

Lecture Notes in Computer Science

74

**Mathematical Foundations
of Computer Science 1979**

Proceedings. Olomouc, Czechoslovakia

Edited by J. Bečvář

TP301
M4
1979

8163359

Lecture Notes in Computer Science

TP301-53
M426
1979

Edited by G. Goos and J. Hartmanis

74

Mathematical Foundations of Computer Science 1979

Proceedings, 8th Symposium,
Olomouc, Czechoslovakia, September 3-7, 1979



Edited by J. Bečvář



E8163359

Springer-Verlag
Berlin Heidelberg New York 1979

Editorial Board

P. Brinch Hansen D. Gries C. Moler G. Seegmüller
J. Stoer N. Wirth

Editor

Jiří Bečvář

Mathematical Institute

Czechoslovak Academy of Sciences

Žitná 25

115 67 Prague 1/Czechoslovakia

74



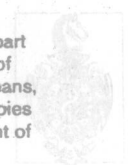
AMS Subject Classifications (1970): 02E10, 02E15, 02F10, 02F15,
68A05, 68A20, 68A25, 68A30, 68A45, 68A50
CR Subject Classifications (1974):

ISBN 3-540-09526-8 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-09526-8 Springer-Verlag New York Heidelberg Berlin

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically those of translation, reprinting, re-use of illustrations, broadcasting, reproduction by photocopying machine or similar means, and storage in data banks. Under § 54 of the German Copyright Law where copies are made for other than private use, a fee is payable to the publisher, the amount of the fee to be determined by agreement with the publisher.

© by Springer-Verlag Berlin Heidelberg 1979
Printed in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210





FOREWORD

This volume contains the papers which were selected for presentation at the symposium on Mathematical Foundations of Computer Science - MFCS '79, held in Olomouc, Czechoslovakia, September 3 - 7, 1979.

The symposium is the eighth in a series of annual international meetings which take place alternately in Czechoslovakia and Poland. It has been organized by the Mathematical Institute of the Czechoslovak Academy of Sciences, Prague, the Faculty of Mathematics and Physics of Charles University, Prague, and the Faculty of Natural Sciences of Palacký University, Olomouc, in co-operation with the Federal Ministry for Technical and Investment Development, the Technical University, Prague, the Computing Research Centre, Bratislava, the Faculty of Natural Sciences of Komenský University, Bratislava, and the Faculty of Natural Sciences of Šafárik University, Košice.

The articles in these Proceedings include invited papers and short communications. The latter were selected from among 95 extended abstracts submitted in response to the call for papers. Selection was made on the basis of originality and relevance to theoretical computer science by the following Program Committee: J. Bečvář /Chairman/, J. Gruska, P. Hájek, M. Chytil, J. Král, M. Novotný, B. Rován. A number of referees helped the Program Committee in the evaluation of the abstracts.

The papers included in these Proceedings were not formally refereed. It is anticipated that most of them will appear in a published and completed form in scientific journals.

The organizers of the symposium are much indebted to all those who contributed to the program, in particular to the authors of the papers. Special thanks are due to the referees of the abstracts. Thanks are also due to all the above mentioned co-operating institutions for their valuable assistance and support, and to all the persons who helped in organizing the symposium. The Organizing Committee consisted of J. Bečvář, J. Gregor, J. Gruska, P. Hájek, I. Havel, Š. Hudák, M. Chytil, J. Král, F. Krutský, B. Miniberger, M. Novotný, A. Rázek, Z. Renc, B. Rován, and M. Vlach /Chairman/. The Program Chairman acknowledges with gratitude the extensive assistance of I.M. Havel, F. Pudlák, and S. Žák in editing this volume.

The organizers of the symposium wish to express their thanks to the representatives of the Palacký University in Olomouc for their support and interest in the symposium.

Finally, the help of the Springer-Verlag in the timely publication of this volume is highly appreciated.

Prague, May 1979

Jiří Bečvář

C O N T E N T SInvited lectures

J.W. de Bakker A sound and complete proof system for partial program correctness	1
J.M. Barzdin The problem of reachability and verification of programs	13
A.J. Blikle Assertion programming	26
R.V. Book Complexity classes of formal languages	43
R. Freivalds Fast probabilistic algorithms	57
J. Hartmanis and T.P. Baker Relative succinctness of representations of languages and separation of complexity classes	70
I.M. Havel On two types of loops	89
M.C.B. Hennessy and G.D. Plotkin Full abstraction for a simple parallel programming language	108
H.A. Maurer On some developments in cryptography and their applications to computer science	121
K. Mehlhorn Searching, sorting and information theory	131
R. Milner LCF: a way of doing proofs with a machine	146
V.R. Pratt Axioms or algorithms	160



A. Salomaa
Power from power series 170

A.O. Slisenko
Computational complexity of string and graph identification 182

D. Wood
A survey of grammar and L forms - 1978 191

Communications

A. Adachi, T. Kasai and E. Moriya
A theoretical study on the time analysis of programs 201

H. Andréka, I. Németi and I. Sain
Completeness problems in verification of programs
and program schemes 208

J.-M. Autebert
Relationships between AFDL's and cylinders 219

G. Comyn and G. Werner
Computable data types 228

G. Cousineau and P. Enjalbert
Program equivalence and provability 237

K. Culik II and J. Karhumäki
Interactive L systems with almost interactionless behaviour 246

R.P. Daley
On the simplification of constructions in degrees
of unsolvability via computational complexity 258

W. Damm
An algebraic extension of the Chomsky-hierarchy 266

M.I. Dekhtjar
Bounds on computational complexity and approximability
of initial segments of recursive sets 277

T. Fischer
On the weighted path length of binary search trees
for unknown access probabilities 284

G.V. Gens and E.V. Levner Computational complexity of approximation algorithms for combinatorial problems	292
A. Goralčíková and V. Koubek A reduct-and-closure algorithm for graphs	301
L. Gregušová and I. Korec Small universal Minski machines	308
T. Kamimura and G. Slutzki Parallel and two-way recognizers of directed acyclic graphs	317
A. Kanda Fully effective solutions of recursive domain equations	326
I. Kramosil A note on computational complexity of a statistical deducibility testing procedure	337
M. Kudlek Context free normal systems	346
M. Linna and M. Penttonen New proofs for jump DPDA's	354
A. de Luca and A. Restivo Synchronization and maximality for very pure subsemigroups of a free semigroup	363
G.B. Marandžjan On the sets of minimal indices of partial recursive functions	372
K. Mehlhorn Some remarks on Boolean sums	375
G. Mirkowska On the propositional algorithmic logic	381
A. Nijholt and E. Soisalon-Soininen Ch(k) grammars: a new characterization of LL(k) languages	390
T. Ottmann and D. Wood A uniform approach to balanced binary and multiway trees	398

G. Păun	
On the generative capacity of some classes of grammars with regulated rewriting	408
P. Ružička	
Validity test for Floyd's operator-precedence parsing algorithms	415
P.H. Starke	
On the languages of bounded Petri nets	425
M. Tegze	
Dyck language D_2 is not absolutely parallel	434
J. Tiuryn	
Fixed points in power-set algebra of infinite trees	443
B.A. Trakhtenbrot	
On relaxation rules in algorithmic logic	453
V. Trnková	
L-fuzzy functorial automata	463
G.E. Tseytlin	
Schematics of structural parallel programming and its applications	474
M.K. Valiev	
On axiomatization of deterministic propositional dynamic logic	482
K. Wagner	
Bounded recursion and complexity classes	492
W. Wechler	
Characterization of rational and algebraic power series	499
G. Wechsung	
A crossing measure for 2-tape Turing machines	508
J. Wiedermann	
The complexity of lexicographic sorting and searching	517
J. Winkowski	
An algebraic approach to concurrence	523

H. Yamasaki	
On multitape automata	533
S. Žák	
A Turing machine oracle hierarchy	542

Appendix *

G. Berry and J.-J. Lévy	
A survey of some syntactic results in the λ -calculus	552
G. Cousineau and M. Nivat	
On rational expressions representing infinite rational trees:	
Application to the structure of flow charts	567

* Manuscript received too late to be placed correctly in the alphabetic listing.

A SOUND AND COMPLETE PROOF SYSTEM FOR
PARTIAL PROGRAM CORRECTNESS

J.W. de Bakker
Mathematical Centre
2^e Boerhaavestraat 49, Amsterdam

1. Introduction

We investigate soundness and completeness of a proof system dealing with partial correctness of programs in a language with assignment, composition, conditionals, block structure, subscripted variables and (possibly recursive) procedures with the parameter mechanisms of call-by-value and call-by-address (call-by-variable in PASCAL, call-by-reference in FORTRAN). The paper is a continuation of Apt & de Bakker [3] presented at MFCS '76, and of its successor Apt & de Bakker [4]. In the meantime various problems not yet well-understood at that time have been pursued further and, we hope, solved.

Section 2 presents syntax and (denotational) semantics of our language; in section 3 we are confronted with an unpleasant consequence of our way of defining the semantics of a block b new x; S e, and propose as solution to restrict our correctness considerations to programs obeying the restriction that all such local variables x be *initialized*. Section 4 introduces the proof system; in the course of trying to prove its soundness we were somewhat shocked by the discovery that essential rules such as, for example, the familiar composition rule turned out to be invalid with respect to the natural validity definition, requiring a complicated refinement of that definition to remedy the situation. Section 5, finally, discusses the completeness of the system.

All proofs are omitted in this paper; they are scheduled to appear in a forthcoming publication.

Our paper could not have been written without Apt [1]. Though the technical results differ (e.g., [1] does not treat parameter mechanisms, nor does it impose the initialization requirement), there are many similarities, in particular concerning the validity definition and soundness proof. Also, the problem signalled at the beginning of section 3 was found by K.R. Apt. Various other approaches to the topic of our paper have appeared in the literature (Cartwright & Oppen [6], Clarke [7], Cook [8], Gorelick [9], to mention only a small selection; for a detailed survey see Apt [2]). However, our treatment of both soundness and completeness of the proposed proof system differs substantially from the techniques used elsewhere; in particular, we have not encountered any analogue of our validity definition in its refined form.

Acknowledgement. As should be clear from the above, our paper owes a lot to the work of K.R. Apt. J.I. Zucker contributed many helpful comments on a previous version.

2. Syntax and semantics

Convention. By writing "Let $(a \in) V$ be the set such that ..." we introduce a set V , with variable a ranging over V , such that $;$.

2.1. *Syntax.* " \equiv " denotes identity between two syntactic objects. Let $(n, m, \epsilon) ICon$ be the set of integer constants, let $(x, y, z, \epsilon) Svar$, $(a \in) Avar$, $(P, Q, \epsilon) Pvar$ be the (infinite, well-ordered) sets of *simple variables* (s.v.), *array variables* and *procedure variables*.

Let $(v, w \epsilon) IVar$ be the set of *integer variables* defined by $v ::= x | a[s]$

...	$(s, t \epsilon) Iexp$...	<i>integer expressions</i>	...	$s ::= n v s_1 + s_2 \text{if } b \text{ then } s_1 \text{ else } s_2 \text{ fi}$
	$(b \epsilon) Bexp$		<i>boolean expressions</i>		$b ::= \text{true} \text{false} s_1 = s_2 \neg b b_1 > b_2$
	$(S \epsilon) Stat$		<i>statements</i>		$S ::= v := s S_1 ; S_2 \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}$ $\quad \underline{b} \text{ new } x; S \underline{e} P(t, v)$
	$(D \epsilon) Decl$		<i>declarations</i>		$D ::= P_1 \Rightarrow B_1, \dots, P_n \Rightarrow B_n, n \geq 0$, and, for $1 \leq i, j \leq n, (P_i \equiv P_j) \Rightarrow (i=j)$
	$(B \epsilon) Pbod$		<i>procedure bodies</i>		$B ::= \langle \text{val } x, \text{ add } y : S \rangle, x \neq y$
	$(R \epsilon) Prog$		<i>programs</i>		$R ::= \langle D : S \rangle$
	$(p, q, r \epsilon) Assn$		<i>assertions</i>		$p ::= \text{true} \text{false} s_1 = s_2 \neg p p_1 > p_2 \exists x[p]$
	$(f \epsilon) Form$		<i>correctness formulae</i> (c.f.)		$f ::= p \{p\} S \{q\} f_1 \wedge f_2$
	$(g \epsilon) Gfor$		<i>generalized c.f.</i>		$g ::= \langle D : f_1 \Rightarrow f_2 \rangle$

Remarks.

1. We write $\langle D | S \rangle$ instead of $\langle D : S \rangle$, and similarly in B and g . If $R \equiv \langle D | S \rangle$, with $D \equiv \langle P_i \Rightarrow B_i \rangle_{i=1}^n$, and all procedure variables occurring in S or any of the B_i , $i = 1, \dots, n$, are included in $\{P_1, \dots, P_n\}$, we call R *closed*.
2. Our statements have local s.v. declarations, but, for simplicity's sake, no local array or procedure declarations, nor array *bounds*.
3. In $B \equiv \langle \text{val } x, \text{ add } y : S \rangle$, x is the formal value parameter and y the formal address parameter, and, in $P(t, v)$, t is the actual value par. and y the actual address par. (cf. also the definition of syntactic application in 2.2).
4. $\langle D | f \rangle$ is short for $\langle D | \text{true} \Rightarrow f \rangle$.
5. For the intended meaning of " \Rightarrow " in a formula g , cf. the remark on the validity of $\langle D | f_1 \Rightarrow f_2 \rangle$ versus the soundness of $\frac{\langle D | f_1 \rangle}{\langle D | f_2 \rangle}$ below.

2.2. Substitution and syntactic application

Substitution is denoted by $[\dots / \dots]$, e.g. we use notations such as $s[t/x]$, $p[t/x]$, $S[v/x]$, $S[a'/a]$, $S[Q/P]$, etc. In case a construct contains a variable binding operator (in $\exists x[p]$ and $\underline{b} \text{ new } x; S \underline{e}$, occurrences of x in p and S are bound) the usual precautions preventing clashes between free and bound variables apply. A notation such as

$s[y_i/x_i]_{i=1}^n$ implies that, for $1 \leq i, j \leq n$, $(x_i \equiv x_j) \Rightarrow (i=j)$. Substitution in a procedure call only affects its parameters (i.e., $P(t,v)[w/x] \equiv P(t[w/x],v[w/x])$), but not the procedure body (possibly) associated with P in the accompanying declaration D . $svar(s)$, $svar(p)$, $avar(S)$, $pvar(f)$, etc., denote the set of all *free* simple variables of s , of p , all array variables of S , all procedure variables of f , etc. Note that $svar(\langle \text{val } x, \text{ add } y | S \rangle) = svar(S) \setminus \{x, y\}$. Notations such as $svar(D, p, S, q)$ should be clear. We also employ the substitution $s[w/v]$ etc., for the definition of which we refer to de Bakker [5]. Constructs which differ at most in their bound variables are called *congruent*. The congruence relation is denoted by " \equiv ".

Syntactic application is a technique of associating with a procedure body B and two actual parameters t, v , a piece of program text $B(t,v)$ such that, for B the body of procedure P , $B(t,v)$ embodies the meaning of $P(t,v)$ according to the customary semantics of the parameter mechanism of call-by-value and call-by-address: let

$B \equiv \langle \text{val } x, \text{ add } y | S \rangle$.

- (i) $v \equiv z$. $B(t,z) \equiv \underline{b} \text{ new } u; u:=t; S[u/x][y/z] \underline{e}$, where u is the first s.v. not in $svar(x,y,z,t,S)$
- (ii) $v \equiv a[s]$. $B(t,a[s]) \equiv \underline{b} \text{ new } u_1, u_2; u_1:=t; u_2:=s; S[u_1/x][a[u_2]/y] \underline{e}$, where u_1 (u_2) is the first (second) s.v. not in $svar(x,y,s,t,S)$

2.3. Domains

A cpo $(\mathcal{X}, \sqsubseteq)$ is a partially ordered set with least element $\perp_{\mathcal{X}}$ such that each (ascending) chain $\langle x_i \rangle_{i=0}^{\infty}$ has a lub $\bigsqcup_i x_i$. Let C_1, C_2 be cpo's. If $f_1, f_2: C_1 \rightarrow C_2$, we put $f_1 \sqsubseteq f_2$ iff $f_1(x) \sqsubseteq f_2(x)$, all $x \in C_1$. For $f: C_1 \rightarrow C_2$, we call f *monotonic* if $x_1 \sqsubseteq x_2 \Rightarrow f(x_1) \sqsubseteq f(x_2)$. A monotonic function is called *continuous* if, for each chain $\langle x_i \rangle_i$, $f(\bigsqcup_i x_i) = \bigsqcup_i f(x_i)$. Each continuous function $f: C \rightarrow C$ has a *least fixed point* $\mu f = \bigsqcup_i f^i(\perp_C)$.

Let V_0 be the set of integers, $W_0 = \{tt, ff\}$ the set of truth-values, and E_0 the (infinite, well-ordered) set of addresses. Let $(\alpha \in) V = V_0 \cup \{\perp_V\}$, $(\beta \in) W = W_0 \cup \{\perp_W\}$, $(\epsilon \in) E = E_0 \cup \{\perp_E\}$, with $\alpha_1 \sqsubseteq \alpha_2$ iff $\alpha_1 = \perp_V$ or $\alpha_1 = \alpha_2$, and similarly for β, ϵ . Let $(\xi \in) Intv = Svar \cup (Avar \times V_0)$ be the set of *intermediate variables*, and let $intv(s) = svar(s) \cup \{ \langle a, \alpha \rangle \mid a \in avar(s), \alpha \in V_0 \}$, etc.. Let $(\epsilon \in) Env = Intv \rightarrow E$ be the set of *environments* which are required to satisfy: let $dom(\epsilon) = \{ \xi \in Intv \mid \epsilon(\xi) \neq \perp_E \}$, $range(\epsilon) = \{ e \in E \mid \epsilon(\xi) = e \text{ for some } \xi \in dom(\epsilon) \}$. Then (i) ϵ is 1-1 on its domain (ii) $\epsilon(a, \alpha) \neq \perp_E$ for some $\alpha \in V_0 \Leftrightarrow \epsilon(a, \alpha) \neq \perp_E$ for all $\alpha \in V_0$ (iii) $Intv \setminus dom(\epsilon) \neq \emptyset$, $E_0 \setminus range(\epsilon) \neq \emptyset$. Let, for $y \notin dom(\epsilon)$, $e \in E_0 \setminus range(\epsilon)$, $\epsilon' \stackrel{df.}{=} \epsilon \cup \langle y, e \rangle$ denote the extension of ϵ such that $\epsilon'(y) = e$, and similarly for $\epsilon \cup \langle y_1, e_1 \rangle_{i=1}^n$, $\epsilon \cup \langle \langle a, \alpha \rangle, e_{\alpha} \rangle_{\alpha \in V_0}$, etc. Let $(\sigma \in) \Sigma = (E+V) \cup \{\perp_{\Sigma}\}$ be the set of *states*, where, for $\sigma \in E+V$, $\sigma(e) = \perp_V$ iff $e = \perp_E$, and \perp_{Σ} (\perp , for short) $\stackrel{df.}{=} \lambda e. \perp_V$. Let $\sigma[\alpha/e] \stackrel{df.}{=} \perp$, if $\sigma = \perp$, and $\lambda \bar{e}. \text{ if } \bar{e} = e \text{ then } \alpha \text{ else } \sigma(\bar{e}) \text{ fi}$, otherwise. Let $(\phi \in) M \stackrel{df.}{=} Env \rightarrow (\Sigma \rightarrow \Sigma)$, $(\eta \in) H \stackrel{df.}{=} Iexp \times Ivar \rightarrow M$, $(\gamma \in) \Gamma \stackrel{df.}{=} Pvar \rightarrow H$, and let $\gamma\{\eta_i/P_i\}_i$ be defined similarly to $\sigma[\alpha/e]$. Let, finally $(\phi \in) H^n \rightarrow H$.

2.4. Semantics

The functions $R: Iexp \rightarrow (\Sigma \rightarrow V)$, $L: Ivar \rightarrow (\Sigma \rightarrow E)$, $W: Bexp \rightarrow (\Sigma \rightarrow W)$, $N: Stat \rightarrow (\Sigma \rightarrow M)$, $M: Prog \rightarrow (\Sigma \rightarrow M)$, $T: Assn \rightarrow (Env \rightarrow (\Sigma \rightarrow \{tt, ff\}))$, $F: Form \rightarrow (\Gamma \rightarrow (Env \rightarrow (\Sigma \rightarrow \{tt, ff\})))$, and $G: Gfor \rightarrow (\Gamma \rightarrow \{tt, ff\})$ are defined by

- a. If $intv(s) \notin \text{dom}(\epsilon)$ or $\sigma = \perp$ then $R(s)(\epsilon)(\sigma) = \perp_V$. Otherwise, $R(v)(\epsilon)(\sigma) = \sigma(L(v)(\epsilon)(\sigma))$, $R(m)(\epsilon)(\sigma) = \alpha$, where α is the integer denoted by the integer constant m , $R(s_1 + s_2)(\epsilon)(\sigma) = R(s_1)(\epsilon)(\sigma) + R(s_2)(\epsilon)(\sigma)$, $R(\text{if } b \text{ then } s_1 \text{ else } s_2 \text{ fi})(\epsilon)(\sigma) = \text{if } W(b)(\epsilon)(\sigma) \text{ then } R(s_1)(\epsilon)(\sigma) \text{ else } R(s_2)(\epsilon)(\sigma)$.
- b. If $intv(v) \notin \text{dom}(\epsilon)$ or $\sigma = \perp$ then $L(v)(\epsilon)(\sigma) = \perp_E$. Otherwise, $L(x)(\epsilon)(\sigma) = \epsilon(x)$, $L[a[s]](\epsilon)(\sigma) = \epsilon(a, R(s)(\epsilon)(\sigma))$.
- c. $W(b)$. Omitted.
- d. If $intv(S) \notin \text{dom}(\epsilon)$ or $\sigma = \perp$ then $N(S)(\gamma)(\epsilon)(\sigma) = \perp$. Otherwise, $N(v := s)(\gamma)(\epsilon)(\sigma) = \sigma(R(s)(\epsilon)(\sigma) / L(v)(\epsilon)(\sigma))$, $N(S_1; S_2)(\gamma)(\epsilon)(\sigma) = N(S_2)(\gamma)(\epsilon)(N(S_1)(\gamma)(\epsilon)(\sigma))$, $N(\text{if } \dots \text{ fi}) = \dots$, $N(\text{b new } x; S_e)(\gamma)(\epsilon)(\sigma) = N(S[y/x])(\gamma)(\epsilon \cup \langle y, e \rangle)(\sigma)$, where y is the first s.v. not in $\text{dom}(\epsilon)$ and e is the first address in $E_0 \setminus \text{range}(\epsilon)$. (*Remark.* The use of a new s.v. y ensures that we obtain the *static scope* rule for procedures.) $N(P(t, v))(\gamma)(\epsilon)(\sigma) = \gamma(P)(t, v)(\epsilon)(\sigma)$.
- e. If $intv(R) \notin \text{dom}(\epsilon)$ or $\sigma = \perp$ then $M(R)(\gamma)(\epsilon)(\sigma) = \perp$. Otherwise, let $R \equiv \langle D | S \rangle$, $D \equiv \langle P_i \leftrightarrow B_i \rangle_{i=1}^n$. $M(R)(\gamma)(\epsilon)(\sigma) = N(S)(\gamma(\eta_i / P_i)_{i=1}^n)(\epsilon)(\sigma)$, where $\langle \eta_1, \dots, \eta_n \rangle = \mu[\phi_1, \dots, \phi_n]$, and, for $j = 1, \dots, n$, $\phi_j = \lambda \eta_1' \dots \lambda \eta_n' \cdot \lambda t \cdot \lambda v$, $N(B_j(t, v))(\gamma(\eta_i / P_i)_i)$.
- f. If $intv(p) \notin \text{dom}(\epsilon)$ or $\sigma = \perp$ then $T(p)(\epsilon)(\sigma) = ff$. Otherwise, $T(\text{true})(\epsilon)(\sigma) = tt, \dots, T(\exists x[p])(\epsilon)(\sigma) = \exists \alpha [T(p[y/x])(\epsilon \cup \langle y, e \rangle)(\sigma(\alpha/e))]$, with $\langle y, e \rangle$ as in part d.
- g. If $intv(f) \notin \text{dom}(\epsilon)$ or $\sigma = \perp$ then $F(f)(\gamma)(\epsilon)(\sigma) = ff$. Otherwise, $F(p)(\gamma)(\epsilon)(\sigma) = T(p)(\epsilon)(\sigma)$, $F(\{p\}S\{q\})(\gamma)(\epsilon)(\sigma) = \forall \sigma' [T(p)(\epsilon)(\sigma) \wedge \sigma' = N(S)(\gamma)(\epsilon)(\sigma) \wedge \sigma' \neq \perp \Rightarrow T(q)(\epsilon)(\sigma')]$, $F(f_1 \wedge f_2)(\gamma)(\epsilon)(\sigma) = F(f_1)(\gamma)(\epsilon)(\sigma) \wedge F(f_2)(\gamma)(\epsilon)(\sigma)$.
- h. Let $g \equiv \langle D | f_1 \Rightarrow f_2 \rangle$, with $D \equiv \langle P_i \leftrightarrow B_i \rangle_{i=1}^n$. Let $\bar{\gamma} = \gamma(\eta_i / P_i)_{i=1}^n$, with η_i as in part e. $G(g)(\gamma) = [\forall \epsilon \text{ such that } intv(D, f_1) \subseteq \text{dom}(\epsilon), \sigma \neq \perp [F(f_1)(\bar{\gamma})(\epsilon)(\sigma) \Rightarrow \forall \epsilon \text{ such that } intv(D, f_2) \subseteq \text{dom}(\epsilon), \sigma \neq \perp [F(f_2)(\bar{\gamma})(\epsilon)(\sigma)]]]$.

Validity and soundness (first definition, to be modified below).

- a. $\vdash g$ (g is valid) iff $G(g)(\gamma) = tt$ for all $\gamma \in \Gamma$
- b. An inference $\frac{g_1, \dots, g_n}{g}$ is called *sound* whenever $\vdash g_1, \dots, \vdash g_n$ implies $\vdash g$.

Remark. Observe the difference between the validity of $\langle D | f_1 \Rightarrow f_2 \rangle$ and soundness of $\frac{\langle D | f_1 \rangle}{\langle D | f_2 \rangle}$. Putting (i) $\frac{df.}{\langle D | f_2 \rangle} \forall \epsilon \text{ such that } intv(D, f_1) \subseteq \text{dom}(\epsilon), \text{ all } \sigma \neq \perp [F(f_1)(\gamma)(\epsilon)(\sigma)]$, $i = 1, 2$, we have that the former corresponds to $\forall \gamma [(1) \Rightarrow (2)]$, whereas the latter corresponds to the *weaker* fact that $\forall \gamma [(1)] \Rightarrow \forall \gamma [(2)]$.

2.5. *Lemmas.* A number of lemmas stating properties of our various constructs will be used below. First we have a lemma relating substitution to state modification.

LEMMA 2.1.

- a. If $\text{intv}(s, t, v) \subseteq \text{dom}(\epsilon)$ then $R(s[t/v])(\epsilon)(\sigma) = R(s)(\epsilon)(\sigma(R(t)(\epsilon)(\sigma)/L(v)(\epsilon)(\sigma)))$
 b. Similarly for $b \in \text{Bexp}$, $p \in \text{Assn}$.

END 2.1.

Next, we have a useful property of *closed* programs, asserting that such programs only affect the values of variables occurring in them:

- LEMMA 2.2.** If $\langle D|S \rangle$ is closed, $\gamma \in \Gamma$, $\bar{\gamma}$ as usual (cf. 2.4h) then, if (i) $\text{intv}(D, S) \subseteq \text{dom}(\epsilon)$, (ii) $\xi \in \text{dom}(\epsilon) \setminus \text{intv}(D, S)$, (iii) $\sigma' = N(S)(\bar{\gamma})(\epsilon)(\sigma)$, $\sigma' \neq 1$, then, (iv) $\sigma'(\epsilon(\xi)) = \sigma(\epsilon(\xi))$

END 2.2.

The last lemma is rather technical, and foreshadows a property of statements to be discussed in section 3. *Notation.* For $\delta \subseteq \text{Intv}$, $(\sigma \circ \epsilon)|\delta$ denotes the function composed of σ and ϵ restricted to δ .

LEMMA 2.3.

- a. Let $m, n \geq 0$. If (i) $\text{intv}(s) \setminus \{x_i\}_{i=1}^n \setminus \{ \langle a_j, \alpha \rangle_{\alpha \in V_0} \}_{j=1}^m \subseteq \delta \subseteq \text{dom}(\epsilon) \cap \text{dom}(\bar{\epsilon})$, (ii) $(\sigma \circ \epsilon)|\delta = (\bar{\sigma} \circ \bar{\epsilon})|\delta$, (iii) For $i = 1, \dots, n$, either $\sigma(e_i) = \bar{\sigma}(\bar{e}_i)$, or $x_i \notin \text{svar}(s)$, and, for $j = 1, \dots, m$, either, for all $\alpha \in V_0$, $\sigma(e_{\alpha, j}) = \bar{\sigma}(\bar{e}_{\alpha, j})$, or $a_j \notin \text{avar}(s)$, then, (iv) $R(s[y_i/x_i][a_j/a_j])(\epsilon \cup \langle y_i, e_i \rangle_i \cup \langle \langle a_j, \alpha \rangle, e_{\alpha, j} \rangle_{\alpha, j})(\sigma) = R(s[z_i/x_i][a_j/a_j])(\bar{\epsilon} \cup \langle z_i, \bar{e}_i \rangle_i \cup \langle \langle a_j, \alpha \rangle, \bar{e}_{\alpha, j} \rangle_{\alpha, j})(\bar{\sigma})$

- b. Similarly for $b \in \text{Bexp}$ and $p \in \text{Assn}$.

END 2.3.

3. Initialization

The validity definition as given in 2.4 is, though rather natural, not satisfactory for our purposes. First, it implies the validity of formulae such as (*): $\langle \{\text{true}\} \underline{b} \text{ new } x; x:=0 \underline{e}; \underline{b} \text{ new } x; y:=x \underline{e}\{y=0\} \rangle$, or (**): $\langle \{\text{true}\} \underline{b} \text{ new } x; y:=x \underline{e}; \underline{b} \text{ new } x; z:=x \underline{e}\{y=z\} \rangle$. The source of this problem is that our semantics is overspecified in that, when declaring a new local s.v., we want its initial value to be some *arbitrary* integer. Now in def. 2.4d, we take for this the value stored at the first free address and, in a situation such as (*), upon entry of the second block we find, as after-effect of the first block, 0 stored at this address. ((**) can be explained similarly.) A solution to this problem is *either* to change the semantics (ensuring by some flag-mechanism that no address is ever used twice as first free address), which we do not adopt mainly because of severe technical complications, *or* to restrict our correctness considerations to programs in which all local s.v. are initialized. The second solution is the one elaborated below (also motivated by the idea that the correctness of programs containing uninitialized local s.v. is probably not very interesting anyway). A second problem with the validity definition is the following: For reasons to be explained below we have to consider in a formal correctness proof also *non-closed* programs in which case counter examples can be found to the validity of quite natural c.f. such as $\langle D|\{p\}S_1\{q\} \wedge \{q\}S_2\{r\} \Rightarrow \{p\}S_1; S_2\{r\} \rangle$. The second problem is dealt with

in section 4; we now define the notion of initialization and state the main theorem concerning it.

DEFINITION 3.1. (initialized s.v.)

a. The set $init(R)$ of all s.v. initialized in R is the smallest subset of $Svar$ satisfying

- (i) If $x \notin svar(s)$ then $x \in init(<D|x:=s>)$.
- (ii) If $x \in init(<D|S_1>)$, or $x \notin svar(S_1)$ and $x \in init(<D|S_2>)$ then $x \in init(<D|S_1;S_2>)$.
- (iii) If $x \notin svar(b)$, $x \in init(<D|S_i>)$, $i = 1, 2$, then $x \in init(<D|if\ b\ then\ S_1\ else\ S_2\ fi>)$.
- (iv) If $x \neq y$, $x \in init(<D|S>)$ then $x \in init(<D|b\ new\ y; S\ e>)$.
- (v) If $D \equiv \langle P_i \rightarrow B_i \rangle_{i=1}^n$ then, for $i = 1, \dots, n$, if $B_i \equiv \langle val\ x_i, add\ y_i | S_i \rangle$, $x \notin svar(t)$, $x \equiv v$, and $y_i \in init(<D|S_i>)$, then $x \in init(<D|P_i(t,v)>)$.

b. All local s.v. in a program $\langle D|S \rangle$, with $D \equiv \langle P_i \rightarrow B_i \rangle_{i=1}^n$, $B_i \equiv \langle val\ x_i, add\ y_i | S_i \rangle$, are initialized whenever for each statement $b\ new\ x; S_0\ e$ occurring as substatement of S or any of the S_i , $1 \leq i \leq n$, we have that $x \in init(<D|S_0>)$.

END 3.1.

For an initialized local s.v., the value associated with it through def. 2.4d is irrelevant. This is one of the (somewhat hidden) messages of

THEOREM 3.2. Let $\langle D|S \rangle$ be a closed program in which all local s.v. are initialized.

Let $n, m \geq 0$, and let $\gamma, \bar{\gamma}$ be as usual. If

- (i) $intv(D) \cup (intv(S) \setminus \{x_i\}_i \setminus \{a_j, \alpha_j\}) \subseteq \delta \subseteq \text{dom}(\epsilon) \cap \text{dom}(\bar{\epsilon})$
 - (ii) $(\sigma \circ \epsilon) | \delta = (\bar{\sigma} \circ \bar{\epsilon}) | \delta$
 - (iii) For $i = 1, \dots, n$, either $\sigma(e_i) = \bar{\sigma}(\bar{e}_i)$, or $x_i \notin svar(S)$, or $x_i \in init(<D|S>)$.
For $j = 1, \dots, m$, either, for all $\alpha \in V_0$, $\sigma(e_{\alpha,j}) = \bar{\sigma}(\bar{e}_{\alpha,j})$, or $a_j \notin svar(S)$.
 - (iv) $N(S[y_i/x_i]_i [a_j/a_j]_j)(\bar{\gamma})(\epsilon \cup \langle y_i, e_i \rangle_i \cup \langle \langle a_j, \alpha \rangle, e_{\alpha,j} \rangle_j)(\sigma) = \sigma'$
 $N(S[z_i/x_i]_i [a_j/a_j]_j)(\bar{\gamma})(\bar{\epsilon} \cup \langle z_i, \bar{e}_i \rangle_i \cup \langle \langle a_j, \alpha \rangle, \bar{e}_{\alpha,j} \rangle_j)(\bar{\sigma}) = \bar{\sigma}'$
- then
- (v) $(\sigma' \circ \epsilon) | \delta = (\bar{\sigma}' \circ \bar{\epsilon}) | \delta$
 - (vi) For $i = 1, \dots, n$, either $\sigma'(e_i) = \bar{\sigma}'(\bar{e}_i)$, or $x_i \notin svar(S)$.
For $j = 1, \dots, m$, either, for all $\alpha \in V_0$, $\sigma'(e_{\alpha,j}) = \bar{\sigma}'(\bar{e}_{\alpha,j})$, or $a_j \notin svar(S)$.

END 3.2.

In section 4 two special cases of this theorem are of interest, mentioned in

COROLLARY 3.3. Let $\langle D|S \rangle$, $m, n; \gamma, \bar{\gamma} \dots$ be as in theorem 3.2.

- a. If (i) $intv(D, S) \subseteq \delta \subseteq \text{dom}(\epsilon) \cap \text{dom}(\bar{\epsilon})$, (ii) $(\sigma \circ \epsilon) | \delta = (\bar{\sigma} \circ \bar{\epsilon}) | \delta$, (iii) $N(S)(\bar{\gamma})(\epsilon)(\sigma) = \sigma'$, $N(S)(\bar{\gamma})(\bar{\epsilon})(\bar{\sigma}) = \bar{\sigma}'$, then (iv) $(\sigma' \circ \epsilon) | \delta = (\bar{\sigma}' \circ \bar{\epsilon}) | \delta$.
- b. If (i) $intv(D) \cup (intv(S) \setminus \{x_i\}_i \setminus \{a_j, \alpha_j\}) \subseteq \text{dom}(\epsilon)$, (ii) $(S[y_i/x_i]_i [a_j/a_j]_j)(\bar{\gamma})(\epsilon \cup \langle y_i, e_i \rangle_i \cup \langle \langle a_j, \alpha \rangle, e_{\alpha,j} \rangle_j)(\sigma) = \sigma'$, $N(S[z_i/x_i]_i [a_j/a_j]_j)(\bar{\gamma})(\bar{\epsilon} \cup \langle z_i, \bar{e}_i \rangle_i \cup \langle \langle a_j, \alpha \rangle, \bar{e}_{\alpha,j} \rangle_j)(\bar{\sigma}) = \bar{\sigma}'$, then (iii) $(\sigma' \circ \epsilon) | \text{dom}(\epsilon) = (\bar{\sigma}' \circ \bar{\epsilon}) | \text{dom}(\bar{\epsilon})$.

END 3.3.

Remark. Let us call a pair $\langle \sigma, \varepsilon \rangle$, $\langle \bar{\sigma}, \bar{\varepsilon} \rangle$ *matching* with respect to δ if it satisfies condition (ii) of part a. We see that a program satisfying the indicated requirements preserves the property of matching. Cor. 3.3b tells us that substituting different fresh s.v. y, z (since $y, z \notin \text{dom}(\varepsilon)$, $y, z \notin \text{intv}(D) \cup (\text{intv}(S) \setminus \dots)$) for some x makes no (essential) difference in the outcome, provided that they are associated with the same address.

4. A sound proof system

The following proof system will be considered:

A. Rules about " \Rightarrow ".

1. $\langle D \mid f \Rightarrow \text{true} \rangle$ (strengthening)
2. $\frac{\langle D \mid f_1 \Rightarrow f_2 \rangle}{\langle D \mid f_1 \wedge f_3 \Rightarrow f_2 \rangle}$ (weakening)
3. $\frac{\langle D \mid f_1 \Rightarrow f_2 \rangle, \langle D \mid f_2 \Rightarrow f_3 \rangle}{\langle D \mid f_1 \Rightarrow f_3 \rangle}$ (transitivity)
4. $\frac{\langle D \mid f \Rightarrow f_1 \rangle, \langle D \mid f \Rightarrow f_2 \rangle}{\langle D \mid f \Rightarrow f_2 \wedge f_1 \rangle}$ (collection)
5. $\langle D \mid f_1 \wedge \dots \wedge f_n \Rightarrow f_i \rangle, n \geq 1, 1 \leq i \leq n$ (selection)

B. Rules about programming concepts.

6. $\langle D \mid \{p[t/v]\}; v := t \{p\} \rangle$ (assignment)
7. $\langle D \mid \{p\} S_1 \{q\} \wedge \{q\} S_2 \{r\} \Rightarrow \{p\} S_1 ; S_2 \{r\} \rangle$ (composition)
8. $\langle D \mid \{p \wedge b\} S_1 \{q\} \wedge \{p \wedge \neg b\} S_2 \{q\} \Rightarrow \{p\} \text{ if } \dots \text{ fi } \{q\} \rangle$ (conditional)
9. $\langle D \mid \{p\} S[y/x] \{q\} \Rightarrow \{p\} \underline{b} \text{ new } x; S \underline{e} \{q\} \rangle$ (s.v. declaration)
provided that $y \notin \text{sva}r(D, p, S, q)$

10. Let Ω be a procedure constant such that $N(\Omega) = \lambda\gamma \cdot \lambda t \cdot \lambda v \cdot \lambda e \cdot \lambda \sigma \cdot 1$.

$$\frac{\langle \langle P_i \Rightarrow B_i \rangle_i \mid f[\Omega/Q_i]_i \rangle, \langle \langle P_i \Rightarrow B_i \rangle_i \mid f \Rightarrow f[B'_i/Q_i]_i \rangle}{\langle \langle P_i \Rightarrow B_i \rangle_i \mid f[P_i/Q_i]_i \rangle} \quad (\text{induction})$$

where $Q_i \notin \text{pva}r(P_1, \dots, P_n, B_1, \dots, B_n)$, and $B'_i \equiv B_i[Q_j/P_j]_j$,
 $i = 1, \dots, n$.

C. Auxiliary rules

11. $\langle D \mid (p \supset p_1) \wedge \{p_1\} S \{q_1\} \wedge (q_1 \supset q) \Rightarrow \{p\} S \{q\} \rangle$ (consequence)