),\ldots,t_2(CM_n)]$.

LEMMA 2.5*. *Let* S *be a closed statement scheme,* CM *be a computation model for* x S y *containing no executable occurrences of* P_j, $j \in J$, *and* $W_j \in SS$ *be for every* $j \in J$ *of the same type as* P_j, *then transformation* t_1 *is a morphism (in the sense indicated above) of the algebra of computation models (defined in def. A.1.1) into itself, which transforms* CM *into a computation model for* $\tilde{S}[W_j/X_j]_{j\in J}$.

---

*) These formulae hold only in case W is closed.

Proof. By induction on the complexity of the statement schemes concerned. We use the notation indicated above in our analysis of the notion "to identify".

a. $S = R$, $R \in A \cup C$ ($R \in X$ does not apply, S being closed): Obvious from definitions 2.2 and 2.6.

b. $S = P_j$: Does not apply as CM contains no executable occurrences of P_j.

c. $S = V_1;W_1$: Step 1 of t_1 results in considering for all $j \in J$ those occurrences of P_j in CM which are identified by occurrences of P_j in $V_1;W_1$. These occurrences are:

   (1) The occurrences of P_j in CM identified by occurrences of P_j in V_1. These correspond exactly with the occurrences of P_j in CM_1 identified by occurrences of P_j in V_1 in CM_1.

   (2) The occurrences of P_j in CM identified by occurrences of P_j in W_1 as contained in $V_1;W_1$. These are:

      (2a) The occurrences of P_j in CM corresponding with the occurrences of P_j in CM_2 identified by occurrences of P_j in W_1 in CM_2.

      (2b) The remaining occurrences of P_j in cs_1^* identified by occurrences of P_j in W_1 as contained in $V_1;W_1$.

   Then step 2 is performed; the occurrences of group 1 above are replaced by W_j - this corresponds exactly with $t_1(CM_1)$ - then the occurrences of group 2a are replaced by W_j - this corresponds exactly with $t_1(CM_2)$ - and finally the occurrences of group 2b are replaced by W_j - corresponding exactly with the extra occurrences of $\tilde{W}_1[W_j/X_j]_{j\in J}^{*)}$ necessary for the construction of $t_1(CM_1);t_1(CM_2)$ from $t_1(CM_1)$ and $t_1(CM_2)$.
   It follows that $t_1(CM) = t_1(CM_1);t_1(CM_2)$.
   By the induction hypothesis $t_1(CM_1)$ and $t_1(CM_2)$ are computation models for $x\ \tilde{V}_1[W_j/X_j]_{j\in J}\ z$ and $z\ \tilde{W}_1[W_j/X_j]_{j\in J}\ y$ for appropriate z, whence, by definitions 2.2 and 2.6, $t_1(CM)$ is a computation model for $\widetilde{(V_1;W_1)}[W_j/X_j]_{j\in J}$.

d. $S = (V_1);W_1$: Step 1 of t_1 results in considering for all $j \in J$ those occurrences of P_j in CM which are identified by occurrences of P_j in $(V_1);W_1$. These are:

   (1) The occurrences of P_j in CM_1 identified by occurrences of P_j in V_1.

   (2) The occurrences of P_j in cs_2^* or CM_2 identified by occurrences of P_j in W_1 - these correspond exactly with the occurrences of P_j in CM_2

identified by occurrences of P_j in W_1 in CM_2.

(3) The occurrences of P_j in $(V_1);W_1$.

Then step 2 is applied; the occurrences of group 1 above are replaced by W_j - this corresponds exactly with $t_1(CM_1)$ - then the occurrences of group 2 are replaced by W_j - this corresponds exactly with $t_1(CM_2)$ - and finally the occurrences of group 3 are replaced by W_j - corresponding exactly with the occurrence of $\widetilde{((V_1);W_1)}[W_j/X_j]^{*)}_{j\in J}$ necessary for the construction of $(t_1(CM_1));t_1(CM_2)$ from $t_1(CM_1)$ and $t_1(CM_2)$.

It follows that $t_1(CM) = (t_1(CM_1));t_1(CM_2)$.

By the induction hypothesis $t_1(CM_1)$ and $t_1(CM_2)$ are computation models for $x\ \tilde{V}_1[W_j/X_j]_{j\in J}\ z$ and $z\ \tilde{W}_1[W_j/X_j]_{j\in J}\ y$ for appropriate z, whence, by definitions 2.2 and 2.6, $t_1(CM)$ is a computation model for $\widetilde{((V_1);W_1)}[W_j/X_j]_{j\in J}$.

e. $S = (p \to V_1,V_2)$ or $S = [V_1,\ldots,V_n]$: Similar to above. □

COROLLARY: LEMMA 2.5.

LEMMA 2.6*. *Let* S *be a closed statement scheme and* CM *be a computation model for* x S y, *then* t_2 *is a morphism (in the sense indicated above) of the algebra of computation models (defined in definition A.1.1) into itself, which transforms* CM *into a computation model for* $x\ S^{[1]}\ y$.

Proof. By induction on the complexity of CM.

We use the notation indicated in our analysis of the notions "to identify" and "executable occurrence".

a. $S = R$, $R \in A \cup C$ ($R \in X$ does not apply, S being closed): Obvious from definitions 2.2 and 2.6.

b. $S = P_j$: CM has the following form: $\overset{\longleftarrow\ cs'\longrightarrow}{\langle x\ P_j\ x\ S_j\ \ldots\ y,\ CM\rangle}$.

Thus $t_2(CM) = \langle cs', CM\rangle$, as in step 1 only the first occurrence of P_j is considered, which is executable, whence in step 2 this occurrence is marked, step 3 does not apply, and step 4 results in the deletion of the part P_j^* x.

c. $S = V_1;W_1$: Step 1 of t_2 results in considering for all $j \in J$ those occurrences of P_j in CM which are identified by occurrences of P_j in $V_1;W_1$.

*) The reader should not be confused in case $1 \in J$.

These occurrences are:

(1) The occurrences of P_j in CM identified by occurrences of P_j in V_1. These correspond exactly with the occurrences of P_j in CM_1 identified by occurrences of P_j in V_1 in CM_1.

(2) The occurrences of P_j in CM identified by occurrences of P_j in W_1 as contained in $V_1;W_1$. These are:

(2a) The occurrences of P_j in CM corresponding with the occurrences of P_j in CM_2 identified by occurrences of P_j in W_1 in CM_2.

(2b) The remaining occurrences of P_j in cs_1^* identified by occurrences of P_j in W_1 as contained in $V_1;W_1$, which are all non-executable.

Next step 2 is performed: the executable occurrences of groups 1 and 2a above are marked, group 2b containing no executable occurrences.
Hence we obtain

$$\langle x_1\ V_1^*;W_1\ x_2\ V_2^*;W_1\ \cdots\ x_n\ V_n^*;W_1\ x_{n+1}\ W_1^*\ y_2\ W_2^*\ \cdots\ y_m\ W_m^*\ y_{m+1},\ CM_1^* \cup CM_2^* \rangle,$$

with V_k^*, W_1^* and CM_i^* indicating the result of marking the executable occurrences of P_j in V_k, W_1 and CM_i, $k = 1,\ldots,n$, $1 = 1,\ldots,m$, $i = 1,2$, which are considered in step 1.
Then step 3 is performed, whence we obtain

$$\langle x_1\ \underbrace{V_1^*[S_j/P_j]_{j\in J};W_1^{[1]}\ x_2\ V_2^*[S_j/P_j]_{j\in J};W_1^{[1]}\ \cdots\ x_n\ V_n^*[S_j/P_j]_{j\in J};W_1^{[1]}}_{cs_1^{**}}\ \cdots$$

$$\cdots\ \underbrace{x_{n+1}\ W_1^*[S_j/P_j]_{j\in J}\ y_2\ W_2^*[S_j/P_j]_{j\in J}\ \cdots\ y_m\ W_m^*[S_j/P_j]_{j\in J}\ y_{m+1}}_{cs_2^{**}},\ CM_1^{**} \cup CM_2^{**} \rangle,$$

with $V_k^*[S_j/P_j]_{j\in J}$, $W_1^*[S_j/P_j]_{j\in J}$ and CM_i^{**} indicating the result of replacing the non-executable (unmarked) occurrences of P_j considered in step 1 by S_j, in V_k^*, W_1^* and CM_i^*, $k = 1,\ldots,n$, $1 = 1,\ldots,m$, $i = 1,2$.
The problem with the construct obtained in step 3 is that parts occur of the form $\ldots\ z_1\ V;S_j\ z_{1+1}\ P_j^*\ z_{1+2}\ S_j\ \ldots$, violating definition 2.4 of computation model (e.g., if $V_1 = W_1 = P_j$, then $W_1^{[1]} = S_j$ but $W_1^*[S_j/P_j]_{j\in J} = P_j^*$).
In step 4 these parts are deleted in order to obtain a proper computation model.
Finally step 4 is performed:
Application of this step to cs_1^{**} and CM_1^{**} results in

$x_{i_1}\ V^*_{i_1}[S_j/P_j]_{j\in J};W^{[1]}\ x_{i_2}\ V^*_{i_2}[S_j/P_j]_{j\in J};W^{[1]} \cdots x_{i_s}\ V^*_{i_s}[S_j/P_j]_{j\in J};W^{[1]}\ x_{i_s+1}$

and CM'_1,

with

$t_2(CM_1) = \langle x_{i_1}\ V^*_{i_1}[S_j/P_j]_{j\in J}\ x_{i_2}\ V^*_{i_2}[S_j/P_j]_{j\in J} \cdots x_{i_s}\ V^*_{i_s}[S_j/P_j]_{j\in J}\ x_{i_s+1}, CM'_1\rangle$

by the induction hypothesis, whence $V^*_{i_1}[S_j/P_j]_{j\in J} = V_1^{[1]}$, $x_{i_1} = x$ and $x_{i_s} = x_{n+1}$, as the set of indices k for which parts $V^*_k[S_j/P_j]_{j\in J};W_1^{[1]}\ x_{k+1}$ are deleted from cs_1^{**} is the *same set* as the set of indices k for which parts $V^*_k[S_j/P_j]_{j\in J}\ x_{k+1}$ are deleted from $x_1\ V_1^*[S_j/P_j]_{j\in J}\ x_2\ V_2^*[S_j/P_j]_{j\in J} \cdots x_n\ V_n^*[S_j/P_j]_{j\in J}\ x_{n+1}$, the result of applying steps 1, 2 and 3 to cs_1.

Application of step 4 to cs_2^{**} and CM_2^{**} results by the induction hypothesis in

$y_{j_1}\ W^*_{j_1}[S_j/P_j]_{j\in J}\ y_{j_2}\ W^*_{j_2}[S_j/P_j]_{j\in J} \cdots y_{j_t}\ W^*_{j_t}[S_j/P_j]_{j\in J}\ y_{j_t+1}$ and CM'_2,

the two constituent parts of $t_2(CM_2)$, whence $y_{j_1} = x_{n+1}$, $y_{j_t+1} = y$ and $W^{[1]}_{j_1} = W^{[1]}_1$. Thus we conclude that $t_2(CM) = t_2(CM_1);t_2(CM_2)$. As $V_1^{[1]};W_1^{[1]} = (V_1;W_1)^{[1]}$ by definitions 2.2 and 2.6, $t_2(CM)$ is a computation model for $x\ S^{[1]}\ y$.

d. $S = (V_1;V_2);V_3$, $(p \to V_1,V_2)$ or $[V_1,\ldots,V_n]$: Proved similarly. □

COROLLARY: LEMMA 2.6: *Let* CM *be a computation model for* x S y, *with* S *closed and with constituent sequence* $x_1\ V_1\ x_2\ V_2 \cdots x_n\ V_n\ x_{n+1}$. *If for some* $j \in J$ *at least one occurrence of* P_j *in* V_1 *identifies an executable occurrence of* P_j, $t_2(CM)$ *is a computation model for* $x\ S^{[1]}\ y$ *which contains at least one executable occurrence of* P_j *less than* CM.

Proof. Follows from lemma 2.6* by a simple induction argument, as t_2 is a morphism. □

LEMMA 2.7. *Let* CM *be a computation model for* x S y *and* S *be closed. Then there exists for some* k *a computation model for* $x\ S^{(k)}\ y$.

Proof. By applying lemma 2.6 n times in succession one obtains a computation model for x $S^{[n]}$ y; this follows from lemma 2.4 ($S^{[m][1]} = S^{[m+1]}$) and the fact that, if $S^{[m]}$ is closed, $S^{[m+1]}$ is also closed.
Let 1 be the smallest number such that $S^{[1]}$ contains no executable occurrences of P_j. This number exists as every application of lemma 2.6 decreases the number of executable occurrences of P_j, if any. Then the conditions of lemma 2.5 are satisfied, whence some computation model for x $\widetilde{S^{[1]}}[\Omega_j/X_j]_{j\in J}$ y exists.
As by lemma 2.4 $\widetilde{S^{[1]}}[\Omega_j/X_j]_{j\in J} = S^{(1+1)}$, it suffices to take 1+1 for k. □

APPENDIX 2: PROOFS OF MONOTONICITY, CONTINUITY AND SUBSTITUTIVITY FOR *MU*

LEMMA 3.1. (Monotonicity). *Let* J *be any index set,* $X_j \in X$ *for all* $j \in J$, $\sigma \in T$ *be syntactically continuous in all* X_j, $j \in J$, *and variable valuations* v_1 *and* v_2 *satisfy*

(1) $v_1(X_j) \subseteq v_2(X_j)$, $j \in J$,

(2) $v_1(X) = v_2(X)$, $X \in X - \{X_j\}_{j\in J}$,

then the following holds:

$$\phi\langle\sigma\rangle(v_1) \subseteq \phi\langle\sigma\rangle(v_2).$$

Proof. By induction on the complexity of σ.

a. $\sigma \in A \cup B \cup C \cup X$: Obvious.

b. $\sigma = \sigma_1;\sigma_2$, $\sigma_1 \cup \sigma_2$, $\sigma_1 \cap \sigma_2$, $\breve{\sigma}_1$:
$\phi\langle\sigma_1;\sigma_2\rangle(v_1) = \phi\langle\sigma_1\rangle(v_1);\phi\langle\sigma_2\rangle(v_1)$ and $\langle x,y\rangle \in \phi\langle\sigma_1\rangle(v_1);\phi\langle\sigma_2\rangle(v_1)$ iff
$\exists z[\langle x,z\rangle \in \phi\langle\sigma_1\rangle(v_1)$ and $\langle z,y\rangle \in \phi\langle\sigma_2\rangle(v_1)]$.
By the induction hypothesis, $\phi\langle\sigma_i\rangle(v_1) \subseteq \phi\langle\sigma_i\rangle(v_2)$, $i = 1,2$.
Thus $\langle x,y\rangle \in \phi\langle\sigma_1\rangle(v_1);\phi\langle\sigma_2\rangle(v_1)$ implies $\langle x,y\rangle \in \phi\langle\sigma_1\rangle(v_2);\phi\langle\sigma_2\rangle(v_2)$,
whence $\phi\langle\sigma_1;\sigma_2\rangle(v_1) \subseteq \phi\langle\sigma_1;\sigma_2\rangle(v_2)$ follows from the definitions.
The cases $\sigma = \sigma_1 \cup \sigma_2$, $\sigma_1 \cap \sigma_2$ and $\breve{\sigma}_1$ are proved similarly.

c. $\sigma = \overline{\sigma_1}$: By syntactic continuity of σ in all X_j, $j \in J$, no X_j occurs in σ_1 for any $j \in J$, whence $\phi\langle\sigma_1\rangle(v_1) = \phi\langle\sigma_1\rangle(v_2)$.
Therefore $\phi\langle\overline{\sigma_1}\rangle(v_1) = \overline{\phi\langle\sigma_1\rangle(v_1)} = \overline{\phi\langle\sigma_1\rangle(v_2)} = \phi\langle\overline{\sigma_1}\rangle(v_2)$.

d. $\sigma = \mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]$:

$$\phi\langle\sigma\rangle(v_2) =$$

$$(\cap\{\langle v_2'(X_1)\rangle_{1=1}^{n} \mid \phi\langle\sigma_1\rangle(v_2') \subseteq v_2'(X_1),\ 1 = 1,\ldots,n, \text{ and}$$

$$v_2'(X) = v_2(X),\ X \in X - \{X_1,\ldots,X_n\}\})_k \quad \ldots \text{ (a.2.1)}$$

Let v_2' satisfy the conditions mentioned in (a.2.1).
Define v_1' by: $v_1'(X_1) = v_2'(X_1)$, $1 = 1,\ldots,n$, and $v_1'(X) = v_1(X)$,
$X \in X - \{X_1,\ldots,X_n\}$.
Then the conditions for monotonicity, w.r.t. the index set $J \cup \{1,\ldots,n\}$, and v_1' and v_2', are fulfilled, whence by the induction hypothesis:

$$\phi\langle\sigma_1\rangle(v_1') \subseteq \text{(monotonicity)}\ \phi\langle\sigma_1\rangle(v_2') \subseteq v_2'(X_1) = v_1'(X_1), \quad 1 = 1,\ldots,n.$$

Thus,

$$\cap\{\langle v_1'(X_1)\rangle_{1=1}^{n} \mid \phi\langle\sigma_1\rangle(v_1') \subseteq v_1'(X_1),\ 1 = 1,\ldots,n, \text{ and } v_1'(X) = v_1(X),\ X \in X - \{X_1,\ldots,X_n\}\} \subseteq$$

$$\subseteq\cap\{\langle v_2'(X_1)\rangle_{1=1}^{n} \mid \phi\langle\sigma_1\rangle(v_2') \subseteq v_2'(X_1),\ 1 = 1,\ldots,n, \text{ and } v_2'(X) = v_2(X),\ X \in X - \{X_1,\ldots,X_n\}\},$$

whence

$$\phi\langle\mu_k X_1\ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v_1) \subseteq \phi\langle\mu_k X_1\ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v_2). \quad \square$$

LEMMA 3.2. (Continuity). *Let* J *be any index set,* $X_j \in X$ *for all* $j \in J$, $\sigma \in T$ *be syntactically continuous in all* X_j, $j \in J$, v *and* v_i $(i \in N)$ *be variable valuations satisfying, for* $i \in N$ *and* $j \in J$,

(1) $v(X_j) = \bigcup_{i=0}^{\infty} v_i(X_j)$,

(2) $v_i(X_j) \subseteq v_{i+1}(X_j)$,

(3) $v(X) = v_i(X)$ for $X \in X - \{X_j\}_{j\in J}$,

then the following holds:

$$\phi\langle\sigma\rangle(v) = \bigcup_{i=0}^{\infty} \phi\langle\sigma\rangle(v_i).$$

Proof. $\supseteq$: By monotonicity (lemma 3.1).

$\subseteq$: By induction on the complexity of σ.

a. $\sigma \in A \cup B \cup C \cup X$: Obvious.

b. $\sigma = \sigma_1;\sigma_2, \sigma_1 \cup \sigma_2, \sigma_1 \cap \sigma_2, \breve{\sigma}_1$:

$\phi\langle\sigma_1;\sigma_2\rangle(v) = \phi\langle\sigma_1\rangle(v);\phi\langle\sigma_2\rangle(v) \subseteq$ (induction hypothesis)

$\bigcup_{i=0}^{\infty} \phi\langle\sigma_1\rangle(v_i);\bigcup_{j=0}^{\infty} \phi\langle\sigma_2\rangle(v_j) = \underbrace{\bigcup_{i=0}^{\infty}\bigcup_{j=0}^{\infty} \phi\langle\sigma_1\rangle(v_i);\phi\langle\sigma_2\rangle(v_j)}_{E_1}$,

by a property of relations.

$\bigcup_{i=0}^{\infty} \phi\langle\sigma_1\rangle(v_i);\phi\langle\sigma_2\rangle(v_i) \subseteq E_1$ is obvious and

$\bigcup_{i=0}^{\infty} \phi\langle\sigma_1\rangle(v_i);\phi\langle\sigma_2\rangle(v_i) \supseteq E_1$ follows from

$\phi\langle\sigma_1\rangle(v_i);\phi\langle\sigma_2\rangle(v_j) \subseteq$ (monotonicity) $\phi\langle\sigma_1\rangle(v_{\max(i,j)});\phi\langle\sigma_2\rangle(v_{\max(i,j)})$.

Thus, $\phi\langle\sigma_1;\sigma_2\rangle(v) \subseteq \bigcup_{i=0}^{\infty} \phi\langle\sigma_1;\sigma_2\rangle(v_i)$.

The cases $\sigma = \sigma_1 \cup \sigma_2$, $\sigma_1 \cap \sigma_2$ and $\breve{\sigma}_1$ are proved similarly.

c. $\sigma = \overline{\sigma_1}$: By syntactic continuity of σ in all X_j, $j \in J$, no X_j occurs in σ_1 for any $j \in J$, whence $\phi\langle\sigma_1\rangle(v) = \phi\langle\sigma_1\rangle(v_i)$.
Therefore $\phi\langle\overline{\sigma_1}\rangle(v) = \overline{\phi\langle\sigma_1\rangle(v)} = \overline{\phi\langle\sigma_1\rangle(v_i)} = \phi\langle\overline{\sigma_1}\rangle(v_i)$ for all $i \in N$,
whence $\phi\langle\overline{\sigma_1}\rangle(v) = \bigcup_{i=0}^{\infty} \phi\langle\overline{\sigma_1}\rangle(v_i)$.

d. $\sigma = \mu_k X_1 \ldots X_n[\sigma_1,\ldots,\sigma_n]$:
Define V_i, for all $i \in N$, by

$$V_i = \{v_i' \mid \phi\langle\sigma_1\rangle(v_i') \subseteq v_i'(X_1),\ 1 = 1,\ldots,n, \text{ and}$$
$$v_i'(X) = v_i(X),\ X \in X - \{X_1,\ldots,X_n\}\}.$$

Then $\bigcup_{i=0}^{\infty} \phi\langle\sigma\rangle(v_i) = \bigcup_{i=0}^{\infty} (\cap\{v_i'(X_k) \mid v_i' \in V_i\})$. ... (a.2.2)

Next define valuation v_i^*, for $i \in N$, by:

$v_i^*(X_1) = \cap\{v_i'(X_1) \mid v_i' \in V_i\}$, $1 = 1,\ldots,n$, and $v_i^*(X) = v_i(X)$,
for $X \in X - \{X_1,\ldots,X_n\}$.

Then $\bigcup_{i=0}^{\infty} \phi\langle\sigma\rangle(v_i) = \bigcup_{i=0}^{\infty} v_i^*(X_k)$ follows directly from (a.2.2). ... (a.2.3)

Let E_2 be defined by

$$E_2 = \cap\{\bigcup_{i=0}^{\infty} v_i'(X_k) \mid \text{for all } i \in N,\ v_i' \in V_i\}. \quad \ldots \text{(a.2.4)}$$

First we shall prove that $\bigcup_{i=0}^{\infty} v_i^*(X_k) = E_2$, and then that $\phi\langle\sigma\rangle(v) \subseteq E_2$, whence the result follows from (a.2.3).

$\bigcup_{i=0}^{\infty} v_i^*(X_k) = E_2$:

$\subseteq$: Since for $v_i' \in V_i$ and $i \in N$ $v_i^*(X_k) \subseteq v_i'(X_k)$ holds, we have
$\bigcup_{i=0}^{\infty} v_i^*(X_k) \subseteq \bigcup_{i=0}^{\infty} v_i'(X_k)$; hence
$\bigcup_{i=0}^{\infty} v_i^*(X_k) \subseteq \cap\{\bigcup_{i=0}^{\infty} v_i'(X_k) \mid v_i' \in V_i\} = E_2$ follows.

$\supseteq$: We prove below that $v_i^* \in V_i$, for $i \in N$, whence $E_2 \subseteq \bigcup_{i=0}^{\infty} v_i^*(X_k)$ follows from E_2's definition.
By definition, for $i \in N$, $v_i^*(X_1) \subseteq v_i'(X_1)$, for $1 = 1,\ldots,n$ and $v_i' \in V_i$, and $v_i^*(X) = v_i(X)$, for $X \in X - \{X_1,\ldots,X_n\}$.
Therefore the conditions for applying monotonicity (lemma 3.1) are fulfilled, whence $\phi\langle\sigma_1\rangle(v_i^*) \subseteq \phi\langle\sigma_1\rangle(v_i') \subseteq v_i'(X_1)$ for all $v_i' \in V_i$, and $\phi\langle\sigma_1\rangle(v_i^*) \subseteq \bigcap_{v_i' \in V_i} v_i'(X_1) = v_i^*(X_1)$, $1 = 1,\ldots,n$.

Moreover, $v_i^*(X) = v_i(X)$ for $X \in X - \{X_1,\ldots,X_n\}$. Hence $v_i^* \in V_i$ follows, for $i \in N$.

$\phi\langle\sigma\rangle(v) \subseteq E_2$:

First we demonstrate that one can restrict oneself in (a.2.4) to intersections of unions of $v_i'(X_1)$ such that $v_i'(X_1) \subseteq v_{i+1}'(X_1)$, $1 = 1,\ldots,n$:
Let $\langle v_i'\rangle_{i=0}^{\infty}$ be a sequence consisting of valuations which satisfy for every $i \in N$, $\phi\langle\sigma_1\rangle(v_i') \subseteq v_i'(X_1)$, $1 = 1,\ldots,n$, and $v_i'(X) = v_i(X)$, for $X \in X - \{X_1,\ldots,X_n\}$.
Define $\langle v_i''\rangle_{i=0}^{\infty}$ as follows:
For every $i \in N$, $v_i''(X_1) = \bigcap_{j=i}^{\infty} v_j'(X_1)$, $1 = 1,\ldots,n$, and $v_i''(X) = v_i'(X)$, $X \in X - \{X_1,\ldots,X_n\}$.
This sequence of valuations satisfies the following properties:

1. For every $i \in N$, $\phi\langle\sigma_1\rangle(v_i'') \subseteq v_i''(X_1)$, $1 = 1,\ldots,n$.
   This can be deduced from the fact that, for all $j \geq i$,

   $$\phi\langle\sigma_1\rangle(v_i'') \subseteq \text{(monotonicity)}\ \phi\langle\sigma_1\rangle(v_j') \subseteq v_j'(X_1),\quad 1 = 1,\ldots,n.$$

2. For every $i \in N$, $v_i''(X_1) \subseteq v_{i+1}''(X_1)$, $1 = 1,\ldots,n$.
3. $\bigcup_{i=0}^{\infty} v_i''(X_1) \subseteq \bigcup_{i=0}^{\infty} v_i'(X_1)$, $1 = 1,\ldots,n$.

Therefore, as every n-tuple $\langle \bigcup_{i=0}^{\infty} v_i'(X_1)\rangle_{1=1}^{n}$ with $\langle v_i'\rangle_{i=0}^{\infty}$ satisfying the conditions mentioned above *contains* coordinatewise an n-tuple $\langle \bigcup_{i=0}^{\infty} v_i''(X_1)\rangle_{1=1}^{n}$ with $\langle v_i''\rangle_{i=0}^{\infty}$ also satisfying these conditions, in addition to the extra condition $v_i''(X_1) \subseteq v_{i+1}''(X_1)$, $1 = 1,\ldots,n$, $i \in N$, one can restrict oneself in (a.2.4) to k-th components of intersections of the latter.
Define v'' by $v''(X_1) = \bigcup_{i=0}^{\infty} v_i''(X_1)$, $1 = 1,\ldots,n$, and $v''(X) = v(X)$, $X \in X - \{X_1,\ldots,X_n\}$.
Then the conditions for continuity, w.r.t. the index set $J \cup \{1,\ldots,n\}$, and v'' and $\langle v_i''\rangle_{i=0}^{\infty}$, are fulfilled, whence by the induction hypothesis:

$$\phi\langle\sigma_1\rangle(v'') = \text{(continuity)}\ \bigcup_{i=0}^{\infty} \phi\langle\sigma_1\rangle(v_i'') \subseteq \text{(point 1 above)}$$

$$\bigcup_{i=0}^{\infty} v_i''(X_1) = v''(X_1), \qquad \text{for } 1 = 1,\ldots,n.$$

Hence,

$$\phi\langle\mu_k X_1\ldots X_n[\sigma_1,\ldots,\sigma_n]\rangle(v) =$$

$= (\cap\{<v'(X_1)>_{1=1}^{n} \mid \phi<\sigma_1>(v') \subseteq v'(X_1),\ 1 = 1,\ldots,n,$ and

$v'(X) = v(x),\ X \in X - \{X_1,\ldots,X_n\}\})_k \subseteq$ (from above)

$(\cap\{<\bigcup_{i=0}^{\infty} v_i''(X_1)>_{1=1}^{n} \mid$ for $i \in N$, $\phi<\sigma_1>(v_i'') \subseteq v_i''(X_1),\ 1 = 1,\ldots,n,$

$v_i''(X_1) \subseteq v_{i+1}''(X_1),\ 1 = 1,\ldots,n,$ and $v_i''(X) = v_i(X),\ X \in X - \{X_1,\ldots,X_n\}\})_k =$

$=$ (from above) E_2. □

LEMMA 3.3. (Substitutivity). *Let* J *be any index set,* $\sigma \in T$, $X_j \in X$ *and* $\tau_j \in T$ *be of the same type for* $j \in J$, *and variable valuations* v_1 *and* v_2 *satisfy*

(1) $v_1(X) = v_2(X)$, $X \in X - \{X_j\}_{j\in J}$,

(2) $v_1(X_j) = \phi<\tau_j>(v_2)$, $j \in J$,

then the following holds:

$$\phi<\sigma>(v_1) = \phi<\sigma[\tau_j/X_j]_{j\in J}>(v_2).$$

Proof. By induction on the complexity of σ.

We only consider the case $\sigma = \mu_m X_1\ldots X_n[\sigma_1,\ldots,\sigma_n]$.

By definition,

$$\mu_m X_1\ldots X_n[\sigma_1,\ldots,\sigma_n][\tau_j/X_j]_{j\in J} =$$

$$= \mu_m Y_1\ldots Y_n[\sigma_1[Y_1/X_1]_{1=1,\ldots,n}[\tau_j/X_j]_{j\in J^*},\ldots$$

$$\ldots,\sigma_n[Y_1/X_1]_{1=1,\ldots,n}[\tau_j/X_j]_{j\in J^*}],$$

with $J^* = J - \{1,\ldots,n\}$ and $Y_1,\ldots,Y_n$ relation variables different from X_j, $j \in J$, and not occurring in σ_k, $k = 1,\ldots,n$, or τ_j, $j \in J^*$.

Let

$$E_1 \equiv$$

$$(\cap\{<v_1''(X_k)>_{k=1}^{n} \mid \phi<\sigma_k>(v_1'') \subseteq v_1''(X_k),\ k = 1,\ldots,n, \text{and}$$

$$v_1''(X) = v_1(X),\ X \in X - \{X_1,\ldots,X_n\}\})_m,$$

$$E_2 \equiv$$

$$(\cap\{<v_1'(Y_k)>_{k=1}^{n} \mid \phi<\sigma_k[Y_1/X_1]_{1=1,\ldots,n}>(v_1') \subseteq v_1'(Y_k),\ k = 1,\ldots,n,\ \text{and}$$

$$v_1'(X) = v_1(X),\ X \in \mathcal{X} - \{Y_1,\ldots,Y_n\}\})_m$$

and

$$E_3 \equiv$$

$$(\cap\{\langle v_2'(Y_k)\rangle_{k=1}^n \mid \underbrace{\phi\langle\sigma_k[Y_1/X_1]_{1=1,\ldots,n}[\tau_j/X_j]_{j\in J^*}\rangle(v_2')}_{\sigma_k'} \subseteq v_2'(Y_k),$$

$$k = 1,\ldots,n, \text{ and } v_2'(X) = v_2(X),\ X \in \mathcal{X} - \{Y_1,\ldots,Y_n\}\})_m.$$

In order to prove $\phi\langle\sigma\rangle(v_1) = \phi\langle\sigma[\tau_j/X_j]_{j\in J}\rangle(v_2)$, that is $E_1 = E_3$, we first prove $E_2 = E_3$ and then $E_1 = E_2$:

$E_2 = E_3$:

$\subseteq$: Let v_2' satisfy $v_2'(X) = v_2(X)$, for $X \in \mathcal{X} - \{Y_1,\ldots,Y_n\}$, and $\phi\langle\sigma_k'\rangle(v_2') \subseteq v_2'(Y_k)$, $k = 1,\ldots,n$.
Define v_1' by $v_1'(X) = v_2'(X)$ for $X \in \mathcal{X} - \{X_j\}_{j\in J}$ and $v_1'(X_j) = \phi\langle\tau_j\rangle(v_2')$, for $j \in J$, and define v_1'' by $v_1''(X) = v_2'(X)$ for $X \in \mathcal{X} - \{X_j\}_{j\in J^*}$ and $v_1''(X_j) = \phi\langle\tau_j\rangle(v_2')$, for $j \in J^*$.

By the induction hypothesis, $\phi\langle\sigma_k[Y_1/X_1]_{1=1,\ldots,n}\rangle(v_1'') = \phi\langle\sigma_k'\rangle(v'_2)$.
As $X_1,\ldots,X_n$ do not occur in $\sigma_k[Y_1/X_1]_{1=1,\ldots,n}$, $\phi\langle\sigma_k[Y_1/X_1]_{1=1,\ldots,n}\rangle(v_1'') = \phi\langle\sigma_k[Y_1/X_1]_{1=1,\ldots,n}\rangle(v_1')$.
Moreover $\phi\langle\sigma_k'\rangle(v_2') \subseteq v_2'(Y_k) = v_1'(Y_k)$, $k = 1,\ldots,n$, as $\{X_j\}_{j\in J} \cap \{Y_1,\ldots,Y_n\} = \emptyset$.
Thus $\phi\langle\sigma_k[Y_1/X_1]_{1=1,\ldots,n}\rangle(v_1') \subseteq v_1'(Y_k)$, $k = 1,\ldots,n$.
Furthermore $v_1'(X_j) = \phi\langle\tau_j\rangle(v_2') = (Y_1,\ldots,Y_n$ do not occur in $\tau_j)$ $\phi\langle\tau_j\rangle(v_2) = v_1(X_j)$, $j \in J$, and $v_1'(X) = v_2'(X) = v_2(X) =$ (assumption) $v_1(X)$ for $X \in \mathcal{X} - \{X_j\}_{j\in J} - \{Y_1,\ldots,Y_n\}$, whence v_1' satisfies the conditions mentioned in E_2.

As $\langle v_1'(Y_k)\rangle_{k=1}^n = \langle v_2'(Y_k)\rangle_{k=1}^n$, we obtain $E_2 \subseteq E_3$.

$\supseteq$: Let v_1' satisfy $v_1'(X) = v_1(X)$, $X \in \mathcal{X} - \{Y_1,\ldots,Y_n\}$ and $\phi\langle\sigma_k[Y_1/X_1]_{1=1,\ldots,n}\rangle(v_1') \subseteq v_1'(Y_k)$, $k = 1,\ldots,n$.
Define v_2' by $v_2'(Y_k) = v_1'(Y_k)$, $k = 1,\ldots,n$, and $v_2'(X) = v_2(X)$, otherwise.
Now (1) $v_1'(X_j) = v_1(X_j) = \phi\langle\tau_j\rangle(v_2) = (Y_1,\ldots,Y_n$ do not occur in $\tau_j)$ $\phi\langle\tau_j\rangle(v_2')$, $j \in J$,

(2) $v_1'(X) = v_1(X) = v_2(X) = v_2'(X)$, $X \in X - \{X_j\}_{j\in J} - \{Y_1,\ldots,Y_n\}$, and

(3) $v_1'(Y_k) = v_2'(Y_k)$, $k = 1,\ldots,n$,

imply together that the induction hypothesis may be applied, whence

$$\phi<\sigma_k[Y_1/X_1]_{1=1,\ldots,n}[\tau_j/X_j]_{j\in J}>(v_2') = \phi<\sigma_k[Y_1/X_1]_{1=1,\ldots,n}>(v_1').$$

Since $\sigma_k[Y_1/X_1]_{1=1,\ldots,n}[\tau_j/X_j]_{j\in J} = \sigma_k[Y_1/X_1]_{1=1,\ldots,n}[\tau_j/X_j]_{j\in J}{}^* \equiv \sigma_k'$, as no $X_1,\ldots,X_n$ occur in $\sigma_k[Y_1/X_1]_{1=1,\ldots,n}$,

$$\phi<\sigma_k'>(v_2') = \phi<\sigma_k[Y_1/X_1]_{1=1,\ldots,n}>(v_1') \subseteq v_1'(Y_k) = v_2'(Y_k)$$

follows, $k = 1,\ldots,n$. As $v_2'(X) = v_2(X)$, $X \in X - \{Y_1,\ldots,Y_n\}$, it can be deduced that $E_2 \supseteq E_3$.

$E_1 = E_2$:

$\supseteq$: Let v_1'' satisfy $\phi<\sigma_k>(v_1'') \subseteq v_1''(X_k)$, $k = 1,\ldots,n$, and $v_1''(X) = v_1(X)$, $X \in X - \{X_1,\ldots,X_n\}$.
Define v_1' by $v_1'(Y_k) = v_1''(X_k)$, $k = 1,\ldots,n$, and $v_1'(X) = v_1(X)$, $X \in X - \{Y_1,\ldots,Y_n\}$.
By the induction hypothesis, $\phi<\sigma_k>(v_1'') = \phi<\sigma_k[Y_1/X_1]_{1=1,\ldots,n}>(v_1')$.
Therefore, $\phi<\sigma_k[Y_1/X_1]_{1=1,\ldots,n}>(v_1') = \phi<\sigma_k>(v_1'') \subseteq v_1''(X_k) = v_1'(Y_k)$, $k = 1,\ldots,n$. As $v_1'(X) = v_1(X)$, $X \in X - \{Y_1,\ldots,Y_n\}$, it can be deduced that $E_1 \supseteq E_2$ holds.

$\subseteq$: As $\sigma_k[Y_1/X_1]_{1=1,\ldots,n}[X_1/Y_1]_{1=1,\ldots,n} = \sigma_k$, the proof of this part is similar to the proof above. □

APPENDIX 3: PROOF OF TARSKI'S "UNPROVABLE ASSERTION"

THEOREM.

$$\vdash X_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2 \subseteq$$

$$X_1;(\breve{X}_1;Y_1 \cap X_2;\breve{Y}_2 \cap (\breve{X}_1;Z_1 \cap X_2;\breve{Z}_2);(\breve{Z}_1;Y_1 \cap Z_2;\breve{Y}_2));Y_2.$$

Proof.

A. $[X_1,Y_1,Z_1] = [X_1,Y_1,Z_1];(\pi_1;\breve{X}_1;Y_1;\breve{\pi}_2 \cap E).$

$\supseteq$: trivial.

$\subseteq$: $[X_1,Y_1,Z_1] =$ (lemma 4.3.c) $[X_1,Y_1,Z_1];\ (\overbrace{[X_1,Y_1,Z_1]}\circ E)$

$\equiv [X_1,Y_1,Z_1];\{(\pi_1;\breve{X}_1 \cap \pi_2;\breve{Y}_1 \cap \pi_3;\breve{Z}_1);U \cap E\}$

$\subseteq [X_1,Y_1,Z_1];\{(\pi_1;\breve{X}_1 \cap \pi_2;\breve{Y}_2);U \cap E\}$

$=$ (lemma 4.5.c) $[X_1,Y_1,Z_1];(\pi_1;\breve{X}_1;Y_2;\breve{\pi}_2 \cap E).$

B. Hence, $X_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2 = [X_1,Y_1,Z_1];(\pi_1;X_2 \cap \pi_2;Y_2 \cap \pi_3;Z_2)$

$=$ (by part A) $[X_1,Y_1,Z_1];(\pi_1;\breve{X}_1;Y_1;\breve{\pi}_2 \cap E);(\pi_1;\breve{X}_1;Z_1;\breve{\pi}_3 \cap E);$

$(\pi_3;\breve{Z}_1;Y_1;\breve{\pi}_2 \cap E);(\pi_1;X_2;\breve{Y}_2;\breve{\pi}_2 \cap E);(\pi_1;X_2;\breve{Z}_2;\breve{\pi}_3 \cap E);$

$(\pi_3;Z_2;\breve{Y}_2;\breve{\pi}_2 \cap E);(\pi_1;X_2 \cap \pi_2;Y_2 \cap \pi_3;Z_2)$

$= [X_1,Y_1,Z_1];(\pi_1;\breve{X}_1;Y_1;\breve{\pi}_2 \cap \pi_1;X_2;\breve{Y}_2;\breve{\pi}_2 \cap \pi_1;\breve{X}_1;Z_1;\breve{\pi}_3 \cap \pi_1;X_2;\breve{X}_2;\breve{\pi}_3 \cap$

$\pi_3;\breve{Z}_1;Y_1;\breve{\pi}_2 \cap \pi_3;Z_2;\breve{Y}_2;\breve{\pi}_2 \cap E);(\pi_1;X_2 \cap \pi_2;Y_2 \cap \pi_3;Z_2)$

$=$

$$[X_1,Y_1,Z_1];\overbrace{\{\pi_1;(\breve{X}_1;Y_1 \cap X_2;\breve{Y}_2);\breve{\pi}_2 \cap \pi_1;(\breve{X}_1;Z_1 \cap X_2;\breve{Z}_2);\breve{\pi}_3 \cap \pi_3;(\breve{Z}_1;Y_1 \cap Z_2;\breve{Y}_2);\breve{\pi}_2 \cap E\}}^{E_1};$$

$$(\pi_1;X_2 \cap \pi_2;Y_2 \cap \pi_3;Z_2).$$

C. $E_1 = (\pi_1;(\breve{X}_1;Y_1 \cap X_2;\breve{Y}_2);\pi_2 \cap E);\underbrace{(\pi_1;(\breve{X}_1;Z_1 \cap X_2;\breve{Z}_2);\breve{\pi}_3 \cap E);(\pi_3;(\breve{Z}_1;Y_1 \cap Z_2;\breve{Y}_2);\breve{\pi}_2 \cap E)}_{E_2},$

$E_2 \subseteq \pi_1;(\breve{X}_1;Z_1 \cap X_2;\breve{Y}_2);(\breve{Z}_1;Y_1 \cap Z_2;\breve{Y}_2);\breve{\pi}_2 \cap E.$

Hence,

$$E_1 \subseteq \underbrace{\pi_1;(\breve{X}_1;Y_1 \cap X_2;\breve{Y}_2 \cap (\breve{X}_1;Z_1 \cap X_2;\breve{Z}_2);(\breve{Z}_1;Y_1 \cap Z_2;\breve{Y}_2));\breve{\pi}_2 \cap E.}_{E_3}$$

D. By (B) and (C),

$$X_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2 \subseteq [X_1,Y_1,Z_1];E_3;(\pi_1;X_2 \cap \pi_2;Y_2 \cap \pi_3;Z_2) \subseteq$$

$$[X_1,Y_1,Z_1];[\pi_1,\pi_1;(\check{X}_1;Y_1 \cap X_2;\check{Y}_2 \cap (\check{X}_1;Z_1 \cap X_2;\check{Z}_2);(\check{Z}_1;Y_1 \cap Z_2;\check{Y}_2)),\pi_3];$$

$$(\pi_1;X_2 \cap \pi_2;Y_2 \cap \pi_3;Z_2) =$$

$$[X_1,Y_1,Z_1];[\pi_1;X_2 \cap \pi_1;(\check{X}_1;Y_1 \cap X_2;\check{Y}_2 \cap (\check{X}_1;Z_1 \cap X_2;\check{Z}_2);$$

$$(\check{Z}_1;Y_1 \cap Z_2;\check{Y}_2));Y_2 \cap \pi_3;Z_2)$$

$$\subseteq$$

$$X_1;X_2 \cap X_1;(\check{X}_1;Y_1 \cap X_2;\check{Y}_2 \cap (\check{X}_1;Z_1 \cap X_2;\check{Z}_2);(\check{Z}_1;Y_1 \cap Z_2;\check{Y}_2));Y_2 \cap Z_1;Z_2 \subseteq$$

$$X_1;(\check{X}_1;Y_1 \cap X_2;\check{Y}_2 \cap (\check{X}_1;Z_1 \cap X_2;\check{Z}_2);(\check{Z}_1;Y_1 \cap Z_2;\check{Y}_2));Y_2. \quad \square$$

REFERENCES

[1] BEKIĆ, H., *Definable operations in general algebra, and the theory of automata and flowcharts*, Report IBM Laboratory Vienna, 1969.

[2] BEKIĆ, H., *Towards a mathematical theory of processes*, Technical Report TR 25.125, IBM Laboratory Vienna, 1971.

[3] BLIKLE, A., *An algebraic approach to programs and their computations*, in: Proc. of the Symposium and Summer School on the Mathematical Foundations of Computer Science, High Tatras, Czechoslovakia, 1973.

[4] BLIKLE, A. & A. MAZURKIEWICZ, *An algebraic approach to the theory of programs, algorithms, languages and recursiveness*, in: Proc. of an International Symposium and Summer School on the Mathematical Foundations of Computer Science, Warsaw-Jablonna, 1972.

[5] BURSTALL, R.M., *Proving properties of programs by structural induction*, Comput. J., 12 (1969) 41-48.

[6] BURSTALL, R.M. & J.W. THATCHER, *The algebraic theory of recursive program schemes*, in: Category theory applied to computation and control, E.G. Manes (ed.), University of Massachusetts, 1974.

[7] CADIOU, J.M., *Recursive definitions of partial functions and their computations*, Thesis, Stanford University, 1972.

[8] DE BAKKER, J.W., *Semantics of programming languages*, Advances in Information Systems Science, Vol. 2, Plenum Press, 1969.

[9] DE BAKKER, J.W., *Recursive procedures*, Mathematical Centre Tracts 24, Amsterdam, 1971.

[10] DE BAKKER, J.W., *Recursion, induction and symbol manipulation*, in: Proc. MC-25 Informatica Symposium, Mathematical Centre Tracts 37, Amsterdam, 1971.

[11] DE BAKKER, J.W. & W.P. DE ROEVER, *A calculus for recursive program schemes*, in: Proc. IRIA Symposium on Automata, Formal languages and Programming, M. Nivat (ed.), North-Holland, Amsterdam, 1972.

[12] DE BAKKER, J.W. & L.G.L.T. MEERTENS, *Simple recursive program schemes and inductive assertions*, Mathematical Centre Report MR 142/72, Amsterdam, 1972.

[13] DE BAKKER, J.W. & L.G.L.T. MEERTENS, *On the completeness of the inductive assertion method*, Prepublication, Mathematical Centre Report IW 12/73, Amsterdam, 1973.

[14] DE BAKKER, J.W., *Least fixed points revisited*, Mathematical Centre Report IW 22/74, Amsterdam, 1974.

[15] DE ROEVER, W.P., *A formalization of various parameter mechanisms as products of relations within a calculus of recursive program schemes*, in: Séminaires IRIA, théorie des algorithmes, des langages et de la programmation, 1972, pp.55-88.

[16] DE ROEVER, W.P., *Recursion and parameter mechanisms: an axiomatic approach*, in: Automata, Languages and Programming, 2nd Colloquium, University of Saarbrücken, July 29 - August 2, 1974, Edited by J. Loeckx, Lecture Notes in Computer Science no. 14, Springer Verlag, Berlin, etc.

[17] DE ROEVER, W.P., *Call-by-value versus call-by-name: a proof-theoretic comparison*, in: Proc. of the third Symposium and Summer School "Mathematical Foundations of Computer Science", Jadwisin, 1974, Lecture Notes in Computer Science, Springer Verlag, Berlin, etc.

[18] DIJKSTRA, E.W., *Notes on structured programming*, in: C.A.R. Hoare, E.W. Dijkstra & O.J. Dahl, *Structured Programming*, Academic Press, New York, 1972.

[19] DIJKSTRA, E.W., *A short introduction to the art of programming*, Report EWD 316, Technological University Eindhoven, 1971.

[20] DIJKSTRA, E.W., *A simple axiomatic basis for programming language constructs*, Indagationes Mathematicae, 36 (1974) 1-15.

[21] FLOYD, R.W., *Assigning meanings to programs*, in: Proc. of a Symposium in Applied Mathematics, Vol. 19, *Mathematical Aspects of Computer Science*, J.T. Schwartz (ed.), AMS, Providence, R.I., 1967.

[22] FOKKINGA, M.M., *Inductive assertion patterns for recursive procedures*, in: Proc. of Symposium on Programming, Paris, April 9-11, 1974 (to appear).

[23] GARLAND, S.J. & D.C. LUCKHAM, *Translating recursion schemes into program schemes*, in: Proc. of an ACM Conference on Proving Assertions about Programs, Las Cruces, New Mexico, January 6-7, 1972.

[24] GOGUEN, J.A. & J.W. THATCHER, *Initial algebra semantics*, in: Proc. of the 15th conference on Switching and Automata Theory, New Orleans, pp. 63-77, 1974.

[25] GUESSARIAN, I., *Sur une réduction des schémas de programmes polyadiques à des schémas monadiques et ses application*, Memo GRIT no. 73.05, Université de Paris, 1973.

[26] HINDLEY, J.R., B. LERCHER & J.P. SELDIN, *Introduction to combinatory logic*, London Mathematical Society Lecture Note Series 7, Cambridge University Press, 1972.

[27] HITCHCOCK, P., *An approach to formal reasoning about programs*, Thesis, University of Warwick, Coventry, England, 1973.

[28] HITCHCOCK, P. & D. PARK, *Induction rules and proofs of termination*, in: Proc. IRIA Symposium on Automata, Formal Languages and Programming, M. Nivat (ed.), North-Holland, Amsterdam, 1972.

[29] HOARE, C.A.R., *An axiomatic basis for computer programming*, Comm. ACM, 12 (1969) 576-583.

[30] HOARE, C.A.R., *Proof of a program: FIND*, Comm. ACM, 14 (1971) 39-45.

[31] HOTZ, G., *Eindeutigkeit und Mehrdeutigkeit formaler Sprachen*, Electron. Informationsverarbeit. Kybernetik, 2 (1966) 235-246.

[32] KAHN, G., *A preliminary theory of parallel programs*, Rapport LABORIA, IRIA, 1973.

[33] KARP, R.M., *Some applications of logical syntax to digital computer programming*, Thesis, Harvard University, 1959.

[34] KING, J.C., *A program verifier*, Thesis, Carnegie-Mellon University, 1969.

[35] KLEENE, S.C., *Introduction to Metamathematics*, North-Holland, Amsterdam, 1952.

[36] KNUTH, D.E., *The Art of Computer Programming*, Vol. 1, *Fundamental Algorithms*, Addison Wesley, Reading (Mass.), 1968.

[37] LYNDON, R.C., *The representation of relational algebras*, Ann. of Math. (2), 51 (1950) 707-729.

[38] LYNDON, R.C., *The representation of relational algebras, II*, Ann. of Math. (2), 63 (1956) 294-307.

[39] MANNA, Z., *The correctness of programs*, JCSS, 3 (1969) 119-127.

[40] MANNA, Z., *Properties of programs and the first-order predicate calculus*, JACM, 16 (1969) 244-255.

[41] MANNA, Z. & J.M. CADIOU, *Recursive definitions of partial functions and their computations*, in: Proc. of an ACM Conference on Proving Assertions about Programs, Las Cruces, New Mexico, January 6-7, 1972.

[42] MANNA, Z., NESS, S. & J. VUILLEMIN, *Inductive methods for proving properties of programs*, ibidem.

[43] MANNA, Z. & J. VUILLEMIN, *Fixpoint approach to the theory of computation*, Comm. ACM, 15 (1972) 528-536.

[44] MAZURKIEWICZ, A., *Proving properties of processes*, PRACE CO PAN CC PAS Reports 134, Warsaw, 1973.

[45] McCARTHY, J., *A basis for a mathematical theory of computation*, in: *Computer Programming and Formal Systems*, pp.33-70, P. Braffort and D. Hirschberg (eds.), North-Holland, Amsterdam, 1963.

[46] MILNER, R., *Algebraic theory of computable polyadic functions*, Computer Science Memorandum 12, University College of Swansea, 1970.

[47] MILNER, R., *Implementation and application of Scott's logic for computable functions*, in: Proc. of an ACM Conference on Proving Assertions about Programs, Las Cruces, New Mexico, January 6-7, 1972.

[48] MILNER, R., *An approach to the semantics of parallel programs*, Edinburgh Technical Memo, University of Edinburgh, 1973.

[49] MORRIS JR., J.H., *Lambda-calculus models of programming languages*, Ph.D. Thesis, M.I.T., December 1968.

[50] MORRIS JR., J.H., *Another recursion induction principle*, Comm. ACM, 14 (1971) 351-354.

[51] PARK, D., *Fixpoint induction and proof of program semantics*, in: *Machine Intelligence*, Vol. 5, pp.59-78, B. Meltzer and D. Michie (eds.), Edinburgh University Press, Edinburgh, 1970.

[52] PARK, D., *Notes on a formalism for reasoning about schemes*, Unpublished notes, University of Warwick, 1970.

[53] ROSEN, B.K., *Tree-manipulating systems and Church-Rosser theorems*, J. Assoc. Comput. Mach., 20 (1973) 160-187.

[54] SCOTT, D., *Models for the λ-calculus*, Unpublished notes, University of Oxford, 1969.

[55] SCOTT, D., *Outline of a mathematical theory of computation*, in: Proc. of the Fourth Annual Princeton Conference on Information Sciences and Systems, pp.169-176, Princeton, 1970.

[56] SCOTT, D., *Mathematical concepts in programming language semantics*, in: Proc. Spring Joint Computer Conference 1972, pp.225-234.

[57] SCOTT, D., *Data types as lattices*, Unpublished lecture notes, Mathematical Centre, 1973.

[58] SCOTT, D., *Data types as lattices*, in: Lecture Notes of the Kiel Summer School, Springer Lecture Notes, Springer Verlag, Berlin, etc., 1974.

[59] SCOTT, D. & J.W. DE BAKKER, *A theory of programs*, Unpublished notes, IBM Seminar, Vienna, 1969.

[60] SCOTT, D. & C. STRACHEY, *Towards a mathematical semantics for computer languages*, in: Proc. of the Symposium on Computers and Automata, Microwave Research Institute Symposia Series Vol. 21, Polytechnic Institute of Brooklyn, 1972.

[61] TARSKI, A., *On the calculus of relations*, J. Symbolic Logic, 6 (1941) 73-89.

[62] VUILLEMIN, J., *Proof techniques for recursive programs*, Thesis, Stanford University, 1972.

[63] WEYRAUCH, R.W. & R. MILNER, *Program correctness in a mechanized logic*, in: Proc. of the First USA-JAPAN Computer Conference, 1972, pp.384-390.

[64] WIRTH, N., *Program development by stepwise refinement*, Comm. ACM, 14 (1971) 221-227.

[65] WRIGHT, J.B., *Characterization of recursively enumerable sets*, J. Symbolic Logic, 37 (1972) 507-511.

[66] DE ROEVER, W.P.,*First-order reduction of call-by-name to call-by-value*, in: Lecture Notes in Computer Science no. 32, *Mathematical Foundations of Computer Science* 1975, Springer Verlag, Berlin, etc.

OTHER TITLES IN THE SERIES MATHEMATICAL CENTRE TRACTS

A leaflet containing an order-form and abstracts of all publications mentioned below is available at the Mathematisch Centrum, Tweede Boerhaavestraat 49, Amsterdam-1005, The Netherlands. Orders should be sent to the same address.

MCT 1 T. VAN DER WALT, *Fixed and almost fixed points*, 1963. ISBN 90 6196 002 9.

MCT 2 A.R. BLOEMENA, *Sampling from a graph*, 1964. ISBN 90 6196 003 7.

MCT 3 G. DE LEVE, *Generalized Markovian decision processes, part I: Model and method*, 1964. ISBN 90 6196 004 5.

MCT 4 G. DE LEVE, *Generalized Markovian decision processes, part II: Probabilistic background*, 1964. ISBN 90 6196 006 1.

MCT 5 G. DE LEVE, H.C. TIJMS & P.J. WEEDA, *Generalized Markovian decision processes, Applications*, 1970. ISBN 90 6196 051 7.

MCT 6 M.A. MAURICE, *Compact ordered spaces*, 1964. ISBN 90 6196 006 1.

MCT 7 W.R. VAN ZWET, *Convex transformations of random variables*, 1964. ISBN 90 6196 007 X.

MCT 8 J.A. ZONNEVELD, *Automatic numerical integration*, 1964. ISBN 90 6196 008 8.

MCT 9 P.C. BAAYEN, *Universal morphisms*, 1964. ISBN 90 6196 009 6.

MCT 10 E.M. DE JAGER, *Applications of distributions in mathematical physics*, 1964. ISBN 90 6196 010 X.

MCT 11 A.B. PAALMAN-DE MIRANDA, *Topological semigroups*, 1964. ISBN 90 6196 011 8.

MCT 12 J.A.TH.M. VAN BERCKEL, H. BRANDT CORSTIUS, R.J. MOKKEN & A. VAN WIJNGAARDEN, *Formal properties of newspaper Dutch*, 1965. ISBN 90 6196 013 4.

MCT 13 H.A. LAUWERIER, *Asymptotic expansions*, 1966, out of print; replaced by MCT 54.

MCT 14 H.A. LAUWERIER, *Calculus of variations in mathematical physics*, 1966. ISBN 90 6196 020 7.

MCT 15 R. DOORNBOS, *Slippage tests*, 1966. ISBN 90 6196 021 5.

MCT 16 J.W. DE BAKKER, *Formal definition of programming languages with an application to the definition of ALGOL 60*, 1967. ISBN 90 6196 022 3.

MCT 17 R.P. VAN DE RIET, *Formula manipulation in ALGOL 60, part 1*, 1968. ISBN 90 6196 025 8.

MCT 18 R.P. VAN DE RIET, *Formula manipulation in ALGOL 60, part 2*, 1968. ISBN 90 6196 038 X.

MCT 19 J. VAN DER SLOT, *Some properties related to compactness*, 1968. ISBN 90 6196 026 6.

MCT 20 P.J. VAN DER HOUWEN, *Finite difference methods for solving partial differential equations*, 1968. ISBN 90 6196 027 4.

MCT 21 E. WATTEL, *The compactness operator in set theory and topology*, 1968. ISBN 90 6196 028 2.

MCT 22 T.J. DEKKER, *ALGOL 60 procedures in numerical algebra, part 1*, 1968. ISBN 90 6196 029 0.

MCT 23 T.J. DEKKER & W. HOFFMANN, *ALGOL 60 procedures in numerical algebra, part 2*, 1968. ISBN 90 6196 030 4.

MCT 24 J.W. DE BAKKER, *Recursive procedures*, 1971. ISBN 90 6196 060 6.

MCT 25 E.R. PAERL, *Representations of the Lorentz group and projective geometry*, 1969. ISBN 90 6196 039 8.

MCT 26 EUROPEAN MEETING 1968, *Selected statistical papers, part I*, 1968. ISBN 90 6196 031 2.

MCT 27 EUROPEAN MEETING 1968, *Selected statistical papers, part II*, 1969. ISBN 90 6196 040 1.

MCT 28 J. OOSTERHOFF, *Combination of one-sided statistical tests*, 1969. ISBN 90 6196 041 X.

MCT 29 J. VERHOEFF, *Error detecting decimal codes*, 1969. ISBN 90 6196 042 8.

MCT 30 H. BRANDT CORSTIUS, *Excercises in computational linguistics*, 1970. ISBN 90 6196 052 5.

MCT 31 W. MOLENAAR, *Approximations to the Poisson, binomial and hypergeometric distribution functions*, 1970. ISBN 90 6196 053 3.

MCT 32 L. DE HAAN, *On regular variation and its application to the weak convergence of sample extremes*, 1970. ISBN 90 6196 054 1.

MCT 33 F.W. STEUTEL, *Preservation of infinite divisibility under mixing and related topics*, 1970. ISBN 90 6196 061 4.

MCT 34 I. JUHÁSZ, A. VERBEEK & N.S. KROONENBERG, *Cardinal functions in topology*, 1971. ISBN 90 6196 062 2.

MCT 35 M.H. VAN EMDEN, *An analysis of complexity*, 1971. ISBN 90 6196 063 0.

MCT 36 J. GRASMAN, *On the birth of boundary layers*, 1971. ISBN 90 6196 064 9.

MCT 37 J.W. DE BAKKER, G.A. BLAAUW, A.J.W. DUIJVESTIJN, E.W. DIJKSTRA, P.J. VAN DER HOUWEN, G.A.M. KAMSTEEG-KEMPER, F.E.J. KRUSEMAN ARETZ, W.L. VAN DER POEL, J.P. SCHAAP-KRUSEMAN, M.V. WILKES & G. ZOUTENDIJK, *MC-25 Informatica Symposium*, 1971. ISBN 90 6196 065 7.

MCT 38 W.A. VERLOREN VAN THEMAAT, *Automatic analysis of Dutch compound words*, 1971. ISBN 90 6196 073 8.

MCT 39 H. BAVINCK, *Jacobi series and approximation*, 1972. ISBN 90 6196 074 6.

MCT 40 H.C. TIJMS, *Analysis of (s,S) inventory models*, 1972. ISBN 90 6196 075 4.

MCT 41 A. VERBEEK, *Superextensions of topological spaces*, 1972. ISBN 90 6196 076 2.

MCT 42 W. VERVAAT, *Success epochs in Bernoulli trials (with applications in number theory)*, 1972. ISBN 90 6196 077 0.

MCT 43 F.H. RUYMGAART, *Asymptotic theory of rank tests for independence*, 1973. ISBN 90 6196 081 9.

MCT 44 H. BART, *Meromorphic operator valued functions*, 1973. ISBN 90 6196 082 7.

MCT 45 A.A. BALKEMA, *Monotone transformations and limit laws*, 1973. ISBN 90 6196 083 5.

MCT 46 R.P. VAN DE RIET, *ABC ALGOL, A portable language for formula manipulation systems, part 1: The language*, 1973. ISBN 90 6196 084 3.

MCT 47 R.P. VAN DE RIET, *ABC ALGOL A portable language for formula manipulation systems part 2: The compiler*, 1973. ISBN 90 6196 085 1.

MCT 48 F.E.J. KRUSEMAN ARETZ, P.J.W. TEN HAGEN & H.L. OUDSHOORN, *In ALGOL 60 compiler in ALGOL 60, Text of the MC-compiler for the EL-X8*, 1973. ISBN 90 6196 086 X.

MCT 49 H. KOK, *Connected orderable spaces*, 1974. ISBN 90 6196 088 6.

MCT 50 A. VAN WIJNGAARDEN, B.J. MAILLOUX, J.E.L. PECK, C.H.A. KOSTER, M. SINTZOFF, C.H. LINDSEY, L.G.L.T. MEERTENS & R.G. FISKER (eds.), *Revised report on the algorithmic language ALGOL 68*. ISBN 90 6196 089 4.

MCT 51 A. HORDIJK, *Dynamic programming and Markov potential theory*, 1974. ISBN 90 6196 095 9.

MCT 52 P.C. BAAYEN (ed.), *Topological structures*, 1974. ISBN 90 6196 096 7.

MCT 53 M.J. FABER, *Metrizability in generalized ordered spaces*, 1974. ISBN 90 6196 097 5.

MCT 54 H.A. LAUWERIER, *Asymptotic analysis, part 1*, 1974. ISBN 90 6196 098 3.

MCT 55 M. HALL JR. & J.H. VAN LINT (eds.), *Combinatorics, part 1: Theory of designs finite geometry and coding theory*, 1974. ISBN 90 6196 099 1.

MCT 56 M. HALL JR. & J.H. VAN LINT (eds.), *Combinatorics, part 2: Graph theory; foundations, partitions and combinatorial geometry*, 1974. ISBN 90 6196 100 9.

MCT 57 M. HALL JR. & J.H. VAN LINT (eds.), *Combinatorics, part 3: Combinatorial group theory*, 1974. ISBN 90 6196 101 7.

MCT 58 W. ALBERS, *Asymptotic expansions and the deficiency concept in statistics*, 1975. ISBN 90 6196 102 5.

MCT 59 J.L. MIJNHEER, *Sample path properties of stable processes*, 1975. ISBN 90 6196 107 6.

MCT 60 F. GÖBEL, *Queueing models involving buffers*. ISBN 90 6196 108 4.

* MCT 61 P. VAN EMDE BOAS, *Abstract resource-bound classes, part 1*. ISBN 90 6196 109 2.

* MCT 62 P. VAN EMDE BOAS, *Abstract resource-bound classes, part 2*. ISBN 90 6196 110 6.

MCT 63 J.W. DE BAKKER (ed.), *Foundations of computer science*, 1975. ISBN 90 6196 111 4.

MCT 64 W.J. DE SCHIPPER, *Symmetrics closed categories*, 1975. ISBN 90 6196 112 2.

MCT 65 J. DE VRIES, *Topological transformation groups 1 A categorical approach*, 1975. ISBN 90 6196 113 0.

* MCT 66 H.G.J. PIJLS, *Locally convex algebras in spectral theory and eigenfunction expansions*. ISBN 90 6196 114 9.

* MCT 67 H.A. LAUWERIER, *Asymptotic analysis, part 2.* ISBN 90 6196 119 X.

* MCT 68 P.P.N. DE GROEN, *Singulary pertlibed differential operators of second order.* ISBN 90 6196 120 3.

* MCT 69 J.K. LENSTRA, *Sequencing by enumerative methods.* ISBN 90 6196 125 4.

MCT 70 W.P. DE ROEVER JR., *Recursive program schemes: semantics and proof theory.* ISBN 90 6196 127 0.

* MCT 71 J.A.E.E. VAN NUNEN, *Contracting Markov decision processes.* ISBN 90 6196 129 7.

* MCT 72 J.K.M. JANSEN, *Simple periodic and nonperiodic Lamé functions and their applications in the theory of eletromagnetism.* ISBN 90 6196 130 0.

* MCT 73 D.M.R. Leivant, *Absoluteness of intuitionistic logic.* ISBN 90 6196 122 x.

* MCT 74 H.J.J. Te Riele, *A theoretical and computational study of generalized aliquot sequences.* ISBN 90 6196 131 9.

An asterisk before the number means "to appear".