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
# CONTENTS

<table>
<thead>
<tr>
<th>Contents</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>vii</td>
</tr>
<tr>
<td>Abstract</td>
<td>ix</td>
</tr>
</tbody>
</table>

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 MU
  3.1. Definition of MU            | 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 MU
  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 "\(n\)" 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
   6.2. Properties of head and tail
   6.3. Correctness of the TOWERS OF HANOI
       6.3.a. Informal part
       6.3.b. An axiomatic correctness proof for the TOWERS OF HANOI

7. ASSESSMENT

APPENDIX 1: SOME TOOLS FOR REASONING ABOUT COMPUTATION MODELS

APPENDIX 2: PROOFS OF MONOTONICITY, CONTINUITY AND SUBSTITUTIVITY

APPENDIX 3: PROOF OF TARKSI'S "UNPROVABLE ASSERTION"

REFERENCES
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.


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. Gensing, 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 "="-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 schemes, 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 conceptualise 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:
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 \((A_1A_2A_3)\) 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 \( \lambda \)-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 "\( \varepsilon \)" between relational terms and predicates: \( \xi \) satisfies \( X@p \) iff \( X \) terminates for input \( \xi \) with output \( \eta \) and output \( \eta \) 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 \( \sigma \) of the program schemes of \( PL \), consisting of proofs for

1. \( \sigma \) 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 $\phi$ of the recursive program schemes defined in chapter 2 to the mathematical interpretation of certain terms of $\mathcal{M}U$, 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 $\mathcal{M}U$ in four successive stages: (1) a many-sorted version of Tarski's axioms for binary relations; derivation of, amongst others, the fundamental lemma $\vdash R;S \cap T = R;(\hat{R};S \cap T) \cap T,$ (2) axiomatization of boolean relation constants; derivation of the properties of the "*" 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 $\mu_1,$ 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, generalised 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 $\mu$-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 derivation 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 $\mu$-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 $\mu$-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 $\mu$-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 Forkinga [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 Manca, Cadiou, Ness and VUILLEMIN in a number of papers: Manca and Cadiou [41], Manca, Ness and VUILLEMIN [42], Manca and VUILLEMIN [43], Cadiou [7] and VUILLEMIN [62]. In section 3.3 we demonstrate that Manca 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 [25, 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 "c" 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 "c" 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 ∅ over D.

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

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

$$R_1 \circ R_2 = \{ <x,y> \mid \exists z <x,z> \in R_1 \text{ and } <z,y> \in R_2 \}. $$

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 \( \pi_i : D^n \rightarrow D \) be the projection function of \( D^n \) on its \( i \)-th component, \( i = 1, \ldots, n \). Let the converse \( \tilde{R} \) of a relation \( R \) be defined by 
\[ \tilde{R} = \{ \langle y, x \rangle \mid \langle y, x \rangle \in R \} \]
and let \( R_1, \ldots, R_n \) be arbitrary relations over \( D \). Consider
\[ R_i; \pi_i^{-1} \cap \ldots \cap R_n; \pi_n^{-1} \]
\((*)\). 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 \( ML \), 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 \( ML \) results in a calculus for recursive programs.

The problem which correctness properties of programs can be formulated within \( ML \) 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. \( E \subseteq R; \bar{R} \) and \( E \subseteq R; U \) both express totality of \( R \).

c. \( R; R \subseteq R \) expresses transitivity of \( R \).

d. \( R; R \subseteq E \) expresses that \( R \) describes the graph of a function, i.e., functionality of \( R \).

e. \( R; \bar{R} \cap E = \{ <x, y> \mid <x, y> \in E \text{ and } <x, y> \in R; \bar{R} \} \)

\[ = \{ <x, y> \mid x = y \text{ and } \exists z[<x, z> \in R \text{ and } <z, y> \in \bar{R}] \} \]

\[ = \{ <x, y> \mid \exists z[<x, z> \in R] \}. \]

Hence \( R; \bar{R} \cap E \) determines that subset of \( E \) which consists of all pairs \( <x, y> \) such that there exists some \( z \) with \( <x, z> \in R \); this indicates a correspondence with a predicate expressing the domain of convergence of \( R \). Note that \( R; \bar{R} \cap E = R; U \cap E \).

f. Let \( p \subseteq E \). Then \( p; U \cap U; p \subseteq p \) expresses that \( p \) contains at most one pair \( <a, a> \) only. This can be understood by deriving a contradiction from the assumption that both \( <a, a> \in p \) and \( <b, b> \in p \) for different \( a \) and \( b \); for that implies that both \( <a, b> \in p; U \) and \( <a, b> \in U; p \), whence \( <a, b> \in p; U \cap U; p \) and therefore \( <a, b> \in p \) 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 \( \mathcal{HU} \) 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]):

\[ C_1 : \pi_1; \bar{\pi}_1 \cap \ldots \cap \pi_n; \bar{\pi}_n = E \]

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

*) This observation is due to Peter VAN EMDE BOAS.
with \( \pi_i \) denoting the projection function of an \( n \)-fold cartesian product on its \( i \)-th component, \( i = 1, \ldots, n \), and \( E \) the identity relation over this product.

In the resulting formal system one can derive properties such as
\[
R = (R_1 \sqcap \neg E) \sqcap R, \quad \text{obtained from example 1.2.e, and } R_1 \sqcap \neg \neg 1 \sqcap R_2 \sqcap \neg 2 = (R_1 \sqcap \neg 1 \sqcap \neg E) \sqcap (R_2 \sqcap \neg 2 \sqcap \neg E) \sqcap (R_1 \sqcap \neg 1 \sqcap \neg R_2 \sqcap \neg 2), \quad \text{obtained by combining examples 1.1.d and 1.2.e.}
\]

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 \( \mu U \) 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

\[
a \subseteq E \quad \text{and} \quad a;\sqcap U; a; \sqsubseteq a.
\]

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. \( a_i;\sqcap U; a_i; \sqsubseteq a_i, \quad i = 1, \ldots, n, \)
2. \( a_1 \sqcup a_2 \sqcup \ldots \sqcup a_n = E, \)
3. \( a_i \sqcap a_j = \Omega, \quad 1 < i < j \leq n, \)
4. \( U \subseteq U; a_i; U, \quad i = 1, \ldots, n. \)

**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 (for the empty, identity, and universal relations \( \emptyset, I, U \) and projection functions \( \pi_i \)) and relation variables.

**Compound terms** are constructed by means of the operators ";'" (relational or Peirce product), "\( \cup \)" (union), "\( \cap \)" (intersection), "\( \rightarrow \)" (converse) and "\( \leftarrow \)" (complementation) and the least fixed point operators "\( \mu_1 \)"; which bind for \( i = 1, \ldots, n \), \( n \) different relation variables in \( n \)-tuples of terms provided none of these variables occur 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 \( \phi \models \psi \), where \( \phi \) and \( \psi \) 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 \(<p,p'>\) of boolean relation constants as pairs \(<m(p),m(p')>\) 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 "\(\wedge\)" and "\(\lor\)" as usual,

(3) interpreting each \(\nu\)-term \(\nu X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n]\) as the \(i\)-th component of the least fixed point of the functional \(<\sigma_1, \ldots, \sigma_n>\) 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 \(\Phi\) are satisfied by \(m\), then the inclusions contained in \(\Psi\) 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 \(\text{tr}\) between schemes and terms we prove that \(\text{tr}\) 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

\[
\begin{array}{c}
T \mapsto \text{tr}(T) \\
\vdots \\
\vdots \\
\downarrow \\
o(T) = m(\text{tr}(T)).
\end{array}
\]

Thus we conclude that, in order to prove properties of \(T\), it suffices to prove properties of \(\text{tr}(T)\), 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 \(o(T) = m(\text{tr}(T))\).
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 **M**U. 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

\[ \forall x, y [p(x) \land x \text{tr}(T) y \rightarrow q(y)] \]

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 \(<p, p'>\) satisfying \(p \subseteq E, p' \subseteq E\) and \(p \cap p' = \Omega\), and \(\text{tr}(T)\) describe program T, then

\[ p \subseteq \text{tr}(T); \text{tr}(T) \]

\(^*)\) This corresponds with \(p(T)q\) in Hoare's notation and with \((p)T(q)\) in Dijkstra's notation (cf. [19]).
or, equivalently,

\[ p \leq \text{tr}(T); U \]

both express (b) (note that \[ p \leq R; K \] is equivalent to \[ p \leq R; U \]).

As to (c): Let \( p \) and \( q \) be represented by \( \langle p, p' \rangle \) and \( \langle q, q' \rangle \), then (c) is expressed by

\[ p; \text{tr}(T) \leq \text{tr}(T); q. \]

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 "\( * \)" operator defined by

\[ Vx[(X*p)(x) \leftrightarrow \exists y [x Yy \text{ and } p(y)]]. \quad (*) \]

Accordingly, if \( X = \text{tr}(T) \) then \((\text{tr}(T)*p)(x)\) is \text{true} iff \( T \) produces for input \( x \) some output \( y \) which satisfies \( p \).

In the present formalism \( X*p \) can be expressed by

\[ X*p = X; p; U \cap E. \]

In example 1.2 we showed that \( \bar{X} \bar{X} \cap E = X; U \cap E = X*E \) describes the domain of convergence of \( X \). Thus \( X*E \) is the least predicate \( p \) satisfying \( X = p; X \).

In chapter 4 we obtain the following characterization of \( X*p \):

\[ X*p = \{ q \mid X; p \leq q ; X \}. \]

*) Let \( X \) denote the function \( f \), then \((X*p)(x) = p(f(x))\).
Therefore \( Xp \) is the least predicate \( q \), sometimes called the \textit{weakest precondition}, satisfying \( Xp \subseteq q; X \).

This observation raises the following question:

When does

\[
X; p = Xp ;X
\]

hold?

We shall prove that \((\ast)\) holds if \( X; X \subseteq E \), i.e., \( X \) denotes the graph of a function.

Therefore the translation theorem implies that

\textit{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 \( <n, \xi> \) in order to distinguish between input domain \( D_n \) and output domain \( D_\xi \). 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 \( <y_1, \ldots, y_n> \) composed of the individual outputs of \( S_i \), \( i = 1, \ldots, n \), and is used to describe the assignment statement and the call-by-value parameter mechanism as follows:

**Assignment statement.** An assignment statement \( x_1 := f(x_{i_1}, \ldots, x_{i_m}) \) occurring in an environment \( x_1, \ldots, x_n \) of variables is expressed by \( [\tau_1, \ldots, \tau_{i-1}, \tau_{i_1}, \ldots, \tau_{i_m}; S_i, \tau_{i+1}, \ldots, \tau_n] \), where \( S \) denotes \( f \).

**Call-by-value parameter mechanism.** A procedure call \( \text{proc}(f_1(x_1, \ldots, x_m), \ldots, f_n(x_1, \ldots, x_m)) \) with parameters which are called-by-value is expressed by \( [S_1, \ldots, S_n]; P \), were \( S_k \) denotes \( f_k \), for \( k = 1, \ldots, n \), 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 \( \{a, a_1, \ldots, b, b_1, \ldots\} \) 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 \( \langle n, \xi \rangle \) 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 \( \langle n, \xi \rangle \) elements denoted by \( A_n^\xi, A_n^\xi, \ldots \)

b. The class of predicate symbols \( B \) contains for all \( n \) elements denoted by \( p_n^\xi, p_n^\xi, \ldots, q_n^\xi, q_n^\xi, \ldots \)

c. The class of constant symbols \( C \) contains the symbols \( a_n^\xi, \xi \) for all types \( \langle n, \xi \rangle, \langle n^1, \ldots, n^m, \xi \rangle \) for all \( n \) and \( n^1, \ldots, n^m \) for all types \( n^1, \ldots, n^m \).

d. The class of variable symbols \( X \), introduced for purposes of substitution, contains for all types \( \langle n, \xi \rangle \) elements denoted by 
\( x_n^\xi, x_n^\xi, \ldots, y_n^\xi, y_n^\xi, \ldots \).

**Procedure symbols.** The class of procedure symbols \( P \) contains for all types \( \langle n, \xi \rangle \) the symbols \( p_n^\xi, p_n^\xi, \ldots \).

**Schemes.**

a. **Statement schemes.** The class of statement schemes \( SS \) (arbitrary elements 
\( s_n^\xi, s_n^\xi, \ldots, \forall n^\xi, \ldots, \forall n^\xi, \ldots \)) is the smallest collection satisfying:

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

2. If \( S_1^s, s_2^s, \xi \in SS \) then \( (S_1,s_2)^n, \xi \in SS \). **)

3. If \( p_n^s, n \in B \) and \( s_1^s, s_2^s, \xi \in SS \) then \( \langle p \rightarrow S_1,s_2 \rangle^n, \xi \in SS \).

4. If \( S_1, \ldots, S_n \in SS \) then \( [S_1, \ldots, S_n]^n, (\xi, \ldots, \xi) \in SS \).

*) 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 \( A_1; (A_2; A_3) \) 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 $DS$ (arbitrary elements $D, D_1, \ldots$) contains all sets $\{P_j^{\alpha_j}\}^\alpha_j\subseteq S_j^{\beta_j}$, $j \in J$ with $J$ any index set, and, for each $j \in J$, $P_j \subseteq P$ and $S_j \subseteq SS$, such that no $S_j$ contains any $X \in X$.

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

**Assertions.** An atomic formula is of the form $T_1 \subseteq T_2$ with $T_1, T_2 \in PS$. A formula is a set of atomic formulas $\{T_1, T_2 \subseteq T_2, l \in L\}$" with $L$ any index set. An assertion is of the form $\phi \vdash \psi$ with $\psi$ and $\phi$ formulas.

**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 $S \subseteq SS$ and $J$ be any nonempty index set such that, for $j \in J$, $(R_j, V_j) \subseteq X \cup P$ 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 $S = S_1; S_2$, $(p + S_1, S_2)$ or $[S_1, \ldots, S_n]$, then $S[V_j/R_j]_{j \in J} = S_1[V_j/R_j]_{j \in J}; S_2[V_j/R_j]_{j \in J}$, $(p + S_1[V_j/R_j]_{j \in J}; S_2[V_j/R_j]_{j \in J})$ or $[S_1[V_j/R_j]_{j \in J}, \ldots, S_n[V_j/R_j]_{j \in J}]$, respectively.

$\exists S$ 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 $X \in X$ occurs in $S \subseteq SS$, $S$ 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 $R_j \in X$, for $j \in J$, and (b) reserve the "**" operator for substitution with $R_j \in P$ 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 generalised to formulae by writing
\( \{V_j, j \in J\} \leftarrow \{V_j, j \in J\} \) for \( \{V_j, j \in J\} \subseteq \{V_j, j \in J\} \), 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 \( S(V_j, \ldots, V_n) \). If \( J = \{1\} \) we also use \( S[V/\Sigma] \).

4. \( S[V_j/X_j]_{j \in J} \) is defined according to the complexity of \( S \). Therefore properties such as the chain rule, \( S[V_j/X_j]_{j \in J} S[V_j/X_j]_{j \in J} = S[V_j/X_j]_{j \in J} S[V_j/X_j]_{j \in J} \), 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 \( D_\eta \), and for each compound domain type \( (\eta_1 \times \cdots \times \eta_n) \), \( o_0(\eta_1 \times \cdots \times \eta_n) \) is the cartesian product of \( o_0(\eta_1), \ldots, o_0(\eta_n) \).

b. For \( A^n, C \in A \) and \( X^n, \xi \in X \), \( o_0(A^n, C) \) and \( o_0(X^n, \xi) \) are subsets of \( o_0(\eta) \times o_0(\xi) \).

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

d. For each projection function symbol \( \pi^n_1 \), \( o_0(\pi^n_1) \) is the projection function of \( o_0(\eta_1) \times \cdots \times o_0(\eta_n) \) on its \( i \)-th constituent coordinate.

e. For all constants \( a^n, C \) and \( x^n, \eta \), \( o_0(a^n, C) \) and \( o_0(x^n, \eta) \) 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 *true* or *false* 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 \( x \Delta y \) is pair \( \langle x_1 S_1 x_2 \ldots x_n S_{n+1} C M \rangle \) 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 \( S_1 = R \) or \( S_1 = R; V \) with \( R \in A \cup C \cup X \), \( x_i x_{i+1} : e o_0(R) \) and \( i = n \) or \( S_{i+1} = V \), respectively.

b. If \( S_i = P_j \) or \( S_i = P_j; V \) and \( P_j \leftarrow S_j \in D \), 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 \( o_0(p)(x_i) = \text{true} \) then \( S_{i+1} = V_1 \) or \( S_{i+1} = V_2 \); if \( o_0(p)(x_i) = \text{false} \) then \( S_{i+1} = V_3 \), 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 \( x_i V_1 y_1 \), for \( i = 1, \ldots, k \), and \( i = n \) or \( S_{i+1} = V \), respectively.

**Remark.** A computation model represents the entire computation of program \( <D, S> \) 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 \( \Omega \). However the set \( \{ xP[x][P, P]; \pi_1, \langle x, \pi_1 \rangle \} \) is a computation model for \( xTx \) in the sense of this definition (P. Van Emde Boas).

**Definition 2.5.**

**Operational interpretation.** Let \( T = <D, S^\pi, \xi> \) 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 \( \langle x, y \rangle \in o_0(\mathcal{E}) \times o_0(\mathcal{E}), \langle x, y \rangle \in o(T) \) 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. \( \phi \) satisfies \( o \) (is valid) iff all its inclusions satisfy \( o \) (are valid).
c. An assertion $\forall \vdash \forall$ such that, for all $\sigma$, if $\phi$ satisfies $\sigma$, then $\forall$
satisfies $\sigma$, is called valid.

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

2.2. The union theorem

First we mention properties of the operational interpretation $\sigma$ such as
$\sigma(S_1; S_2) = \sigma(S_1); \sigma(S_2)$, $\sigma(p \rightarrow S_1; S_2) = m(p); \sigma(S_1) \cup m(p'); \sigma(S_2)$,
$\sigma([S_1, \ldots, S_n]) = \sigma(S_1); \sigma(S_1) \cap \ldots \cap \sigma(S_n); \sigma(S_n)$, 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 pro-
gram 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. $\sigma(p \rightarrow S_1; S_2) = m(p); \sigma(S_1) \cup m(p'); \sigma(S_2)$, with $m(p)$ and $m(p')$ defined as
follows: $\langle x, x \rangle < m(p)$ iff $\sigma_0(p)(x) = \text{true}$ and $\langle x, x \rangle < m(p')$ iff
$\sigma_0(p)(x) = \text{false}$.

d. $\sigma([S_1, \ldots, S_n]) = \sigma(S_1); \sigma(S_1) \cap \ldots \cap \sigma(S_n); \sigma(S_n)$.

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 $\emptyset \subseteq S$ and $B; S = S$ 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).

\[ \{ V_{1,j} \leq V_{2,j} \}_{j \in J} \vdash S[V_{1,j}/X_{j}]_{j \in J} \leq S[V_{2,j}/X_{j}]_{j \in J}. \]

Proof. By induction on the complexity of \( S \).

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

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 \( o((S_1; S_2)[V_{1,j}/X_{j}]_{j \in J}) = \)

\[ = o(S_1[V_{1,j}/X_{j}]_{j \in J}; S_2[V_{1,j}/X_{j}]_{j \in J}) = \text{(lemma 2.1)} \]

\[ o(S_1[V_{1,j}/X_{j}]_{j \in J}; o(S_2[V_{1,j}/X_{j}]_{j \in J}) \leq \text{(induction hypothesis)} \]

\[ = o(S_1[V_{1,j}/X_{j}]_{j \in J}; o(S_2[V_{1,j}/X_{j}]_{j \in J}) = \text{(lemma 2.1)} \]

\[ = o(S_1[V_{2,j}/X_{j}]_{j \in J}; S_2[V_{2,j}/X_{j}]_{j \in J}) = o((S_1; S_2)[V_{2,j}/X_{j}]_{j \in J}). \]

d. \( S = (p - S_1, S_2) \) or \( S = [S_1, \ldots, S_n] \), similar to c. \( \square \)

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 \( S \), \( \{ V_{j} \leq P_{j} \}_{j \in J} \vdash \tilde{S}[P_{j}/X_{j}]_{j \in J}; [V_{j}/X_{j}]_{j \in J} \leq S[V_{j}/X_{j}]_{j \in J}. \)

c. For arbitrary \( S \), \( \tilde{S}[V_{j}/X_{j}]_{j \in J} = \tilde{S}[V_{j}/X_{j}]_{j \in J}. \)

Proof. Follows from the definitions, properties of substitution and monotonicity, by induction on the complexity of \( S \). \( \square \)

Informally, if a recursive procedure \( P_{n; r}^{m; z} \) 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 \( P_{n; r}^{m; z} \) are not executed again, whereas they may be replaced by \( \Omega_{n; r}^{m; z} \) without affecting the computation.

This process of replacement can be generalized to calls of simultaneously declared recursive procedures: Let \( S_{1}^{m; z} \) be a statement scheme. Then \( S_{1}^{m; z} \)
is obtained from $S$ by uniformly replacing calls of $P_j^n$ at level $n$ by $\tilde{a}^n$, for $j \in J$ with $S(0)$ defined as $\tilde{a}^0$. 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

$$x \ o(S) \ y \iff \exists n[x \ o(S(n)) \ y].$$

**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 preceding 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 $i$, $1 \leq i \leq n$, $S_i = P_j$ 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 $S_i$, $1 \leq i \leq n$. This occurrence directly identifies the corresponding occurrence of $P_j$ in $S$ occurring in $S_{i+1}$ or $S'_i$ 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) $S_i = (S); V_j$ and $S$ occurs as first statement $S'_i$ of the associated computation model for $x_1 S x_{i+1}$.

*) 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) \( S_i = (p \rightarrow S_i; V) \) or \( S_i = (p \rightarrow S_i; v_i; S) \), and \( S_{i+1} = S \),
(d2) \( S_i = (p \rightarrow S_i; V_1); V_2 \) or \( S_i = (p \rightarrow V_i, S); V_2 \), and \( S_{i+1} = S_i; V_2 \),
(d3) \( S_i = (p \rightarrow V_1, V_2); S \) and \( S_{i+1} = V_1; S \) or \( S_{i+1} = V_2; S \),
(e1) \( S_i = [V_1, \ldots, V_m] \) or \( S_i = [V_i, \ldots, V_m]; V \), and \( S = V_k \) for some \( k \),
\( 1 \leq k \leq m \), CM contains an associated computation model CM' for
\( x_i; S_{i+1}; k \), and \( S \) occurs as first statement \( S_i \) of the constituent com-
putation 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. *)
\[ s^{[n]}(0) = S, s^{[k+1]} = s^{[k]} / X_j \] for \( k = 0, 1, 2, \ldots \).
\[ s^{[n]}, s^{(0)} = S, s^{(k+1)} = s^{[k]} / X_j \] for \( k = 0, 1, 2, \ldots \).

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

**Lemma 2.4.** Let \( n \) be a natural number. Then \( S^{(n+1)} = S^{(n)} \) and \( s^{[k+1]} = s^{[k]}(1) \).

**Proof.** We prove the second result only. Use induction on \( n \).
1. \( k = 0 \).
\[ s^{(1)} = 0 / X_j \] for \( j \in J \).

2. Assume the result for \( n = k \). We have
\[ s^{[k+1]} / X_j \] for \( j \in J \).
\[ s^{[k]} / X_j \] for \( j \in J \).
\[ s^{[k]} / X_j \] for \( j \in J \).
\[ s^{[k]} / X_j \] for \( j \in J \).
\[ s^{[k]} / X_j \] for \( j \in J \).

In order to prove \( o(S) \leq \bigcup_{n=0}^{\infty} o(S^{(n)}) \) we shall transform a computation model for \( xS \) into a computation model for \( xS^{(n)} \).
Let \( S \) be closed and CM be a computation model for \( xS \) with constituent sequence \( x_1, x_2, \ldots, x_n, S_{n+1} \). If \( n \) occurrences of \( S_j \) in \( S \) are executed to compute \( y \), all occurrences of \( S_j \) identified by occurrences of \( S_j \) in \( S_i \)

*) Hence, if \( S_i = P_j \) or \( S_i = P_j; V \), the only or first occurrence, respec-
tively, 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 \( xS[V_j/x_j] \), \( j \in J \).

Observe as a corollary that by choosing \( V \) 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 \( x_j \) \( S_j \) 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 \( xS^{[n]}[G_j/P_j], j \in J \) 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 \( \sum_{n=0}^{\infty} o(S^{(n)}) \leq o(S) \):

First we show that for each \( j \in J \) and each \( k \), \( P_j^{(k)} \leq P_j \). Use induction on \( k \).
1. \( k = 0 \). Clear.

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

Next we show that \( S_j^{(k)} \leq S : S_j^{(k)} = \tilde{s}_j[S_j^{(k-1)}/x_j]_{j \in J} = \tilde{s}_j[P_j^{(k)}/x_j]_{j \in J} \leq\)
\( \leq (\text{lemma } 2.2) \tilde{s}_j[P_j/x_j]_{j \in J} = (\text{lemma } 2.3) S.\)

Thus \( \bigcup_{n=0}^{\infty} S(n) \subseteq S \) follows. \( \square \)

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 lfp) 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 \( P_j^{(k+1)} = S_j^{(k)} = \tilde{s}_j[P_j^{(k)}/x_j]_{j \in J} \leq\)
\( \leq (\text{induction hypothesis}) \tilde{s}_j[V_j/x_j]_{j \in J} \subseteq V_j. \quad \square \)

Remark. Combination of the fixed point and least fixed point properties yields, for all \( i \in J,\)
\( o(p_i) = \bigcap_{o(V_i) \mid \tilde{s}_j[V_j/x_j]_{j \in J} \subseteq o(V_k), \text{ for all } k \in J}.\)

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:

\( o(p_i) = \bigcap_{\forall k \in J} o(V_i) \text{ for all } k \in J, o(\tilde{s}_k[V_j/x_j]_{j \in J}) \subseteq o(V_k) \).

This characterization of \( o(p_i) \) is the key to the definition of the mathematical interpretation of \( \mu \)-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 1fpp (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 $(j,k) \in J \times K$, $S_{j,k}$ be closed statement schemes the procedure symbols of which are indexed by J, then

$$<p_j \equiv \bar{S}_j[S_j, \{x_j \mid j \in J\}]_{j \in J}> =<p_j, k \equiv \bar{S}_j[S_j[S_j, \{x_j \mid j \in J\}]_{j \in J}]_{j \in J}, k \equiv \bar{S}_j[S_j[S_j, \{x_j \mid j \in J\}]_{j \in J}]_{j \in J}>$$

is valid.

**PROOF.** The case $J = \{0\}$ and $K = \{1,2\}$ is considered to be representative. Then one has to prove $<p_0 \equiv S_0(S_0(P_0), S_0(P_0)), P_0>$ = $<p_1 \equiv S_1(S_0(P_0, P_2), P_2 \equiv S_2(S_0(P_1, P_2)), S_0(P_1, P_2)>)$.

Consider the following declaration scheme:

$$<p_0 \equiv S_0(S_0(P_0), S_0(P_0)), P_1 \equiv S_1(S_0(P_1, P_2), P_2 \equiv S_2(S_0(P_1, P_2)), P_3 \equiv S_0(P_1, P_2), P_4 \equiv S_1(P_0), P_5 \equiv S_2(P_0)>$$

With respect to this declaration scheme one proves $p_0 = p_3$ by applying 1fpp 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., $S_0(S_1(P_2), S_2(P_3)) \subseteq P_3$ is derived by $S_0(S_1(P_2), S_2(P_3)) =$

$= (fpp \text{ and substitution rule}) S_0(S_1(S_0(S_0(P_1, P_2)), S_2(S_0(P_1, P_2)))) = (\text{similarly}) S_0(P_1, P_2) = (fpp) P_3$.

As $P_3 = (fpp) S_0(P_1, P_2)$, the desired result is obtained by deleting declarations for uncalled procedures. $\Box$

Let us introduce the following convention. Calls of recursive procedures P, with P declared by $P \equiv (p; S; P; E)$, 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:

\[
\langle P \iff A ; (s \rightarrow S ; P ; F , E ) ; (b \rightarrow B ; P , E ) , P \rangle = \\
= \langle P \iff A ; (s \rightarrow S ; P ; b \star (B ; P ) ; F , E ) , P ; b \star (B ; P ) \rangle . "
\]

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 $A_1^{\eta_1}, A_2^{\eta_2}, \ldots, A_k^{\eta_k}$.

b. The class of boolean relation constant symbols $B$ contains for all $\eta$ elements denoted by $p_1^{\eta_1}, p_2^{\eta_2}, \ldots, q_1^{\eta_1}, q_2^{\eta_2}, \ldots$.

c. The class of logical relation constant symbols $C$ contains for all types $\eta_1, \ldots, \eta_n$ concerned the symbols $x_1^{\eta_1}, x_2^{\eta_2}, \ldots, x_i^{\eta_i}, \ldots$.

d. The class of relation variable symbols $X$ contains for all types $\langle \eta, \xi \rangle$ elements denoted by $x_1^{\eta_1}, x_2^{\eta_2}, \ldots, x_i^{\eta_i}, \ldots$.

Terms. The class of terms $T$, with arbitrary elements $\sigma_1^{\eta_1}, \sigma_2^{\eta_2}, \ldots, \sigma_i^{\eta_i}, \ldots$ is the smallest collection satisfying:

a. $A \cup B \cup C \cup X \subseteq T$
b. If $\sigma^n_\xi \in T$, then $\sigma^n_\xi \eta$ and $\sigma^n_\xi \xi \in T$.

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

d. If $\sigma^n_1, \ldots, \sigma^n_n \in T$ and $X^n_1, \ldots, X^n_n$ denote pairwise distinct relation variables then $\nu_i X^n_1 \ldots X^n_n[\sigma^n_1, \ldots, \sigma^n_n] \in T$, for $i = 1, \ldots, n$.

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

**Syntactically continuous.** A term $\sigma$ is syntactically continuous in $X$ if no free occurrence of $X$ in $\sigma$ lies within any subterm $\tau$.

**Well-formed terms.** A term $\sigma$ is well-formed if, for all terms $\nu_i X^n_1 \ldots X^n_n[\sigma^n_1, \ldots, \sigma^n_n]$ occurring as subterms of $\sigma$, each $\sigma^n_j$ is syntactically continuous in each $X^n_k$, $j, k = 1, \ldots, n$.

**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 $\{\sigma^n_{1, l} \subseteq \sigma^n_{2, l}\}_{l \in L}$ with $L$ any index set. An assertion is of the form $\phi \models \psi$ with $\phi$ and $\psi$ formulae.

**Remarks.**
1. $\sigma_1 = \sigma_2$ is an abbreviation for $\sigma_1 \subseteq \sigma_2, \sigma_2 \subseteq \sigma_1$ and $\nu_i X^n_1[\sigma_1]$, written as $\nu X[\sigma]$.
2. For empty $\emptyset$, $\phi \models \psi$ is written as $\models \psi$.

**DEFINITION 3.2.** (Substitution)

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

*) In accordance with the convention, that "$\tau$" binds stronger than "$\sigma^n" and "$\sigma^n" binds stronger than "$\nu^n", the parentheses around $\sigma \tau, \sigma \eta \tau$ 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 \( \sigma = \sigma_1 \) or \( \sigma_1 \) then \( \sigma[\tau_j/X_j]_{j \in J} = \sigma_1[\tau_j/X_j]_{j \in J} \) or \( \sigma[\tau_j/X_j]_{j \in J} \), respectively.

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

e. If \( \sigma = \mu_1 Z_n \ldots Z_n [\sigma_1, \ldots, \sigma_n] \) then, for \( i = 1, \ldots, n \),
\[
\sigma[\tau_j/X_j]_{j \in J} = \mu_1 Y_1 \ldots Y_n \sigma_1[Y_j/Z_j, \ldots \sigma_1[M_1 \ldots M_n \tau_j/X_j]_{j \in J}, \ldots, \mu_n Y_1 \ldots Y_n \sigma_n[Y_j/Z_j, \ldots \sigma_n[M_1 \ldots M_n \tau_j/X_j]_{j \in J},\ldots,\sigma_n[M_1 \ldots M_n \tau_j/X_j]_{j \in J}, \ldots],
\]
where \( J^* = J - \{ j \mid j \in J \text{ and } (\exists i [i \leq n \text{ and } Z_i = X_i]) \} \), whence
\[
\{X_j\}_{j \in J^*} = \{X_j\}_{j \in J} - \{X_j \mid j \in J \text{ and } (\exists i [i \leq n \text{ and } Z_i = X_i]) \},
\]
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 \( \sigma_k \) (\( k = 1, \ldots, n \)), nor in any \( \tau_j \) (\( j \in J^* \)),
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 \( \sigma \) by simultaneous substitution of \( \tau_j \) for \( X_j \), replacing bound variables whenever necessary in order to prevent binding of free occurrences of \( X_k \) in any substituted \( \tau_j \), 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
\[
\{\sigma_1, \ldots, \sigma_l\}_{l \in L} \ldots \{\tau_j/X_j\}_{j \in J} \quad \text{for} \quad \{\sigma_1, \ldots, \sigma_l\}_{l \in L} \subseteq \{\sigma_1[\tau_j/X_j]_{j \in J} \}_{l \in L},
\]
3. Properties involving the substitution operator such as the chain rule can be proved by induction on the complexity of \( \sigma \).

4. If \( J = \{1, \ldots, n\} \), \( \sigma[\tau_j/X_j]_{j \in J} \) is written as \( \sigma[\tau_1/X_1]_{j=1}, \ldots, n \) or \( \sigma(\ldots, \tau_n) \). If \( J = \{1\} \) we also use \( \sigma(\ldots, \tau) \).
Compared with the everyday relational language the $\nu$-terms
\[ \nu_1X_1 \ldots X_n[\tau_1 \ldots \tau_n] \] represent the only new feature of $\nu$ and its predecessors (c.f. Scott and de Bakker [59], de Bakker [9] and de Rijke [11]). In order to explain their interpretation we first describe the concept of continuity.

A term $\tau$ 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 $\tau$. Therefore interpretations of variables, called variable valuations $\nu$, have to be separated from interpretations of constants, called initial interpretations $\lambda$. Thus a pair $<\lambda,\nu>$ determines a functional; this functional is called model function and denoted by $\phi_{\lambda}^{(\nu)}$.

Continuity of $\phi_{\lambda}^{(\nu)}$ in $X_1, \ldots, X_n$ can now be defined as follows: Let $\tau$ be a term, $X_1, \ldots, X_n$ be variables, $\lambda$ be an initial interpretation and $\nu$ and, for each $j \in N$, $v_j$ be variable valuations satisfying, for $i = 1, \ldots, n,$
\[ v_i(X_i) = \bigcup_{j=0}^m v_j(X_i), \quad v_j(X_i) \subseteq v_{j+1}(X_i) \] and $v(X) = v_j(X)$ for $X$ different from $X_i$, for all $j$. Then $\phi_{\lambda}^{(\nu)}$ is continuous in $X_1, \ldots, X_n$ iff $\phi_{\lambda}^{(\nu)}(v) = \lim_{j \to \infty} \phi_{\lambda}^{(\nu_j)}(v)$ for all $\nu$ and $\nu_j$ considered above and all $\lambda$.

This concept derives its importance from the fact that only if $\phi_{\lambda}^{(\tau_1)}, \ldots, \phi_{\lambda}^{(\tau_n)}$ are continuous in $X_1, \ldots, X_n$, is Scott’s induction rule for establishing properties of $\phi_{\lambda}^{(\nu_1X_1 \ldots X_n[\tau_1, \ldots, \tau_n]})(\nu)$ valid.

A syntactically sufficient, although not necessary condition for continuity of $\phi_{\lambda}^{(\nu)}$ in $X_1, \ldots, X_n$, is the following one: free occurrences of $X_1, \ldots, X_n$ are not contained in complemented subterms cf $\tau$, i.e., $\tau$ is syntactically continuous in $X_1, \ldots, X_n$.

We therefore define the interpretation of $\nu_1X_1 \ldots X_n[\tau_1, \ldots, \tau_n]$ 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 $\nu$)

*Assignment of types.* An initial assignment of types is a function $t_0 \cdot G \rightarrow D$, where $G$ is the collection of possibly subscripted greek letters and $D$ 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 $D_n$, 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 $D_{n_1} \times \ldots \times D_{n_n}$.

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

a. $\iota(A^{n_1} \times \xi) \subseteq t(\eta) \times t(\xi)$.

b. For $p^{n_1} \eta, p^{n_2} \eta \in B$, $\iota(p^{n_1} \eta)$ and $\iota(p^{n_2} \eta)$ are disjoint subsets of the identity relation over $t(\eta)$.

c. $\iota(A^{n_1} \times \xi)$ is the empty subset of $t(\eta) \times t(\xi)$, $\iota(E^{n_1} \eta)$ is the identity relation over $t(\eta)$, $\iota(u^{n_1 \xi})$ is $t(\eta) \times t(\xi)$ itself and $\iota(\tau^{n_1 \times \ldots \times n_n \eta \xi})$ 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 $v : X \rightarrow \bigcup_{D_1, D_2 \in D} D_1 \times D_2$, satisfying $v(X^{n_1} \times \xi) \subseteq t(\eta) \times t(\xi)$ for all $X^{n_1} \times \xi \in X$.

Model function. Relative to a given assignment of types $t$ and an initial interpretation $\iota$, the model function $\phi_t^{(n_1 \times \xi)} : V \rightarrow 2^{n_1 \times D_2}$ is defined as follows for well-formed terms $\sigma^{n_1 \times \xi}$:

a. $\phi_t^{(R)}(v) = \iota(R)$, $R \in A \cup B \cup C$.

b. $\phi_t^{(X)}(v) = v(X)$, $X \in X$.

c. $\phi_t^{(f_1 \circ f_2)}(v) = \phi_t^{(f_1)}(v) \circ \phi_t^{(f_2)}(v)$, $\phi_t^{(f_1 \cup f_2)}(v) = \phi_t^{(f_1)}(v) \cup \phi_t^{(f_2)}(v)$, $\phi_t^{(f_1)}(v) = \phi_t^{(f_1)}(v)$, $\phi_t^{(f_1 \leq f_2)}(v) = \phi_t^{(f_1)}(v)$, $\phi_t^{(f_1 \triangleright f_2)}(v) = \phi_t^{(f_1)}(v)$.

d. $\phi_t^{(u_{1 \times n} X_{\sigma_1}, \ldots, X_{\sigma_n})}(v) = \langle n(v'(X_k))_{k=1}^n \mid \phi_t^{(\sigma_k)}(v') \leq v'(X_k), k = 1, \ldots, n, \text{ and } v'(X) = v(X) \text{ for } X \in \{X_1, \ldots, X_n\} \rangle$.

*) Cf. remark on page 29.
Interpretation of terms. An interpretation of terms is a triple \( \langle \tau_0, \iota, \nu \rangle \) where each term \( \sigma \) is interpreted as \( \Phi_{\iota}(\nu) \). This triple will often be referred to as \( m \). Then \( \Phi_{\iota}(\nu) \) 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 \( \{ \sigma_{1,l} \subseteq \sigma_{2,l} \}_{1 \in \mathcal{L}} \) satisfies an interpretation of terms \( m \) iff \( \sigma_{1,l} \subseteq \sigma_{2,l} \) satisfies \( m \) for all \( l \in \mathcal{L} \).

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

Remark. The definition of \( \nu \)-terms can be straightforwardly generalized to the case where the \( \nu \)-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 induction theorem for \( \mathcal{M} \) 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 induction 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, \( X_j \subseteq X \) for all \( j \in J \), \( \sigma \in \tau \) 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 \( X \in X - \{ X_j \}_{j \in J} \). Then the following holds:
\[
\Phi_{\iota}(v_1) \subseteq \Phi_{\iota}(v_2).
\]

\(^*)\) In the sequel \( m \) is often called the mathematical interpretation, as opposed to \( \sigma \), the operational interpretation.

\(^**)\) Reference to a given initial interpretation is tacitly assumed. Accordingly, \( \Phi_{\iota} \) will be written as \( \Phi_{\iota}(v) \).
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 $\tau$ and $\nu_i$ $(i \in N)$ be variable valuations which satisfy, for $i \in N$ and $j \in J$, (1) $\nu(X_j) = \bigcup_{i=0}^{\infty} \nu_i(X_j)$, (2) $v_i(X_j) \subseteq v_{i+1}(X_j)$ and (3) $\nu(X) = \nu_1(X)$ for $X \in X - \{X_j\}_{j \in J}$. Then the following holds:

$$\phi_{<\sigma>(v)} = \bigcup_{i=0}^{\infty} \phi_{<\sigma_i>(v)}.$$

LEMMA 3.3. (Substitutivity). Let $J$ be any index set, $\sigma \in T$, $X_j \in X$ and $\tau_j \in \tau$ for $j \in J$, and variable valuations $\nu_1$ and $\nu_2$ satisfy (1) $\nu_1(X_j) = \phi_{<\tau_j>(v_1_i)}$ for $j \in J$ and (2) $\nu_2(X) = \nu_2(X)$ for $X \in X - \{X_j\}_{j \in J}$. Then the following holds:

$$\phi_{<\sigma>(v_1)} = \phi_{<\sigma \tau_j \tau_j >}(v_2).$$

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

$$\phi_{<u \in X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n]>(v)} = \phi_{<u \in Y_1 \ldots Y_n [\sigma_1[Y_1/X_1]_{1=1} = 1, \ldots, \sigma_n[Y_n/X_n]_{1=1} = \ldots, \ldots]>(v)}.$$

Proof. Follows by definition 3.2 from lemma 3.3. □

The union theorem for $MU$ states that least fixed points

$$\phi_{<u \in X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n]>(v)}, \ldots, \phi_{<u \in X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n]>(v)}$$

of continuous functionals $\lambda \nu \phi_{<\sigma>(v)}$, $\ldots, \phi_{<\sigma>(v)}$ can be obtained as unions of sequences of finite approximations $\phi_{<\sigma^i_u>(v)}$, $\ldots, \phi_{<\sigma^i_u>(v)}$, $i = 0, 1, \ldots$, with $\sigma^i_k$ similarly defined as $S_k^i$, $k = 1, \ldots, n$, cf. definition 2.6.

DEFINITION 3.4. $\sigma^i_k$. Let $\eta_1^{\xi_1}, \ldots, \eta_n^{\xi_n} \in X$ be the free variables in $\sigma_1^{\xi_1}, \ldots, \sigma_n^{\xi_n} \in T$, then $\sigma^i_k$ is defined by (1) $\sigma^0_k = \eta_k^{\xi_k}$ and

$$(2) \sigma_{k+1}^i = \sigma_k^i[\sigma_k^{i+1}[\eta_k^{\xi_k}]_{1=1} = 1, \ldots, n], \text{ for } k = 1, \ldots, n.$$

THEOREM 3.1. (Union theorem for $MU$). Let $\eta_1, \ldots, \eta_n \in T$ be syntactically continuous in $X_1, \ldots, X_n \in X$. Then the following holds for all variable valuations $\nu$:
Proof. The proof splits into three parts. In the first part we prove
\[ \phi_{\varsigma_i}^i(v) \subseteq \phi_{\varsigma_i}^{i+1}(v) \] for \( i \in N \), in the second part
\[ \phi_{\varsigma_i \ldots \varsigma_n}^i(v) \subseteq \bigcup_{i=0}^n \phi_{\varsigma_i}^i(v) \], and in the third part
\[ \phi_{\varsigma_i \ldots \varsigma_n}^i(v) \supseteq \bigcup_{i=0}^n \phi_{\varsigma_i}^i(v) \] (the reverse inclusion).

Part 1. By induction on \( i \). Obviously, \( \phi_{\varsigma_k}^0(v) \subseteq \phi_{\varsigma_k}^0(v) \).
Assume by hypothesis \( \phi_{\varsigma_k}^{i-1}(v) \subseteq \phi_{\varsigma_k}^i(v) \) and prove \( \phi_{\varsigma_k}^i(v) \subseteq \phi_{\varsigma_k}^{i+1}(v) \),
\( k = 1, \ldots, n \). Define variable valuation \( v_i \) by \( v_i(X_k) = \phi_{\varsigma_k}^i(v) \) for
\( k = 1, \ldots, n \) and \( v_i(X) = v(X) \), otherwise.
Then \( \phi_{\varsigma_k}^{i+1}(v) = \phi_{\varsigma_k}^{i+1}(v) \) (substitutivity) \( \phi_{\varsigma_k}^i(v) \).
Similarly, \( \phi_{\varsigma_k}^{i+1}(v) = \phi_{\varsigma_k}^{i+1}(v) \) with \( v_2 \) defined by \( v_2(X_k) = \phi_{\varsigma_k}^{i-1}(v) \) for
\( k = 1, \ldots, n \) and \( v_2(X) = v(X) \), otherwise.
As \( \varsigma_1, \ldots, \varsigma_n \) are syntactically continuous, \( \phi_{\varsigma_k}^i(v) = \phi_{\varsigma_k}^{i+1}(v) \) \( \subseteq \)
( monotonicity and hypothesis ) \( \phi_{\varsigma_k}^i(v) = \phi_{\varsigma_k}^{i+1}(v) \), for \( k = 1, \ldots, n \).

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

Part 3. \( \supseteq \): Let \( v \) satisfy \( \phi_{\varsigma_k}^i(v) \subseteq v(X_k) \) for \( k = 1, \ldots, n \) and \( v(X) = v(X) \), otherwise.
Then we prove \( \phi_{\varsigma_k}^i(v) \subseteq v(X_k) \) for \( i \in N \) by induction on \( i \). Obviously,
\( \phi_{\varsigma_k}^i(v) \subseteq v(X_k) \).
Assume by hypothesis \( \varphi_\alpha^i(v') \subseteq v'(X_k) \) and prove \( \varphi_\alpha^{i+1}(v') \subseteq v'(X_k) \), \( k=1,\ldots,n \).

Define variable valuation \( v'' \) by \( v''(X_k) = \varphi_\alpha^1(v') \) for \( k=1,\ldots,n \) and \( v''(X) = v'(X) \), otherwise.

Then \( \varphi_\alpha^{i+1}(v') = \varphi_\alpha^{i}[X_1^i/X_1]_{i=1},\ldots,\varphi_\alpha^{i}(v') = (\text{substitutivity}) \varphi_\alpha^{i}(v'') \subseteq \quad \)

(\( \text{(monotonicity, as} \quad v''(X_k) = \varphi_\alpha^{i}(v') \subseteq v'(X_k) \text{ by hypothesis and } v''(X) = v'(X), \text{ otherwise}) \quad \)

Thus \( \bigcup_{i=0}^{N} \varphi_\alpha^{i}(v) = (X_1,\ldots,X_n \text{ not occurring in } \alpha, \quad \)

\( \forall \varphi_\alpha^{i}(v) \subseteq \quad \)

\( \langle \forall (v'(X_k)^{i}_{i=1} \mid \varphi_\alpha^{i}(v') \subseteq v'(X_k), \quad 1=1,\ldots,n, \text{ and } v'(X) = v(X) \rangle \quad \)

for \( X \in X - \{X_1,\ldots,X_n\} \rangle \).

\( \square \)

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 (Burstable [5]) and computational induction (Manna and Vuijlemin [43]).

Its formulation is given by

\[
\begin{align*}
\phi & \vdash \forall[\alpha[k,\beta_k/X_k]_{k=1},\ldots,n] \\
\phi, \forall & \vdash \forall[\alpha[k,\beta_k/X_k]_{k=1},\ldots,n] \\
I: & \\
\phi & \vdash \forall[\alpha[X_1,\ldots,X_n,\beta_k/X_k]_{k=1},\ldots,n] \\
\end{align*}
\]

where \( \phi \) has no free occurrences of \( X_1 \), and \( \forall \) only contains occurrences of \( X_1 \) which are not contained in complemented subterms.

**THEOREM 3.2. (Validity of Scott's induction rule, I).** If \( \phi \) and \( \forall \) are formulae such that \( \phi \) does not contain any free occurrence of \( X_k \), \( k=1,\ldots,n \), and all terms contained in \( \forall \) are syntactically continuous in \( X_k \), \( k=1,\ldots,n \), then \( I \) is valid.
Proof. Let \( v \) be any variable valuation satisfying \( \varphi \), let \( v' \) be defined by
\[
v'(x_k) = \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v)
\]
for \( k = 1, \ldots, n \) and \( v'(X) = v(X) \), otherwise, and let \( \tau_{1,1} \leq \tau_{2,1} \) be any atomic formula contained in
\[
\varphi = (\tau_{1,1} \cup \tau_{2,1}) \star \Xi.
\]
We prove \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v) \leq \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v) \).
By substitutivity, \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v) = \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v) \), \( j = 1, 2 \).

By the union theorem for \( MU \), \( v'(X_k) = \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v) = \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v) \).
Let variable valuations \( v_i \) be defined by \( v_i(x_k) = \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v) \), \( i = 0 \).
Then \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v) = \bigcup_{i=0}^\infty \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i) \), \( j = 1, 2 \), by continuity.

Therefore we must prove \( \bigcup_{i=0}^\infty \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i) \leq \bigcup_{i=0}^\infty \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i) \) in order to obtain the desired result.

It is sufficient to prove \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i) \leq \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i) \) by induction on \( i \).
For \( i = 0 \), \( v_0 = \{ x_k \} \), whence \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_0) \leq \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_0) \) follows by substitutivity from validity of \( \varphi \models v \models v_0 \).

Assume as hypothesis \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i) \leq \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i), \quad i \in L \), and prove \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i+1) \leq \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i+1), \quad i \in L \).

Validity of \( \varphi \models v \models v_i \) implies in particular that if \( \sigma \) and \( \psi \) satisfy \( v_i \), \( v_i \models v_i \) satisfies \( v_i \). Now \( \sigma \) satisfies \( v_i \) by an argument similar to the one above for \( i = 0 \). By hypothesis, \( \varphi \) satisfies \( v_i \).

Therefore we conclude that \( \varphi v_i \sigma \models v_i \models v_i \) satisfies \( \varphi \) in particular \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i) \leq \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i) \) by definitions.

Thus we conclude \( \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i+1) \leq \varphi_{x_k} x_1 \ldots x_n [\sigma_1, \ldots, \sigma_n](v_i+1), \quad i \in L \).

Finally we define the mapping \( \text{tr} : PL \to MU \) (compare section 1.2) and prove the translation theorem.
DEFINITION 3.5. (tr). The mapping $\text{tr}$ of program schemes of PL into terms of $\mathcal{M}_t$ is defined as follows: consider a program scheme $T = \langle \mathcal{P}_k \leftarrow \mathcal{S}_k \rangle_{k=1, \ldots, n}$, with all procedure symbols contained in $S$ amongst those denoted by $\mathcal{P}_1, \ldots, \mathcal{P}_n$, then $\text{tr}(T)$ is inductively defined by

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

b. $\text{tr}(\mathcal{P}_i) = \mu X_1 \ldots X_n [\text{tr}(\mathcal{S}_i_1), \ldots, \text{tr}(\mathcal{S}_n_1)]$, $i = 1, \ldots, n$.

c. $\text{tr}(\mathcal{S}_1; \mathcal{S}_2) = \text{tr}(\mathcal{S}_1); \text{tr}(\mathcal{S}_2)$, $\text{tr}((p + \mathcal{S}_1, \mathcal{S}_2) = p; \text{tr}(\mathcal{S}_1) \cup \text{tr}(\mathcal{S}_2)$ and

\[ \text{tr}(\mathcal{S}_1, \ldots, \mathcal{S}_n) = \text{tr}(\mathcal{S}_1); \ldots; \text{tr}(\mathcal{S}_n) \]

with $\tau_i$ of type $\langle \mathcal{E}_1 \times \ldots \times \mathcal{E}_n \rangle$, $i = 1, \ldots, n$.

THEOREM 3.3. (Translation theorem). Let $o$ be an operational interpretation of PL, $m$ be a mathematical interpretation of $\mathcal{M}_t$, 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 $o(p)(x) = \text{true}$ iff $\langle x, x \rangle \in m(p)$ and $o(p)(x) = \text{false}$ iff $\langle x, x \rangle \in m(p')$. Then $o(T) = m(\text{tr}(T))$ for all $T \in \mathcal{P}_t$, i.e., $\text{tr}$ 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 $D = \{ \mathcal{P}_j \leftarrow \mathcal{S}_j \}_{j=1, \ldots, n}$. 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 $c : \mathcal{P}_t \rightarrow N \cup N \times \{0\}$, defined by

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

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

c. If $S = \mathcal{S}_1; \mathcal{S}_2$, $S = (p + \mathcal{S}_1, \mathcal{S}_2)$, let $x$ or $\langle x, 0 \rangle$ be the maximum of $c(\mathcal{S}_1)$ and $c(\mathcal{S}_2)$ under the well-order $\prec$. Then $c(\mathcal{S}_1; \mathcal{S}_2)$ and $c(p + \mathcal{S}_1, \mathcal{S}_2)$ are defined as $x + 1$ or $\langle x + 1, 0 \rangle$.

d. If $S = [\mathcal{S}_1, \ldots, \mathcal{S}_n]$ let $x$ or $\langle x, 0 \rangle$ be the maximum of $c(\mathcal{S}_1), \ldots, c(\mathcal{S}_n)$ under the well-order $\prec$. Then $c(\mathcal{S}_1, \ldots, \mathcal{S}_n)$ is defined as $x + 1$ or $\langle x + 1, 0 \rangle$.

Thus $c(\mathcal{S}_1) \nsubseteq c(\mathcal{S}_1; \mathcal{S}_2)$ and $c(\mathcal{S}_1) \nsubseteq c(p + \mathcal{S}_1, \mathcal{S}_2)$ for $i = 1, 2$,

c(\mathcal{S}_1) \nsubseteq c(\mathcal{S}_1, \ldots, \mathcal{S}_n)$, $i = 1, \ldots, n$, and $c(S(k)) \nsubseteq c(\mathcal{P}_j)$ 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 \( S \in A \lor C \lor X \) then \( o(S) = m(tr(S)) \) is obvious.

b. If \( S = S_1;S_2 \) then \( o(S_1;S_2) = (\text{lemma 2.1}) o(S_1);o(S_2) = (\text{induction hypothesis}) m(tr(S_1);m(tr(S_2)) = m(tr(S_1);tr(S_2)) = m(tr(S_1;S_2)). \)

c. If \( S = p + S_1;S_2 \) then \( o(p + S_1;S_2) = (\text{lemma 2.1}) m(p);o(S_1 \lor S_2) = m(p);m(tr(S_1);m(tr(S_2)) = m(tr(p + S_1;S_2)). \)

d. If \( S = S_1 \ldots S_n \) then \( o(S) = (\text{lemma 2.1}) \bigwedge_{i=1}^n o(S_i);o(\bar{x}_1 \wedge \ldots \wedge \bar{x}_n) \text{ (induction hypothesis)} m(tr(S_1);m(tr(S_2);\ldots;m(tr(S_n);m(\bar{x}_1 \wedge \ldots \wedge \bar{x}_n) = m(tr(S_1;\ldots;S_n)). \)

e. If \( S = P_j \) then \( o(P_j) = \bigwedge_{i=0}^\infty o(P_j(i)) = (\text{lemma 2.4}) \)

\[ \bigwedge_{i=0}^\infty o(S_j(i)) = (\text{induction hypothesis}) \bigwedge_{i=0}^\infty m(tr(S_j(i))). \]

\( \bigwedge_{i=0}^\infty m(tr(S_j(i))) = \bigwedge_{i=0}^\infty m(tr(\bar{x}_j)^i) = (\text{union theorem for } MU) \)

\( m(\bigcup_{i=0}^\infty x_1 \ldots x_n [tr(\bar{x})_1;\ldots;tr(\bar{x})_n] = m(tr(P_j)), j = 1,\ldots,n. \quad \square \)

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 $\nu$-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 $\forall^n, \exists^n, E^n$ and $\forall^n, \exists^n$ 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 ",", ",", "," and "," operators, i.e., the "," 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 $\forall$ and $\exists$).

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

   $T_1 : \vdash (x^n; y^n; z^n; \varepsilon; \xi) z^n, \xi = x^n; y^n; z^n; \varepsilon; \xi$
   $T_2 : \vdash x^n, \xi = x^n, \xi$
\( T_4 : \vdash x^\delta \in E^\cdot ; E^\cdot ; E^\cdot = x^{\delta, i} \)

\( T_5 : (x^n, \theta ; y^n, \theta ) \in z^n, \xi = \nu^n, \xi \vdash (y^n, \xi ; z^n, \xi ) \in \xi^n, \theta = \nu^n, \theta \)

3. \( u : \vdash u^n, \xi \leq y^n, \theta ; y^n, \xi \)

4. The substitution rule:

\[
\text{if } \varphi \vdash \psi \text{ then } \forall \{ \tau_j \} \in J \vdash \forall \{ \tau_j \} \in J,
\]

for all suitable \( \varphi, \psi, J, \{ \tau_j \} \in J \) 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 "\n", which has in turn priority over "\u".

**Lemma 4.1.**

a. \( x^n, \xi \leq y^n, \xi \vdash x^n, \xi , \xi , z^n, \theta \leq x^n, \xi , z^n, \theta , z^n, \theta , x^n, \xi \leq x^n, \theta , y^n, \xi \)

b. \( \vdash \nu^n, \xi , x^n, \xi \in \xi^n, \xi \theta \leq \nu^n, \xi , \xi , \theta = \nu^n, \xi \)

c. \( \vdash E^n, \xi , x^n, \xi = x^n, \xi \)

d. \( \vdash y^n, \xi , y^n, \xi = y^n, \theta \)

e. \( \vdash y^n, \xi = x^n, \theta \epsilon, \nu^n, \xi = \nu^n, \xi \)

f. \( \vdash x^n, \xi ; (y^n, \theta ; z^n, \theta ) = x^n, \xi , y^n, \theta , z^n, \theta = x^n, \theta , z^n, \theta , y^n, \theta = x^n, \theta , z^n, \theta , y^n, \theta = x^n, \theta , z^n, \theta , y^n, \theta \)

g. \( \vdash (x^n, \xi \cup y^n, \xi ) = x^n, \xi \cup y^n, \xi , (x^n, \xi \cap y^n, \xi ) = x^n, \xi \cap y^n, \xi , (x^n, \xi , y^n, \xi ) = x^n, \xi , y^n, \xi , x^n, \xi = x^n, \xi \)

**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 TARKSI [61].

Lemma 4.1.a expresses monotonicity of "\n" and "\;". Together with the obvious monotonicity of "\u" and "\n", 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.** \( \models x ; y \cap z = x; (\bar{x}; z \cap y) \cap z^* \).

*Proof.* \( x ; y \cap z = x; (u \cap y) \cap z = x; ((\bar{x}; z \cup \bar{x}; z) \cap y) \cap z = \{ x; (\bar{x}; z \cap y) \cap z \} \cup \{ x; (\bar{x}; z \cap y) \cap z \} \). Also \( \bar{x}; z \cap \bar{x}; z = \emptyset \), whence by \( T_2 \), \( x; (\bar{x}; z^* \cap y) \cap z = \emptyset \). Therefore, \( x; (\bar{x}; z \cap y) \cap z = \emptyset \), whence \( x ; y \cap z = x; (\bar{x}; z \cap y) \cap z \) 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 \( x* e \) has been defined as \( x; u \cap e \) (section 1.3). By convention the "*" operator has a higher priority than the ";" operator.

**Lemma 4.3.**

a. \( \bar{x}; z \subseteq e \models x; y \cap z = x; y \cap x; z \)

b. \( x \subseteq e \models x = \bar{x} \)

c. \( x = x* e \cap x ; x = x; \bar{x} \cap e \subseteq x ; x ; u = x* e ; u \)

d. \( x \subseteq y \subseteq y \models x* e ; y = x \)

e. \( \bigcap_{i=1}^{n} x_i; y_i = x_1; e \cap \ldots \cap x_n* e ; (\bigcap_{i=1}^{n} x_i; y_i); \bar{y}_1; e \cap \ldots \cap \bar{y}_n* e \).

*Proof.*

a. \( \models x ; y \cap z = (\text{lemma 4.2}) x; (\bar{x}; z \cap y) \cap z \subseteq (\text{assumption}) x; (y \cap z) \).

b. \( x = x \cap e = (\text{lemma 4.2}) x; (\bar{x}; e \cap e) \cap e \subseteq x; \bar{x} \subseteq \bar{x} \). Thus \( x \subseteq \bar{x} \), whence \( \bar{x} \subseteq x \).

c. \( x = x* e \cap x ; x = (\text{lemma 4.2}) \bar{x} ; (x; u \cap e) \cap u = \bar{x} ; (x; u \cap e) \).

Thus, by \( T_2 \), \( x = (x; u \cap e) \cap x = (\text{part b}) x* e \cap x \).

\( x* e = x; \bar{x} \cap e \). Direct from lemma 4.2.

\( x; u = x* e \cap u \cap x; u = (\text{from above}) (x; u \cap e) ; x; u \subseteq (\text{lemma 4.1}) x* e \cap u \subseteq x; u ; u = x ; u \).

d. \( x \subseteq y \models \bar{y}; x \subseteq \bar{y}; y \subseteq (\text{assumption}) e \), whence \( x; \bar{x}; y \subseteq (\text{part b}) \) and \( T_2 \) \( x \) and \( (x; \bar{x} \cap e) ; y \subseteq x; \bar{x}; y \subseteq x \).

e. We prove \( x ; y \cap z = x* e ; (x; y \cap z) \) only. \( \models \). Obvious.

\( x ; y \cap z = (\text{part c}) x* e \cap x ; y \cap z = (\text{part b and lemma 4.2}) x* e \cap (x* e \cap z \cap y) \cap z \subseteq x* e ; (x; y \cap z) \). 

*) This assertion can also be proved without making use of the "*" 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

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

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.\(^\text{a)}\) Therefore this assertion does not follow from our axiomatization of $\mu_0$\(^\text{**}\) 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 $\langle p^{n,n}, p^{n,n} \rangle$, whose interpretation is restricted to pairs of disjoint subsets of the identity relation corresponding to inverse images of true and false. $\mu_1$ 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 $\mu_0$:

$$\text{P}_1 : \vdash p^{n,n} \subseteq E^{n,n}, \ p^{n,n} = E^{n,n},$$

$$\text{P}_2 : \vdash p^{n,n} \cap p^{n,n} = \emptyset^{n,n}.$$

\(^\text{a)}\) Properly speaking, LYNDON constructs a relation algebra in which

if $X_1 X_2 \cap Y_1 Y_2 \cap Z_1 Z_2 \neq \emptyset$ then

$(\tilde{X}_1 Y_1 \cap \tilde{X}_2 \tilde{Y}_2 \cap (\tilde{X}_1 Z_1 \cap \tilde{X}_2 \tilde{Z}_2) ; (\tilde{Z}_1 Y_1 \cap \tilde{Z}_2 \tilde{Y}_2)) \neq \emptyset$

does not hold.

\(^\text{**}\) However, as demonstrated in appendix 3, this assertion can be proved using our axiomatization of $\mu_2$.
The translation theorem implies \( o(p \times S_1, S_2) = m(p \times \epsilon(S_1) \cup p \times \epsilon(S_2)) \), provided \( o(S_i) = m(\epsilon(S_i)) \), \( i = 1, 2 \), and \( o(p) \) is represented by \( \langle m(p), m(p') \rangle \). Thus axiomatization of \( M_1 \) 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.** \( \vdash \rho = p \), \( p \cap q = p \cap q \).

*Proof.* \( \rho = p \): Follows from lemma 4.3. b, and axiom \( P_1 \).

\( p \cap q \leq p \cap q \): Since \( \vdash p \leq E \), \( q \leq E \), monotonicity implies \( \vdash p \cap q \leq p \). Thus \( \vdash p \cap q \leq p \cap q \).

\( 2. \) \( \vdash p \cap q = (\text{lemma 4.2}) p ; p \cap q \cap E \cap q \leq p ; (p \cap q \cap E) \leq p ; p \cap q \leq p ; q \).

**Corollary 4.1.** Using the notation \( p \times X, Y = p \times Y \), we have

\( \vdash (p + (p \times X, Y, Z)) = (p + X, Z, (p + X, (p + Y, Z))) =

= (p + X, Z, (p + (q + X_1, X_2), (q + (Y + Y_1, Y_2)) = (q + (p + X_1, Y)) + (p + X_2, Y_2)) \).

*Proof.* Immediate from lemma 4.4, using \( P_1 \) and \( P_2 \).

**Corollary 4.2.** \( \vdash p \cap X \cap Y = p ; (X \cap Y) \).

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

\( p ; Y \cap p ; X = (\text{lemma 4.3.a}) p ; (X \cap Y) \).

In section 1.3 we already mentioned the "*" operator, defined by \( X \ast p = X ; p \cup U \cap E \). The basic properties of this operator are collected in \(*\)

**Lemma 4.5.**

\( a. \) \( \vdash (X ; Y) \ast p = X \ast (Y \ast p) \)

\( b. \) \( \vdash (X \cup Y) \ast p = X \ast p \cup Y \ast p \)

\( c. \) \( \vdash (X \cap Y) \ast p = X ; p ; Y \cap E \)

\( d. \) \( \vdash X ; p \geq X \ast p ; X \) \( **\)

\( e. \) \( X \cap X \leq E \vdash X ; p = X \ast p ; X \)

\( f. \) \( X ; p \geq q ; X \vdash X \ast p \leq q \)

*) Some connections between \( \mu \)-terms and the "*" operator are collected in section 5.3.

**) Henk Goman has observed that one can prove \( X ; p = X \ast p ; X \iff X \ast p ; X \leq p ; U ; p \).
Proof. a. By definition, \((X;Y;p) = X;Y;p;U \cap E \text{ and } X;(Y;p) = X;(Y;p;U \cap E);U\), the result follows.

b. Immediate from the definitions and lemma 4.1.

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

d. Applying lemma 4.3.c we obtain \(X;p = (X;p;U \cap E);X;\leq (X;p;U \cap E);X = X;p \cap X\).

e. \(\subseteq\) by part d above.

\[\leq (\text{lemmas } 4.2 \text{ and } 4.4) \ X;p;X;X \leq (\text{lemma } 4.3.c) \ X;U \subseteq (\text{assumption}) \ X;p;U \subseteq (\text{corollary } 4.2) X;p\).

f. Assume \(X;p \leq q;X\). Then \(X;p = X;p;U \leq q;X;U \subseteq (\text{corollary } 4.2) q\).

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):

\[X;p = \Pi (q \mid X;p \leq q;X)\]

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 \(i = 1, \ldots, n\), projection function symbols \(\pi_n^I\) to the basic terms of \(\mu_1\), for all types concerned. \(\mu_2\) is axiomatized by adding the following two axioms to the axioms of \(\mu_1\):

\[
\begin{align*}
C_1 : & \vdash \pi_1^1 \pi_1^2 \ldots \pi_1^n = E \\
C_2 : & \vdash X_1;Y_1 \ldots \pi_1^n;Y_1 = (X_1;X_2 \ldots \pi_1^n;X_2);(\pi_1^1;Y_1 \ldots \pi_1^n;Y_1),
\end{align*}
\]

where \(\pi_i^I\) is of type \(\eta_i^1 \times \ldots \times \eta_i^n, \eta_i^1 \times \ldots \times \eta_i^n\) and \(X_1\) and \(Y_1\) are of types \(\langle 0, \eta_i^1 \rangle\) and \(\langle 1, \xi \rangle\), respectively.

An assignment \(x_i : = f(x_1, \ldots, x_n)\) is expressed by a statement scheme \(V\) of the form \([\pi_i^1, \ldots, \pi_i^n, 0, x_i+1, \ldots, x_n]\). Hence Hoare's axiom for the assignment (cf. [29])

\(\ast\) Note added in print: By a conjecture of PARL 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.
\[ \vdash (p(x_1, \ldots, x_{i-1}, f(x_i, \ldots, x_n), x_{i+1}, \ldots, x_n)) \iff f(x_1, \ldots, x_n) (p(x_1, \ldots, x_n)) \]

corresponds with the assertion \[ \vdash \text{tr}(V) \cdot p : \text{tr}(V) \subseteq \text{tr}(V) ; p, \] as \( q_1(V) q_2 \) is expressed by \( q_1 \text{tr}(V) \subseteq \text{tr}(V) ; q_2 \), and \( (\text{tr}(V) ; p)(x_1, \ldots, x_n) = p(x_1, \ldots, x_{i-1}, f(x_i, \ldots, x_n), x_{i+1}, \ldots, x_n) \) (compare section 1.3). As functionality of \( f \) implies \( \text{tr}(V) ; V \subseteq E \) 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 \( i = 1, \ldots, n \):

a. \[ \vdash \ \eta_1^{x_1} \ldots \eta_n^{x_n} \eta_i^n \eta_i^n \in \text{E}_i \eta_n \eta_1^{x_1} \ldots \eta_n \eta_n \]

b. \[ \vdash \ \eta_1^{x_1} \ldots \eta_n^{x_n} \eta_i^{x_i} \in \text{U} \eta_n \eta_1^{x_1} \ldots \eta_n \eta_n \]

c. \[ \vdash \ \eta_1^{x_1} \eta_2^{x_2} \ldots \eta_n^{x_n} \eta_i^{x_i} \eta_j^n \eta_j^n \in \text{E} \eta_n \eta_1^{x_1} \ldots \eta_n \eta_n \]

d. \[ \vdash \ \eta_1^{x_1} \eta_2^{x_2} \ldots \eta_n^{x_n} \eta_i^{x_i} \eta_j^n \eta_j^n \eta_j^n \in \text{U} \eta_n \eta_1^{x_1} \ldots \eta_n \eta_n \]

**Proof.** a. Let \( n \eta_n \eta_i^n \eta_i^n \), then \( E_n = (C_1) \eta_i^n \eta_i^n \eta_i^n \subseteq E_n \)

b. \( \eta_i^n \eta_i^n \in (\text{U} \eta_n \eta_1^{x_1} \ldots \eta_n \eta_n) \)

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

\[ (\eta_1^{x_1} \eta_1^{x_1} \eta_1^{x_1} \eta_1^{x_1} \eta_2^{x_2} \eta_2^{x_2} \eta_2^{x_2} \eta_2^{x_2}) = (\text{Lemma 4.1 and part b above}) \eta_1^{x_1} \eta_2^{x_2} \]

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

\[ \eta_1^{x_1} \eta_2^{x_2} \in (\text{U} \eta_1^{x_1} \eta_1^{x_1} \eta_2^{x_2} \eta_2^{x_2} \eta_2^{x_2} \eta_2^{x_2}) \]

\[ = (\text{Lemma 4.1 and part b above}) \eta_1^{x_1} \eta_2^{x_2} \]

\[ \square \]
Already in example 1.1 we signalled the analogy between $\prod_{i=1}^{n} \vec{X}_{i}$; $\vec{Y}_{1}$ and a list of parameters called by value. From this point of view properties such as $\prod_{i=1}^{n} X_{i}, \vec{Y}_{1} = \prod_{i=1}^{n} X_{i}, \vec{Y}_{1}$ - the computation of such a list terminates iff the computations of its individual members terminate - and $\prod_{i=1}^{n} X_{i}, \vec{Y}_{1} = (\prod_{i=1}^{n} X_{i}, \vec{Y}_{1}) \vdash \prod_{i=1}^{n} X_{i}, \vec{Y}_{1}$ - 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 $k, l \leq n$,

$$\vdash X_{1} \circ E; \ldots; X_{k} \circ E; (\prod_{i=1}^{n} X_{i}; \prod_{i=1}^{l} Y_{i}) = \prod_{i=1}^{l} X_{i} \circ E; \ldots; \prod_{i=1}^{l} Y_{i} \circ E; \vdash Y_{l} = \prod_{i=1}^{l} X_{i} \circ E; \ldots; \prod_{i=1}^{l} Y_{i} \circ E;$$

for $k \leq l$, $j = 1, \ldots, k$, $i = 1, \ldots, l$, $t = 1, \ldots, n$, with $X_{i}$ of type $\langle \eta_{1}, \ldots, \eta_{n}, \eta_{l} \rangle$, and $X_{i}$ $Y_{i}$ of types $\langle \eta_{i}, \xi_{1} \rangle$ and $\langle \eta_{i}, \xi_{2} \rangle$, respectively.

**Proof.** The case of $n = 3$, $k = 1 = 2$, $i_{1} = 1$, $i_{2} = 2$, $s_{1} = 2$, $s_{2} = 3$ is representative. Hence we prove

$$X_{1} \circ E; X_{2} \circ E; X_{2}; Y_{2}; \prod_{i=1}^{3} \vec{Y}_{i} \circ E = (X_{1}; X_{1} \circ X_{2}; X_{2}; Y_{2}) \circ (X_{1}; X_{2}; Y_{2}) \circ \prod_{i=1}^{3} \vec{Y}_{i} \circ E.$$ 

By lemma 4.6,

$$X_{1} \circ E; X_{2} \circ E; X_{2}; Y_{2} \circ E = (X_{1}; X_{1} \circ X_{2}; X_{2}; Y_{2}) \circ \prod_{i=1}^{3} \vec{Y}_{i} \circ E \quad \text{and}$$

$$X_{1} \circ E; X_{2} \circ E; X_{2}; Y_{2} \circ E = (X_{1}; X_{1} \circ X_{2}; X_{2}; Y_{2}) \circ \prod_{i=1}^{3} \vec{Y}_{i} \circ E \quad \text{and}$$

whence

$$(X_{1}; X_{1} \circ X_{2}; X_{2}; Y_{2}) \circ (X_{1}; X_{1} \circ X_{2}; X_{2}; Y_{2}) \circ \prod_{i=1}^{3} \vec{Y}_{i} \circ E.$$

By corollary 4.2, $X_{1} \circ E; U_{i} \circ E; X_{2}; Y_{2} \circ E; \prod_{i=1}^{3} \vec{Y}_{i} \circ E = X_{1} \circ E; X_{2}; Y_{2} \circ E ; \vec{Y}_{3} \circ E,$

whence the result follows by lemma 4.4. □
COROLLARY 4.3. \( \vdash ( \bigcap_{i=1}^{n} X_{i}; Y_{i}) = ( \bigcap_{i=1}^{n} \alpha_{i}; \beta_{i} \mid \gamma_{i}) = (C_{1}) \bigcap_{i=1}^{n} X_{i}; Y_{i} \), with \( X_{i} \) of type \( \langle \theta, \eta \rangle \) and \( p_{i} \) of type \( \langle \eta_{1}, \eta \rangle \).

Proof. \( ( \bigcap_{i=1}^{n} X_{i}; Y_{i}) = ( \bigcap_{i=1}^{n} \alpha_{i}; \beta_{i} \mid \gamma_{i}) = (C_{1}) \bigcap_{i=1}^{n} X_{i}; Y_{i} \), with \( X_{i} \) of type \( \langle \theta, \eta \rangle \) and \( p_{i} \) of type \( \langle \eta_{1}, \eta \rangle \).

\( = (\text{lemma 4.6.b}) ( \bigcap_{i=1}^{n} X_{i}; Y_{i}) \bigcap_{i=1}^{n} \alpha_{i}; \beta_{i} \mid \gamma_{i} \), with \( X_{i} \) of type \( \langle \theta, \eta \rangle \) and \( p_{i} \) of type \( \langle \eta_{1}, \eta \rangle \).

\( = (\text{lemma 4.7}) (X_{1}; p_{1}) \mid \epsilon \ldots ; (X_{n}; p_{n}) \mid \epsilon \mid X_{1}; p_{1} \bigcap_{i=1}^{n} \alpha_{i}; \beta_{i} \mid \gamma_{i} \), with \( X_{i} \) of type \( \langle \theta, \eta \rangle \) and \( p_{i} \) of type \( \langle \eta_{1}, \eta \rangle \).

\( = (\text{corollary 4.2 and lemma 4.5.a}) X_{1}; p_{1} \mid \epsilon \ldots ; X_{n}; p_{n} \). □

One of the consequences of lemma 4.7 is

\( \vdash ( \bigcap_{i=1}^{n-1} X_{i}; Y_{i}) = ( \bigcap_{i=1}^{n-1} \alpha_{i}; \beta_{i} \mid \gamma_{i}) = (C_{1}) \bigcap_{i=1}^{n-1} X_{i}; Y_{i} \), with \( X_{i} \) and \( Y_{i} \) of types \( \langle \eta_{1}, \ldots, \eta_{n}, \eta_{1} \rangle \), \( \langle \theta, \eta \rangle \) and \( \langle \eta_{1}, \eta \rangle \), respectively.

Assume \( \eta_{1} = \eta_{2} = \ldots = \eta_{n} \) for simplicity, then, apart from the intended interpretation of \( \pi_{x} \) as special subset of \( D^{n} \times n \),

"axiom \( C_{2} \) for \( n=1 \), in which \( \eta_{1}, \ldots, \eta_{n-1} \) are interpreted as subsets of \( D^{n-1} \times D \) "follows from" axiom \( C_{2} \) for \( n, n \geq 2 \).

This line of thought may be pursued as follows:

Change the definition of type in that only compounds \( (\eta_{1}, \ldots, \eta_{n}) \) are considered, and introduce projection function symbols \( \pi_{1}^{(n \times \xi)}, \pi_{2}^{(n \times \xi)} \), \( \xi \) only. For \( n > 2 \) define \( (\eta_{1}, \ldots, \eta_{n}) \) as \((\ldots ((\eta_{1}, \eta_{2}), \eta_{3}), \ldots, \eta_{n})\) and \( \pi_{1}^{(n \times \eta_{1})}, \ldots, \pi_{n}^{(n \times \eta_{n})} \) as

e.g., for \( n = 3 \) and \( i = 1, 2, 3 \), \( \pi_{1}^{(n \times \eta_{1})}, \pi_{2}^{(n \times \eta_{2})}, \pi_{3}^{(n \times \eta_{3})} \), \( \pi_{1}^{(n \times \eta_{1})}, \pi_{2}^{(n \times \eta_{2})}, \pi_{3}^{(n \times \eta_{3})} \).

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 "\( n \)" 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.** \( \models \tilde{x} = \tilde{y}_2 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2) \).

**Proof.** We prove \( X = \tilde{y}_1 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2) \). The result then follows by lemma 4.3.b.

\[
\tilde{y}_1 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2) = (C_1 \tilde{y}_1 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2)) = (\text{lemma 4.3.a}) \tilde{y}_1 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2)
\]

Hence,

\[
\tilde{y}_1 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2) = (C_1 \tilde{y}_1 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2)) = (\text{lemma 4.3.a}) \tilde{y}_1 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2)
\]

Therefore, \( \tilde{y}_1 \circ (E \circ \tilde{y}_1 \circ \tilde{y}_2) = X \) follows.

4.4. **Axiomatization of the "u" operators**

\( MU \) is obtained from \( MU_2 \) by introducing the "\( u \)" 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 \( MU_2 \):

\[
M : \vdash \{u_j X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n/\tilde{y}_1] i=1, \ldots, n \} \subseteq u_j X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n] j=1, \ldots, n
\]

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 \( Y \),

\[ \text{\( \tau \in B_1, \ldots, B_{n+1} \text{ leads to } \tau(X_1, \ldots, X_n, Y) \subseteq \tau(B_1, \ldots, B_{n+1}, i=1, \ldots, n) \)). \]

Then \( Y_1 \leq Y_2 \models (u_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)]) \leq \]

\[ \leq u_j X_1 \ldots X_n[\tau_1(X_1, \ldots, X_n, Y_2), \ldots, \tau_n(X_1, \ldots, X_n, Y_2)])_{j=1, \ldots, n}. \]

b. (Monotonicity). *If \( \tau(X_1, \ldots, X_n) \) is syntactically continuous in

\( X_1, \ldots, X_n \), then \( \tau \) is monotonic in \( X_1, \ldots, X_n \), i.e.,

\( X_1 \leq Y_1, \ldots, X_n \leq Y_n \models \tau(X_1, \ldots, X_n) \leq \tau(Y_1, \ldots, Y_n). \)

c. (Fixed point property). *\( \models \{x_j [u_j X_1 \ldots X_n[\tau_1, \ldots, \tau_n]/X_i]_{i=1, \ldots, n} = \]

\[ = u_j X_1 \ldots X_n[\tau_1, \ldots, \tau_n]_{j=1, \ldots, n}. \]

d. (Least fixed point property, PARK [51]).

\( \models \{x_j [Y_1 \ldots Y_n] \leq Y_j \}_{j=1, \ldots, n} \)

\( \models \{u_j X_1 \ldots X_n[\tau_1, \ldots, \tau_n] \leq Y_j \}_{j=1, \ldots, n}. \)

**Proof.** a. Use I, taking \( \{Y_1 \leq Y_2\} \) for \( \phi \) and

\( \{X_j \leq u_j X_1 \ldots X_n[\tau_1(X_1, \ldots, X_n, Y_2), \ldots, \tau_n(X_1, \ldots, X_n, Y_2)]\}_{j=1, \ldots, n} \) for \( \psi \),

and \( \tau_j(X_1, \ldots, X_n, Y_j) \) for \( \sigma_j \), \( j = 1, \ldots, n. \)

1. \( \models \{\sigma_j \leq u_j X_1 \ldots X_n[\tau_1(X_1, \ldots, X_n, Y_2), \ldots, \tau_n(X_1, \ldots, X_n, Y_2)]\}_{j=1, \ldots, n}. \)

Obvious.

2. \( \phi, \psi \models \{x_j [X_1 \ldots X_n, Y_1] \leq u_j X_1 \ldots X_n[\tau_1(X_1, \ldots, X_n, Y_2), \ldots, \tau_n(X_1, \ldots, X_n, Y_2)]\}_{j=1, \ldots, n}. \)

By monotonicity of \( \tau_j \) in \( X_1, \ldots, X_n \) and \( Y_j \), and \( M. \)

b. Follows by induction on the complexity of \( \tau \), using lemma 4.1.a. and

part a above.

c. \( \equiv \). Use I, with \( \phi \) empty and taking \( \{X_j \leq u_j X_1 \ldots X_n[\tau_1, \ldots, \tau_n]\}_{j=1, \ldots, n} \)

for \( \psi \), proving the induction step with part b above.

d. \( \equiv \). Use I, taking \( \{x_j [Y_1 \ldots Y_n] \leq Y_j \}_{j=1, \ldots, n} \) for \( \phi \) and \( \{X_j \leq Y_j \}_{j=1, \ldots, n} \)

for \( \psi \), proving the induction step with part b above. \( \square \)

Modularity is but one of the many consequences of the generalized iteration lemma below. For \( m = 1 \) this lemma asserts that simultaneous minimization by \( \mu_1 \)-terms is equivalent to successive singular minimization by \( \mu \)-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 \( J \equiv \{q_1, \ldots, q_n\} \), where \( k = m + n \). Then the following holds:

For \( p_i \in I \),

\[
\vdash_{\mu_{p_1}} Z_{m+n} [\rho_1, \ldots, \rho_{m+n}]
\]

\[
= \mu_{i_1} X_{i_1} \ldots X_{i_m} \left[ \sigma_{i_1} \left[ u_{j_1} \ldots u_{j_{n+m}} \left[ \sigma_{m+n} \right] / Y_{i_1} \right] \left[ \sigma_{m+n} \right] / Y_{j_1} \right] \left[ \sigma_{m+n} \right] / Y_{j_2} \ldots
\]

\[
\ldots \left[ \sigma_{m+n} \right] / Y_{j_{n+m}} \left[ \sigma_{m+n} \right] / Y_{j_{n+m+1}} \ldots
\]

where \( \sigma_s \equiv \rho_{p_s} [X_{i_1}/Z_{p_i}] p_i \in I [Y_{j_1}/Z_{q_j}] q_j \in J \), \( s = 1, \ldots, m \), and

\( \sigma_{m+n} \equiv \rho_{q_s} [X_{i_1}/Z_{p_i}] p_i \in I [Y_{j_1}/Z_{q_j}] q_j \in J \), \( s = 1, \ldots, n \).

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 \( I \equiv \{i, \ldots, m\} \) and \( J \equiv \{m+1, \ldots, m+n\} \); the general version should be clear.

We use the following notation:

\( \mu_i \equiv \mu_{i_1} Z_{m+n} [\rho_1, \ldots, \rho_{m+n}] \), \( i = 1, \ldots, m+n \),

\( \theta_i X \equiv \mu_{i_1} X_{i_1} \ldots X_{i_m} \left[ \sigma_{i_1} \left[ u_{j_1} \ldots u_{j_{n+m}} \left[ \sigma_{m+n} \right] / Y_{i_1} \right] \left[ \sigma_{m+n} \right] / Y_{j_1} \right] \left[ \sigma_{m+n} \right] / Y_{j_2} \ldots
\]

\[
\ldots \left[ \sigma_{m+n} \right] / Y_{j_{n+m}} \left[ \sigma_{m+n} \right] / Y_{j_{n+m+1}} \ldots
\]

where \( \sigma_s \equiv \rho_{p_s} [X_{i_1}/Z_{p_i}] p_i \in I [Y_{j_1}/Z_{q_j}] q_j \in J \), \( s = 1, \ldots, m \), and

\( \sigma_{m+n} \equiv \rho_{q_s} [X_{i_1}/Z_{p_i}] p_i \in I [Y_{j_1}/Z_{q_j}] q_j \in J \), \( s = 1, \ldots, n \).

\( \theta_j Y(X_{i_1}, \ldots, X_{i_m}) \equiv \nu_{j_1} Y_{j_1} [\sigma_{m+1}, \ldots, \sigma_{m+n}] \), \( j = 1, \ldots, n \).

First we notice that \( \theta_j Y(\mu_{i_1}, \ldots, \mu_{i_m}) \subseteq \nu_{m+j} \), \( j = 1, \ldots, n \), follows by the least fixed point property (lfp, lemma 4.9.d) from

\( \sigma_{m+j}(\mu_{i_1}, \ldots, \mu_{m+j}, \ldots, \mu_{m+n}) = (\text{fixed point property, fpp, lemma 4.9.c}) \nu_{m+j} \), \( j = 1, \ldots, n \).

Then the result follows also by least fixed point property from

\( \sigma_{i}(\mu_{i_1}, \ldots, \mu_{m+i}, \ldots, \mu_{m+n}) = (\text{monotonicity, lemma 4.9.b}) \sigma_{i}(\mu_{i_1}, \ldots, \mu_{m+i}, \ldots, \mu_{m+n}) = (\text{fpp}) \theta_i \), \( i = 1, \ldots, m \).
COROLLARY 4.4. (Modularity). For $i = 1, \ldots, n$,

\[ \mu_i \subseteq \beta_i X, \quad i = 1, \ldots, m, \quad \text{and} \quad \mu_{i+j} \subseteq \beta_j Y(\beta_1 X, \ldots, \beta_m X), \quad j = 1, \ldots, n. \]

Follows by the least fixed point property from

\[
\sigma_i(\beta_i X, \ldots, \beta_m X, \beta_i Y(\beta_1 X, \ldots, \beta_m X), \ldots, \beta_n Y(\beta_1 X, \ldots, \beta_m X)) = (\text{fpp}) \beta_i X, \quad i = 1, \ldots, m, \quad \text{and} \]

\[
\sigma_{m+j}(\beta_1 X, \ldots, \beta_m X, \beta_1 Y(\beta_1 X, \ldots, \beta_m X), \ldots, \beta_n Y(\beta_1 X, \ldots, \beta_m X)) = (\text{fpp}) \beta_j Y(\beta_1 X, \ldots, \beta_m X), \quad j = 1, \ldots, n. \]

\[ \Box \]

Proof. We use the following notation:

\[
u_i = \mu_i X_1 \ldots X_n [\sigma_i(\tau_{i1}(X_1, \ldots, X_n), \ldots, \tau_{im}(X_1, \ldots, X_n)), \ldots, \tau_{nm}(X_1, \ldots, X_n)] = \sigma_i(\nu_i X_1, \ldots, X_n) \]

\[
u_{i+j} = \nu_{i+m+j} X_1 \ldots X_n [\sigma_i(\tau_{i1}(X_1, \ldots, X_n), \ldots, \tau_{im}(X_1, \ldots, X_n)), \ldots, \tau_{nm}(X_1, \ldots, X_n)], \quad i = 1, \ldots, m, \quad \text{and} \quad j = 1, \ldots, n.

\[ \Box \]

We have to prove: \[ \mu_i = \sigma_i(\nu_i), \quad i = 1, \ldots, n. \]

This result is a straightforward consequence of parts a, b and c below.

a. \[ \nu_i = \beta_i X. \]

(Generalized iteration, lemma 4.10)

\[
u_i X_1 \ldots X_n [\sigma_i(\mu_i X_1 \ldots X_n) \ldots, \tau_{nm}(X_1, \ldots, X_n)] y_j^m = \tau_{i1}(X_1, \ldots, X_n), \ldots, \tau_{im}(X_1, \ldots, X_n)]/y_j^m, \quad j = 1, \ldots, m.

\[ = (\text{fpp}, \text{lemma 4.9.c}) \nu_i. \]

b. \[ \beta_i = (\text{fpp}) \sigma_i(\beta_i X_1, \ldots, \beta_i X_m). \]

c. \[ \beta_{i+j} = \nu_{i+j}. \]

(Generalized iteration)

\[
u_{i+j} X_1 \ldots X_n [\sigma_i(\tau_{i1}(X_1, \ldots, X_n), \ldots, \tau_{im}(X_1, \ldots, X_n))] y_j^m = \tau_{i1}(X_1, \ldots, X_n), \ldots, \tau_{im}(X_1, \ldots, X_n)]/y_j^m, \quad j = 1, \ldots, m.

\[ = (\text{fpp}) \nu_{i+j}. \]
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 $\text{ML}_n$, is a straightforward transformation of the proof given at the end of section 2.2, which uses modularity in $\text{PL}$.

**Corollary 4.5.**

$$\vdash \{u_i X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n] \}^\gamma = u_i X_1 \ldots X_n [\sigma_1(X_1, \ldots, X_n)^\gamma, \ldots, \sigma_n(X_1, \ldots, X_n)^\gamma]_{i=1, \ldots, n}.$$

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

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

The last lemma of this chapter states some sufficient conditions for provability of $\phi \vdash \beta; \rho \subseteq E$, i.e., functionality of $\sigma$, and is frequently applied in combination with lemma 4.5.e $(X; X \subseteq E \vdash X p = X p ; X)$.

**Lemma 4.11.** (Functionality). The assertion $\phi \vdash \beta; \rho \subseteq E$ is provable if one of the following assertions is provable:

a. If $\sigma = \bigcup_{i=1}^n \sigma_i$, then $\phi \vdash \{\sigma_i \in E ; \sigma_j = \sigma_i \mid i < j \leq n \} \cup \{\tilde{\sigma}_i \in E \}_{i=1, \ldots, n}$

b. If $\sigma = \sigma_1 \tilde{\sigma}_1 n \ldots n \sigma_n \tilde{\sigma}_n$, then $\phi \vdash \{\tilde{\sigma}_i \sigma_i \subseteq E \}_{i=1, \ldots, n}$

c. If $\sigma = \sigma_1 \sigma_2$, then $\phi \vdash \tilde{\sigma}_1 \sigma_2 \subseteq E, \tilde{\sigma}_2 \sigma_2 \subseteq E$.

d. If $\sigma = \sigma_1 n \sigma_2$, then $\phi \vdash \tilde{\sigma}_1 \sigma_2 \subseteq E$ or $\phi \vdash \tilde{\sigma}_2 \sigma_2 \subseteq E$ or $\phi \vdash \tilde{\sigma}_1 \sigma_2 \subseteq E$

or $\phi \vdash \tilde{\sigma}_2 \sigma_2 \subseteq E$.

e. If $\sigma = u_i X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n]$, then $\phi \vdash (X_1; X \subseteq E)_{i=1, \ldots, n} \vdash \{\tilde{\sigma}_i \sigma_i \subseteq E \}_{i=1, \ldots, n}$

**Proof.** Straightforward. $\square$

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

1. $[\sigma_1, \ldots, \sigma_n]$ for $\sigma_1 \tilde{\sigma}_1 n \ldots n \sigma_n \tilde{\sigma}_n$.
2. $[\sigma_1 | \ldots | \sigma_n]$ for $\sigma_1 \sigma_1 \tilde{\sigma}_1 n \ldots n \sigma_n \sigma_n \tilde{\sigma}_n$. 
5. APPLICATIONS

5.1. An equivalence due to Morris

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

\[ f(x, y) \iff \text{if } p(x) \text{ then } y \text{ else } h(z(k(x), y)) \]

and

\[ g(x, y) \iff \text{if } p(x) \text{ then } y \text{ else } g(k(x), h(y)). \]

We present a proof in our framework.

The following equivalence is stated without proof:

**Lemma 5.1.** \( \vdash [A_1 | \ldots | A_{i-1} | A_i | A_{i+1} | \ldots | A_n]; \pi_1 = [A_1 | \ldots | A_{i-1} | E | A_{i+1} | \ldots | A_n]; \pi_1 ; A_i. \)

**Theorem 5.1.** (MORRIS)

Let \( F \equiv \nu X [[p,E]; \pi_2 \cup [p' | E]; [K | E]; X; H] \) and \( G \equiv \nu X [[p,E]; \pi_2 \cup [p' | E]; [K | H]; X]. \)

Then

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

**Proof.** Let \( \emptyset \) 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

\[ \vdash \psi(\nu X[\sigma(X)], \nu Y[\tau(Y)]) \ldots \tag{5.1.1} \]

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 \( \mu \)-term.

However, the observation that
\[
\vdash \mu_1 \xy[s(x), \tau(y)] = \mu [s(x)]
\]
and
\[
\vdash \mu_2 \xy[s(x), \tau(y)] = \mu [\tau(y)]
\]
are straightforward applications of the iteration lemma (lemma 4.10), gives us the equivalent assertion
\[
\vdash \forall (\mu_1 \xy[s(x), \tau(y)], \mu_2 \xy[s(x), \tau(y)])
\]
to which Scott's induction rule does apply.

Henceforth, such transitions will be tacitly assumed.

Thus, we have to prove:

1. \( \vdash \forall (\emptyset, \emptyset) \). Obvious.

2. \( X = Y, [E|H]; Y = Y; H \vdash s(x) = \tau(y), [E|H]; \tau(y) = \tau(y); H \).

a. \( s(x) = \tau(y) : [p|E]; \pi_2 \cup [p'|E]; [K|E]; X; H = (\text{hyp.}) \)
\[ [p|E]; \pi_2 \cup [p'|E]; [K|E]; Y; H = (\text{hyp.}) \]
\[ [p|E]; \pi_2 \cup [p'|E]; [K|E]; [E|H]; Y = (C_2) \]
\[ [p|E]; \pi_2 \cup [p'|E]; [K|H]; Y. \]

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

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]:

\[ \mu X[A_1; X \cup A_2; X \cup E] = A_1; (A_2; \mu X[A_1; X \cup E]) \ast E, \quad \ldots \tag{5.2.1} \]

where \( \ast E \) stands for \( \mu X[A; X \cup E] \) and "\( \ast \)" 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 \( \sigma_1(X, Y) = A_1; X \cup Y \) and \( \sigma_2(X) = A_2; X \cup E \), leads to
   \[ \mu \lambda X[A_1; X \cup Y, A_2; X \cup E] = (\text{iteration}) \mu X[A_1; X \cup \mu Y[A_2; X \cup E]] = \quad \]
   \[ = (\text{fpp}) \mu X[A_1; X \cup A_2; X \cup E]. \]

2. \( A_1; \mu X[A_2; X \ast E] = (\mu X[A_1; X \cup E, A_2; X \cup E]; \mu X[A_1; X \cup E, A_2; X \cup Y \cup E]), \)
   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.** \( \mu_1 = \mu_2 = \mu_2 \),

with \( \mu_1 = \mu_1 X[A_1; X \cup Y, A_2; X \cup E] \) and \( \mu_2 = \mu_1 X[A_1; X \cup E, A_2; X; Y \cup E] \), \( i = 1, 2 \).

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

a. \( \sigma_1(\mu_1; \mu_2, \mu_2) = A_1; \mu_1; \mu_2 \cup \mu_2 = (A_1; \mu_1 \cup E); \mu_2 = (\text{fpp}) \mu_1; \mu_2, \)

b. \( \sigma_2(\mu_1; \mu_2) = A_2; \mu_1; \mu_2 \cup E = (\text{fpp}) \mu_2. \)

\( \vDash \): We prove \( \mu_1; \mu_2 \leq \mu_1, \mu_2 \leq \mu_2 \),

with \( \mu_1; \mu_2 \leq \mu_1; \mu_2 \leq \mu_1 \) as obvious consequence.

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

1. \( \vDash \Omega \leq \mu_2, \Omega; \mu_2 \leq \mu_1 \), Obvious.
2. \( X; \nu_2 \subseteq \nu_1, Y \subseteq \nu_2 \vdash \tau_1(X); \nu_2 \subseteq \nu_1, \tau_2(X,Y) \subseteq \nu_2 \).
   a. \( \tau_1(X); \nu_2 = (A_1; X \cup E); \nu_2 \leq (\text{hyp.}) \lambda_1; \nu_1 \cup \nu_2 = (\text{fpp}) \nu_1 \).
   b. \( \tau_2(X,Y) = A_2; X; Y \cup E \leq (\text{hyp.}) A_2; X; \nu_2 \cup E \leq (\text{hyp.}) A_2; \nu_1 \cup E = \text{fpp} \nu_2 \).

5.3. Wright's regularization of linear procedures

In [65] WRIGHT obtains the following results:

a. The class of recursively enumerable subsets of \( \mathbb{N}^2 \) is the smallest class of sets with the successor relation \( S \) as member and closed under the operations "\( \sim \)", "\( ; \)" and "\( \mu(XQ \cup P_1X; R) \)". Where \( Q, P \) and \( R \) are subsets of \( \mathbb{N}^2 \) 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 \( D \). This generalization is:

\[ \vdash \mu(Q \cup P; X; R) = \nu_1; \nu \subseteq \mu([F; \bar{R}]; Y \subseteq E \cap \nu_1; Q; \bar{Y}_2); \nu_2 \quad (5.3.1) \]

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

The following two auxiliary lemmas are needed:

**Lemma 5.2.** \( \vdash [A; B] = \mu(E \cap \nu_1; A; \bar{Y}_1; P; \nu_2; \bar{Y}_2; \bar{Y}_2). \)

**Proof.** Straightforward from lemma 4.5.c. \( \Box \)

**Lemma 5.3.** \( \vdash \mu(A; X \cup B; \nu) = \mu(A; X \cup B; \nu) \).

**Proof.** Amounts to a straightforward application of Scott's induction rule. \( \Box \)

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

**Theorem 5.3.** (Wright)

\[ \vdash \mu(Q \cup P; X; R) = \nu_1; \mu([E \cap \nu_1; Q; \bar{Y}_2] \cup [F; \bar{R}]; X) = \nu_2. \]

\[ L \quad R \]
Proof. \( \leq \): Follows by the least fixed point property from:

\[ \varpi_1; R \circ E \vdash \pi_2 \triangleq (fpp) \varpi_1; ((E \cap \pi_1; Q; \varpi_2) \cup [P[R] ; R] \circ E) \vdash \pi_2 = (\text{lemma 4.5.a}) \]
\[ \varpi_1; (E \cap \pi_1; Q; \varpi_2) \vdash \pi_2 \cup \varpi_1; [P[R] ; R \circ E] \vdash \pi_2 = (\text{lemma 4.8}) \]
\[ Q \cup \varpi_1; [P[R] ; R \circ E] \vdash \pi_2 = (\text{lemma 5.2}) \]
\[ Q \cup \varpi_1; (E \cap \pi_1; P; \varpi_1; R; \varpi_2) \vdash \pi_2 = (\text{lemma 4.8}) \]
\[ Q \cup P; \varpi_1; R \circ E \vdash \pi_2 \leq R. \]

\( \geq \): One derives by similar techniques:

\[ \varpi_1; ((E \cap \pi_1; Q; \varpi_2) \cup [P[R] ; E \cap \pi_1; L; \pi_2]) \vdash \pi_2 = L, \]

whence by lemmas 4.8 and 5.2

\[ (E \cap \pi_1; Q; \varpi_2) \cup [P[R] ; E \cap \pi_1; L; \varpi_2) \leq E \cap \pi_1; L; \varpi_2, \]

and by the least fixed point property

\[ R \circ E \leq E \cap \pi_1; L; \varpi_2 \leq \pi_1; L; \varpi_2. \]

By lemma 4.6.c one therefore obtains

\[ \varpi_1; R \circ E \vdash \pi_2 \leq L. \]

The reader might notice that \( \varpi_1; iuX(\pi_1; Q; \varpi_2 \cap E) \cup [P[R] ; X] \circ E \vdash \pi_2 \) does not correspond with any program scheme. Using work of GARLAND and LUCKHAM [23] this has been remedied in I. GUESARIAN [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 \( p_0^{n,n} \) and an individual relation constant \( S^{n,n} \). These satisfy:

\[
\begin{align*}
N_1 & : \vdash S;S \cap p_0 = \emptyset. \\
N_2 & : \vdash S;S \subseteq E, \\
N_3 & : \vdash S;S = E , \\
N_4 & : \vdash E \subseteq \nu X[p_0 \cup S;S].
\end{align*}
\]

Clearly, the intended interpretation of \( p_0 \) is \( \langle 0,0 \rangle \) and of \( S \) is \( \langle n,n+1 \rangle \mid n \in N \). 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 \( \bigcup_{j \in J} N_j \) of \( |J| \) copies of \( N \), \( m_j(p_0) \) be \( \langle \langle 0, j \rangle, \langle 0, j \rangle \rangle \mid j \in J \) and \( m_j(S) \) be \( \langle \langle n, j \rangle, \langle n+1, j \rangle \rangle \mid n \in N, j \in J \).

Then \( \langle D_j, m_j(p_0), m_j(S) \rangle \) satisfies \( N_1, N_2, N_3 \) and \( N_4 \).

Let \( R^* \equiv \nu X[R;X \cup E] \). Note that

\[
\vdash \nu X[R;X \cup E] = \nu X[X;R \cup E] \tag{5.4.1}
\]

is a consequence of Scott's induction rule.

Then we exclude disjoint copies of \( N \) from being models by replacing \( N_4^* \) by

\[
N_4^* : \vdash \nu \subseteq S^*;p_0;S^*.
\]
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 $<a, b> \in \mathbb{U}$, whence $N_4$ implies $<a, b> \in \mathbb{S}^*; p_0; S^*$. By $N_1$
and $N_2$, $<a, a> \in \mathbb{S}^*$ and $<b, b> \in \mathbb{S}^*$ are the only pairs contained
in $\mathbb{S}^*$ and $S^*$ with $a$ as first and $b$ as second element, respectively.
Therefore, by definition of ";", $<a, b> \in p_0$, and this contradicts
$p_0 \in \mathbb{E}$.

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 atmicity of $p_0$ is derived. Following example 1.2.f this is expressed by

**LEMMA 5.4.** $\vdash p_0; U \cap U; p_0 \equiv p_0^*$.  

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

Secondly, $N_4^*$ follows from

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

**Proof.** $\vdash \mathbb{E} = \mu X[p_0 \cup \mathbb{S}; X; S]$ by Scott's induction rule.

Then the result follows from $N_4^*$.

We prove

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

As

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

the proof of this splits into two parts:

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

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

*) 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.
\[ = (N_1) \tilde{s}; s \cap \tilde{s}; x; x; s^*; s \subseteq (\text{hyp.}, \text{lemma 4.3.a}) \tilde{s}; u; x; [p_0 \cup \tilde{s}; x; x]; s \subseteq (\text{fpp}) u; x; [p_0 \cup \tilde{s}; x; x]. \]

\[ \vdash \text{Straightforward from Scott's induction rule.} \]

Let \( eq \) stand for \( u; x; [p_0 \uparrow p_0] \vee [\tilde{s}; \tilde{s}]; x; [s, s] \).

Clearly, \( \langle n, m \rangle, \langle n, m \rangle \in eq \iff n = m \). In relational formulation, this amounts to

**Lemma 5.6.** \( \vdash eq; \pi_1 = \pi_2 \) \hfill (5.4.2)

**Proof.** First we prove \( \vdash [p_0 \uparrow p_0]; \pi_1 = [p_0 \uparrow p_0]; \pi_2 \) \hfill (5.4.3)

a. \( [p_0 \uparrow p_0]; \pi_1 = (\text{lemma 4.6.b}) (\pi_1 \uparrow p_0; \pi_1 \cap \pi_2 \uparrow p_0; \pi_2); (\pi_1 \cap \pi_2; u) = \)

\[ = (C_2) (\pi_1 \uparrow p_0 \cap \pi_2 \uparrow p_0; \pi_2) = \text{(lemma 4.3.e)} \pi_1 \uparrow p_0 \cap \pi_2 \uparrow p_0; \pi_0 = \]

\[ = \text{(lemma 5.4 and monotonicity)} \pi_1 \uparrow p_0 \cap \pi_2 \uparrow p_0. \]

b. \( [p_0 \uparrow p_0]; \pi_2 = \pi_1 \uparrow p_0 \cap \pi_2 \uparrow p_0 \) is similarly derived.

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

Next we prove (5.4.2).

\[ \vdash \text{Use Scott's induction rule on eq. By lemma 5.5 we have to prove parts d and e below:} \]

\[ \vdash [p_0 \uparrow p_0]; \pi_1 = [p_0 \uparrow p_0]; \pi_2 \subseteq \pi_2. \]

\[ \vdash x; s_1 \subseteq \pi_2 \vdash [\tilde{s}; \tilde{s}]; x; [s, s]; s \subseteq \pi_2. \]

\[ [\tilde{s}; \tilde{s}]; x; [s, s]; s; \pi_2 = [\tilde{s}; \tilde{s}]; x; \pi_1; s \subseteq [\tilde{s}; \tilde{s}]; \pi_2; s = \pi_2 \tilde{s}; s \subseteq \pi_2. \]

\[ \vdash \text{Similarly.} \]

#### 5.5. The primitive recursion theorem

This is the following theorem:

**Theorem 5.4.** Let \( G: N^2 \to N \) 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 \( x_1, \ldots, x_n, y \in N \):

\[ F(x_1, \ldots, x_n, y) = \begin{cases} y & \text{if } y = 0 \\ \text{G}(x_1, \ldots, x_n) \end{cases} \text{ else} \]
\[ H(x_1, \ldots, x_n, y-1, p(x_1, \ldots, x_n, y-1)) \]...

(5.5.1)

**Proof.** To simplify the notation we take \( n = 1 \).

The minimal solution of (5.5.1) is

\[
\mu \mathcal{X}[[E][P_0]; \pi_1; G \cup \{ \pi_1, \pi_2; \tilde{S}, [E][\tilde{S}]; H \}; \mu].
\]

We prove below that \( \mu \) is total. By the least fixed point property, then certainly \( \mu \in \mathcal{F} \), if \( \mathcal{F} \) is any solution of (5.5.1). If \( \mathcal{F} \) is a function, then \( \mu \in \mathcal{F} \) implies by lemma 4.3.4 that \( \mu = \mu \circ \mathcal{F} \), whence \( \mu = \mathcal{F} \) follows from totality of \( \mu \). It remains to be demonstrated that such an \( \mathcal{F} \) exists, i.e., \( \mu \) is functional; this follows from Scott's induction rule by repeated application of lemma 4.11. \( \square \)

**Lemma 5.7.** \( G \circ E^{1,1} = E^{1,1} \), \( H \circ E^{1,1} = E^{3,3} \) \( \vdash E^2,2 \leq \mu \circ U^{1,2} \),

with \( o^{j,k} \in o^{N \times N, \ldots, N, N \times N, \ldots, N} \),

\( j \) times \( k \) times

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

(5.5.2)

Then

\( \vdash E^2,2 = [E^{1,1} \mathcal{X}[P_0 \cup \tilde{S}; S]] \)

holds by lemma 5.5 and

\( \vdash [E^{1,1} \mathcal{X}[P_0 \cup \tilde{S}; S]] \leq \mu \circ U^{1,2} \)

follows from Scott's induction rule as proved below, whence the result.

We prove the induction step only:

\[ [E^{1,1}[x] \leq \mu \circ U^{1,2} \vdash [E^{1,1}[P_0 \cup \tilde{S}; S]] \leq \mu \circ U^{1,2}. \]

\( \mu \circ U^{1,2} = (fpp) [E][P_0]; \pi_1; G; U^{1,2} \cup \{ \pi_1, \pi_2; \tilde{S}, [E][\tilde{S}]; \mu \}; H; U^{1,2} \)

\( = (\text{lemma 4.3.4 by totality of } \pi_1, G \text{ and } H) \)

\[ [E][P_0]; U^{2,2} \cup \{ \pi_1, \pi_2; [E][\tilde{S}]; \mu \}; U^{3,2} \]

\( = (\text{lemma 4.6.b}) \)

\[ [E][P_0]; U^{2,2} \cup \{ \pi_1, \pi_2; [E][\tilde{S}]; \mu \}; \pi_1; U^{1,2} \land \pi_2; U^{1,2} \land \pi_3; U^{1,2} \]
\[
= [E|p_0];u^2,2 \cup (\tau_2;S;u^1,2 \cap [E|\tilde{S}];\tau;u^1,2)
\geq [E|p_0];u^2,2 \cup [E|\tilde{S}];\tau;u^1,2;[E|S]
\geq \text{(hypo.)} [E|p_0] \cup \tilde{S}X;S].
\]

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 \( \text{car}(t) \) and \( \text{cdr}(t) \), if possible; otherwise, \( t \) is an atom and satisfies the predicate \( \text{at} \), whence \( \text{at}(t) = t \),

(2) constructing a new binary tree from two old ones by application of the function \( \text{cons} \),

where \( \text{car} \), \( \text{cdr} \) and \( \text{cons} \) are related by \( \text{car} = \text{cons}; 1 \) and \( \text{cdr} = \text{cons}; 2 \).

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

\[
\begin{align*}
L_1 : & \vdash \text{cons}; \text{cons} = \text{E}^{\text{\( \eta \)}}
L_2 : & \vdash \text{cons}; \text{cons} \subseteq \text{E}^{\text{\( \eta \)}}
L_3 : & \vdash \text{at} \land \text{cons}; \text{cons} = \text{\( \Omega \)}^{\text{\( \eta \)}}
L_4 : & \vdash \text{E}^{\text{\( \eta \)}} \leq \mu X [\text{at} \lor (\text{cons}; 1; \text{X}; \text{cons}; 2; \text{X}); \text{cons}].
\end{align*}
\]

Remark. 1. \( L_1 \) implies that \( \text{cons} \) is total and also that \( \text{cons} \) is a function; hence \( \text{cons}; 1 \) and \( \text{cons}; 2 \) are also functions. \( L_2 \) yields that \( \text{cons} \) 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 $<D_n, \text{at}, \text{cons}>$ as a structure of lists. This leads us to introduce a new type, $L$, reserved for lists, resulting in $<L, L>$ and $<L \times L>$ 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 $\text{car}(l)$ is always an atom.

Thus we obtain axioms for linear lists by replacing $L_1$ by

$$L L_1 : \vdash \text{cons}; \text{cons} = [\pi_1; \text{at}, \pi_2],$$

and postulating $L_2$, $L_3$ and $L_4$.

The reader may wonder why we didn't replace $L_4$ by

$$\vdash \text{E}^n \cdot \pi \leq \mu X\left[\text{at} \cup \left[\text{car}, \text{cdr}; X\right]; \text{cons}\right].$$

The reason for this is, that $L L_1$, $L_2$ and $L_3$ imply

$$\vdash \mu X\left[\text{at} \cup \left[\text{car}; X, \text{cdr}; X\right]; \text{cons}\right] = \mu X\left[\text{at} \cup \left[\text{car}, \text{cdr}; X\right]; \text{cons}\right], \ldots (6.1.2)$$

cf. the proof of lemma 6.1.b.

$L L$ 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 $\mu X[\text{cons} \cup [\pi_1; \text{car}, [\pi_1; \text{cdr}, \pi_2]; X]; \text{cons}]$, i.e.,

$$\text{conc}(l_1, l_2) \iff \begin{cases} \text{if} \ \text{atom}(l_1) \ \text{then} \ \text{cons}(l_1, l_2) \ \text{else} \ \text{cons}(\text{car}(l_1), \text{conc}((\text{cdr}(l_1), l_2))), \ldots (6.1.3) \end{cases}$$

then conc is associative, i.e., $\text{conc}((\text{conc}(l_1, l_2), l_3)) = \text{conc}(l_1, (\text{conc}(l_2, l_3)))$, cf. McCarthy [45],

(2) if first and last stand for $(\text{at} \cup \text{car})$ and $\mu X[\text{at} \cup \text{cdr}; X]$, \ldots (6.1.4)
respectively, then conc\1/first = π₁;first and conc\1/last = π₂;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 analogous 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 \(l_1\) and \(l_2\): \(l_1 \prec l_2\), but introduce the characteristic predicate \(\prec\) of this relation: \(< l_1,l_2 > \iff l_1 \prec l_2\), i.e., \(\prec = \pi_1;\prec'_2 \cap E\). ... (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 \(\text{conc}\cdot E = \prec\), i.e., \(\text{conc}(l_1,l_2)\) is defined iff \(l_1 \prec l_2\).

In order to axiomatize ordered linear lists we introduce therefore a boolean constant \(\prec^{\mathbf{n}\mathbf{n}}\), replace \(LL_1\) by \(\vdash \text{cons};\text{cons} = [\pi_1;\text{at},\pi_2] \prec\), i.e., \(\text{car}(l),\text{cdr}(l) \prec \text{car}(l),\text{cdr}(l)\), and stipulate that \(\text{at}_i;\text{at}_{i+1} \prec \prec \text{at}_i;\text{at}_{i+1}\) holds for all subsequent atoms \(\text{at}_i\) and \(\text{at}_{i+1}\) which constitute an ordered linear list. This leads to the following axioms for ordered linear lists:

\[
\begin{align*}
\text{OLL}_1 : & \quad \vdash \text{cons};\text{cons} = [\pi_1;\text{at},\pi_2] \prec \\
\text{OLL}_2 : & \quad \vdash \text{cons};\text{cons} \subseteq E^{\mathbf{n}\mathbf{n}} \\
\text{OLL}_3 : & \quad \vdash \text{at} \cap \text{cons};\text{cons} = \emptyset^{\mathbf{n}\mathbf{n}} \\
\text{OLL}_4 : & \quad \vdash E^{\mathbf{n}\mathbf{n}} \subseteq \mu \chi \text{at} \cup \text{car},\text{cdr}X;\text{cons} \quad *) \\
\text{OLL}_5 : & \quad \vdash \prec = [\pi_1;\text{last},\pi_2;\text{first}] \prec,
\end{align*}
\]

with last and first as defined in (6.1.4).

Remarks. OLL is introduced as type for ordered linear lists and \((\text{at} \cup [\text{car},\text{cdr};X];\text{cons})\) will be referred to as \(\tau_{\text{OLL}}\). Then \(\text{OLL}_4\) reads as \(\vdash E^{\mathbf{n}\mathbf{n}} \subseteq \mu \chi \tau_{\text{OLL}}\).

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 \((\text{car}, \text{cdr})\);cons (or \(\text{cons};\text{cons}\), which is equivalent) then the following properties hold for

a. Lists: \(\vdash E = \mu X[\text{at} \cup (\text{car}, \text{cdr}, X); \text{cons}],\) at \(\cup\) at' = E, cons;at' = cons, cons;at = \(\Omega\).

b. Linear lists: \(\vdash E = \mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}],\) cons;cons = \(\pi_1\);at, car;at = car, car;at' = \(\Omega\).

c. Ordered linear lists: \(\vdash\) cons;cons = \(\pi_1\);at;\<.

Proof. a. \(E = \mu X[\text{at} \cup (\text{car}; \text{cdr}; X); \text{cons}],\) \(\leq\). Axiom \(L_4\).

\(\geq\). Use I with \(\emptyset\) empty, taking \(X \subseteq E\) for \(\forall\) and \((\text{at} \cup (\text{car}; \text{cdr}; X); \text{cons})\) for \(\forall\).

at \(\cup\) at' = E : \(E = \mu X[\text{at} \cup (\text{car}; \text{cdr}; X); \text{cons}] = (\text{fpp})\) at \(\cup (\text{car}, \text{cdr}; X); \text{cons}\).

cons;at' = cons : cons;at' = cons;cons;cons = \(L_1\) cons.

cons;at = \(\Omega\) : cons;at = cons;cons;cons;E;at = \(L_2\) cons;cons;cons;cons;\(\cap\)at = \(L_3\) \(\Omega\).

b. cons;cons = \(\pi_1\);at : Obvious from \(L_1\).

car;at = car : cons;\(\pi_1\);at = (lemma 4.5.e) cons;cons;E;\(\pi_1\);at = \(\pi_1\);at = (from above) cons;cons;cons;E;\(\pi_1\) = cons;\(\pi_1\).

car;at' = \(\Omega\) : cons;\(\pi_1\);at' = cons;\(\phi_1\);at;\(\pi_2\);\(\pi_1\);at' = cons;\(\phi_1\);(at \(\cap\) at') = \(L_3\) \(\Omega\).

E = \(\mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}]\): Prove (6.1.2),
\(\mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}] = \mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}] = \mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}]\)
\(= \mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}]\)

first, using \(\text{fpp}\) (lemma 4.3.c) in both directions:

\(\leq\) at \(\cup\) [(\text{car}; \mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}]\)]cons = \(L_1\) and lemma 4.3.c)

\(\leq\) at \(\cup\) [(\text{car}; \mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}]\)]cons = (fpp and part a above)

at \(\cup\) [(\text{car}; \mu X[\text{at} \cup (\text{car}, \text{cdr}; X); \text{cons}]\)]cons = (from above)

\(\geq\) Similarly.

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

c. cons;cons = \(\pi_1\);at;\< : Obvious from \(\emptyset L_1\). □

In the proofs of this chapter the following property, lemma 4.5.e, is
often implicitly applied: \( \tilde{X};X \subseteq E \vdash X;\tilde{p} = X;p \vdash X \). Functionality of the terms involved is proved by repeated application of lemma 4.11 and may require in the induction steps \( \tilde{X};X \subseteq E \) as additional hypothesis and \( \text{OLL}(X);\text{OLL}(X) \subseteq E \) as additional conclusion.

Next we establish an auxiliary lemma.

**Lemma 6.2.** \( \vdash [[\pi_1;\text{at},\pi_2];\text{cons},\pi_3];\text{conc} = [[\pi_1;\text{at},\pi_2];\text{conc} \triangleleft \pi_3];\text{conc} \); cons.

**Proof.** \( \vdash [[\pi_1;\text{at},\pi_2];\text{cons},\pi_3];\text{conc} = [[\pi_1;\text{at},\pi_2];\text{cons},\pi_3];\text{conc} \); cons = \( \vdash [[\pi_1;\text{at},\pi_2];\text{cons},\pi_3];\text{conc} \); cons = \( \vdash [[\pi_1;\text{at},\pi_2];\text{cons},\pi_3];\text{conc} \); cons, as may be proved using \( C_2 \) and (6.1.1),

\[ ... = (\text{OLL}) [[\pi_1;\text{at},\pi_2];\text{conc} \triangleleft \pi_1;[[\pi_1;\text{at},\pi_2];\text{conc} \triangleleft \pi_1;\pi_3];\text{conc} \); cons, whence by
t lemma 4.5.e and cor. 4.2 the result follows. \( \square \)

The fundamental theorem of this section is

**Theorem 6.1.** \( \vdash \text{conc};\text{first} = \triangleleft;\pi_1;\text{first};\text{conc};\text{last} = \triangleleft;\pi_2;\text{last} \).

**Proof.** We derive \( \vdash \text{conc};\text{first} = \triangleleft;\pi_1;\text{first} \) as an example; the proof of \( \vdash \text{conc};\text{last} = \triangleleft;\pi_2;\text{last} \) uses similar techniques.

By lemma 6.1 it is sufficient to prove \( \vdash [\pi_1;\mu X.(\text{oll})_X,\pi_2];\text{conc};\text{first} = [\pi_1;\mu X.(\text{oll})_X,\pi_2];\text{conc};\text{first} \).

Use \( I \) with \( \phi \) empty, taking

\( \{[\pi_1;X,\pi_2];\text{conc};\text{first} = [\pi_1;X,\pi_2];\text{conc};\text{first} \} \) for \( \psi \) and \( \text{OLL} \) for \( \sigma \).

\( \vdash \psi(X). \) Obvious.

\( \psi(X) \vdash \psi(\text{OLL}(X)). \)

1. \( [\pi_1;\text{at},\pi_2];\text{cons};\text{first} = (\text{lemma 6.1})\ [\pi_1;\text{at},\pi_2];\text{cons};\text{car} = (\text{OLL}) \ [\pi_1;\text{at},\pi_2];\text{conc};\text{first} = [\pi_1;\text{at},\pi_2];\text{conc};\text{first} = \).

2. The nucleus of the proof:

\( [\pi_1;\text{car},[\pi_1;\text{cdr},X,\pi_2];\text{conc} \triangleleft = (\text{OLL}) [\pi_1;\text{car},[\pi_1;\text{cdr},X,\pi_2];\text{conc} \triangleleft = [\pi_1;\text{car},[\pi_1;\text{cdr},\pi_2];[\pi_1;X,\pi_2];\text{conc};\text{first} \triangleleft = (\text{induction hypothesis}) \)

\(*\) This corresponds with structural induction on the first coordinate, cf. section 5.5.
\[
\begin{align*}
&[[p_1; \text{car}, p_1; \text{cdr}, p_2]; [p_1; \text{X}, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] \Rightarrow = \\
&= [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] \Rightarrow = \text{(lemma 4.5.e)} \\
&[p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] \Rightarrow = \text{(cor. 4.2)} \\
&[p_1; \text{cdr}, X, p_2] \Rightarrow [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] \Rightarrow = \\
&\quad = [p_1; \text{cdr}, X, p_2] \Rightarrow [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] \Rightarrow = \text{(lemma 4.5.a)} \\
&\quad = [p_1; \text{cdr}, X, p_2] \Rightarrow [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] \Rightarrow = \text{(OLL}_5) \\
&\quad = [p_1; \text{car}, p_1; \text{cdr}, X] \Rightarrow [p_1; \text{cdr}, X, p_2] \Rightarrow.
\end{align*}
\]

\[3. \quad [[[p_1; \text{car}, p_1; \text{cdr}, X]; \text{cons}, p_2]; \text{conc}; \text{first}] = (\text{lemmas 6.1 and 6.2})
\]

\[
\begin{align*}
&[[p_1; \text{car}, p_1; \text{cdr}, X]; \text{cons}, p_2] \Rightarrow [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] = \\
&\quad = (\text{using cons; first} = \Rightarrow [p_1; \text{at}, \text{lemma 4.5.e and part 2}) \\
&\quad = [p_1; \text{car}, p_1; \text{cdr}, X] \Rightarrow [p_1; \text{cdr}, X, p_2] \Rightarrow [p_1; \text{car}.
\end{align*}
\]

\[4. \quad [[[p_1; \text{car}, p_1; \text{cdr}, X]; \text{cons}, p_2]; \text{conc}, \text{first}] = (\text{lemma 4.5.e})
\]

\[
\begin{align*}
&[[p_1; \text{car}, p_1; \text{cdr}, X]; \text{cons}, p_2] \Rightarrow [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] = \\
&\quad = (\text{using cons; first} = \Rightarrow [p_1; \text{at}, \text{lemma 4.5.e and cor. 4.2}) \\
&\quad = [p_1; \text{car}, p_1; \text{cdr}, X] \Rightarrow [p_1; \text{cdr}, X, p_2] \Rightarrow [p_1; \text{car}.
\end{align*}
\]

\[5. \quad [[[p_1; \text{car}, p_1; \text{cdr}, X]; \text{cons}, p_2]; \text{conc}, \text{first}] = (\text{OLL}_5 \text{ and cor. 4.2})
\]

\[
\begin{align*}
&[[p_1; \text{car}, p_1; \text{cdr}, X]; \text{cons}, p_2]; \text{conc}; \text{first}] = \\
&\quad = [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] = (\text{part 3}) \\
&\quad = [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}] = (\text{parts 4 and 5}) \\
&\quad = [p_1; \text{car}, [p_1; \text{cdr}, X, p_2]; \text{cons}, p_2]; \text{conc}; \text{first}. \quad \Box
\end{align*}
\]

We apply this theorem for the first time in

**Theorem 6.2.** \( \vdash \text{conc}; \text{E} \Rightarrow \).

**Proof.**

1. \( \text{conc}; \text{E} = (\text{fpp}) \)

\[
((p_1; \text{at}, p_2); \text{cons} \cup [p_1; \text{car}, [p_1; \text{cdr}, p_2]; \text{conc}; \text{cons}]) \Rightarrow \text{E}.
\]

2. \( [[p_1; \text{at}, p_2]; \text{cons}] \Rightarrow [p_1; \text{at}, p_2] \Rightarrow \).

3. \( [[[p_1; \text{car}, p_1; \text{cdr}, p_2]; \text{conc}; \text{cons}]; \text{E}] = \\
\quad = (\text{OLL}_5 \text{ and theorem 6.1}) \quad [p_1; \text{car}, [p_1; \text{cdr}, p_2]; \text{cons}, p_2] \Rightarrow = \\
\quad = [p_1; \text{car}, [p_1; \text{cdr}, p_2]; \text{cons}, p_2] \Rightarrow = \\
\quad = [p_1; \text{car}, [p_1; \text{cdr}, p_2]; \text{cons}, p_2] \Rightarrow \).
\]
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).**
\[ \vdash [[\pi_1;\pi_2];\text{conc},\pi_3];\text{conc} = [[\pi_1;\pi_2];\text{conc}];\text{conc}. \]

**Proof.** By lemma 6.1 it is sufficient to prove
\[ \vdash [[\pi_1;\mu X[\text{OLL}],\pi_2];\text{conc},\pi_3];\text{conc} = [[\pi_1;\mu X[\text{OLL}],\pi_2];\text{conc}];\text{conc}. \]
Use I with \( \emptyset \) empty, taking \([[\pi_1;\mu X,\pi_2];\text{conc},\pi_3];\text{conc} = [[\pi_1;\mu X,\pi_2];\text{conc}];\text{conc}\) for \( \Psi \) and \( \text{OLL} \) for \( \sigma \).

\[ \vdash \Psi(\Gamma). \text{ Obvious.} \]

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

1. Lemma 6.2 and theorem 6.1 imply \([[\pi_1;\text{at},\pi_2];\text{conc},\pi_3];\text{conc} = [[\pi_1;\text{at},\pi_2];\text{conc},\pi_3];\text{conc}.
2. \([[\pi_1;\text{car},\pi_1;\text{cdr},X];\text{conc},\pi_2];\text{conc},\pi_3];\text{conc} = (\text{frp, OLL}_5, \text{theorem 6.1}) \] \([[\pi_1;\text{car},\pi_1;\text{cdr},X,\pi_2];\text{conc},\pi_3];\text{conc} = \]
(\text{similarly}) \([[\pi_1;\text{car},\pi_1;\text{cdr},X,\pi_2];\text{conc},\pi_3];\text{conc} = \]
(\text{hypothesis}) \([[\pi_1;\text{car},\pi_1;\text{cdr},X,\pi_2];\text{conc},\pi_3];\text{conc} = \]
([[\pi_1;\text{car},\pi_1;\text{cdr},X];\text{conc},\pi_2,\pi_3];\text{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\(_1\),...,OLL\(_5\) of \( E^{\text{NN}},E^{\text{NN}} \) for \( E^{\text{NN}},E^{\text{NN}} \)
results in LL\(_1\),...,LL\(_4\) \( \vdash E^{\text{NN}},E^{\text{NN}} = [\pi_1;\text{last},\pi_2;\text{first}]=E^{\text{NN}},E^{\text{NN}} \),
which is proved by \([\pi_1;\text{last},\pi_2;\text{first}]=E^{\text{NN}},E^{\text{NN}} = \) (corollary 4.3)
\( (\pi_1;\text{last})=E^{\text{NN}};(\pi_2;\text{first})=E^{\text{NN}}=\pi_1^*(\text{last}=E^{\text{NN}});\pi_2^*(\text{first}=E^{\text{NN}}) = \)
(\text{part 1 above}) \( \pi_1^*E^{\text{NN}};\pi_2^*E^{\text{NN}} = \text{lemma 4.6} E^{\text{NN}},E^{\text{NN}}. \)

Hence we have, a fortiori,

\( \) By (6.1.2).
LEMMA 6.3. Any property of ordered linear lists holds upon substitution of \(<\) by \(\ll \ll \ll , \ll \ll \ll \) for linear lists.

6.2. Properties of head and tail

The head and tail functions \(hd\) and \(tl\), both of type \(<N^*, OLL, OLL>\), where \(N^*\) is the type of the positive natural numbers and \(OLL\) the type of ordered linear lists, are defined by

1. \(hd(n, 1)\) is the ordered linear list of \(n\) elements which constitutes the initial part of \(1\) of length \(n\), if extant, and
2. \(tl(n, 1)\) is the ordered linear list which constitutes the remainder of \(1\), after \(hd(n, 1)\) has been chopped off, if possible.

If both sides are defined, clearly properties such as

\[
\text{conc}(hd(n, 1), tl(n, 1)) = 1, \quad tl(n+1, 1) = \text{cdr}(tl(n, 1)),
\]

\[
\text{conc}(hd(n, 1), \text{car}(tl(n, 1))) = hd(n+1, 1), \quad tl(n, \text{conc}(hd(n, 1), 1_2)) = 1_2 \quad \text{and}
\]

\[
hd(n, \text{conc}(hd(n, 1), 1_2)) = hd(n, 1_1) \quad \text{are valid and therefore amenable to proof within our system.}
\]

First we observe that the axioms for \(N^*\) are the axioms for \(N\) which are modified by "renaming" \(p_0\) as \(p_1\) (\(p_0^1\) is renamed as \(p_0^1\), too).

Next we introduce some notation:

\[
hd\ \text{denotes } \mu X [\pi_1^1 p_1 ; \pi_2^1 \text{car} \cup [\pi_2^1 \text{car}, [\pi_1^1 S, \pi_2^1 \text{cdr}]; X]; \text{cons}], \quad \ldots \ (6.2.1)
\]

\[
tl\ \text{denotes } \mu X [\pi_1^1 p_1 ; \pi_2^1 \text{cdr} \cup [\pi_1^1 S, \pi_2^1 \text{cdr}]; X], \quad \ldots \ (6.2.2)
\]

\[
[p_1^1, \ldots, p_n^1] \ \text{denotes } [p_1^1, \ldots, p_n^1]. \quad \ldots \ (6.2.3)
\]

Then the above mentioned properties are established in

THEOREM 6.4.

\begin{itemize}
  \item[a.] \( \vdash [hd, tl]; \text{conc} \Rightarrow [hd, tl] \leq \pi_2^1 \), of type \(<N^*, OLL, OLL>\).
  \item[b.] \( \vdash \text{tl}; \text{cdr} \Rightarrow [\pi_1^1 S, p_2^1] tl \), of type \(<N^*, OLL, OLL>\).
  \item[c.] \( \vdash [hd, tl]; \text{car}; \text{conc} \Rightarrow [\pi_1^1 S, p_2^1] hd \), of type \(<N^*, OLL, OLL>\).
  \item[d.] \( \vdash [\pi_1^1, [\pi_1^2; hd, \pi_3^1]; \text{conc}]; \text{tl} = [\pi_1^1, [\pi_1^2; hd, \pi_3^1]; \text{conc}]; \text{tl} \Rightarrow [\pi_1^1, [\pi_1^2; hd, \pi_3^1]; \text{conc}]; \text{tl} \), of type \(<N^*, OLL, OLL>\).
  \item[e.] \( \vdash [\pi_1^1, [\pi_1^2; hd, \pi_3^1]; \text{conc}]; \text{hd} = [\pi_1^1, [\pi_1^2; hd, \pi_3^1]; \text{conc}]; \text{hd} \Rightarrow [\pi_1^1, [\pi_1^2; hd, \pi_3^1]; \text{conc}]; \text{hd} \), of type \(<N^*, OLL, OLL>\).
\end{itemize}
f. $\vdash t_1 \approx E = [h_1, t_1] \approx \pi_2$, of type $\mathcal{N} \times \mathcal{O}, \mathcal{N} \times \mathcal{O}$.

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

a. First we prove $\vdash [h_1, t_1]; \text{conc} \subseteq \pi_2$. Then the result follows from

$[h_1, t_1]; \text{conc} = (\text{lemma 4.3.d}) ([h_1, t_1]; \text{conc}) \approx E \supseteq \pi_2 = (\text{theorem 6.2})$

$[h_1, t_1] \approx \pi_2$.

Apply I, with $\emptyset$ empty and taking $[[h_1, t_1]; \pi \subseteq \pi_2]$ for $\forall$ and

$(\text{cons} \cup [\pi_1; \text{car}, [\pi_1; \text{cdr}, \pi_2]; X]; \text{cons})$ for $a$. Then $\forall(X) \vdash \forall(a(X))$ follows from parts 1 and 2 below.

1. $[h_1, t_1]; \text{cons} = (\text{OLL}) [h_1; \text{at}, t_1]; \text{cons} = (\text{fpp and lemma 6.1})$

$\pi_1 \supseteq \pi_2; [\pi_2; \text{car}, \pi_2; \text{cdr}]; \text{cons} \subseteq (\text{OLL}) \pi_2$.

2. $[h_1, t_1]; [\pi_1; \text{car}, [\pi_1; \text{cdr}, \pi_2]; X]; \text{cons} = [h_1; \text{car}, [h_1; \text{cdr}, t_1]; X]; \text{cons} =$

$(\text{fpp and lemma 6.1})$

$[\pi_2; \text{car}, [\pi_1; \text{car}, \pi_2; \text{cdr}]; h_1; [\pi_1; \text{car}, \pi_2; \text{cdr}, t_1]; X]; \text{cons} \subseteq (\text{hypothesis})$

$[\pi_2; \text{car}, [\pi_1; \text{car}, \pi_2; \text{cdr}, \pi_2]; X]; \text{cons} \subseteq (\text{OLL}) \pi_2$.

c. Apply I, with $\emptyset$ empty, taking $[[\pi_1; [\pi_1, \pi_2; \text{hd}, \pi_3]; \text{conc}]; X =$

$= [\pi_1, \pi_2; \text{hd}, \pi_3] \approx \pi_1, \pi_2; X]$ for $\forall$ and $(\pi_1 \supseteq \pi_2; \text{car} \cup$

$\cup [\pi_2; \text{car}, [\pi_1; \text{car}, \pi_2; \text{cdr}]; X]; \text{cons})$ for $a$. Then $\forall(X) \vdash \forall(a(X))$ follows from

part 1 and 4 below.

1. It follows from lemma 4.3.d that $[\pi_1, \pi_2; \text{hd}, \pi_3]; \text{conc}; \text{car} \subseteq (\text{fpp}) \pi_2; \text{car}$

and $([\pi_1, \pi_2; \text{hd}, \pi_3]; \text{conc}; \text{car}) \approx E = [\pi_1, \pi_2; \text{hd}, \pi_3] \approx (\text{conc} \supseteq \pi_1')$ (fpp)

$[\pi_1, \pi_2; \text{hd}, \pi_3] \approx (\text{conc} \supseteq E) = (\text{theorem 6.2}) [\pi_1, \pi_2; \text{hd}, \pi_3] \approx \pi_2$ together imply

$[\pi_1, \pi_2; \text{hd}, \pi_3]; \text{conc}; \text{car} = [\pi_1, \pi_2; \text{hd}, \pi_3] \approx \pi_1, \pi_2; \text{car}$.

2. $[\pi_1, \pi_2; \text{hd}, \pi_3]; \text{conc}; \text{cdr} =$

$= [\pi_1, \pi_2; \text{hd}, \pi_3] \approx \pi_1, \pi_2; \text{hd}, \pi_3] \supseteq \pi_1, \pi_2; \text{car}$.

3. $\pi_1, \pi_2; \text{hd}; \text{cdr} = (\text{fpp}) [\pi_1; \pi_2; \text{car}; \text{hd}].$

4. $[\pi_1; [\pi_1, \pi_2; \text{hd}, \pi_3]; \text{conc}; \pi_1 \supseteq \pi_1']$;$[\pi_2; \text{car}, [\pi_1; \pi_2; \text{cdr}]; X]; \text{cons} =$

$= (\text{parts 1 and 2})$

$[\pi_1, \pi_2; \text{hd}, \pi_3] \approx \pi_1, \pi_2; \text{car}$

$[\pi_2; \text{car}, [\pi_1; \pi_2; \text{car}, \pi_2; \text{cdr}, \pi_3]; [\pi_1, [\pi_1, \pi_2; \text{hd}, \pi_3]; \text{conc}; X]; \text{cons} =$

$(\text{hypothesis})$

$[\pi_1, \pi_2; \text{hd}, \pi_3] \approx \pi_1, \pi_2; \text{car}$.

$[\pi_2; \text{car}, [\pi_1; \pi_2; \text{car}, \pi_2; \text{cdr}, \pi_3]; [\pi_1, \pi_2; \text{hd}, \pi_3] \approx \pi_1, \pi_2; X]; \text{cons} =$
[\pi_2;ihd,\pi_3] \alpha \text{; } \pi_1 \alpha \pi_1 ' \text{; } \pi_2 ' = \text{(part c)}

[\pi_2,car,([\pi_1;iS,\pi_2;cdr];hd,\pi_3] \alpha \text{; } [\pi_1;iS,\pi_2;cdr,\pi_3];\pi_1,2;iX];\text{cons} =

Since \alpha = \pi_1 ; \alpha ' ; \pi_2 \in E, (6.1.5), transitivity of the relation \alpha ', i.e.,
the property \alpha ' ; \alpha ' \subseteq \alpha ', implies \pi_2,3 \alpha ; \pi_2,3 \alpha \subseteq \pi_2,3 \alpha ', transitivity of the predicate \alpha in its two arguments or transitivity of \alpha, for short. This follows from \pi_2,3 \alpha ; \pi_2,3 \alpha \subseteq \pi_2,3 \alpha = (\pi_1 ; \alpha ; \pi_2 \in E) \Rightarrow \pi_3 \in \pi_1 ; \alpha ' ; \pi_3 \in E \Rightarrow \alpha ; \pi_3 \in E \Rightarrow \alpha .

COROLLARY 6.1. Let \alpha be transitive (in its two arguments), then

a. \vdash [[\pi_1;S,\pi_2];hd,\pi_3] \alpha =
   [\pi_2;ihd,\pi_1;2;tl;car] \alpha \text{; } [\pi_1;iS,\pi_2;car,\pi_3] \alpha = [\pi_1;ihd,\pi_3] \alpha .

b. \vdash (1) [[\pi_1;S,\pi_2];tl;E = [hd,tl;car] \alpha = [tl;car,tl;cdr] \alpha = [hd,tl;cdr] \alpha .

Proof.

a. [[\pi_1;S,\pi_2];hd,\pi_3] \alpha = (theorem 6.4.1) [[\pi_1;2;hd,\pi_1;2;tl;car];conc,\pi_3] \alpha =
   (theorem 6.1) [[\pi_1;2;hd,\pi_1;2;tl;car] \alpha = [\pi_1;2;tl;car,\pi_3] \alpha , whence the result can be deduced from the assumption.

b. (1) [[\pi_1;S,\pi_2];tl;E = (theorem 6.4.1) [[\pi_1;S,\pi_2];hd,\pi_1;S,\pi_2;tl] \alpha =
   (theorems 6.4.b and 6.4.c) [[hd,tl;car];conc,tl;cdr] \alpha = (theorems 6.1 and transitivity of \alpha) [hd,tl;car] \alpha = [tl;car,tl;cdr] \alpha = [hd,tl;cdr] \alpha .

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,1,2,3); integer n,x,y; ordered linear list l1,l2,l3;
if n=1 then MOVE(n,x,y,l1,l2,l3) else
```
begin $n := n - 1$; $y := \text{alt}(x, y)$; TVH$(n, x, y, \ell_1, \ell_2, \ell_3)$;
      $y := \text{alt}(x, y)$; MOVE$(n, x, y, \ell_1, \ell_2, \ell_3)$; $x := \text{alt}(x, y)$;
      TVH$(n, x, y, \ell_1, \ell_2, \ell_3)$; $n := n + 1$; $x := \text{alt}(x, y)$
end;

procedure MOVE$(n, x, y, \ell_1, \ell_2, \ell_3)$; integer $n, x, y$; ordered linear list $\ell_1, \ell_2, \ell_3$;
if $x = 1 \land y = 2$ then begin $\ell_2 := \text{cons}(\text{car}(\ell_1), \ell_2)$; $\ell_1 := \text{cdr}(\ell_1)$ end else
if $x = 1 \land y = 3$ then begin $\ell_3 := \text{cons}(\text{car}(\ell_1), \ell_3)$; $\ell_1 := \text{cdr}(\ell_1)$ end else
if $x = 2 \land y = 3$ then begin $\ell_3 := \text{cons}(\text{car}(\ell_2), \ell_3)$; $\ell_2 := \text{cdr}(\ell_2)$ end else
if $x = 2 \land y = 1$ then begin $\ell_1 := \text{cons}(\text{car}(\ell_2), \ell_1)$; $\ell_2 := \text{cdr}(\ell_2)$ end else
if $x = 3 \land y = 1$ then begin $\ell_1 := \text{cons}(\text{car}(\ell_3), \ell_1)$; $\ell_3 := \text{cdr}(\ell_3)$ end else
if $x = 3 \land y = 2$ then begin $\ell_2 := \text{cons}(\text{car}(\ell_3), \ell_2)$; $\ell_3 := \text{cdr}(\ell_3)$ end else
undefined;

integer procedure alt$(x, y)$; integer $x, y$; if $x \geq 1 \land x \leq 3 \land y \geq 1 \land y \leq 3$ 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$(n, x, y, \ell_1, \ell_2, \ell_3)$, 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 $R(n, x, y, \ell_1, \ell_2, \ell_3)$ by
procedure $R(n,x,y,\ell_1,\ell_2,\ell_3)$; integer $n,x,y$; ordered linear list $\ell_1,\ell_2,\ell_3$;
if $x=1\land y=2$ then begin $\ell_2:=\text{conc}(\text{hd}(n,\ell_1),\ell_2)$; $\ell_1:=\text{tl}(n,\ell_1)$ end else
if $x=1\land y=3$ then begin $\ell_3:=\text{conc}(\text{hd}(n,\ell_1),\ell_3)$; $\ell_1:=\text{tl}(n,\ell_1)$ end else
if $x=2\land y=3$ then begin $\ell_3:=\text{conc}(\text{hd}(n,\ell_2),\ell_3)$; $\ell_2:=\text{tl}(n,\ell_2)$ end else
if $x=2\land y=1$ then begin $\ell_1:=\text{conc}(\text{hd}(n,\ell_2),\ell_1)$; $\ell_2:=\text{tl}(n,\ell_2)$ end else
if $x=3\land y=1$ then begin $\ell_1:=\text{conc}(\text{hd}(n,\ell_3),\ell_1)$; $\ell_3:=\text{tl}(n,\ell_3)$ end else
if $x=3\land y=2$ then begin $\ell_2:=\text{conc}(\text{hd}(n,\ell_3),\ell_2)$; $\ell_3:=\text{tl}(n,\ell_3)$ end else
undefined.

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

$$TVH(n,x,y,\ell_1,\ell_2,\ell_3) = R(n,x,y,\ell_1,\ell_2,\ell_3),$$

provided both sides are defined.

Proof. As $TVH(1,x,y,\ell_1,\ell_2,\ell_3) = R(1,x,y,\ell_1,\ell_2,\ell_3)$ follows from the declarations, we concentrate on the case $n > 1$:

The induction hypothesis is $TVH(n-1,x,y,\ell_1,\ell_2,\ell_3) = R(n-1,x,y,\ell_1,\ell_2,\ell_3)$, provided both sides are defined. Start with statevector $\xi_0 \equiv <n,1,2,\ell_1,\ell_2,\ell_3>$.

1. Execution of $n := n-1$; $y := \text{alt}(x,y)$; $TVH(n,x,y,\ell_1,\ell_2,\ell_3)$ with $\xi_0$ as input results in

$$\xi_1 \equiv <n-1,1,3,\text{tl}(n-1,\ell_1),\ell_2,\text{conc}(\text{hd}(n-1,\ell_1),\ell_3)>,$$

by the induction hypothesis.

2. Execution of $y := \text{alt}(x,y)$; $\text{MOVE}(n,x,y,\ell_1,\ell_2,\ell_3)$ with $\xi_1$ as input results in

$$\xi_2 \equiv <n-1,1,2,\text{cdr}(\text{tl}(n-1,\ell_1)),\text{cons}(\text{car}(\text{tl}(n-1,\ell_1)),\ell_2),$$

$$\text{conc}(\text{hd}(n-1,\ell_1),\ell_3)>.$$

3. Execution of $x := \text{alt}(x,y)$; $TVH(n,x,y,\ell_1,\ell_2,\ell_3)$; $n := n+1$; $x := \text{alt}(x,y)$ with $\xi_2$ as input results in

$$\xi_2 \equiv <n,1,2,\text{cdr}(\text{tl}(n-1,\ell_1)),$$

Expr 1

$$\text{conc}(\text{hd}(n-1,\text{conc}(\text{hd}(n-1,\ell_1),\ell_3))),\text{cons}(\text{car}(\text{tl}(n-1,\ell_1),\ell_2))>,$$

Expr 2

$$\text{tl}(n-1,\text{conc}(\text{hd}(n-1,\ell_1),\ell_3))>.$$

Expr 3
We demonstrate that, provided \( \xi_3 \) is defined, \( \xi_3 \) equals
\(<n,1,2,t1(n,\xi1),\text{conc}(\text{hd}(n,\xi1),\xi2),\xi3>\).

\textbf{Expr 1:} \( \text{cdr}(t1(n-1,\xi1)) = t1(n,\xi1) \) by theorem 6.4.b.

\textbf{Expr 2:} 1. \( \text{hd}(n-1,\text{conc}(\text{hd}(n-1,\xi1),\xi3)) = \text{if} \ \text{hd}(n-1,\xi1) \prec \xi3 \ \text{then} \ \text{hd}(n-1,\xi1) \)
\( \text{else} \ \text{undefined}, \) by theorem 6.4.e.

\begin{enumerate}
\item \( \text{conc}(\text{hd}(n-1,\xi1),\text{cons}(\text{car}(t1(n-1,\xi1)),\xi2)) = \)
\( = \text{conc}(\text{conc}(\text{hd}(n-1,\xi1),\text{car}(t1(n-1,\xi1))),\xi2), \) by associativity of conc, theorem 6.3.
\item \( \text{conc}(\text{hd}(n-1,\xi1),\text{car}(t1(n-1,\xi1))) = \text{hd}(n,\xi1), \) by theorem 6.4.c.
\end{enumerate}

Thus \( \textbf{Expr 2} = \text{if} \ \text{hd}(n-1,\xi1) \prec \xi3 \ \text{then} \ \text{conc}(\text{hd}(n,\xi1),\xi2) \)
\( \text{else} \ \text{undefined}. \)

\textbf{Expr 3:} \( t1(n-1,\text{conc}(\text{hd}(n-1,\xi1),\xi3)) = \text{if} \ \text{hd}(n-1,\xi1) \prec \xi3 \ \text{then} \ \xi3 \)
\( \text{else} \ \text{undefined}, \) by theorem 6.4.d.

Thus \( \xi_3 = \text{if} \ \text{hd}(n-1,\xi1) \prec \xi3 \ \text{then} \ <n,1,2,t1(n,\xi1),\text{conc}(\text{hd}(n,\xi1),\xi2),\xi3> \)
\( \text{else} \ \text{undefined}, \) whence the result. \( \Box \)

\textbf{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 \( \xi_3 \). Furthermore we need the function \( \text{alt} \) of type \( \langle 3,3 \rangle \) defined by:
if \( x \neq y \) then \( \text{alt}(x,y) = (a,b,c) - (x,y) \), and \( \text{alt}(x,y) \) is undefined, otherwise.
Then \( \text{alt} \) has the following properties: \( \text{alt}(x,y) = \text{alt}(y,x) \),
\( \text{alt}(\text{alt}(x,y),x) = y \) and \( \text{alt}(\text{alt}(x,y),y) = x \). The formal definition of \( \text{alt} \),
using the predicates \( a, b \) and \( c \), and the subsequent derivation of these properties is a matter of routine.

\[ \forall i-j \ \text{DEF} \ \forall i,i+1,...,j \]

, for \( i < j \).

Secondly we define \( \text{TVH} \), of type \( \langle N^+ \times 3 \times \text{OLL} \times \text{OLL} \times \text{OLL} \times \text{OLL} \times \text{OLL} \rangle \),
by
\[ \text{TVH} \ \text{DEF} \ \mu X[\xi_1,p_1;\text{MOVE} \cup \xi_1;p_1;\xi_1\xi_1\xi_2\xi_2\xi_2;\xi_3\xi_4;\text{alt}_x\xi_4;\xi_5;X;\xi_0\xi_1] \]
\[ [\pi_{1-3},\pi_{2-3};\text{alt},\pi_{3-6}] \]
\[ \text{MOVE} \{ [\pi_{1-3},\pi_{2-3};\text{alt},\pi_{3-6}] ; X \} \]
\[ \pi_2 \]
\[ [\pi_{1-3},\pi_{2-3};\text{alt},\pi_{3-6}] \]
\[ \pi_3 \]

and

\[ \text{MOVE} \quad \begin{array}{l}
\quad \text{DEF}_{a,b} \{ [\pi_{1-3},\pi_{4};\text{cdr},[\pi_{4};\text{car},\pi_5];\text{cons},\pi_6] \} \cup \\
\quad \quad \cup \quad \text{DEF}_{a,c} \{ [\pi_{1-3},\pi_{4};\text{cdr},[\pi_{5};\text{car},\pi_6];\text{cons}] \} \cup \\
\quad \quad \cup \quad \text{DEF}_{b,c} \{ [\pi_{1-4},\pi_{5};\text{cdr},[\pi_{5};\text{car},\pi_6];\text{cons}] \} \cup \\
\quad \quad \cup \quad \text{DEF}_{b,a} \{ [\pi_{1-3},\pi_{5};\text{cdr},[\pi_{5};\text{car},\pi_6];\text{cons}] \} \cup \\
\quad \quad \cup \quad \text{DEF}_{c,a} \{ [\pi_{1-3},\pi_{5};\text{cdr},[\pi_{5};\text{car},\pi_6];\text{cons}] \} \cup \\
\quad \quad \cup \quad \text{DEF}_{c,b} \{ [\pi_{1-4},\pi_{6};\text{cdr};\text{cons},\pi_6] \} \cup \\
\end{array} \]

with

\[ P_{x,y} \quad \text{DEF} \quad \pi_{2}^{a x} \pi_{3}^{x y} \quad \text{for} \; x, y \in \{ a, b, c \}. \]

Thirdly we define \( p' \), \( 0 \) and \( R \) in order to express correctness of TVH:

\[ p'_e \quad \text{DEF} \quad \psi_{x,y} \quad \text{cf. (6.3.2)}. \]

\[ 0 \quad \text{DEF} \quad \begin{array}{l}
\quad \text{DEF}_{a} \{ [\pi_{1-4};\text{hd},\pi_{5}]^{a \pi} \} \cup \\
\quad \quad \cup \quad \text{DEF}_{b} \{ [\pi_{1-5};\text{hd},\pi_{4}]^{b \pi} \} \cup \\
\quad \quad \cup \quad \text{DEF}_{c} \{ [\pi_{1-6};\text{hd},\pi_{4}]^{c \pi} \} \cup \\
\end{array} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{4}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]

\[ \pi_{1-6} \]

\[ [\pi_{1-6};\text{hd},\pi_{5}]^{c \pi} \]
Then the correctness of TVH is established by

**THEOREM 6.5. (Correctness of TOWERS OF HANOI).** Let \( \preceq \) be transitive (in the sense indicated in (6.2.1)), then

\[
| \vdash p_{eq} \circ 0 ; TVH = p_{eq} \circ 1 ; R. 
\]

**Proof.** The proof of this theorem proceeds by induction on \( N^* \), i.e., we prove

\[
| \vdash p_{eq}^i \circ [r_1 ; \mu X[p_1 \cup \tilde{S}; X; S], \pi_{2-6}] ; 0 ; TVH = p_{eq}^i \circ [r_1 ; \mu X[p_1 \cup \tilde{S}; X; S], \pi_{2-6}] ; 0 ; \tilde{E} 
\]

by applying \( I \) as follows: let \( \emptyset \) be empty, \( \forall \) be

\[
| \vdash p_{eq}^i \circ [r_1 ; \pi_{2-6}] ; 0 ; TVH = p_{eq}^i \circ [r_1 ; \pi_{2-6}] ; 0 ; R \text{ and } \sigma \text{ be } (p_1 \cup \tilde{S}; X; S). \text{ Then the result follows from } \mu X[p_1 \cup \tilde{S}; X; S] = e_{N^*} N^*, \text{ cf. lemma 5.5.} 
\]

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

\[
| \vdash p^i_{eq} \circ [r_1 ; (p_1 \cup \tilde{S}; X; S), \pi_{2-6}] ; 0 ; TVH = (f_{pp}) 
\]

\[
| \vdash p^i_{eq} \circ [r_1 ; \pi_{2-6}] ; 0 ; TVH = p^i_{eq} \circ [r_1 ; \pi_{2-6}] ; 0 ; TVH \circ TVH ; \tau \circ \tau_3 . 
\]

Then our correctness proof consists in proving, with \( \forall \) as hypothesis,

\[
P_0 ; \tau_0 = Q_0 \quad \ldots \quad (6.3.4) 
\]

and

\[
P_1 ; \tau_1 ; TVH ; \tau_2 ; TVH ; \tau_3 = \quad (\text{parts 1 and 2) } Q_1 ; TVH ; \tau_2 ; TVH ; \tau_3 = 
\]
(part 3) \( P_2; \tau_2;TVH; \tau_3 = \)

(= parts 4, 5 and 6) \( Q_2;TVH; \tau_3 = \)

(= part 7) \( p_3; \tau_3 = \) (part 8) \( Q_3, ^*) \) 

\( \) since \( p_0 = [p_1 \vdash p'_1];p_{eq};O, \) \( Q_0 = [p_1 \vdash p'_1];p_{eq};0;R, \) \( p_1 = p'_1;[\pi_1;S;X;S,\pi_2;6];0, \) and \( Q_3 = p'_1;[\pi_1;S;X;S,\pi_2;6];0;R, \) whence (6.3.4) and (6.3.5) together imply

\[ p_{eq};[\pi_1;(p_1 \cup \tilde{S};X;S,\pi_2;6]0;TVH = p_{eq};[\pi_1;(p_1 \cup \tilde{S};X;S,\pi_2;6);0;R. \]

Without loss of generality we prove

\[ p_{eq};[\pi_1;X,\pi_2;6];0;TVH = p_{eq};[\pi_1;X,\pi_2;6];0;R \]

\[ \vdash \pi_1;(p_1 \cup \tilde{S};X;S,\pi_2;6];a,\pi_3;3,b,\pi_4;6]0_a;TVH = \]

\[ = [\pi_1;(p_1 \cup \tilde{S};X;S,\pi_2;6];a,\pi_3;3,b,\pi_4;6]0_a;R. \]

Next terms \( p_i \) and \( Q_i \) are defined as below, \( i = 0, \ldots, 3. \)

Let \( o_a(X) \) be \( [\pi_1;X,\pi_4];hd,\pi_5;< \) \( [\pi_1;X,\pi_4];hd,\pi_6;< \), whence

\( \pi_2;\pi_4;0_a;0_a(\pi) = 0_a \) (see 6.3.3), and let \( O_{a,b} \) be \( [\pi_1,4]hd,\pi_5;< \) and

\( O_{a,c} \) be \( [\pi_1,4]hd,\pi_6;< \), whence \( O_a = \pi_2;\pi_4;0_a;0_a,b;0_a,c; \). For \( O_b \) and \( O_c \) we introduce similar notations.

\( P_0 \) is defined as

\[ [\pi_1;\pi_1,\pi_2;6;\pi_3;6,\pi_4;6];0_a. \]

\( Q_0 \) is defined as

\[ [\pi_1;\pi_1,\pi_2;6;\pi_3;6,\pi_4;6];0_a;R. \]

\( P_1 \) is defined as

\[ [\pi_1;S;X;S,\pi_2;6;\pi_3;6,\pi_4;6];0_a. \]

\( Q_1 \) is defined as

\[ a;[\pi_1;S;X;S,\pi_2;6;\pi_3;6,\pi_4;6];0_a. \]

\( P_2 \) is defined as

\[ a;[\pi_1;\pi_1,\pi_2;6;\pi_3;6,\pi_4;6];0_a. \]

\( Q_2 \) is defined as

\[ a;[\pi_1;\pi_1,\pi_2;6;\pi_3;6,\pi_4;6];0_a. \]

\( P_3 \) is defined as

\[ a;[\pi_1;\pi_1,\pi_2;6;\pi_3;6,\pi_4;6];0_a. \]

\( ^* \) Parts 1 to 8 refer to the formal proof at the end of this section.
Finally we prove the induction step as indicated in (6.3.4) and (6.3.5).
Assume transitivity of \(\prec\), i.e., \(\tau^i_1,\tau^i_2 \prec \tau^i_3 \prec \tau^i_4 \implies \tau^i_1,\tau^i_3 \prec \tau^i_4\), and the induction hypothesis \(\psi\).

The proof of \(P_2:\text{TVH} = Q_0\) is a matter of routine and therefore omitted.

1. \(\tau_1\) = \((S; S = E^+, N^+, \text{ cf. axiom } N_3)\)
   \(\tau_2\) = \((S; S = E^+, N^+, \text{ cf. axiom } N_3)\).

2. \(P_1\) = \(Q_1\) = \((\text{lemma } 4.5.6)\)

3. \(Q_3\) = \((\text{hypothesis})\)

4. \(P_2\) = \(Q_2\) = \((\text{theorem } 6.4.4)\)

5. \(Q_2\) = \((\text{theorem } 6.4.4)\)

6. \(Q_2\) = \((\text{corollary } 6.1)\)

By combining parts 4, 5 and (i), (ii) above, we obtain
\(P_2\) = \(Q_2\) = \(Q_3\) = \(P_3\).
8. (i) $[[\pi_1;X,[[\pi_1;X,\pi_4];\text{hd},\pi_6]];\text{conc}];\text{hd},[[\pi_1;X,\pi_4];\text{tl};\text{car},\pi_5];\text{conc}];\text{conc} =$
  = (\text{theorem 6.4}) $[[\pi_1;X,\pi_4];\text{hd},\pi_6] > <$
  $[[\pi_1;X,\pi_4];\text{hd},[[\pi_1;X,\pi_4];\text{tl};\text{car},\pi_5];\text{conc}];\text{conc} =$
  = (\text{theorems 6.3 and 6.4}) $[[\pi_1;X,\pi_4];\text{hd},\pi_6] > <$
  $[[\pi_1;X;S,\pi_4];\text{hd},\tau_5];\text{conc}.$

(ii) $[[\pi_1;X,[[\pi_1;X,\pi_4];\text{hd},\pi_6]];\text{conc}];\text{tl} = (\text{theorem 6.4})$
  $[[\pi_1;X,\pi_4];\text{hd},\pi_6] > < ;\tau_6.$

(iii) By part 6(ii), $0_a(\tilde{s};X;S);[[\pi_1;\tilde{s},\tau_2];6] = ... [[\pi_1;X,\pi_4];\text{hd},\pi_6] > < .$

By combining parts (i), (ii) and (iii) above, we obtain

$P_3 = 0_a(\tilde{s};X;S);[[\pi_1;\tilde{s},\tau_2];6];$
  $[[\pi_1;X,\tau_2];c,\tau_3;[b,[[\pi_1;X;S,\pi_4];\text{tl},[[\pi_1;X;S,\pi_4];\text{hd},\pi_5];\text{conc},\pi_6]],$

whence $P_3;\tau_3 = [[\pi_1;\tilde{s};X;S,\pi_2];a,\pi_3;b,\pi_4];0_a;R = Q_3.$ □
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 $\sigma$ 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 $\mathcal{T}$ between programs and terms of this mathematical language, i.e., a mapping satisfying

   $$\sigma(T) = m(\mathcal{T}(T))$$

   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 "$e" operator.

3. The "$e" 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 I: 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 \(O_0\) 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 \(O_0\) 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 \(O_0\) and \(D\):

a. Let \(CM_1 = \langle x_1, V_1, x_2, V_2, ..., x_n, V_n, x_{n+1}, CM_1 \rangle\) be a computation model for \(x_1, V_1, x_2, V_2, ..., x_n, V_n, x_{n+1}\) with \(V_1 \in A \cup C \cup X \cup P\), let

\[ CM_2 = \langle y_1, W_1, y_2, W_2, ..., y_m, W_m, y_{m+1}, CM_2 \rangle \]

be a computation model for \(y_1, W_1, y_2, W_2, ..., y_m, W_m, y_{m+1}\), and let \(x_{n+1} = y_1\), then the computation model \(CM_1; CM_2\) is defined by

\[ CM_1; CM_2 = \langle x_1, V_1; W_1, x_2, V_2; W_2, ..., x_n, V_n; W_n, x_{n+1}, W_1, y_2, W_2, ..., y_m, W_m, y_{m+1}, CM_1 \cup CM_2 \rangle. \]

b. Let \(CM_1 = \langle x_1, V_1, x_2, V_2, ..., x_n, V_n, x_{n+1}, CM_1 \rangle\) be a computation model for \(x_1, V_1, x_2, V_2, ..., x_n, V_n, x_{n+1}\) with \(V_1 = V' \cup V''\), for some statement schemes \(V'\) and \(V''\), let

\[ CM_2 = \langle y_1, W_1, y_2, W_2, ..., y_m, W_m, y_{m+1}, CM_2 \rangle \]

be a computation model for \(y_1, W_1, y_2, W_2, ..., y_m, W_m, y_{m+1}\), and let \(x_{n+1} = y_1\), then the computation model \((CM_1); CM_2\) is defined by

\[ (CM_1); CM_2 = \langle x_1, (V_1); W_1, y_1, W_1, ..., y_m, W_m, y_{m+1}, [CM_1] \cup CM_2 \rangle. \]

c. Let \(CM = \langle x_1, V_1, x_2, V_2, ..., x_n, V_n, x_{n+1}, CM \rangle\) be a computation model for \(x_1, V_1, x_2, V_2, ..., x_n, V_n, x_{n+1}\), let \(W_0; \theta\) be an arbitrary statement scheme, and let \(p^{n,n}\) be a predicate symbol. If

(1) \(O_0(p) = true\), then the computation model \((O_0(p) \rightarrow CM, W)\) is defined by \((O_0(p) \rightarrow CM, W) = \langle x_1, (p \rightarrow V_1, W) \times x_1, V_1, ..., x_n, V_n, x_{n+1}, CM \rangle\),

(2) \(O_0(p) = false\), then the computation model \((O_0(p) \rightarrow W, CM)\) is defined by \((O_0(p) \rightarrow W, CM) = \langle x_1, (p \rightarrow W, V_1) \times x_1, V_1, ..., x_n, V_n, x_{n+1}, CM \rangle\).

d. Let for \(j = 1, ..., n, CM_j = \langle x_{j,1}, V_{j,1}, x_{j,2}, V_{j,2}, ..., x_{j,m_j}, V_{j,m_j}, x_{j,m_j+1}, CM_j \rangle\) be computation models for \(x_{j,1}, V_{j,1}, x_{j,2}, V_{j,2}, ..., x_{j,m_j}, V_{j,m_j}, x_{j,m_j+1}\), and let \(x_{1,1} = ... = x_{n,1}\).
then the computation model \([CM_1, \ldots, CM_n]\) is defined by
\[
[CM_1, \ldots, CM_n] = \langle x_{1,1}, \ldots, V_{1,n}, \ldots, x_{m+1}, \ldots, x_{n,\omega+1}\rangle,(CM_1, \ldots, CM_n)\rangle.
\]

**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 \(\psi_D^{S}(o_0)\) of a statement scheme \(S\) relative to the initial interpretation \(\gamma_0\) and the declaration scheme \(D\) is the set
\[
\{CM \mid \exists x,y[CM \text{ is, relative } o_0 \text{ and } D, \text{ a computation model for } x \land S \land y]\}.
\]

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 \(\psi_D\) and \(\psi_D^{S}\) are conceived of as functions.

Definition 2.5 of the operational interpretation \(\sigma(S)\) of a statement scheme \(S\) relative to \(o_0\) and \(D\) can be recovered from \(\psi_D^{S}(o_0)\) by forgetting the internal structure of the computation models constituting \(\psi_D^{S}(o_0)\) and preserving the external input-output relationship of these models.

After defining the appropriate operations one can establish results such as:

\[
\begin{align*}
\psi_D^{S_1; S_2}(o_0) &= \psi_D^{S_1}(o_0) \circ \psi_D^{S_2}(o_0) \\
\psi_D^{S_1; S_2; S_3}(o_0) &= (\psi_D^{S_1; S_2}(o_0)) \circ \psi_D^{S_3}(o_0) \\
\psi_D^{x \rightarrow S_1; S_2}(o_0) &= (\sigma_0(p) \circ \psi_D^{S_1}(o_0), S_2) \cup (\sigma_0(p) \rightarrow S_1, \psi_D^{S_2}(o_0)) \\
\psi_D^{[S_1, \ldots, S_n]}(o_0) &= \{\psi_D^{S_1}(o_0), \ldots, \psi_D^{S_n}(o_0)\},
\end{align*}
\]

from which the proofs of parts b, c and d of lemma 2.1 can be derived.

Let us now analyze how the notions "to identify" and "executable occurrence", defined in def. 2.6, relate to this way of structuring computation models:

**a.** \(CM = CM_1; CM_2\):

\[
CM_1 = \langle x_1, V_1, x_2, V_2, \ldots, x_n, V_n, x_{n+1}, CM_1\rangle,
\]

\[
CM_2 = \langle y_1, W_1, y_2, W_2, \ldots, y_m, W_m, y_{m+1}, CM_2\rangle, x_{n+1} = y_1 \text{ and}
\]

\[
CM = \langle x_1, V_1, W_1, x_2, V_2; W_1, \ldots, x_n, V_n; W_1, x_{n+1}, W_1, y_2, W_2, \ldots, y_m, W_m, y_{m+1}, CM_1 \cup CM_2\rangle.
\]
It follows from the definitions that

(1) Two occurrences of some procedure symbol, which are both contained in $CM_i$, identify each other w.r.t. $CM_i$ iff the corresponding occurrences in $CM_i$, i.e., in $cs_i^*$ or $CM_i^1$, identify each other w.r.t. $CM_i$, $i = 1, 2$; 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 $CM_i^1$ is executable, $i = 1, 2$; these are the only executable occurrences.

b. $CM = (CM_1, CM_2)$:

$CM_1 = \langle x_1, V_1, x_2, V_2, \ldots, x_n, V_n, x_{n+1}, CM_1^1, \rangle, V_1 = V_1; W$ for some statement

$CM_2 = \langle y_1, W_1, y_2, W_2, \ldots, y_m, W_m, y_{m+1}, CM_2^1, \rangle, x_{n+1} = y_1$ and $x_1 \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow$ schemes $V$ and $W$,

$CM = \langle x_1, (V_1) ; W_1, y_1, W_1, \ldots, y_m, W_m, y_{m+1}, \{CM_1^1 \cup CM_2^1\}, \rangle,

\rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \right}
It follows from the definitions that

(1) Two occurrences of some procedure symbol which are both contained in CM, identify each other w.r.t. CM, iff the corresponding occurrences in c_{j,1} or CM 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, W_2).

(2) An occurrence of some procedure symbol contained in CM is executable w.r.t. CM, iff its corresponding occurrence in c_{j,1} or CM is executable w.r.t. CM; these are the only executable occurrences.

d. CM = [CM_1, ..., CM_n]:

CM_j = \langle x_1^{j,1}, V_1, x_1^{j,2}, V_2, ..., x_j^{j,m_j}, V_j, x_{j,m_j+1}^{j}, CM_j, j = 1, ..., n,
CM = \langle x_1^{1,1}, V_1, ..., x_n^{1,m_n+1}, V_n, CM_1, ..., CM_n \rangle

and x_1^{j,1} = x_j^{j,1}, j = 1, ..., n.

It follows from the definitions that

(1) Two occurrences of some procedure symbol both contained in CM, identify each other w.r.t. CM, iff they identify each other w.r.t. CM, j = 1, ..., n; an occurrence of some procedure symbol contained in V_j, as occurring in [V_1, ..., V_n, 1] also identifies the corresponding occurrence of this symbol contained in CM_j, j = 1, ..., n.

(2) An occurrence of some procedure symbol contained in CM is executable w.r.t. CM, iff it is executable w.r.t. CM, j = 1, ..., n; these are the only executable occurrences.

Next we define two transformations of computation models, \( \tau_1 \) and \( \tau_2 \), which are essential to the proofs of lemmas 2.5 and 2.6:

In the following definition \( x_1 \ V_1 \ x_2 \ V_2 \ ... \ x_n \ V_n \ x_{n+1} \) 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 \( \tau_1(CM) \) 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 \( V_j \).

Step 2: Replace all considered occurrences by \( W_j \), for all \( j \in J \).
For arbitrary \( M \), \( t_2(M) \) is obtained from \( M \) by executing the following steps:

**Step 1:** Consider for every \( j \in J \) all occurrences of \( P_j \) in \( M \) identified by occurrences of \( P_j \) in \( V_j \).

**Step 2:** Mark all those considered occurrences which are executable.

**Step 3:** Replace all other considered occurrences of \( P_j \) by \( S_j \) (with \( P_j \models S_j \)).

**Step 4:** Replace every combination \( \ldots x_k \overset{\cdot}{P_j} x_{k+1} S_j x_{k+2} \ldots \) by \( \ldots x_k \overset{\cdot}{S_j} x_{k+2} \ldots \) and every combination \( x_k \overset{\cdot}{P_j} S_j x_{k+1} S_j x_{k+2} \ldots \) by \( \ldots x_k S_j \overset{\cdot}{S_j} x_{k+2} \ldots \), 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. \( t_1(\overbrace{CM_1;CM_2}^n) = t_1(\overbrace{CM_1}^n) \circ t_1(\overbrace{CM_2}^n) \),
   \[ t_1((\overbrace{O_0(p) \rightarrow CM, W}^n)) = (\overbrace{O_0(p) \rightarrow t_1(CM), \overbrace{W[X_j/X_j}_{j \in J}}^n)^* \text{ and } t_1(\overbrace{\overbrace{CM_1; \ldots ; CM_n}^n)) = \overbrace{t_1(CM_1), \ldots , t_1(CM_n)}^n. \]

2. \( t_2(\overbrace{CM_1;CM_2}^n) = t_2(\overbrace{CM_1}^n) \circ t_2(\overbrace{CM_2}^n) \),
   \[ t_2((\overbrace{O_0(p) \rightarrow CM, W}^n)) = (\overbrace{O_0(p) \rightarrow t_2(CM), \overbrace{W[1]}^1})^* \text{ and } t_2(\overbrace{\overbrace{CM_1; \ldots ; CM_n}^n)) = \overbrace{t_2(CM_1), \ldots , t_2(CM_n)}^n. \]

**Lemma 2.5.** Let \( S \) be a closed statement scheme, \( M \) 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 \( M \) into a computation model for \( \overbrace{\overbrace{S[W_j/X_j}_{j \in J}}^n) \).

*) 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 ∈ A ∪ C (R ∈ 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 ∈ 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 W_1[W_j/X_j]_{j∈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 V_1[W_j/X_j]_{j∈J} z and x W_1[W_j/X_j]_{j∈J} y for appropriate z, whence, by definitions 2.2 and 2.6, t_1(CM) is a computation model for (V_1;W_1)[W_j/X_j]_{j∈J}.

d. S = (V_1);W_1: Step 1 of t_1 results in considering for all j ∈ 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_1 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 $(V_1)W_j$.

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 $(V_1)W_j[X_j]_{j \in J}$ 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 $x \bar{V}_1[W_j/X_j]_{j \in J} z$ and $z \bar{V}_1[W_j/X_j]_{j \in J} y$ for appropriate $z$, whence, by definitions 2.2 and 2.6, $t_1(CM)$ is a computation model for $(V_1)W_j[X_j]_{j \in J}$.

e. $S = (p \rightarrow V_1, V_2)$ 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: $<x P_j X S_j \ldots y, CM>$.

Thus $t_2(CM) = <cs', CM>$, 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.

(2) The occurrences of $P_j$ in CM identified by occurrences of $P_j$ in $W_1$ as contained in $V_1W_1$. 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 $V_1W_1$, 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

$$<x_1 V_{1}^{*}; W_1 \ldots x_n V_{n}^{*}; W_1 \ldots y_m W_m^{*}; y_{m+1}, CM_1^{*} \cup CM_2^{*}>,$$

with $V_k^{*}$, $W_1^{*}$ and $CM_1^{*}$ indicating the result of marking the executable occurrences of $P_j$ in $V_k$, $W_1$ and $CM_1$, $k = 1, \ldots, n$, $i = 1, \ldots, m$, $i = 1, 2$, which are considered in step 1.

Then step 3 is performed, whence we obtain

$$<x_1 V_{1}^{*}[S_j^{P_j}]_{j < j \in J}; W_1^{[1]} \ldots x_n V_{n}^{*}[S_j^{P_j}]_{j \in J}; W_1^{[1]} \ldots y_m W_m^{*}[S_j^{P_j}]_{j \in J}; y_{m+1}, CM_1^{**} \cup CM_2^{**}>,$$

with $V_k^{*}[S_j^{P_j}]_{j \in J}$, $W_1^{[1]}$ and $CM_1^{**}$ 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 $CM_1^{*}$, $k = 1, \ldots, n$, $i = 1, \ldots, m$, $i = 1, 2$.

The problem with the construct obtained in step 3 is that parts occur of the form $Z_1 V; S_j z_{1+1} P_j^{*} z_{1+2} S_j \ldots$, violating definition 2.4 of computation model (e.g., if $V_1 = W_1 = P_j$, then $W_1^{[1]} = S_j$ but $W_1^{[1]} = P_j$).

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
$x_1^* V_{i_1}^* [S_j/P_j]_{j \in J}; W_{i_1}^{[1]} x_2^* V_{i_2}^* [S_j/P_j]_{j \in J}; W_2^{[1]} \ldots x_s^* V_{i_s}^* [S_j/P_j]_{j \in J}; W_s^{[1]} x_{i_s+1}^{*}$

and $CM_i^*$,

with

$t_2(CM_i^*) = <x_1^* V_{i_1}^* [S_j/P_j]_{j \in J}, x_2^* V_{i_2}^* [S_j/P_j]_{j \in J}, \ldots, x_s^* V_{i_s}^* [S_j/P_j]_{j \in J}, x_{i_s+1}^{*}, CM_i^*>$

by the induction hypothesis, whence $V_{i_1}^* [S_j/P_j]_{j \in J} = W_{i_1}^{[1]}$, $x_1 = x$ and

$x_{i_s} = x_{n+1}$, as the set of indices $k$ for which parts $V_{i_s}^* [S_j/P_j]_{j \in J}; W_1^{[1]} x_{k+1}$

are deleted from $cs_i$ is the same set as the set of indices $k$ for which

parts $V_{i_s}^* [S_j/P_j]_{j \in J}; x_{k+1}$ are deleted from

$x_1^* V_{i_1}^* [S_j/P_j]_{j \in J} x_2^* V_{i_2}^* [S_j/P_j]_{j \in J} \ldots x_n^* V_{i_n}^* [S_j/P_j]_{j \in J} x_{n+1}^{*}$, the result of applying steps 1, 2 and 3 to $cs_i$.

Application of step 4 to $cs_i$ and $CM_i^*$ results by the induction hypothesis in

$y_1^* V_{j_1}^* [S_j/P_j]_{j \in J} y_2^* V_{j_2}^* [S_j/P_j]_{j \in J} \ldots y_t^* V_{j_t}^* [S_j/P_j]_{j \in J} y_{j+1}^{*}$ and $CM_i^*$.

the two constituent parts of $t_2(CM_i^*)$, whence $y_{j_1} = x_{n+1}$, $y_{j+1} = y$ and

$W_{j_1}^{[1]} = W_1^{[1]}$. Thus we conclude that $t_2(CM) = t_2(CM_1); t_2(CM_2)$. As $v_{i_1}^{[1]} w_1^{[1]} = (v_{i_1}^{[1]} w_1^{[1]}$ by definitions 2.2 and 2.6, $t_2(CM)$ is a computation model for

$x S^{[1]} y$.

d. $S = (V_1, V_2); (p \rightarrow V_1, V_2, V_3)$ 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 sequences $x_1 V_1 x_2 V_2 \ldots x_n V_n x_{n+1}$; 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 $l$ be the smallest number such that $S^{[l]}$ 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 $x S^{[l]} [\sigma_j/X_j]_{j \in J} y$ exists.

As by lemma 2.4 $S^{[l]} [\sigma_j/X_j]_{j \in J} = S^{(l+1)}$, it suffices to take $l+1$ 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 $j \in J$, and $\sigma$ 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. $v_1(X) = v_2(X)$, $X \in X - \{X_j \mid j \in J\}$

then the following holds:

$\phi^{\sigma}(v_1) \subseteq \phi^{\sigma}(v_2)$.

Proof. By induction on the complexity of $\sigma$.

a. $\sigma \in A \cup B \cup C \cup X$: Obvious.

b. $\sigma = \sigma_1 \sigma_2$, $\sigma_1 \cup \sigma_2$, $\sigma_1 \cap \sigma_2$, $\overline{\sigma_1}$:

$\phi^{\sigma_1 \sigma_2}(v_1) = \phi^{\sigma_1}(v_1) \phi^{\sigma_2}(v_1)$ and $\phi^{\sigma_1 \sigma_2}(v_1) \Longleftrightarrow \exists x, y \in \phi^{\sigma_1}(v_1)$ and $\phi^{\sigma_2}(v_1)$.

By the induction hypothesis, $\phi^{\sigma_1}(v_1) \subseteq \phi^{\sigma_1}(v_2)$, $i = 1, 2$.

Thus $x, y \in \phi^{\sigma_1}(v_1) \phi^{\sigma_2}(v_1)$ implies $x, y \in \phi^{\sigma_1}(v_1) \phi^{\sigma_2}(v_2)$, whence $\phi^{\sigma_1 \sigma_2}(v_1) \subseteq \phi^{\sigma_1}(v_2)$ follows from the definitions.

The cases $\sigma = \sigma_1 \cup \sigma_2$, $\sigma_1 \cap \sigma_2$ and $\overline{\sigma_1}$ are proved similarly.

c. $\sigma = \overline{\sigma_1}$: By syntactic continuity of $\sigma$ in all $X_j$, $j \in J$, no $X_j$ occurs in $\sigma_1$ for any $j \in J$, whence $\phi^{\overline{\sigma_1}}(v_1) = \phi^{\overline{\sigma_1}}(v_2)$.

Therefore $\phi^{\overline{\sigma_1}}(v_1) = \phi^{\overline{\sigma_1}}(v_2) = \phi^{\overline{\sigma_1}}(v_2)$.

d. $\sigma = \mu_{X_1} \ldots X_n[\sigma_1, \ldots, \sigma_n]$:

$\phi^{\sigma}(v_2) = \phi^{\overline{\sigma_1}}(v_2)$

$\{v_1(X_j) \mid v_1(X_j) \subseteq v_2(X_j), 1 \leq 1, \ldots, n, \text{ and}

v_1(X) = v_2(X), X \in X - \{X_1, \ldots, X_n\}\}_{k}$

... (a.2.1)

Let $v_1'$ satisfy the conditions mentioned in (a.2.1).

Define $v_1'$ by: $v_1'(X_j) = v_2'(X_j)$, $1 \leq 1, \ldots, n$, and $v_1'(X) = v_1(X)$.

Then the conditions for monotonicity, w.r.t. the index set $J \cup \{1, \ldots, n\}$, and $v_1'$ and $v_1''$, are fulfilled, whence by the induction hypothesis:

$\phi^{\overline{\sigma_1}}(v_1') \subseteq \text{(monotonicity)} \phi^{\overline{\sigma_1}}(v_2') \subseteq v_2'(X_j) = v_1'(X_j), 1 \leq 1, \ldots, n$. 

Thus,

$$\forall \langle v_1^i(X_1) \rangle_{i=1}^n \mid \psi < \sigma_1 > (v_1^i) \subseteq v_1^i(X_1), \ 1 = 1, \ldots, n, \text{ and}$$

$$v_1(X) = v_1(X), \ X \in X - \{X_1, \ldots, X_n\} \}$$

$$\exists \langle v_2^i(X_1) \rangle_{i=1}^n \mid \psi < \sigma_2 > (v_2^i) \subseteq v_2^i(X_1), \ 1 = 1, \ldots, n, \text{ and}$$

$$v_2(X) = v_2(X), \ X \in X - \{X_1, \ldots, X_n\},$$

whence

$$\psi < \mu_k X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n] > (v_1) \subseteq \psi < \mu_k X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n] > (v_2). \ \square$$

**Lemma 3.2.** (Continuity). Let \( J \) be any index set, \( X \subseteq X \) for all \( j \in J \), \( \sigma \in I \) be syntactically continuous in all \( X_j \), \( j \in J \), \( \nu \) 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:

$$\psi < \sigma > (v) = \bigcup_{i=0}^\infty \psi < \sigma > (v_i).$$

**Proof.** ≥: By monotonicity (Lemma 3.1).

≤: By induction on the complexity of \( \sigma \).

a. \( \sigma \in A \cup B \cup C \cup X \): Obvious.

b. \( \sigma = \sigma_1 \sigma_2 \cup \sigma_1 \sigma_2 \cap \sigma_2 \sigma_1 \):

$$\psi < \sigma_1 \sigma_2 > (v) = \psi < \sigma_1 > (v) ; \psi < \sigma_2 > (v) \subseteq (\text{induction hypothesis})$$

$$\bigcup_{i=0}^\infty \psi < \sigma_1 > (v_i) ; \bigcup_{j=0}^\infty \psi < \sigma_2 > (v_j) = \bigcup_{i=0}^\infty \bigcup_{j=0}^\infty \psi < \sigma_1 > (v_i) ; \psi < \sigma_2 > (v_j),$$

by a property of relations.

\( E_1 \)

$$\bigcup_{i=0}^\infty \psi < \sigma_1 > (v_i) ; \psi < \sigma_2 > (v_i) \subseteq E_1 \) is obvious and

$$\bigcup_{i=0}^\infty \psi < \sigma_1 > (v_i) ; \psi < \sigma_2 > (v_i) \supseteq E_1 \) follows from

$$\psi < \sigma_1 > (v_i) ; \psi < \sigma_2 > (v_i) \subseteq (\text{monotonicity}) \psi < \sigma_1 > (v_{\max(i,j)}) ; \psi < \sigma_2 > (v_{\max(i,j)}).$$
Thus, $\phi_{<\sigma_1;\sigma_2>}(v) \subseteq \bigcup_{i=0}^{n} \phi_{<\sigma_1;\sigma_2>}(v_1)$.

The cases $\sigma = \sigma_1 \cup \sigma_2$, $\sigma_1 \cap \sigma_2$ and $\sigma_1$ are proved similarly.

c. $\sigma = \sigma_1^\sim$: By syntactic continuity of $\sigma$ in all $X_j$, $j \in J$, no $X_j$ occurs in $\sigma_1$ for any $j \in J$, whence $\phi_{<\sigma_1^\sim>}(v) = \phi_{<\sigma_1>}(v_1)$.
Therefore $\phi_{<\sigma_1^\sim>}(v) = \phi_{<\sigma_1>}(v_1) = \phi_{<\sigma_1^\sim>}(v_1)$ for all $i \in N$,
whence $\phi_{<\sigma_1^\sim>}(v) = \bigcup_{i=0}^{n} \phi_{<\sigma_1^\sim>}(v_1)$.

d. $\sigma = \mu_X X_1 \ldots X_n [\sigma_1, \ldots, \sigma_n]$:
Define $V_i$, for all $i \in N$, by

$\forall_{i \in N} \left( v_i^i \mid \phi_{<\sigma_i>}(v_i^i) \subseteq v_i^i(X_i), 1 = 1, \ldots, n, \right.$

$v_i^i(X) = v_i(X), X \in X - (X_1, \ldots, X_n)$.

Then $\bigcup_{i=0}^{n} \phi_{<\sigma>}(v_i) = \bigcup_{i=0}^{n} \left( n(v_i^i(X_i) \mid v_i^i \in V_i). \right)$ \hspace{1cm} (a.2.2)

Next define valuation $v_i^*$, for $i \in N$, by:

$v_i^*(X_1) = n(v_i^i(X_1) \mid v_i^i \in V_i), 1 = 1, \ldots, n,$

$v_i^*(X) = v_i(X), X \in X - (X_1, \ldots, X_n)$.

Then $\bigcup_{i=0}^{n} \phi_{<\sigma>}(v_i) = \bigcup_{i=0}^{n} v_i^*(X_1) \text{ follows directly from (a.2.2)} \ldots (a.2.3)$

Let $E_2$ be defined by

$E_2 = n(\bigcup_{i=0}^{n} v_i^i(X_1) \mid \text{for all } i \in N, v_i^i \in V_i).$ \hspace{1cm} (a.2.4)

First we shall prove that $\bigcup_{i=0}^{n} v_i^*(X_1) = E_2$, and then that $\phi_{<\sigma>}(v) \subseteq E_2$,
whence the result follows from (a.2.3).

$\bigcup_{i=0}^{n} v_i^*(X_1) = E_2$:

$\subseteq$: Since for $v_i^i \in V_i$ and $i \in N$ $v_i^*(X_1) \subseteq v_i^i(X_1)$ holds, we have

$\bigcup_{i=0}^{n} v_i^i(X_1) \subseteq \bigcup_{i=0}^{n} v_i^*(X_1)$; hence

$\bigcup_{i=0}^{n} v_i^*(X_1) \subseteq \bigcap_{i=0}^{n} v_i^i(X_1) \mid v_i^i \in V_i) = E_2$ follows.

$\supseteq$: We prove below that $v_i^* \in V_i$, for $i \in N$, whence $E_2 \subseteq \bigcup_{i=0}^{n} v_i^*(X_1)$

follows from $E_2$’s definition.

By definition, for $i \in N$, $v_i^*(X_1) \subseteq v_i^i(X_1)$, for $1 = 1, \ldots, n$ and $v_i^i \in V_i$,
and $v_i^*(X) = v_i(X), X \in X - (X_1, \ldots, X_n)$.

Therefore the conditions for applying monotonicity (lemma 3.1) are
fulfilled, whence $\phi_{<\sigma_i>}(v_1^i) \subseteq \phi_{<\sigma_1>}(v_1^i) \subseteq v_i^i(X_1)$ for all $v_i^i \in V_i$, and

$\phi_{<\sigma_1>}(v_1^i) \subseteq v_i^1 \bigcap_{v_1^i \in V_i} v_i^1(X_1) = v_i^1(X_1), 1 = 1, \ldots, n.$
Moreover, \( v^*_i(X) = v_i(X) \) for \( X \in \mathcal{X} \setminus \{X_1, \ldots, X_n\} \). Hence \( v^*_i \in \mathcal{V}_i \) follows, for \( i \in N \).

\[ \phi_{<\sigma_i}(v) \subseteq E_2 : \]

First we demonstrate that one can restrict oneself in (a.2.4) to intersections of unions of \( v^*_i(X_1) \) such that \( v^*_i(X_1) \subseteq v^*_i(X_{i+1}), I = \{1, \ldots, n\} \):

Let \( \langle v^*_i \rangle_{i=0}^{\infty} \) be a sequence consisting of valuations which satisfy for every \( i \in N \), \( \phi_{<\sigma_i}(v^*_i) \subseteq v^*_i(X_1), \quad I = \{1, \ldots, n\} \), and \( v^*_i(X) = v_i(X) \), for \( X \in \mathcal{X} \setminus \{X_1, \ldots, X_n\} \).

Define \( \langle v^*_{1_i} \rangle_{i=0}^{\infty} \) as follows:

For every \( i \in N \), \( v^*_1(X_1) = \cap_{j=1}^{\infty} v_j(X_1), \quad I = \{1, \ldots, n\} \), and \( v^*_1(X) = v_1(X) \), \( X \in \mathcal{X} \setminus \{X_1, \ldots, X_n\} \).

This sequence of valuations satisfies the following properties:

1. For every \( i \in N \), \( \phi_{<\sigma_i}(v^*_{1_i}) \subseteq v^*_1(X_1), \quad I = \{1, \ldots, n\} \).

   This can be deduced from the fact that, for all \( j \geq i \),
   \[ \phi_{<\sigma_i}(v^*_{1_i}) \subseteq (\text{monotonicity}) \phi_{<\sigma_i}(v^*_{1_j}) \subseteq v^*_1(X_1), \quad I = \{1, \ldots, n\} \).

2. For every \( i \in N \), \( v^*_1(X_1) \subseteq v^*_{1_i}(X_1), \quad I = \{1, \ldots, n\} \).

3. For every \( i \in N \), \( v^*_1(X_1) \subseteq \cup_{i=0}^{\infty} v^*_1(X_1), \quad I = \{1, \ldots, n\} \).

Therefore, as every \( n \)-tuple \( \langle v^*_1(X_1), \ldots, v^*_1(X_1) \rangle_1 \) with \( \langle v^*_1 \rangle_{i=0}^{\infty} \) satisfying the conditions mentioned above contains coordinatewise an \( n \)-tuple \( \langle v^*_1(X_1), \ldots, v^*_1(X_1) \rangle_1 \) with \( \langle v^*_1 \rangle_{i=0}^{\infty} \) also satisfying these conditions, in addition to the extra condition \( v^*_1(X_1) \subseteq v^*_{1_i}(X_1), \quad I = \{1, \ldots, n\}, \quad i \in N \), one can restrict oneself in (a.2.4) to \( k \)-th components of intersections of the latter.

Define \( v^* \) by \( v^*(X_1) = \cup_{i=0}^{\infty} v^*_1(X_1), \quad I = \{1, \ldots, n\} \), and \( v^*(X) = v(X), \quad X \in \mathcal{X} \setminus \{X_1, \ldots, X_n\} \).

Then the conditions for continuity, w.r.t. the index set \( J \cup \{1, \ldots, n\} \), and \( v^* \) and \( \langle v^* \rangle_{i=0}^{\infty} \), are fulfilled, whence by the induction hypothesis:

\[ \phi_{<\sigma_i}(v^*) = (\text{continuity}) \cup_{i=0}^{\infty} \phi_{<\sigma_i}(v^*_1) \subseteq (\text{point 1 above}) \]

\[ \cup_{i=0}^{\infty} v^*_1(X_1) = v^*(X_1), \quad I = \{1, \ldots, n\}. \]

Hence,

\[ \phi_{<\sigma_1, \ldots, \sigma_n}(v) = \]
= (n(<v'_{i1}>_{i=1}^{n}) | \phi_{<\sigma_{i1}>}(v') \subseteq v'(X_{i}), \quad 1 = 1, \ldots, n, \quad \text{and} \\
v'(X) = v(x), \quad x \in X - \{X_{1}, \ldots, X_{n}\}) )_{k} \subseteq \text{(from above)} \\
(\cap_{i=0}^{n} v''_{i1}(X_{i1})_{i=1}^{n} | \text{for } i \in N, \quad \phi_{<\sigma_{i1}>}(v'_{i1}) \subseteq v''_{i1}(X_{i1}), \quad 1 = 1, \ldots, n, \\
v''_{i1}(X_{i1}) \subseteq v''_{i+11}(X_{i+11}), \quad 1 = 1, \ldots, n, \quad \text{and} \quad v''_{1}(X) = v_{1}(X), \quad x \in X - \{X_{1}, \ldots, X_{n}\}) )_{k} = \\
= \text{(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. \( v_{1}(X) = v_{2}(X), \quad X \in X - \{x_{j}\}_{j \in J}, \)
2. \( v_{1}(x_{j}) = \phi_{<\tau_{j}>(v_{2})}, \quad j \in J, \)

then the following holds:

\[ \phi_{<\sigma>}(v_{1}) = \phi_{<[\tau_{j}/x_{j}]_{j \in J}>(v_{2})}. \]

**Proof.** By induction on the complexity of \( \sigma \).

We only consider the case \( \sigma = \mu_{m}X_{1} \ldots X_{n}[\sigma_{1}, \ldots, \sigma_{n}]. \)

By definition,

\[ \mu_{m}X_{1} \ldots X_{n}[\sigma_{1}, \ldots, \sigma_{n}][\tau_{j}/x_{j}]_{j \in J} = \]

\[ = \mu_{m}Y_{1} \ldots Y_{n}[\sigma_{1}[Y_{1}/X_{1}], \ldots, n[\tau_{j}/x_{j}]_{j \in J} = \]

\[ \ldots, \sigma_{n}[Y_{1}/X_{1}], \ldots, n[\tau_{j}/x_{j}]_{j \in J*}, \]

with \( J* = J - \{1, \ldots, n\} \) and \( Y_{1}, \ldots, Y_{n} \) relaton variables different from \( x_{j}, \quad j \in J , \) and not occurring in \( \sigma_{k}, \quad k = 1, \ldots, n, \) or \( \tau_{j}, \quad j \in J*. \)

Let

\[ E_{1} \equiv \]

\[ (n(<v'_{i1}>_{k=1}^{n}) | \phi_{<\sigma_{i1}>}(v'_{i}) \subseteq v'_{i}(X_{i}), \quad k = 1, \ldots, n, \text{and} \\
v'_{i}(X) = v_{1}(X), \quad x \in X - \{X_{1}, \ldots, X_{n}\}) )_{m}, \]

\[ E_{2} \equiv \]

\[ (n(<v'_{i1}>_{k=1}^{n}) | \phi_{<\sigma_{i1}>}(Y_{i1}/X_{1}, \ldots, n[v'_{i}]_{i=1}^{n}) \subseteq v'_{i}(Y_{i}), \quad k = 1, \ldots, n, \text{and} \]
\[ v'_1(X) = v_1(X), \; X \in X - \{Y_1, \ldots, Y_n\} \] and

\[ E_3 = \left( \lambda k. \sigma_k \in \{1, 2, \ldots, n\} \middle| \phi \sigma_k \circ [Y_1/X_1]_{1=1} \ldots [Y_n/X_n]_{n} \circ (v'_2) \subseteq v'_2(Y_k) \right) \]

\[ k = 1, \ldots, n, \text{ and } v'_2(X) = v_2(X), \; X \in X - \{Y_1, \ldots, Y_n\} \]

In order to prove \( \phi \circ \circ (v'_1) = \phi \circ \circ (v'_2) \), that is \( E_1 = E_3 \), we first prove \( E_2 = E_3 \) and then \( E_1 = E_2 \):

\[ E_2 = E_3: \]

\[ \subseteq: \text{Let } v'_2 \text{ satisfy } v'_2(X) = v_2(X), \text{ for } X \in X - \{Y_1, \ldots, Y_n\}, \] and \( \phi \circ \circ (v'_2) \subseteq v'_2(Y_k), \; k = 1, \ldots, n \).

Define \( v'_1 \) by \( v'_1(X) = v'_2(X) \) for \( X \in X - \{X_j\}_{j \in J} \) and \( \phi \circ \circ (v'_1) = \phi \circ \circ (v'_2) \), for \( j \in J \), and define \( v''_1 \) by \( v''_1(X) = v'_2(X) \) for \( X \in X - \{X_j\}_{j \in J^*} \) and \( \phi \circ \circ (v''_1) = \phi \circ \circ (v'_2) \), for \( j \in J^* \).

By the induction hypothesis, \( \phi \circ \circ (v'_1) = \phi \circ \circ (v'_2) \).

As \( X_1, \ldots, X_n \) do not occur in \( \sigma_k \circ [Y_1/X_1]_{1=1} \ldots [Y_n/X_n]_{n} \circ (v'_1) \), for \( k = 1, \ldots, n \), as

\[ (X_1)_{j \in J^*} \cap \{Y_1, \ldots, Y_n\} = \emptyset. \]

Thus \( \phi \circ \circ \circ (v'_2) = \phi \circ \circ \circ (v'_1) \), for \( k = 1, \ldots, n \).

Furthermore, \( v'_1(X_j) = \phi \circ \circ (v'_2) = (Y_1, \ldots, Y_n) \) do not occur in \( \tau_j \), \( \phi \circ \circ (v_2) \) = \( v_1(X_j), \; j \in J \), and \( v'_1(X) = v'_2(X) = v_2(X) \) = (assumption) \( v_1(X) \) for \( X \in X - \{X_j\}_{j \in J}, \) \( \{Y_1, \ldots, Y_n\}, \) whence \( v'_1 \) satisfies the conditions mentioned in \( E_2 \).

As \( \circ \circ \circ (v'_2) = \circ \circ \circ (v'_1) \), we obtain \( E_2 \subseteq E_3 \).

\[ \supseteq: \text{Let } v'_1 \text{ satisfy } v'_1(X) = v_1(X), \text{ for } X \in X - \{Y_1, \ldots, Y_n\} \] and \( \phi \circ \circ \circ (v'_1) \subseteq v'_1(Y_k), \; k = 1, \ldots, n \).

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 \( (1) \; \circ \circ \circ (v'_1) = \circ \circ (v_1) = \circ \circ (v'_2) = (Y_1, \ldots, Y_n) \) do not occur in \( \tau_j \).

\( \circ \circ (v'_2) ) = (Y_1, \ldots, Y_n), \; j \in J, \)
(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

\[ \phi \circ k[Y_1/X_1]_{i=1}^n \tau_{j \in J}^{x_j} (v'_2) = \phi \circ k[Y_1/X_1]_{i=1}^n (v'_1) \]

Since \( \circ k[Y_1/X_1]_{i=1}^n \tau_{j \in J}^{x_j} = \circ k[Y_1/X_1]_{i=1}^n \tau_{j \in J}^{x_j} = \circ k \), as no \( X_1, \ldots, X_n \) occur in \( \circ k[Y_1/X_1]_{i=1}^n \),

\[ \phi \circ k > (v'_2) = \phi \circ k[Y_1/X_1]_{i=1}^n > (v'_1) \leq v'_1(Y_k) = v'_2(Y_k) \]

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 \( E_2 \geq E_3 \).

\[ E_1 = E_2 \]

\[ \geq \]: Let \( v''_1 \) satisfy \( \phi \circ k > (v''_1) \leq v''_1(X_k) \), \( k = 1, \ldots, n \), 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, \( \phi \circ k > (v''_1) = \phi \circ k[Y_1/X_1]_{i=1}^n > (v'_1) \).

Therefore, \( \phi \circ k[Y_1/X_1]_{i=1}^n > (v'_1) = \phi \circ k > (v''_1) \leq v''_1(Y_k) = v'_1(Y_k) \), \( k = 1, \ldots, n \). As \( v'_1(X) = v_1(X) \), \( X \in X - \{Y_1, \ldots, Y_n\} \), it can be deduced that \( E_1 \geq E_2 \) holds.

\[ \leq \]: As \( \circ k[Y_1/X_1]_{i=1}^n > (X_1/X_1)_{i=1}^n = \circ k \), the proof of this part is similar to the proof above. ∎
APPENDIX 3: PROOF OF TARKSI'S "UNPROVABLE ASSERTION"

THEOREM.
\[ \vdash X_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2 \subseteq \]
\[ X_1; (X_1;Y_1 \cap X_2;Z_2 \cap Z_1;Z_2) \cap (Z_1;Y_1 \cap Z_2;Y_2); Y_2. \]

Proof.
A. \[ [X_1;Y_1;Z_1] = [X_1;Y_1;Z_1];([Y_1;Y_1;Y_2] \cap E). \]
\[ \subseteq \text{ trivial.} \]
\[ \subseteq [X_1;Y_1;Z_1] = \text{(lemma 4.3.c)} [X_1;Y_1;Z_1];([X_1;Y_1;Y_2] \cap E) \]
\[ \subseteq [X_1;Y_1;Z_1];([Y_1;Y_1;Y_2] \cap E) \subseteq [X_1;Y_1;Z_1];([Y_1;Y_1;Y_2] \cap E) \]
\[ \subseteq [X_1;Y_1;Z_1] = \text{(lemma 4.5.c)} [X_1;Y_1;Z_1];([Y_1;Y_1;Y_2] \cap E). \]

B. Hence, \[ X_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2 = [X_1;Y_1;Z_1];([Y_1;Y_1;Y_2] \cap E); \]
\[ (\pi_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2) \]
\[ = \text{(by part A)} [X_1;Y_1;Z_1];([Y_1;Y_1;Y_2] \cap E);([Y_1;X_2;Y_2]; \cap E);([Y_1;Y_2;Z_2] \cap E); \]
\[ (\pi_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2) \]
\[ = [X_1;Y_1;Z_1];([Y_1;Y_1;Y_2] \cap E);([Y_1;X_2;Y_2]; \cap E);([Y_1;Y_2;Z_2] \cap E); \]
\[ \subseteq E_1 \]
\[ = \text{[X_1;Y_1;Z_1];([Y_1;Y_1;Y_2] \cap E);([Y_1;X_2;Y_2]; \cap E);([Y_1;Y_2;Z_2] \cap E); \]
\[ (\pi_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2). \]

C. \[ E_1 = (\pi_1;([X_1;Y_1 \cap X_2;Z_2]; [Y_1;Y_1;Y_2] \cap E);([Y_1;X_2;Y_2]; \cap E);([Y_1;Y_2;Z_2] \cap E); \]
\[ E_2 \subseteq (\pi_1;([X_1;Y_1 \cap X_2;Z_2]; [Y_1;Y_1;Y_2] \cap E);([Y_1;X_2;Y_2]; \cap E);([Y_1;Y_2;Z_2] \cap E); \]
Hence,
\[ E_1 \subseteq \pi_1;([X_1;Y_1 \cap X_2;Z_2]; [Y_1;Y_1;Y_2] \cap E);([Y_1;X_2;Y_2]; \cap E);([Y_1;Y_2;Z_2] \cap E); \]
\[ E_3 \]

\[ E_2 \subseteq (\pi_1;([X_1;Y_1 \cap X_2;Z_2]; [Y_1;Y_1;Y_2] \cap E);([Y_1;X_2;Y_2]; \cap E);([Y_1;Y_2;Z_2] \cap E); \]

\[ E_3 \]

Hence,
D. By (B) and (C),

\[ X_1;X_2 \cap Y_1;Y_2 \cap Z_1;Z_2 \subseteq [X_1;X_1;Z_1];E_3;([\pi_1;X_2 \cap \pi_2;Y_2 \cap \pi_3;Z_2]) \subseteq [X_1;Y_1;Z_1];[\pi_1;\pi_1;\pi_1;X_1;Y_1;X_2;Y_2 \cap (\tilde{X}_1;Z_1 \cap X_2;\tilde{Y}_2);(\tilde{Z}_1;Y_1 \cap Z_2;\tilde{Y}_2)]_{\pi_3};\]

\[ (\pi_1;X_2 \cap \pi_2;Y_2 \cap \pi_3;Z_2) = [X_1;Y_1;Z_1];[\pi_1;X_2 \cap \pi_1;\tilde{X}_1;Y_1 \cap X_2;\tilde{Y}_2 \cap (\tilde{X}_1;Z_1 \cap \tilde{X}_2;\tilde{Y}_2);(\tilde{Z}_1;Y_1 \cap \tilde{Z}_2;\tilde{Y}_2)]_{\pi_3};\]

\[ (\tilde{X}_1;Y_1 \cap \tilde{Z}_2;\tilde{Y}_2);Y_2 \cap \pi_3;Z_2) \subseteq X_1;X_2 \cap X_1;\tilde{X}_1;Y_1 \cap X_2;\tilde{Y}_2 \cap (\tilde{X}_1;Z_1 \cap X_2;\tilde{Y}_2);(\tilde{Z}_1;Y_1 \cap Z_2;\tilde{Y}_2);Y_2 \cap Z_1;Z_2 \subseteq X_1;\tilde{X}_1;Y_1 \cap X_2;\tilde{Y}_2 \cap (\tilde{X}_1;Z_1 \cap X_2;\tilde{Y}_2);(\tilde{Z}_1;Y_1 \cap Z_2;\tilde{Y}_2);Y_2. \]
REFERENCES


[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.


[40] MANNA, Z., Properties of programs and the first-order predicate calculus, JACM, 16 (1969) 244-255.


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 7 W.R. VAN ZWET, Convex transformations of random variables, 1964. ISBN 90 6196 007 X.
MCT 13 H.A. LAUWERIER, Asymptotic expansions, 1966, out of print; replaced by MCT 54.
MCT 18 R.P. VAN DER RIET, Formula manipulation in ALGOL 60, part 2, 1968. ISBN 90 6196 038 X.

MCT 22  T.J. DEKKER, ALGOL 60 procedures in numerical algebra, part 1, 1968. ISBN 90 6196 029 0.


MCT 28  J. OOSTERHOFF, Combination of one-sided statistical tests, 1969. ISBN 90 6196 041 X.


MCT 42  W. VERVAAT, Success epochs in Bernoulli trials (with applications in number theory), 1972. ISBN 90 6196 077 0.


ISBN 90 6196 083 5.

MCT 46  R. P. van de Riet, ABC ALCOL, A portable language for formula manipula-

MCT 47  R. P. van de Riet, ABC ALCOL A portable language for formula manipula-

MCT 48  F. E. J. Kroese Arentz, P. J. W. ten Hagen & H. L. Oudshoorn, In ALCOL
        60 compiler in ALCOL 60, Text of the MC-compiler for the
        EL-20, 1973. ISBN 90 6196 086 X.


MCT 50  A. van Wijngaarden, B. J. Mailoux, J. E. L. Peck, C. H. A. Roster,
        (eds.), Revised report on the algorithmic language ALCOL 88.


MCT 53  M. J. Faber, Metrizability in generalised ordered spaces, 1974.
        ISBN 90 6196 097 5.


MCT 55  M. Hall Jr. & J. H. van Lint (eds.), Combinatorics, part 1: Theory
        of designs, finite geometry and coding theory, 1974.

MCT 56  M. Hall Jr. & J. H. van Lint (eds.), Combinatorics, part 2: Graph
        theory, foundations, partitions and combinatorial geometry,

MCT 57  M. Hall Jr. & J. H. van Lint (eds.), Combinatorics, part 3: Combinato-

MCT 58  W. Albers, Asymptotic expansions and the deficiency concept in sta-


* MCT 61  P. van Emde Boas, Abstract resource-bound classes, part 1.


MCT 64  W. J. de Schipper, Symmetries closed categories, 1975. ISBN 90 6196
        112 2.

MCT 65  J. de Vries, Topological transformation groups 1 A categorical ap-
        proach, 1975. ISBN 90 6196 113 0.

* MCT 66  H. G. J. Piela, Locally convex algebras in spectral theory and eigen-
* MCT 67  H.A. LAWERIER, Asymptotic analysis, part 2.
   ISBN 90 6196 119 X.

* MCT 68  P.P.N. DE GROEN, Singulary perturbed differential operators of
   second order. ISBN 90 6196 120 3.

* MCT 69  J.K. LENSTRA, Sequencing by enumerative methods.

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.

* MCT 72  J.K.M. JANSEN, Simple periodic and nonperiodic Lamé functions and
   their applications in the theory of electromagnetism.
   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

An asterisk before the number means "to appear".