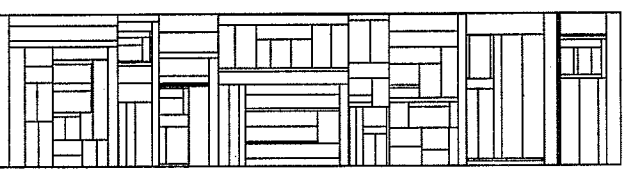
AN INVESTIGATION INTO DIFFERENT SEMANTIC APPROACHES

by

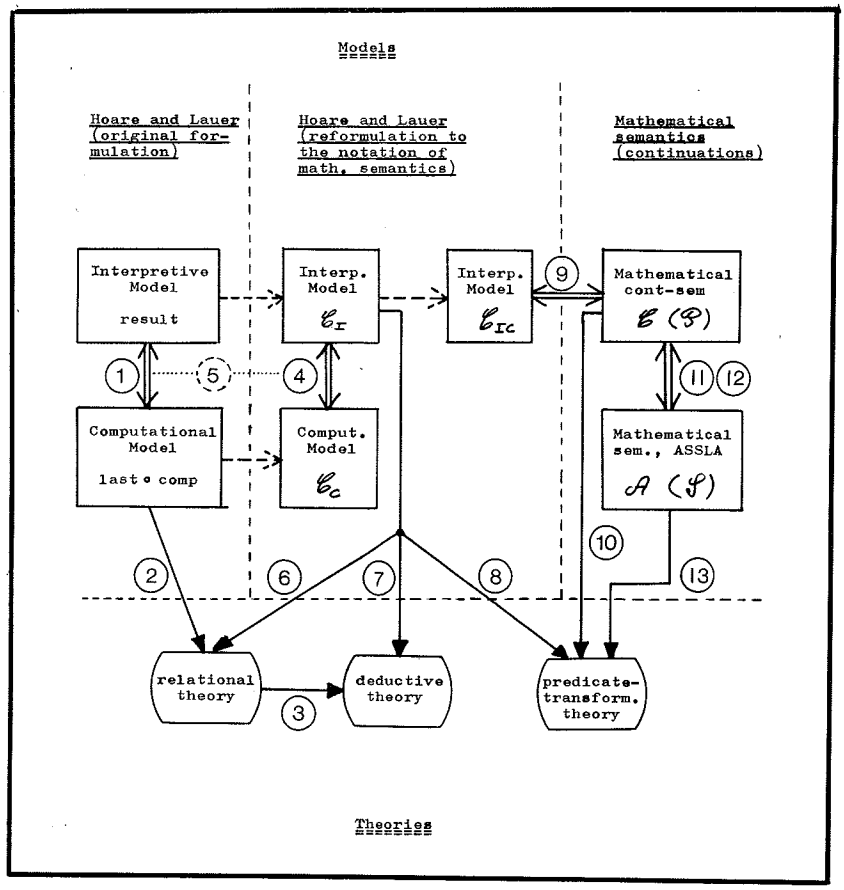
Kurt Jensen

DAIMI PB-61
June 1976

Institute of Mathematics University of Aarhus
DEPARTMENT OF COMPUTER SCIENCE
Ny Munkegade - 8000 Aarhus C - Denmark
Phone 06-12 83 55



AN INVESTIGATION INTO DIFFERENT SEMANTIC APPROACHES



contents:

acknowledgement

I want to express the warmest thanks to my adviser BRIAN MAYOH who did an excellent work in providing new ideas and inspiration for this paper.

0 INTRODUCTION

In this paper I shall investigate different semantic approaches. I shall use them all on an example language called SNAIL. This language is small enough to make such an investigation possible. On the other hand SNAIL is strong enough to represent most problems met in pratical programming languages. Chapter 1 is an informal description of SNAIL. This paper makes very heavy use of the mathematical semantic as developed by Strachey, Scott and others at Oxford University. A short review of this description-method is given in chapter 2. Chapter 3 gives a mathematical semantics for SNAIL.

Chapter 4 describes some ideas given by Hoare and Lauer at IBM laboratory Vienna. Two different models and two different theories are defined for SNAIL. The models are proved equivalent - and they are proved to satisfy the theories.

Chapter 5 reformulates the result and proposals from chapter 4. The notation is now following the line of the Oxford-approach. This gives a more straightforward and convenient notation. Moreover chapter 5 introduces a third theory - predicate-transformer theory proposed by Dijkstra in Eindhoven.At last it is shown that the addition of command-continuations eases the description - especially for this last theory.

The connection between continuation-semantics and Dijkstra's predicate-transformer theory are further developed in chapter 6.

Chapter 7 describes a method for proving compiler correctness using mathematical semantics. A very simple example language is used.

Chapter 8 defines the mathematical semantics for an assembly language specially designed to fit as target-language for a SNAIL-compiler. Chapter 9 defines such a compilation and in chapter 10 this compilation is proved correct in a certain specified sense.

The existence of the compilation from SNAIL to ASSLA opens a new possibility for interpreting Dijkstra's theory since this now can be done via the compilation. This approach is described and verified in chapter 11.

Chapter 12 is a conclusion of the derived results.

Appendix A describes the connection between sideeffects, infinite loops and abnormal termination on the one side and two nondeterministic commandtypes, proposed by Dijkstra and enclosed in SNAIL in a deterministic version, on the other side.
Appendix B is merely a collection of the given definitions.
Appendix C draws a connection between Dijkstra's guarded IF-command and an unguarded IF-command proposed by Scott.

It should be mentioned, that to keep this paper on a reasonable lenght, I have not included in considerations the recent letters between Robert Milne and Tony Hoare about predicate-transformers as a tool in semantic descriptions. Nore have I included any comments about the work done at Copenhagen University by Dines Bjørner and others. I have tried to include the formalism suggested by de Bakker in /B/, but without much progress. It looks as this formalism transformed to the notation of mathematical semantics gives very little which is not already provided by the deductive theory. Moreover the axioms for de Bakkers theory becomes more clumsy and inelegant than the corresponding Hoare-axioms.

Dijkstra' s new book /D4/ became avaibale so late that it had very little influence on the work in this paper. It should be stressed that /D4/ contains a much more profound discussion of the weakest predicate-transformer theory and the nondeterministic IF- and DO-commands.

1 A LANGUAGE CALLED SNAIL - INFORMAL DESCRIPTION.

In this paper I shall always use the same example-language called SNAIL (simple nontrivial algorithmic language).

SNAIL will incorporate many different commandclasses, but inside each such class only one or two representatives will be dealt with. For example SNAIL includes WHILE-commands, but not UNTIL-, REPEAT-, REPEATWHILE-, or REPEATUNTIL-commands (speaking in BCPL-terminology).

The commandclasses incorporated in SNAIL are:

1. Assignment.
   Simple assignment, where left hand side is an identifier and right hand side an arbitrary expression.

2. Deterministic conditional.
   TEST (IF-THEN-ELSE). Neither of the conditional commands can be omitted, but both may be the empty string, e. If the expression neither evaluates to true or false the entire command is undefined (program abortion).

3. Nondeterministic conditional.
   IF-command. This is described in Dijkstra /D1/ and /D2/. An IF-command consists of one or more guarded commands (enclosed by IF and FI and separated by #). Each guarded command consists of a guard (boolean expression) followed by -> and a command. The idea is now to do a nondeterministic choice between those commands whose guards evaluate to true. Only one command is executed. If all guards evaluate to false the IF-command yields program abortion.

   In a deterministic language as SNAIL some rules most be imposed on the selection between guards evaluating to true. For a discussion of the most obvious selection rules please see appendix A. In the rest of this paper I shall use the following rule:

The guards will be evaluated in order of appearence until the first true guard is found. Then evaluation of guards stops and the command corresponding to this guard is executed.

Please note that the TEST-command mentioned above can be regarded as syntactic suggaring since it is easy to verify

$$\text{TEST } \varepsilon \text{ DO } \gamma_1 \text{ OR } \gamma_2 \text{ TSET} \equiv \text{IF } \varepsilon \rightarrow \gamma_1 \# \neg\varepsilon \rightarrow \gamma_2 \text{ FI}$$

where ≡ denotes program equivalence.
It is here essential that evaluation of ε does not have sideeffects.

4. Deterministic loop.
WHILE-command.

5. Nondeterministic loop.
DO-command. This is described in Dijkstra /D1/ and /D2/.
It is the repetitive counterpart of the IF-command.
The syntax is identical to that used for the IF-command except that the enclosing brackets now are DO and OD.
The semantic meaning of this command is to do a repetitive execution of the corresponding IF-command until a situation is reached where all guards evaluate to false. Then control is passed to the next command in the program. Thus the DO-command can be viewed as a conjunction of the ideas behind the IF- and WHILE-command.

In the same manner as TEST was as special case of IF, we have that WHILE is a special case of DO:

$$\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD} \equiv \text{DO } \varepsilon \rightarrow \gamma \text{ OD}$$

but this time the equivalence also holds if evaluation of ε has sideeffects.

6. Blocks.

These are used to declare identifiers. Since SNAIL follows BCPL in having only one single type (bitpattern), the main purpose of a declaration is to define the scope for the identifier in question. The scope rules are as for ALGOL-60:

> An identifiers scope is the block in which it is defined and all inner blocks, which does not declare an identifier of the same name.

Besides defining the scope a declaration initializes the identifier, but this is only simple syntactic suggaring: Let BEGIN DEC $\xi;\gamma$ END be semantic equivalent to BEGIN DEF $\xi:=\varepsilon;\gamma$ END except that the former does not initialize ξ .

Then we will have

$$\begin{aligned} &\text{BEGIN DEF } \xi=\varepsilon;\gamma \text{ END} \equiv \\ \equiv\ &\xi':=\varepsilon \quad \text{BEGIN DEC } \xi;\xi:=\xi';\gamma \text{ END} \end{aligned}$$

where ξ' is a new identifier (declared in a block sourrounding this construction and used no other places).

If ε does not involve ξ we simply have

$$\begin{aligned} &\text{BEGIN DEF } \xi=\varepsilon;\gamma \text{ END} \\ \equiv\ &\text{BEGIN DEC } \xi;\xi:=\varepsilon;\gamma \text{ END} \quad . \end{aligned}$$

Each block has exactly one declaration (which may be empty). This is done to avoid using too much effort on problems of technical nature such as indexing etc. It is no severe restriction since a command of the type

$$\text{BEGIN DEF } \xi_1=\varepsilon_1,\ldots\ldots,\xi_n=\varepsilon_n;\gamma \text{ END}$$

(equipped with a semantic which is a straightforward generalisation of the semantic for BEGIN DEF $\xi=\varepsilon;\gamma$ END) can be shown equivalent to

```
ξ1' := ε1;
   .
   .
ξn' := εn;
BEGIN
   DEF ξ1 = ξ1';
   BEGIN
      DEF ξ2 = ξ2';
         .
         .
                  BEGIN
                  DEF ξn = ξn';
                  .
                  END γ
         .
         .
      END
   END
END
```

or, in the case where $\varepsilon_1, \ldots \varepsilon_n$ does not involve the identifiers $\xi_1, \ldots \xi_n$.

```
BEGIN
   DEF ξ1 = ε1;
   BEGIN
      DEF ξ2 = ε2;
         .
         .
            BEGIN
               DEF ξn = εn;
               .
            END γ
         .
         .
      END
   END
END
```

To avoid confusion the reader should notice that only BEGIN DEF ξ:=ε; γ END (and not BEGIN DEC ξ; γ END) is incorporated into SNAIL. This guarantees that all defined identifiers actually have an assigned value at all times.

7. Input-output.
   READ- and WRITE-commands. Simple sequential input and output. These commands has many properties which resembles those of assignment. This commandclass is incorporated in SNAIL to give it a smooth , gentle surface. The presence of these commands allows us to define the semantic meaning og a SNAIL-program as a function defined on inputs and with values in the domain of outputs.

Not included in SNAIL are commandclasses as:

1. Procedures.
   For non recursive procedures this is only syntactic suggaring - but a very convenient help for fast and understandable programming.

2. Transfer of control.
   GOTO-, RETURN-, BREAK-, ENDCASE-, RESULTIS-commands (nor is VALOF-expressions included in SNAIL).

An <u>expression</u> in SNAIL will be either

1 a constant
2 an identifier
3 $\neg\mathcal{E}$ where $\mathcal{E}$ allready is a SNAIL-expression
4 $\mathcal{E}_1 \oplus \mathcal{E}_2$ where $\mathcal{E}_1$ and $\mathcal{E}_2$ allready are SNAIL-expressions and $\oplus$ is either one of the four arithmetic operators $(+,-,*,/)$ or one of the dyadic operators from propositional logic $(\wedge, \vee, \supset, \equiv)$.

Since SNAIL only has one type there is no destinction between boolean and arithmetic expressions. This allows us to use such cumbersome expressions as:

$$(1 \wedge 3) + TT$$

(provided 1, 3 and TT are constants).
In the rest of this paper TT and FF will denote TRUE and FALSE respectively.

In most programming languages evaluation of expressions involve sideeffects. This is for instance the situation when procedures, functions or input-output routines are incorporated in expressions. By this reason this paper to maintain generality does not use that SNAIL-expressions are evaluated without sideeffects, unless explicitely mentioned.

Evaluation of dyadic operators follow the weak truth-tables, which yields an undefined result whenever at least one of the operands are undefined. For the arithmetic operators this is the only reasonable possibility. This is not the case for the propositional operators.

$\wedge$ (and) has at least 3 possibilities :

	T	F	⊥
T	T	F	⊥
F	F	F	⊥
⊥	⊥	⊥	⊥

SNAIL

	T	F	⊥
T	T	F	⊥
F	F	F	F
⊥	⊥	⊥	⊥

LISP

	T	F	⊥
T	T	F	⊥
F	F	F	F
⊥	⊥	F	⊥

most powerfull, but difficult to implement

where the leftmost coloumn indicate the value of the leftside operand and the upper row indicate the value of the rightside operand - $\perp$ denotes undefined.

In fact SNAIL is a whole class of languages since the value domains V and L representing storable values and locations as well as the function $\mathcal{K}$ assigning values to constants are left undefined together with the syntactic domains of identifiers and constants. This part of a semantic description is normally machine dependent.

In many of the proofs in this paper I shall restrict myself to a subset of SNAIL called MINI-SNAIL. This contains:

assignment
IF-command
WHILE-command
declaration (not empty)

This subset represents in most aspects the whole language. Besides the empty command and the empty declaration 4 different commands are omitted:

TEST-command ,but this is a special case of IF
DO-command, but this gives no problems which is not already given by WHILE or IF. (see appendix C page 154).
READ and WRITE, but in most questions they behave exactly as assignment.

2 MATHEMATICAL SEMANTIC - STRUCTURE AND CONVENTIONS

Whenever giving a mathematical semantic for a language I shall use the notation developed in Oxford by Strachey , Scott and others.It will be assumed that the reader is familiar to this type of notation as described in the Oxford Technical Monographs /S/, /S1/, /S2/, /S4/ and /M3/.

I shall always follow Ligler /L2/ in dividing such a description into 6 different parts:

1. Syntactic domains
   The detailed nature of these domains can be derived from the syntax-part (paragraph 3) but this part establish the mnemonics used. There is 2 kinds of syntactic domains. Elementary domains such as identifiers and constants. Most times these will not be detailed specified in a general approach as this, since they constitute a more machine dependent part. All other syntactic domains are build from these elementary ones by using the domain operators $+$, $\times$, $*$ and $\rightarrow$.

2. Value domains
   These are all other domains used in the semantic equations. Again some of these will be considered elementary and all others will be build from these and from the syntactic domains by using the same domain operators as above. As examples of elementary domains can be mentioned locations and storable values (which clearly are machine dependent).

3. Syntax
   BNF-like description, where nonterminals are indicated by greek letters as introduced in paragraph 1.

4. Semantic functions
   Describes the names and functionality of the used functions.

5. Semantic equations

This is the most important part of the semantic description. All other paragraphs can be seen as auxilliary.
I shall (briefly) descibe some of the conventions used:

a) programtext are enclosed in $[\![\ ]\!]$ -parentheses
b) application of functions associates from left to right: f g a = (f(g))(a)
c) continuations will often be enclosed in $\{\}$ -parentheses
d) lambda-notation is used
e) $\varsigma[\alpha/\xi]$ denotes the enviroment identical to ς except that the identifier ξ now is bound to location α.
f) FIX $\theta.\ = Y(\lambda\theta.$ where Y is the least-fixpoint operator.
g) $\oplus$ is used in two different ways. It denotes elements in the syntactic domain DOP, but is also used as an operator in the metalanguage.
h) $\langle a_1, \ldots, a_n \rangle$ denotes tuples

$$\langle a_1, \ldots, a_n \rangle \downarrow i = a_i \qquad 1 \le i \le n$$

$$\langle a_1, \ldots, a_n \rangle \uparrow i = \langle a_i, a_{i+1}, \ldots, a_n \rangle \qquad 1 \le i \le n$$

i) "|" denotes domain-projection.
j) At the other hand "in" should be used to lift an element from one domain to another containing the first as an addend. Since this does not give the reader any usuable information it will be omitted.
For a detailed description of domain structures and the suitable operators see Peter Mosses /M4/.

6. Auxilliary functions.

In this paragraph is described functions, which perform elementary operations such as updating store, input-output and so on. These functions can so to speak be appended to the total semantic description at run-time. One could say that they constitute the connection between a machine independent description (paragraph 1-5) and the actual physical machine. By drawing these functions outside the semantic equations machine independence can be maintained as long as possible.

3 MATHEMATICAL SEMANTIC FOR SNAIL

In this section I shall give an exact and unambigious description of SNAIL using the notation described in the last section.

1. Syntactic domains

$\gamma \in$	PROG	programs
$\gamma \in$	COM	commands
$\delta \in$	DEC	declarations
$\omega \in$	GBLOCK	guarded blocks
$\varepsilon \in$	EXP	expressions
$\xi \in$	ID	identifiers
$k \in$	CONST	constants
$\oplus \in$	DOP	dyadic operators

2. Value domains

$\psi \in R = S \times In \times Out$	configurations
$\sigma \in S = [L \to V]$	states
$\mu_I \in In = V^*$	inputs
$\mu_O \in Out = V^*$	outputs
$\theta \in C = [R \to A]$	command continuations
$\kappa \in K = [V \to C]$	expression cont.
$\chi \in X = [ENV \to C]$	declaration cont.
$\eta \in G = [W^* \to C]$	guardedblock cont.
$\varsigma \in Y = [L \to C]$	location cont.
$\beta \in V$	values
$\alpha \in L$	locations
$\rho \in ENV = [Id \to D]$	enviroments
$D = L + C$	denotations

A	answers
$r \in H = [K \to C]$	guards
$i \in I = [C \to C]$	guarded lines
$W = H \times I$	guard components

where a configuration $\psi = \langle z, \mu_I, \mu_O \rangle$ describes

- z – current state (store)
- μ_I – remaining input
- μ_O – output until now

3. Syntax

$$\gamma ::= \gamma_1 ; \gamma_2 \mid \xi := \varepsilon \mid$$

TEST ε DO γ_1 OR γ_2 TSET |
IF ω FI | DO ω OD |
BEGIN δ; γ END |
WHILE ε DO γ OD
READ (ξ) | WRITE (ε) | e

$$\delta ::= \text{DEF } \xi = \varepsilon \mid e$$

$$\omega ::= \varepsilon_1 \to \gamma_1 \# \cdots\cdots \# \varepsilon_n \to \gamma_n \qquad n \geq 1$$

$$\varepsilon ::= \text{TRUE} \mid \text{FALSE} \mid \xi \mid k \mid \neg \varepsilon \mid \varepsilon_1 \oplus \varepsilon_2$$

$$\oplus ::= + / - \mid * \mid / \mid \wedge \mid \vee \mid \supset \mid \equiv$$

4. Semantic functions

$\mathcal{P}$: PROG → IN → OUT

$\mathcal{C}$: COM → ENV → C → C

$\mathcal{D}$: DEC → ENV → X → C

$\mathcal{E}$: EXP → ENV → K → C

$\mathcal{G}$: GLOCK → ENV → G → C

$\mathcal{K}$: CONST → V

5. Semantic equations

$$\mathcal{P}[\![\gamma]\!]\,\kappa_I = (\mathcal{C}[\![\gamma]\!]\,\varsigma^*_{IN}\,\theta^*_{IN}\,\psi_{IN})\downarrow 3$$

$$\varsigma^*_{IN} = \lambda\xi.\,\bot \qquad \psi_{IN} = \langle z_{IN}, \kappa_I, \kappa_{O_{IN}}\rangle$$

$$\theta^*_{IN} = \lambda\psi.\,\psi \qquad z_{IN} = \lambda\alpha.\,\bot$$

$$\kappa_{O_{IN}} = NIL$$

The initial situation can be characterized by

1. Enviroment where all identifiers are undefined.
2. State, where the contents of all locations are undefined.
3. Empty outputfile.
4. Identityfunction as continuation. This shows that a SNAIL-program is considered to be a closed unit, after which nothing remains to be done. An alternative view would be to consider programcontinuations, processcontinuations or even operating system continuations.
   For a discussion of this see /S4/ p. 9.

It is possible to give a more elegant definition of $\mathcal{P}$.
We can use the initial command-continuation θ_{IN} to express that we only are interested in the third component of the final configuration. We would then have

$$\mathcal{P}[\![\gamma]\!]\kappa_I = \mathcal{C}[\![\gamma]\!]\sigma^*_{IN}\theta^*_{IN}\psi_{IN}$$
$$\theta^*_{IN} = \lambda\psi.\ (\psi\downarrow 3)$$

where all other symbols are as before.

Later on I shall give a proof for correctness of a SNAIL-compiler. In this proof it is convenient to consider the whole configuration and not only its third component.

$$\mathcal{E}[\![\gamma_1;\gamma_2]\!]\varsigma\theta = \mathcal{E}[\![\gamma_1]\!]\varsigma\,\{\,\mathcal{E}[\![\gamma_2]\!]\varsigma\theta\,\}$$

$$\mathcal{E}[\![\xi := \varepsilon]\!]\varsigma\theta = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\,\lambda\beta.\ Update\ (\varsigma[\![\xi]\!]L, \beta)\theta\,\}$$

$$\mathcal{E}[\![\text{TEST } \varepsilon \text{ DO } \gamma_1 \text{ OR } \gamma_2 \text{ TSET}]\!]\varsigma\theta = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\,Cond\ (\mathcal{E}[\![\gamma_1]\!], \mathcal{E}[\![\gamma_2]\!])\varsigma\theta\,\}$$

$$\mathcal{E}[\![\text{IF } \omega \text{ FI}]\!]\varsigma\theta = \mathcal{G}[\![\omega]\!]\varsigma\,\{\,Oracle\ (\theta, \bot)\,\}$$

$$\mathcal{E}[\![\text{DO } \omega \text{ OD}]\!]\varsigma\theta = \text{FIX}\ \ \theta'.\ \mathcal{G}[\![\omega]\!]\varsigma\,\{\,Oracle\ (\theta', \theta)\,\}\,)$$

$$\mathcal{E}[\![\text{BEGIN } \delta;\gamma \text{ END}]\!]\varsigma\theta = \mathcal{D}[\![\delta]\!]\varsigma\,\{\,\lambda\varsigma.\ \mathcal{E}[\![\gamma]\!]\varsigma\theta\,\}$$

$$\mathcal{E}[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\varsigma\theta = \text{FIX}\ \ \theta'.\ \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\,Cond\ (\mathcal{E}[\![\gamma]\!]\varsigma\theta', \theta)\,\}$$

$$\mathcal{E}[\![\text{READ } (\xi)]\!]\varsigma\theta = Read\ (\varsigma[\![\xi]\!]L)\,\theta$$

$$\mathcal{E}[\![\text{WRITE } (\varepsilon)]\!]\varsigma\theta = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\,\lambda\beta.\ Write\ (\beta)\theta\,\}$$

$$\mathcal{E}[\![e]\!]\varsigma\theta = \theta$$

$$\mathcal{E}[\![\text{TRUE}]\!]\varsigma\kappa = \kappa\,(TT)$$

$$\mathcal{E}[\![\text{FALSE}]\!]\varsigma\kappa = \kappa\,(FF)$$

$$\mathcal{E}[\![\xi]\!]\varsigma\kappa = \lambda\psi.\ \kappa\,(Contents\ (\varsigma[\![\xi]\!]L)\,(\psi\downarrow 1))$$

$$\mathcal{E}[\![k]\!]\varsigma\kappa = \kappa\,(\mathcal{K}[\![k]\!])$$

$$\mathcal{E}[\![\neg\varepsilon]\!]\varsigma\kappa = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\,\lambda\beta.\ \kappa(\neg\beta)\,\}$$

$$\mathcal{E}[\![\varepsilon_1 \oplus \varepsilon_2]\!]\varsigma\kappa = \mathcal{E}[\![\varepsilon_1]\!]\varsigma\,\{\,\lambda\beta_1.\ \mathcal{E}[\![\varepsilon_2]\!]\varsigma\,\{\,\lambda\beta_2.\ \kappa(\beta_1 \oplus \beta_2)\,\}\,\}$$

$$\mathcal{D}[\![\text{DEF } \xi = \varepsilon]\!]\varsigma\chi = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\,\lambda\beta.\ New(\varsigma)\,\{\,\lambda\alpha.\ Update\ (\alpha,\beta)\,\{\,\chi(\varsigma[\alpha/\xi])\,\}\,\}\,\}$$

$$\mathcal{D}[\![e]\!]\varsigma\chi = \chi(\varsigma)$$

$$\mathcal{G}[\![\varepsilon_1 \to \gamma_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to \gamma_n]\!]\varsigma\eta$$
$$= \eta\langle\langle \mathcal{E}[\![\varepsilon_1]\!]\varsigma, \mathcal{E}[\![\gamma_1]\!]\varsigma\rangle, \ldots\ldots, \langle \mathcal{E}[\![\varepsilon_n]\!]\varsigma, \mathcal{E}[\![\gamma_n]\!]\varsigma\rangle\rangle$$

6. Auxilliary functions

Update: $L \times V \to C \to C$

$Update\,(\alpha,\beta)\,\theta = \lambda\langle z, \kappa_I, \kappa_o\rangle .\ \theta\langle Assign\,(\alpha,\beta)\,z, \kappa_I, \kappa_o\rangle)$

Assign: $L \times V \to S \to S$

$z' = Assign\,(\alpha,\beta)\,z$

$\Updownarrow$

$$z'(\alpha') = \begin{cases} z(\alpha') & \text{for } \alpha' \neq \alpha \\ \beta & \text{for } \alpha' = \alpha \end{cases}$$

if α or β is equal to $\perp$ assign yields $\perp$.

Contents: $L \to S \to V$

$Contents\,(\alpha)(z) = z(\alpha)$

Cond: $F \times F \to T \times F$

where F is any domain

$$Cond\,(f_1, f_2)(t) = t \to f_1, f_2 = \begin{cases} f_1 & \text{for } t = TT \\ f_2 & \text{for } t = FF \\ \perp & \text{else} \end{cases}$$

New: $ENV \to Y \to C$

$New\,(\varsigma)(\varsigma) = \varsigma(\alpha)$

where α is a new location not bound to any identifier by ς.

Read: $L \to C \to C$

$$Read(\alpha)\,\Theta = \Theta \circ Rd(\alpha)$$

Rd: $L \to R \to R$

$$\langle z', \mu_I', \mu_O' \rangle = Rd(\alpha)\,\langle z, \mu_I, \mu_O \rangle$$

$$\Updownarrow$$

$$\left[\begin{array}{l} dim(\mu_I) \geq 1 \\ z' = Assign(\alpha, (\mu_I \downarrow 1))\, z \\ \mu_I' = \mu_I + 2 \\ \mu_O' = \mu_O \end{array}\right] \vee \left[\begin{array}{l} dim(\mu_I) = 0 \\ z' = Readerror \\ \mu_I' = \mu_I \\ \mu_O' = \mu_O \end{array}\right]$$

where it moreover is assumed that all command continuations are strict on Readerror.

Write: $V \to C \to C$

$$Write(\beta)\,\Theta = \Theta \circ Wr(\beta)$$

Wr: $V \to R \to R$

$$\langle z', \mu_I', \mu_O' \rangle = Wr(\beta)\,\langle z, \mu_I, \mu_O \rangle$$

$$\Updownarrow$$

$$\left[\begin{array}{l} z' = z \\ \mu_I' = \mu_I \\ \mu_O' = \mu_O \,@\, (\beta) \end{array}\right]$$

where @ is the append operator for lists.

Oracle: $C \times C \rightarrow W^* \rightarrow C$

$$Oracle\ (\theta_1, \theta_2)\ (\langle\langle \nu_1, i_1 \rangle, \ldots, \langle \nu_n, i_n \rangle\rangle)\ \psi$$

$$= \begin{cases} i_j\ (\theta_1)\ \psi & \text{where j is the least number so that } \nu_j\ (\lambda\beta.\ \lambda\psi.\ \beta)\ \psi = TT \\ \theta_2\ \psi & \text{if such a j does not exists} \end{cases}$$

The auxilliary functions are implicit defined. By this I mean, that I have specified some properties which these functions should posses, but not proposed how this actual should be implemented. This is reasonable since an implementation nescessarely must depend on the actual target machine.

Nevertheless most of these auxilliary functions have straight-forward implementations. For instance on most machines it would be quite natural to consider the domain L as a contigeous block of store equipped by some pointer to indicate the first free cell. Then New would simply return the specified location continuation applied to the value of this pointer (and increment the pointer by 1).

4 HOARE AND LAUER - 4 DIFFERENT SEMANTICS

In /H1/ Hoare and Lauer describes 4 different ways of giving semantic definitions for a language. These four descriptions range from the extreme constructive approach, where a language is described step by step by its effect on a particular (abstract) machine to the extreme implicit approach, where a language is characterized by connecting logical deduction rules to each command type. Lauer and Hoare argues that the different kinds of computer scientists needs different semantic approaches. The implementator of a compilator wants a constructive approach, while the user wanting to prove his program correct needs an implicit approach, where he is not bored by a lot of more or less irrelevant machine-dependent details. By this reason Lauer and Hoare suggests that a language designer should give (at least) 2 different semantic descriptions for his language - and prove them equivalent.

In /H1/ the 4 proposed semantic approaches are used on a very simple language giving a clear and easy understandable picture of their connections. In /L/ Lauer uses the same 4 approaches on a much richer language (same complexity as SNAIL). Now the notation becomes a bit more unapprochable, but still /L/ shows that the 4 approaches are possible for practical programming languages.

In this section I shall use the 4 proposed semantic definition methods on SNAIL. In many of the proofs I shall restrict the argumentation to MINI-SNAIL (see page 12).

To ease later comparison with mathematical semantics I shall use a notation differing a little from Hoare and Lauers.

There is especially one point which may cause some confusion - the binding of identifiers (declarations). In mathematical semantics the standard treatment is to define an enviroment. This is a function from the syntactic domain of identifiers to the value domain of locations and tells which locations are bound to which identifiers. Hoare and Lauer follow

another idea. They speak of the domain of unique identifiers, where uniqueness is maintained by text substitutions when nescessarily. Such an approach is in the terminology of mathematical semantics equivalent to discarding the notion of enviroment and identifying the domains of unique identifiers and locations ($L = ID$, $\xi \in L$).

There is no conceptual difference between the notations used here and the notation used in /L/ and /H1/. The main differences lay in such trivial things as choice of names, order of parameters and so on.

I shall now give a very brief description of the used notation. For a more strict and profound definition of this please see /L/ or /H1/.

1. The CASES-notation is due to Burstall and descriminates on the syntactic structure of the object under consideration.

   e.g. CASES γ IN
   first altern. → first action
   ⋮ ⋮
   last altern. → last action
   ENDCASE

   (Hoare and Lauer omits "IN" and "ENDCASE").

2. $[b_1 \rightarrow A_1, \ \vdots \ , b_n \rightarrow A_n]$

   yields A_i , where i is the minimal number so that b_i = true. If no such i exists the value of this construction is $\perp$.

3. The domain structure and the auxilliary functions used will unless explicitely mentioned be exactly as in the mathematical semantics for SNAIL.

4. New_I : $R \to L$ returns an unused location.
(Hoare and Lauer calls this function "un").

5. $augment_I$: $R \times L \to R$
Since we have no enviroment the state must be viewed as a partial function with domain equal to the used identifiers. $augment_I(\psi, \xi)$ appends a new identifier ξ to the domain of ψ.

6. $\mathcal{E}_I[\![\varepsilon]\!]\psi$ evaluates the expression ε in the configuration ψ. It is supposed in this section, that such an evaluation is done without sideeffects.
A formal definition of $\mathcal{E}_I[\![\varepsilon]\!]\psi$ will be left to the reader. It follows exactly the same lines as the definition of $\mathcal{E}$ on page 19. Since identifiers and locations are identified giving an enviroment equal to the identity function we have in particular:

$$\mathcal{E}_I[\![\xi]\!]\psi = contents(\xi)(\psi \downarrow 1) \quad .$$

I shall often use ε_ψ as an abbreviation of $\mathcal{E}_I[\![\varepsilon]\!]\psi$.
(Hoare and Lauer use $val(\psi, \varepsilon)$).

7. $assign_I$: $L \times V \to R \to R$ is defined in terms of assign (see page 20):

$$assign_I(\alpha, \beta)\langle z, \kappa_I, \kappa_0 \rangle = \langle assign(\alpha, \beta) z, \kappa_I, \kappa_0 \rangle$$

8. Rd_I : $L \to R \to R$
Wr_I : $V \to R \to R$
is exactly as Rd and Wr on page 21.

9. tail is a list function which yields its argument with initial item removed, and is undefined for empty argument.

10. last is a list function yielding the last item of a non-empty list. It is undefined for empty argument.

4.1 Interpretive Model

This represents the extreme constructive approach. It describes exactly how a given (abstract) machine would execute a given program step by step. The method is due to the IBM Laboratory Vienna.

A machine state of this abstract machine is a pair $\langle \psi, \gamma \rangle \in R \times COM$ As usual ψ denotes the current configuration (we shall often use the word state since we are not very concerned about input and output). γ is called the control state and contains those commands remaining to be executed.

The heart of this semantic approach is definition of a function

$$next : R \times COM \rightarrow R \times COM$$

Intuitively one single application of next corresponce to a single step in the execution (altering of machine state).

Having this explanation in mind it cannot be a surprise that the result of interpreting a program γ starting in configuration ψ will be defined as result(ψ, γ), where

$$result : R \times COM \rightarrow R$$

result(ψ, γ) =

CASES γ IN
  NIL $\rightarrow$ ψ
  T $\rightarrow$ result(next(ψ, γ))
ENDCASE

(this specifies an iterated application of next until the control-state becomes empty).

4.1.1 Interpretive model used on SNAIL

$next(\psi, \gamma_0) =$

CASES γ_0 IN

$\xi := \varepsilon; \gamma' \rightarrow (assign_I(\xi, \varepsilon_\psi)\psi, \gamma')$,

TEST ε DO γ_1 OR γ_2 TSET$;\gamma' \rightarrow$ $[\varepsilon_\psi \rightarrow (\psi, \gamma_1;\gamma'),$
$\neg\varepsilon_\psi \rightarrow (\psi, \gamma_2;\gamma')]$,

IF $\varepsilon_1 \rightarrow \gamma_1 \# \cdots \# \varepsilon_n \rightarrow \gamma_n$ FI$;\gamma' \rightarrow$ $[(\varepsilon_1)_\psi \rightarrow (\psi, \gamma_1;\gamma'),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow (\psi, \gamma_n;\gamma')]$,

DO $\varepsilon_1 \rightarrow \gamma_1 \# \cdots \# \varepsilon_n \rightarrow \gamma_n$ OD$;\gamma' \rightarrow$ $[(\varepsilon_1)_\psi \rightarrow (\psi, \gamma_1;\gamma_0),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow (\psi, \gamma_n;\gamma_0),$
$TT \rightarrow (\psi, \gamma')]$,

WHILE ε DO γ OD$;\gamma' \rightarrow$ $[\varepsilon_\psi \rightarrow (\psi, \gamma;\gamma_0),$
$\neg\varepsilon_\psi \rightarrow (\psi, \gamma')]$,

BEGIN DEF $\xi=\varepsilon;\gamma$ END$;\gamma' \rightarrow$ LET $\xi' = New_I(\psi)$ IN
$(augment_I(\psi, \xi'), \xi'=\varepsilon; \gamma(\xi'/\xi); \gamma')$,

BEGIN $e;\gamma$ END$;\gamma' \rightarrow (\psi, \gamma;\gamma')$,

READ $(\xi);\gamma' \rightarrow (Rd_I(\xi)\psi, \gamma')$,

WRITE $(\varepsilon);\gamma' \rightarrow (Wr_I(\varepsilon_\psi)\psi, \gamma')$,

$e;\gamma' \rightarrow (\psi, \gamma')$

ENDCASE

where $\gamma(\xi'/\xi)$ denotes substitution of ξ' for all free ξ .
In this formulation γ' is assumed to be a command or the empty list NIL. Please note that next is undefined if the control state is empty.

4.2 Computational Model

This differs from the previous one in that it discards the notion of control state. Now a program is mapped into a list of machine states representing its entire computation.

The basic function will be

$$comp : R \times COM \rightarrow R^*$$

and it is established by a recursive definition.

The connection between the two models is described by

Theorem 1 : $last \circ comp = result$

As usual ";" acts as append-operator.

4.2.1 Computational model used on SNAIL

$comp\ (\psi, f_0)$ =

CASES f_0 IN

$NIL \rightarrow (\psi)$,

$f_1; f_2 \rightarrow comp(\psi, f_1);\ tail \circ comp(last \circ comp(\psi, f_1), f_2)$,

$e \rightarrow (\psi, \psi)$,

$\xi := \varepsilon \rightarrow (\psi,\ assign_I(\xi, \varepsilon_\psi)\ \psi)$,

TEST ε DO f_1 OR f_2 TSET $\rightarrow$ $[\ \varepsilon_\psi \rightarrow comp(\psi, f_1),\ \neg\varepsilon_\psi \rightarrow comp(\psi, f_2)\ ]$,

IF $\varepsilon_1 \rightarrow f_1 \# \cdots\cdots \# \varepsilon_n \rightarrow f_n$ FI $\rightarrow$ $[\ (\varepsilon_1)_\psi \rightarrow comp(\psi, f_1),\ \vdots\ (\varepsilon_n)_\psi \rightarrow comp(\psi, f_n)\ ]$,

DO $\varepsilon_1 \rightarrow f_1 \# \cdots\cdots \# \varepsilon_n \rightarrow f_n$ OD $\rightarrow$ $[\ (\varepsilon_1)_\psi \rightarrow comp(\psi, f_1; f_0),\ \vdots\ (\varepsilon_n)_\psi \rightarrow comp(\psi, f_n; f_0),\ TT \rightarrow (\psi)\ ]$,

WHILE ε DO f OD $\rightarrow$ $[\ \varepsilon_\psi \rightarrow comp(\psi, f; f_0),\ \neg\varepsilon_\psi \rightarrow (\psi)\ ]$,

BEGIN DEF $\xi = \varepsilon$; f END $\rightarrow$ LET $\xi' = New_I(\psi)$ IN $comp(augment_I(\psi, \xi'),\ \xi' := \varepsilon; f(\xi'/\xi))$,

BEGIN e ; f END $\rightarrow$ $comp(\psi, f)$,

READ(ξ) $\rightarrow$ $(\psi,\ Rd_I(\xi)\ \psi)$,

WRITE(ε) $\rightarrow$ $(\psi,\ Wr_I(\varepsilon_\psi)\ \psi)$

ENDCASE

At a first glance the expression for comp used on a list of two commands, $\gamma_1;\gamma_2$, can look rather complicated. The idea behind the definition of comp is that comp(ψ,γ_0) should be the list of configurations traversed during execution of the command γ_0 starting in the configuration ψ. Having this in mind it is easy to understand the expression:

$$\text{comp}(\psi,\gamma_1;\gamma_2) = \text{comp}(\psi,\gamma_1)\ ;\ \text{tail} \circ \text{comp}(\ \text{last} \circ \text{comp}(\psi,\gamma_1),\gamma_2)$$

We first compute the list comp(ψ,γ_1) of configurations traversed during execution of γ_1 with start-configuration ψ. We next add to this the list of configurations traversed during execution of the command γ_2 with a start-configuration which is identical to the final-configuration for the former execution of γ_1. This configuration is last $\circ$ comp(ψ,γ_1). Tail is used to avoid that this configuration is included twice in the resulting list of configurations.

4.2.2 Proof for Theorem 1 applied to SNAIL

The proof is done by showing, that the two sides of the equalitysign, result and last•comp, has the same recursive definition. In the case of $f_1 f_2$ is used that

$$\text{result}(\psi, f_1 f_2) = \text{result}(\text{result}(\psi, f_1), f_2) \quad .$$

This is not proved in this paper. The proof can be taken directly from Hoare and Lauer /H1/ p. 141-142 and p. 151-152.

result (ψ, f_0) =

CASES f_0 IN

NIL $\rightarrow$ ψ ,

$\xi := \varepsilon$ $\rightarrow$ result (next $(\psi, \xi := \varepsilon)$)

= result ($\text{assign}_I(\xi, \varepsilon_\psi)\psi$, NIL)

= $\text{assign}_I(\xi, \varepsilon_\psi)\psi$,

TEST ε DO f_1 OR f_2 TSET $\rightarrow$ result ($[\varepsilon_\psi \rightarrow (\psi, f_1)$,
$\neg\varepsilon_\psi \rightarrow (\psi, f_2)]$)

= $[\varepsilon_\psi \rightarrow$ result (ψ, f_1),
$\neg\varepsilon_\psi \rightarrow$ result $(\psi, f_2)]$,

IF $\varepsilon_1 \rightarrow f_1 \# \cdots\cdots \# \varepsilon_n \rightarrow f_n$ FI $\rightarrow$ result ($[(\varepsilon_1)_\psi \rightarrow (\psi, f_1)$,
$\vdots$
$(\varepsilon_n)_\psi \rightarrow (\psi, f_n)]$)

= $[(\varepsilon_1)_\psi \rightarrow$ result (ψ, f_1),
$\vdots$
$(\varepsilon_n)_\psi \rightarrow$ result $(\psi, f_n)]$,

$$\text{DO } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ OD} \to \text{result}\big([(\varepsilon_1)_\psi \to (\psi, f_1; f_0), \ldots, (\varepsilon_n)_\psi \to (\psi, f_n; f_0), \text{TT} \to (\psi, \text{NIL})]\big)$$

$$= [(\varepsilon_1)_\psi \to \text{result}(\psi, f_1; f_0), \ldots, (\varepsilon_n)_\psi \to \text{result}(\psi, f_n; f_0), \text{TT} \to \psi] ,$$

$$\text{WHILE } \varepsilon \text{ DO } f \text{ OD} \to \text{result}\big([\varepsilon_\psi \to (\psi, f; f_0), \neg\varepsilon_\psi \to (\psi, \text{NIL})]\big)$$

$$= [\varepsilon_\psi \to \text{result}(\psi, f; f_0), \neg\varepsilon_\psi \to \psi] ,$$

$$\text{BEGIN DEF } \xi = \varepsilon; f \text{ END} \to \text{result}(\text{LET } \xi' = \text{New}_I(\psi) \text{ IN } (\text{augment}_I(\psi, \xi'), \xi' := \varepsilon; f(\xi'/\xi)))$$

$$= \text{LET } \xi' = \text{New}_I(\psi) \text{ IN result}(\text{augment}_I(\psi, \xi'), \xi' := \varepsilon; f(\xi'/\xi)),$$

$$\text{BEGIN } e; f \text{ END} \to \text{result}(\psi, f) ,$$

$$\text{READ}(\xi) \to \text{result}(\text{Rd}_I(\xi)\psi, \text{NIL}) = \text{Rd}_I(\xi)\psi ,$$

$$\text{WRITE}(\varepsilon) \to \text{result}(\text{Wr}_I(\varepsilon_\psi)\psi, \text{NIL}) = \text{Wr}_I(\varepsilon_\psi)\psi ,$$

$$e \to \text{result}(\psi, \text{NIL}) = \psi ,$$

$$f_1; f_2 = \text{result}(\text{result}(\psi, f_1), f_2)$$

ENDCASE

which shall be compared with

$last \circ comp\ (\psi, f_0) =$

CASES f_0 IN

$NIL \rightarrow last(\psi) = \psi$,

$\xi := \varepsilon \rightarrow last\ (comp(\psi, \xi := \varepsilon))$

$= last((\psi, assign_I(\xi, \varepsilon_\psi)\psi)$

$= assign_I(\xi, \varepsilon_\psi)\psi$,

TEST ε DO f_1 OR f_2 TSET $\rightarrow last([\ \varepsilon_\psi \rightarrow comp(\psi, f_1),$
$\neg\varepsilon_\psi \rightarrow comp(\psi, f_2)])$

$= [\ \varepsilon_\psi \rightarrow last \circ comp(\psi, f_1),$
$\neg\varepsilon_\psi \rightarrow last \circ comp(\psi, f_2)\ ]$,

IF $\varepsilon_1 \rightarrow f_1 \# \cdots \# \varepsilon_n \rightarrow f_n$ FI $\rightarrow last([\ (\varepsilon_1)_\psi \rightarrow comp(\psi, f_1),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow comp(\psi, f_n)\ ])$

$= [\ (\varepsilon_1)_\psi \rightarrow last \circ comp(\psi, f_1),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow last \circ comp(\psi, f_n)\ ]$,

DO $\varepsilon_1 \rightarrow f_1 \# \cdots \# \varepsilon_n \rightarrow f_n$ OD $\rightarrow last([\ (\varepsilon_1)_\psi \rightarrow comp(\psi, f_1; f_0),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow comp(\psi, f_n; f_0),$
$TT \rightarrow (\psi)\ ])$

$= [(\varepsilon_1)_\psi \rightarrow last \circ comp(\psi, f_1; f_0),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow last \circ comp(\psi, f_n; f_0),$
$TT \rightarrow \psi\ ]$,

$$\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD} \rightarrow \mathit{last}\,(\,[\,\varepsilon_\psi \rightarrow \mathit{comp}(\psi, \gamma; \gamma_0),\ \neg\varepsilon_\psi \rightarrow (\psi)\,]\,)$$

$$= [\,\varepsilon_\psi \rightarrow \mathit{last}\circ\mathit{comp}(\psi, \gamma; \gamma_0),\ \neg\varepsilon_\psi \rightarrow \psi\,],$$

$$\text{BEGIN DEF } \xi = \varepsilon\,;\ \gamma \text{ END} \rightarrow \mathit{last}\,(\text{LET } \xi' = \mathit{New}(\psi) \text{ IN } (\mathit{comp}\,(\mathit{augment}_I(\psi, \xi'), \xi' := \varepsilon; \gamma(\xi'/\xi))))$$

$$= \text{LET } \xi' = \mathit{New}(\psi) \text{ IN } (\mathit{last}\circ\mathit{comp}(\mathit{augment}_I(\psi, \xi'), \xi' := \varepsilon; \gamma(\xi'/\xi))),$$

$$\text{BEGIN } e; \gamma \text{ END} \rightarrow \mathit{last}\circ\mathit{comp}(\psi, \gamma)$$

$$\text{READ}(\xi) \rightarrow \mathit{last}\,((\psi, \mathit{Rcd}_I(\xi)\,\psi))$$

$$= \mathit{Rcd}_I(\xi)\,\psi\ ,$$

$$\text{WRITE}(\varepsilon) \rightarrow \mathit{last}\,((\psi, \mathit{Wr}_I(\varepsilon_\psi)\,\psi))$$

$$= \mathit{Wr}_I(\varepsilon_\psi)\,\psi\ ,$$

$$e \rightarrow \mathit{last}((\psi, \psi))$$

$$= \psi\ ,$$

$$\gamma_1; \gamma_2 \rightarrow \mathit{last}\,(\mathit{com}(\psi, \gamma_1);\ \mathit{tail}\circ\mathit{comp}(\mathit{last}\circ\mathit{comp}(\psi, \gamma_1), \gamma_2))$$

$$= \mathit{last}\circ\mathit{comp}\,(\mathit{last}\circ\mathit{comp}(\psi, \gamma_1), \gamma_2)$$

ENDCASE

end of proof
for Theorem 1

4.3 Relational Theory

We now move into the implicit approaches by discarding all intermediate steps in the computation. This can be compared with the difference between a procedure in some programming language and a mathematical function. The former contains a detailed specification of each step in an evaluation algorithm, while the letter merely is a relation and says nothing about how the result-value can be obtained from the parameter-values. In proving proterties of programming parts we want the last alternative. If using for example a sine-function we want to know that it computes the right answers. How they actually is found is irrelevant (if one is assured that there exists an evaluation algorithm). For a further discussion of this difference between abstraction and realization please see D. Scott /S/ and C. Strachey /S2/.

In the relational theory each command are connected to a relation in R×R. A straightforward notation is to write

$$\psi \; R_\gamma \; \psi'$$

to indicate that (ψ,ψ') is a member of the relation R_γ connected to the command, γ .

This notation will often be abbreviated (removing the redundant "R"):

$$\psi(\gamma)\psi' \quad .$$

The semantics for a language is given in form of a set of axioms - a theory. It is possible to interpret this theory in the computational model:

$$\psi \; R_\gamma \; \psi' \;\equiv\; \psi' = last \circ comp(\psi,\gamma) \quad .$$

It is now possible to prove the axioms correct (satisfied) under this interpretation:

Theorem 2: Relational theory is satisfied by the computational model

Corellary: The relational theory is satisfied by the interpretive model.

For an introduction to mathematical logics especially aimed for computer scientists see Levin /L1/.

It should be stressed that the relations are total. $\psi(\gamma)\psi'$ can be true or false - but not undefined.

4.3.1 Relational theory used on SNAIL

A1. $\psi(\gamma_1;\gamma_2)\psi' \equiv \exists\psi''[\psi(\gamma_1)\psi'' \wedge \psi''(\gamma_2)\psi']$

A2. $\psi(\xi := \varepsilon)\psi' \equiv \psi' = assign_I(\xi, \varepsilon_\psi)\psi$

A3. ψ (TEST ε DO γ_1 OR γ_2 TSET) ψ'

$\equiv (\varepsilon_\psi \wedge \psi(\gamma_1)\psi') \vee (\neg\varepsilon_\psi \wedge \psi(\gamma_2)\psi')$

A4 ψ (IF $\varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n$ FI) ψ'

$\equiv B_\psi \neq \varnothing \wedge \psi(\gamma_{\min B_\psi})\psi'$

A5 ψ (DO $\varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n$ OD) $\psi' \Rightarrow B_{\psi'} = \varnothing$

A6 $\forall\psi\psi'[I(\psi) \wedge \psi(\gamma_{\min B_\psi})\psi' \Rightarrow I(\psi')] \Rightarrow$

$\forall\psi\psi'[I(\psi) \wedge \psi(DO\ \varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n\ OD)\psi' \Rightarrow I(\psi')]$

A7 ψ (WHILE ε DO γ OD) $\psi' \Rightarrow \neg\varepsilon_{\psi'}$

A8 $\forall\psi\psi'[I(\psi) \wedge \varepsilon_\psi \wedge \psi(\gamma)\psi' \Rightarrow I(\psi')] \Rightarrow$

$\forall\psi\psi'[I(\psi) \wedge \psi(WHILE\ \varepsilon\ DO\ \gamma\ OD)\psi' \Rightarrow I(\psi')]$

A9 ψ (BEGIN DEF $\xi = \varepsilon; \gamma$ END) ψ'

$\equiv \exists\xi'[\xi' = New_I(\psi) \wedge augment_I(\psi,\xi')(\xi' := \varepsilon; \gamma(\xi'/\xi))\psi']$

A10 ψ (BEGIN $e; \gamma$ END) $\psi' \equiv \psi(\gamma)\psi'$

A11 ψ (READ (ξ)) $\psi' \equiv \psi' = Rd_I(\xi)\psi$

A12 ψ (WRITE (ε)) $\psi' \equiv \psi' = Wr_I(\varepsilon_\psi)\psi$

A13 $\psi(e)\psi' \equiv \psi' = \psi$

where $B_\psi = \{i \mid (\varepsilon_i)_\psi\}$

and $B_\psi = \varnothing \Rightarrow \gamma_{\min B_\psi} = e$.

$I(\psi)$ is any first order logical formula possibly depending on ψ.

Before A1 - A13 can be viewed as a theory we must give axioms defining New_I, $augment_I$, $assign_I$, Rd_I and Wr_I.

$$New_I(\psi) = \alpha \Rightarrow \alpha \notin domain(\psi V1)$$

$$augment_I(\langle z, \kappa_I, \kappa_o\rangle, \xi) = \langle z', \kappa_I', \kappa_o'\rangle$$

$$\Updownarrow$$

$$\begin{cases} \xi \in domain(z) \\ domain(z') = domain(z) \cup \{\xi\} \\ z'(\alpha) = NOTINIT \\ \kappa_I' = \kappa_I \\ \kappa_o' = \kappa_o \end{cases}$$

where domain is the operator which gives the definition-domain for partial functions. NOTINIT is an errorvalue in domain V.

$assign_I$, Rd_I and Wr_I are defined by 7 and 8 on page 25 together with page 20-21.

From A4 and A6 follows imediately the weaker axioms:

$$A4' \quad (B_\psi \neq \emptyset \wedge \forall i\ [(\varepsilon_i)_\psi \Rightarrow \psi(f_i)\psi']) \Rightarrow \psi(IF\ \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n\ FI)\psi'$$

$$A6' \quad \forall i\ \forall\psi\psi'\ [I(\psi) \wedge (\varepsilon_i)_\psi \wedge \psi(f_i)\psi' \Rightarrow I(\psi')] \Rightarrow \forall\psi\psi'\ [I(\psi) \wedge \psi(DO\ \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n\ OD)\psi' \Rightarrow I(\psi')].$$

If our approach had been nondeterministic in the selection of guards, the axioms A4 and A6 would have been replaced by A4' (with $\equiv$ substituted for the last $\Rightarrow$) and A6' (no substitution) respectively.

4.3.2 Proof for Theorem 2 applied to SNAIL

To avoid boring the reader (and myself) too much I shall now restrict the examination to MINI-SNAIL (see page 12).

In the proof is used the expression for last ∘ comp derived on page 33.

A1 - Command lists

$$\Updownarrow \quad \psi\,(\zeta_1;\zeta_2)\,\psi'$$

$$\Updownarrow \quad \psi' = last \circ comp\,(\psi, \zeta_1, \zeta_2)$$

$$\Updownarrow \quad \psi' = last\,(comp\,(\psi,\zeta_1);\ tail \circ comp\,(last \circ comp\,(\psi,\zeta_1),\zeta_2))$$

$$\Updownarrow \quad \exists\psi''\,[\,\psi'' = last \circ comp\,(\psi,\zeta_1) \ \wedge \ \psi' = last\,(comp\,(\psi,\zeta_1);\ tail \circ comp\,(\psi'',\zeta_2))\,]$$

$$\Updownarrow \quad \exists\psi''\,[\,\psi'' = last \circ comp\,(\psi,\zeta_1) \wedge \psi' = last \circ comp\,(\psi'',\zeta_2)\,]$$

$$\exists\psi''\,[\,\psi\,(\zeta_1)\,\psi'' \wedge \psi''\,(\zeta_2)\,\psi'\,]$$

where it is used that $comp(\psi'',\zeta_2)$ always has at least two elements - except in trivial cases.

A2 - Assignment

$$\Updownarrow \quad \psi\,(\xi := \varepsilon)\,\psi'$$

$$\Updownarrow \quad \psi' = last \circ comp\,(\psi,\ \xi := \varepsilon)$$

$$\psi' = assign_I\,(\xi, \varepsilon_\psi)\,\psi$$

A4 - IF-command

$$\psi\ (\text{IF } \varepsilon_1 \to f_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to f_n \text{ FI})\ \psi'$$

$$\psi' = last \circ comp(\psi, \text{IF } \varepsilon_1 \to f_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to f_n \text{ FI})$$

$$\psi' = [\ (\varepsilon_1)\psi \to last \circ comp(\psi, f_1),$$
$$\vdots$$
$$(\varepsilon_n)\psi \to last \circ comp(\psi, f_n)\ ]$$

$$B\psi \neq \varnothing \ \wedge\ \psi' = last \circ comp(\psi, f_{min\, B\psi})$$

$$B\psi \neq \varnothing \ \wedge\ \psi\ (f_{min\, B\psi})\ \psi'$$

A7 - A8 - WHILE-command

In both cases we start with the following rewriting:

$$\psi\ (\text{WHILE } \varepsilon \text{ DO } f \text{ OD})\ \psi'$$

$$\psi' = last \circ comp(\psi, \text{WHILE } \varepsilon \text{ DO } f \text{ OD})$$

$$\psi' = [\ \varepsilon_\psi \to last \circ comp(\psi, f; f_0),$$
$$\neg\varepsilon_\psi \to \psi\ ]$$

$$\psi' = [\varepsilon_\psi \to last(comp(\psi, f); tail \circ comp(last \circ comp(\psi, f), f_0)),$$
$$\neg\varepsilon_\psi \to \psi\ ]$$

$$\psi' = [\varepsilon_\psi \to last \circ comp(last \circ comp(\psi, f), f_0)),$$
$$\neg\varepsilon_\psi \to \psi\ ]$$

where $f_0 = \text{WHILE } \varepsilon \text{ DO } f \text{ OD}$

A7:

The proof is now finished by induction after the lenght of comp(ψ , WHILE ε DO γ OD) = comp(ψ,γ_0).

lenght = 1: Then it follows that $\neg \varepsilon_\psi$ and $\psi' = \psi$, which immediately gives the desired result.

lenght > 1: The desired result follows from the inductional hypothesis since the lenght of

$$\text{comp} (\text{last} \circ \text{comp}(\psi,\gamma), \gamma_0)$$

is less than the lenght of comp(ψ,γ_0) whenever the latter is defined.

A8:

Again the proof is done by induction after the lenght of comp(ψ , WHILE ε DO γ OD) = comp(ψ,γ_0).

We first write A8 in our interpretation:

$$\forall\psi\psi' [I(\psi) \wedge \varepsilon_\psi \wedge (\psi' = last \circ comp(\psi,\gamma)) \Rightarrow I(\psi')] \Rightarrow$$
$$\forall\psi\psi' [I(\psi) \wedge (\psi' = last \circ comp(\psi,\gamma_0)) \Rightarrow I(\psi')]$$

Assume:

1. The antecedent of A8:

$$\forall\psi\psi^+ [I(\psi) \wedge \varepsilon_\psi \wedge (\psi^+ = last \circ comp(\psi,\gamma)) \Rightarrow I(\psi^+)]$$

2. $I(\psi) \wedge \psi' = last \circ comp(\psi, \gamma_0)$

lenght = 1: It follows that $\psi' = \psi$ which immediately gives $I(\psi')$.

lenght > 1: From the definition of comp follows: ε_ψ
From this and from assumption 1 and 2 above follows by modus ponens that

*) $I(\psi^+) = I(\mathit{last} \circ \mathit{comp}(\psi, \chi)) = \mathit{true}$.

From our rewriting on page 40 and from ε_ψ follows that

**) $\psi' = \mathit{last} \circ \mathit{comp}(\mathit{last} \circ \mathit{comp}(\psi, \chi), \chi_0)$.

The lenght of comp(last ∘ comp(ψ, χ), χ_0) is less than the lenght of comp (ψ, χ_0) whenever the latter is defined.

The inductional hypothesis together with *) and **) therefore gives the desired $I(\psi')$. - Set ψ = last ∘ comp(ψ, χ) in the consequent of A8.

A9 - DECLARATION

$$
\begin{array}{l}
\Updownarrow \quad \psi\ (\mathit{BEGIN}\ \mathit{DEF}\ \xi = \varepsilon; \chi\ \mathit{END})\ \psi' \\
\Updownarrow \quad \psi' = \mathit{last} \circ \mathit{comp}(\psi,\ \mathit{BEGIN}\ \mathit{DEF}\ \xi = \varepsilon; \chi\ \mathit{END}) \\
\Updownarrow \quad \psi' = \mathit{LET}\ \xi' = \mathit{New}(\psi)\ \mathit{IN}\ (\mathit{last} \circ \mathit{comp}(\mathit{augment}_I(\psi, \xi'), \xi' := \varepsilon; \chi(\xi'/\xi))) \\
\Updownarrow \quad \exists \xi' [\xi' = \mathit{New}(\psi) \wedge \psi' = \mathit{last} \circ \mathit{comp}(\mathit{augment}_I(\psi, \xi'), \xi' := \varepsilon; \chi(\xi'/\xi))) \\
\quad\ \exists \xi' [\xi' = \mathit{New}(\psi) \wedge \mathit{augment}_I(\psi, \xi')(\xi' := \varepsilon; \chi(\xi'/\xi))\ \psi']
\end{array}
$$

end of proof
for Theorem 2

The corollary follows from Theorem 1 and Theorem 2.

4.4 Deductive theory

We now make another big jump toward abstraction by entirely discarding the notion of configurations (states). Instead our relations now are defined over PRED x PRED ,where PRED is the set of all first order logical predicates with free variables in the domains of identifiers, inputs and outputs. We use P, Q or R (possibly indexed) to denote elements of PRED. For a definition of first order logic see Levin /L1/.

As in the relational theory we will connect a relation to each command. In /H/ Hoare suggest the notation:

$$\{P\}\,\gamma\,\{Q\}$$

to indicate that (P,Q) is a member of the relation connected to the command, γ . We shall by convenience instead use the analogue notation:

$$P\,\{\gamma\}\,Q\ .$$

As before the semantics for a language is specified in form of a set of actions and deduction rules. It is possible to interpret these in terms of the relational thery:

$$P\,\{\gamma\}\,Q \equiv \forall\psi\psi'\,[P(\psi)\wedge\psi(\gamma)\psi' \Rightarrow Q(\psi')]$$

which intuitively means that $P\{\gamma\}Q$ is true iff P satisfied before execution of γ implies that Q is satisfied after execution if this execution stops. Since $P\{\gamma\}Q$ = true does not guarantee that execution actually stops we are said to deal with <u>partial correctness</u>. A seperate proof must be done to show that execution of our program really stops.

It is now possible to show

<u>Theorem 3</u>: The deductive theory is conservative as regards the relational theory.

<u>Corollary</u>: The deductive theory is satisfied by the computational model and by the interpretive model.

That one theory is conservative as regards another theory means that everything which can be deduced in the second theory also has a deduction in the first theory. To shown conservatism it is enough to show that all axioms constituting the second theory can be proved in the first theory under the given interpretation. For further information about this see Leven /L1/ page 99.

D1 $$\frac{P\{f_1\}Q \wedge Q\{f_2\}R}{P\{f_1;f_2\}R}$$

D2 $$P(E/\xi)\{\xi := E\}P$$

D3 $$\frac{(P\wedge E)\{f_1\}Q \wedge (P\wedge\neg E)\{f_2\}Q}{P\{\text{TEST } E \text{ DO } f_1 \text{ OR } f_2 \text{ TSET}\}Q}$$

D4 $$\frac{\forall i\le n\,[(P\wedge E_i)\{f_i\}Q]}{P\{\text{IF } E_1\to f_1 \# \cdots \# E_n\to f_n \text{ FI}\}Q}$$

D5 $$\frac{\forall i\le n\,[(P\wedge E_i)\{f_i\}P]}{P\{\text{DO } E_1\to f_1 \# \cdots \# E_n\to f_n \text{ OD}\}(P\wedge\neg(\exists i: E_i))}$$

D6 $$\frac{(P\wedge E)\{f\}P}{P\{\text{WHILE } E \text{ DO } f \text{ OD}\}(P\wedge(\neg E))}$$

D7 $$\frac{P\{\xi := E;\ f(\xi'/\xi)\}Q}{P\{\text{BEGIN DEF } \xi = E;\ f \text{ END}\}Q}$$

where ξ' is a new unused identifier.

D8 $$\frac{P\{f\}Q}{P\{\text{BEGIN } e;\ f \text{ END}\}Q}$$

D9 $$P((\mu_I\downarrow 1),(\mu_I\uparrow 2)\ /\ \xi,\mu_I)\{\text{READ}(\xi)\}P$$

D10 $$P((\mu_0@E)/\mu_0)\{\text{WRITE}(E)\}P$$

D11 $$P\{e\}P$$

4.4.2 Proof for Theorem 3 applied to SNAIL

Once more I shall only consider MINI-SNAIL.

D1 - Command lists

$$\begin{aligned}&(\ \forall\psi\psi'[\,P\psi \wedge \psi(f_1)\psi' \Rightarrow Q\psi'\,] \wedge\\ &\ \ \forall\psi\psi'[\,Q\psi \wedge \psi(f_2)\psi' \Rightarrow R\psi'\,]\ ) \Rightarrow\\ &\ \ \forall\psi\psi'[\,P\psi \wedge \psi(f_1;f_2)\psi' \Rightarrow R\psi'\,]\end{aligned}$$

Proof:

Assume 1. The two antecedents.

2. $P\psi \wedge \psi(f_1;f_2)\psi'$

$$\Updownarrow$$

$$\psi(f_1;f_2)\psi'$$

$$\Updownarrow$$

$$\exists\psi''[\,\psi(f_1)\psi'' \wedge \psi''(f_2)\psi'\,] \qquad \text{by A1.}$$

The first antecedent gives $Q\psi''$
and then the second antecedent gives $R\psi'$.

D2 - Assignment

$$\forall\psi\psi'[\,R(\varepsilon/\xi)\psi \wedge \psi(f)\psi' \Rightarrow R\psi'\,]$$

Proof:

Let $R = R(\xi, \xi^1, \xi^2, \ldots, \xi^n)$, where $\xi, \xi^1, \xi^2, \ldots, \xi^n$ are all free variables in R.

We have

$$\begin{aligned}&\Updownarrow\\ &R(\varepsilon/\xi)\psi\\ &\Updownarrow\\ &R(\varepsilon, \xi^1, \xi^2, \ldots, \xi^n)\psi\\ &\Updownarrow\\ &R(\varepsilon_\psi, \xi^1_\psi, \xi^2_\psi, \ldots, \xi^n_\psi)\end{aligned}$$

and

$$\Updownarrow \quad \psi\,(\xi := \varepsilon)\,\psi'$$

$$\psi' = assign_I(\xi, \varepsilon_\psi)\,\psi \qquad \text{by A2.}$$

The definition of $assign_I$ gives

$$\Updownarrow \quad \mathcal{R}\,\psi'$$

$$\Updownarrow \quad \mathcal{R}\,(\xi, \xi^1, \xi^2, \ldots\ldots, \xi^n)\,\psi'$$

$$\Updownarrow \quad \mathcal{R}\,(\xi_{\psi'}, \xi^1_{\psi'}, \xi^2_{\psi'}, \ldots\ldots \xi^n_{\psi'})$$

$$\Updownarrow \quad \mathcal{R}\,(\varepsilon_\psi, \xi^1_\psi, \xi^2_\psi, \ldots\ldots \xi^n_\psi)$$

We have now proved that

$$\psi\,(\xi := \varepsilon)\,\psi' \Rightarrow (\mathcal{R}(\varepsilon/\xi)\psi \Leftrightarrow \mathcal{R}\psi')$$

which is a stronger result than the desired.

D4 - IF-command

$$\forall\psi\psi'\ \forall i\ [\mathcal{P}\psi \wedge (\mathcal{E}_i)_\psi \wedge \psi\,(f_i)\,\psi' \Rightarrow \mathcal{Q}\psi'] \Rightarrow$$
$$\forall\psi\psi'\ [\mathcal{P}\psi \wedge \psi\,(\text{IF } \mathcal{E}_1 \to f_1 \,\#\, \ldots\ldots \,\#\, \mathcal{E}_n \to f_n \text{ FI})\,\psi' \Rightarrow \mathcal{Q}\psi']$$

Proof:

Assume 1. Antecedent

2. $\mathcal{P}\psi \wedge \psi\,(\text{IF } \mathcal{E}_1 \to f_1 \,\#\, \ldots\ldots \,\#\, \mathcal{E}_n \to f_n \text{ FI})\,\psi'$

$$\Updownarrow \quad \psi\,(\text{IF } \mathcal{E}_1 \to f_1 \,\#\, \ldots\ldots \,\#\, \mathcal{E}_n \to f_n \text{ FI})\,\psi'$$

$$\mathcal{B}_\psi \neq \emptyset \wedge \psi\,(f_{min\,\mathcal{B}_\psi})\,\psi' \qquad \text{by A4.}$$

The antecedent (assumption 1) gives $\mathcal{Q}\psi'$.

D6 - WHILE-command

$$\forall \psi \psi' \; [\; (P \wedge \varepsilon)\psi \;\wedge\; \psi(\gamma)\psi' \Rightarrow P\psi' \;] \Rightarrow$$
$$\forall \psi \psi' \; [\; P\psi \wedge \psi(\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD})\psi' \Rightarrow (P \wedge (\neg \varepsilon))\psi' \;]$$

Proof:
Immediately from A7 and A8.

D7 - Declaration

$$\forall \psi \psi' \; [\; P\psi \wedge \psi(\xi' := \varepsilon;\, \gamma(\xi'/\xi))\psi' \Rightarrow Q\psi' \;] \Rightarrow$$
$$\forall \psi \psi' \; [\; P\psi \wedge \psi(\text{BEGIN DEF } \xi = \varepsilon;\, \gamma \text{ END})\psi' \Rightarrow Q\psi' \;]$$

where ξ' is a new unused identifier.

Proof:
Assume
1. Antecedent
2. $P\psi \wedge \psi(\text{BEGIN DEF } \xi = \varepsilon;\, \gamma \text{ END})\psi'$

$$\Leftrightarrow \quad \psi(\text{BEGIN DEF } \xi = \varepsilon;\, \gamma \text{ END})\psi'$$
$$\exists \xi' \; [\; \xi' = \text{New}(\psi) \wedge \text{augment}_I(\psi, \xi')\; (\xi' := \varepsilon;\, \gamma(\xi'/\xi))\psi'$$

by A9.

The definition of augment_I gives $P(\psi) = P(\text{augment}_I(\psi, \xi'))$ since ξ' is a new unused identifier and therefore cannot be a free variable of P.

The antecedent (assumption 1) gives $Q\psi'$.

end of proof
for Theorem 3.

The corollary follows from Theorem 2, Corollary of Theorem 2 and from Theorem 3.

5 A REFORMULATION OF THE IDEAS GIVEN BY HOARE AND LAUER

At a first glance it may be difficult to see any close connection between the ideas developed by Hoare and Lauer in /H1/ and mathematical semantics as described by Strachey, Scott and Wadsworth in /S/, /S1/, /S2/ and /S3/. A more profound examination however shows, that there is an indeed very strong correspondence between these two apparently different approaches.

In this section I shall rewrite Hoares and Lauers ideas to the notation of mathematical semantics. It turns out that such a change of notation in many aspects makes the theorems and proofs more straightforward and understandable.

As many times before I shall restrict the examination to MINI-SNAIL. The auxilliary functions and the notation used will be as in my description of Hoares and Lauers original formulation.

It may be a help to know that the mnemonic subscripts I, C and A stands for interpretive, computational and auxilliary.

5.1 Interpretive Model

In the original formulation of Hoare and Lauer result is a member of the domain $[[R \times COM] \rightarrow R]$. Since it is trivial to show that this domain is isomorph with $[COM \rightarrow [R \rightarrow R]]$, and since the intuitive meaning of result(ψ, γ) is the configuration reached when the command, γ, is evaluated starting from ψ, it would be natural to try the following substitution.

$$\mathcal{E}_I[\![\gamma]\!]\psi = \text{result}(\psi, \gamma)$$

If this is inserted everywhere in the definition of result and next we get a recursive definition of a function

$$\mathcal{E}_I \in [COM \rightarrow [R \rightarrow R]] .$$

In the following reformulation one further modification occurs. Hoare and Lauer defines a state, s, as a partial function whose domain is the set of used (unique) identifiers. In mathematical semantics it is common to view all functions as total allowing them to take the value, $\perp$, for undefined. It is here important to distinguish beween $\perp$ which means undefined and NOTINIT (Ω in Hoare and Lauer) which means defined but not initialized.

Viewed in this frame the only action taken by augment$_I(\psi, \xi)$ is to change the value of ξ from $\perp$ to NOTINIT. Each declaration in SNAIL assigns an initial value to the declared identifier and therefore any use of augment$_I(\psi, \xi)$ is immediately followed by an operation updating the value of ξ. By this reason augment$_I$ can be omitted simply replacing all occurences of augment$_I(\psi, \xi)$ by ψ.

Command lists

$$\mathcal{C}_I[\![f_1; f_2]\!]\psi$$
$$= result(\psi, f_1; f_2)$$
$$= result(result(\psi, f_1), f_2)$$
$$= \mathcal{C}_I[\![f_2]\!](\mathcal{C}_I[\![f_1]\!]\psi)$$
$$= \mathcal{C}_I[\![f_2]\!] \circ \mathcal{C}_I[\![f_1]\!]\psi$$

Assignment

$$\mathcal{C}_I[\![\xi := \varepsilon]\!]\psi$$
$$= result(\psi, \xi := \varepsilon)$$
$$= result(next(\psi, \xi := \varepsilon))$$
$$= result(assign_I(\xi, \varepsilon_\psi)\psi, e)$$
$$= assign_I(\xi, \mathcal{E}_I[\![\varepsilon]\!]\psi)\psi$$

IF-command

$$\mathcal{C}_I[\![IF\ \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n\ FI]\!]\psi$$
$$= result(next(\psi, IF\ \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n\ FI))$$
$$= result([\ (\varepsilon_1)_\psi \to (\psi, f_1), \ \vdots \ (\varepsilon_n)_\psi \to (\psi, f_n)\ ])$$
$$= [\ (\varepsilon_1)_\psi \to result(\psi, f_1), \ \vdots \ (\varepsilon_n)_\psi \to result(\psi, f_n)\ ]$$
$$= [\ \mathcal{E}_I[\![\varepsilon_1]\!]\psi \to \mathcal{C}_I[\![f_1]\!]\psi, \ \vdots \ \mathcal{E}_I[\![\varepsilon_n]\!]\psi \to \mathcal{C}_I[\![f_n]\!]\psi\ ]$$

WHILE-command

$$\mathcal{E}_I[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\psi$$
$$= \text{result}(\text{next}(\psi, \text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}))$$
$$= \text{result}([\varepsilon_\psi \to (\psi, \gamma; \text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}), \neg\varepsilon_\psi \to (\psi, e)])$$
$$= [\varepsilon_\psi \to \text{result}(\text{result}(\psi, \gamma), \text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}), \neg\varepsilon_\psi \to \text{result}(\psi, e)]$$
$$= [\mathcal{E}_I[\![\varepsilon]\!]\psi \to \mathcal{E}_I[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!] \circ \mathcal{E}_I[\![\gamma]\!]\psi, \neg\mathcal{E}_I[\![\varepsilon]\!]\psi \to \psi]$$
$$= \mathcal{E}_I[\![\varepsilon]\!]\psi \to \mathcal{E}_I[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!] \circ \mathcal{E}_I[\![\gamma]\!]\psi, \psi$$
$$= (\text{FIX } F.(\lambda\psi.\, \mathcal{E}_I[\![\varepsilon]\!]\psi \to F \circ \mathcal{E}_I[\![\gamma]\!]\psi, \psi))\psi$$

Declarations

$$\mathcal{E}_I[\![\text{BEGIN DEF } \xi = \varepsilon; \gamma \text{ END}]\!]\psi$$
$$= \text{result}(\text{next}(\psi, \text{BEGIN DEF } \xi = \varepsilon; \gamma \text{ END}))$$
$$= \text{LET } \xi' = \text{New}_I(\psi) \text{ IN } \text{result}(\psi, \xi' := \varepsilon; \gamma(\xi'/\xi))$$
$$= \text{LET } \xi' = \text{New}_I(\psi) \text{ IN } (\mathcal{E}_I[\![\xi' := \varepsilon; \gamma(\xi'/\xi)]\!]\psi)$$

As can be seen this differs very little from a "standard" mathematical semantics. The main difference is the treatment of declarations (identifier-bindings). Hoare and Lauer uses unique identifiers, where uniqueness is assured by textsubstitution. A "standard" mathematical semantics would use the notion of enviroment.

It should be stressed, that the omitted command-types in SNAIL possesses no further difficulties. They are omitted simply to keep this paper on a resonable lenght.

5.2 Computational Model

A first attempt to make an analogous transformation for this model might be to make the substitution:

$$\mathcal{C}_A[\![\gamma]\!]\psi = \mathrm{comp}(\psi,\gamma) \ .$$

This would give a recursive definition for a function

$$\mathcal{C}_A \in [\mathrm{COM} \to [\mathrm{R} \to \mathrm{R}^*]]$$

This give however a rather nasty equation for simple sequencing:

$$\begin{aligned} & \mathcal{C}_A[\![\gamma_1;\gamma_2]\!]\psi \\ = \ & comp(\psi, \gamma_1;\gamma_2) \\ = \ & comp(\psi,\gamma_1);\ tail \circ comp(last \circ comp(\psi,\gamma_1),\gamma_2) \\ = \ & \mathcal{C}_A[\![\gamma_1]\!]\psi\ ;\ tail(\mathcal{C}_A[\![\gamma_2]\!](last(\mathcal{C}_A[\![\gamma_1]\!]\psi))) \ . \end{aligned}$$

To avoid this we "lift" $\mathcal{C}_A$ to a function

$$\mathcal{C}_C \in [\mathrm{COM} \to [\mathrm{R}^* \to \mathrm{R}^*]]$$

defined by

$$\mathcal{C}_C[\![\gamma]\!]\Psi = \Psi;\ tail(\mathcal{C}_A[\![\gamma]\!]last(\Psi))$$

where

$$\Psi \in R^* .$$

The idea behind this definition is that $\mathcal{C}_C[\![\gamma]\!]\Psi$ appends to Ψ the configurations traversed during execution of $\mathcal{C}_A[\![\gamma]\!]$ on the last element of Ψ . The reader should compare this to the definition of comp on page 29.

Command lists

$$
\begin{aligned}
&\mathcal{C}_c[\![f_1; f_2]\!]\,\Psi \\
&= \Psi;\ tail\,(\mathcal{C}_A[\![f_1; f_2]\!]\ last(\Psi)) \\
&= \Psi;\ tail\,(\mathcal{C}_A[\![f_1]\!]last(\Psi);\ tail(\mathcal{C}_A[\![f_2]\!](last(\mathcal{C}_A[\![f_1]\!]\ last(\Psi))))) \\
&= \Psi;\ tail\,(\mathcal{C}_A[\![f_1]\!]\ last(\Psi));\ tail(\mathcal{C}_A[\![f_2]\!](last(\mathcal{C}_A[\![f_1]\!]\ last(\Psi)))) \\
&= \Psi;\ tail\,(\mathcal{C}_A[\![f_1]\!]\ last(\Psi));\\
&\qquad\qquad tail\,(\mathcal{C}_A[\![f_2]\!](last(\Psi;\ tail(\mathcal{C}_A[\![f_1]\!]\ last(\Psi))))) \\
&= \mathcal{C}_c[\![f_1]\!]\,\Psi;\ tail\,(\mathcal{C}_A[\![f_2]\!](last(\mathcal{C}_c[\![f_1]\!]\,\Psi))) \\
&= \mathcal{C}_c[\![f_2]\!]\,(\mathcal{C}_c[\![f_1]\!]\,\Psi) \\
&= \mathcal{C}_c[\![f_2]\!] \circ \mathcal{C}_c[\![f_1]\!]\,\Psi
\end{aligned}
$$

Assignment

$$
\begin{aligned}
&\mathcal{C}_c[\![\xi := \varepsilon]\!]\,\Psi \\
&= \Psi;\ tail\,(\mathcal{C}_A[\![\xi := \varepsilon]\!]\ last(\Psi)) \\
&= \Psi;\ tail\,(comp(last(\Psi), \xi := \varepsilon)) \\
&= \Psi;\ tail\,(last(\Psi), assign_I(\xi, \mathcal{E}_{last(\Psi)})\ last(\Psi)) \\
&= \Psi;\ assign_I(\xi, \mathcal{E}_I[\![\varepsilon]\!]\ last(\Psi))\ last(\Psi) \\
&= \Psi;\ assign_c(\xi, \mathcal{E}_c[\![\varepsilon]\!]\,\Psi)\,\Psi
\end{aligned}
$$

where $\quad assign_c \in [L \times V \to [R^* \to R]]$

$\qquad\qquad \mathcal{E}_c \in [Exp \to [R^* \to V]]$

is defined by

$$assign_c(\alpha,\beta)\,\Psi = assign(\alpha,\beta)\ last(\Psi)$$

$$\mathcal{E}_c[\![\varepsilon]\!]\,\Psi = \mathcal{E}_I[\![\varepsilon]\!]\ last(\Psi)$$

IF-command

$$\mathcal{C}_C[\![\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}]\!]\,\Psi$$

$$= \Psi;\ tail(\mathcal{C}_H[\![\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}]\!]\,last(\Psi))$$

$$= \Psi;\ tail(comp(last(\Psi), \text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}))$$

$$= \Psi;\ tail\Big([\ (\varepsilon_1)_{last(\Psi)} \to comp(last(\Psi), f_1),\ \vdots\ (\varepsilon_n)_{last(\Psi)} \to comp(last(\Psi), f_n)\ ] \Big)$$

$$= \Psi;\ [\ (\varepsilon_1)_{last(\Psi)} \to tail \circ comp(last(\Psi), f_1),\ \vdots\ (\varepsilon_n)_{last(\Psi)} \to tail \circ comp(last(\Psi), f_n)\ ]$$

$$= [\ (\varepsilon_1)_{last(\Psi)} \to \Psi;\ tail \circ comp(last(\Psi), f_1),\ \vdots\ (\varepsilon_n)_{last(\Psi)} \to \Psi;\ tail \circ comp(last(\Psi), f_n)\ ]$$

$$= [\ \mathcal{E}_I[\![\varepsilon_1]\!]\,last(\Psi) \to \Psi;\ tail(\mathcal{C}_H[\![f_1]\!]\,last(\Psi)),\ \vdots\ \mathcal{E}_I[\![\varepsilon_n]\!]\,last(\Psi) \to \Psi;\ tail(\mathcal{C}_H[\![f_n]\!]\,last(\Psi))\ ]$$

$$= [\ \mathcal{E}_C[\![\varepsilon_1]\!]\,\Psi \to \mathcal{C}_C[\![f_1]\!]\,\Psi,\ \vdots\ \mathcal{E}_C[\![\varepsilon_n]\!]\,\Psi \to \mathcal{C}_C[\![f_n]\!]\,\Psi\ ]$$

WHILE-command

$$\mathcal{E}_C[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\,\Psi$$

$$= \Psi;\ tail(\mathcal{E}_H[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\ last(\Psi))$$

$$= \Psi;\ tail(comp(last(\Psi), \text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}))$$

$$= \Psi;\ tail([\ \varepsilon_{last(\Psi)} \to comp(last(\Psi), \gamma; \text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}),\ \neg\varepsilon_{last(\Psi)} \to last(\Psi)\ ])$$

$$= [\ \varepsilon_{last(\Psi)} \to \Psi;\ tail(comp(last(\Psi), \gamma; \text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD})),\ \neg\varepsilon_{last(\Psi)} \to \Psi;\ tail(last(\Psi))\ ]$$

$$= [\ \mathcal{E}_I[\![\varepsilon]\!]\ last(\Psi) \to \Psi;\ tail(\mathcal{E}_H[\![\gamma; \text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\ last(\Psi)),\ \neg\mathcal{E}_I[\![\varepsilon]\!]\ last(\Psi) \to \Psi\ ]$$

$$= [\ \mathcal{E}_C[\![\varepsilon]\!]\ \Psi \to \mathcal{E}_C[\![\gamma; \text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\ \Psi,\ \neg\mathcal{E}_C[\![\varepsilon]\!]\ \Psi \to \Psi\ ]$$

$$= \mathcal{E}_C[\![\varepsilon]\!]\ \Psi \to \mathcal{E}_C[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!] \circ \mathcal{E}_C[\![\gamma]\!]\ \Psi,\ \Psi$$

$$= (FIX\ \ F.(\lambda\Psi.\ \mathcal{E}_C[\![\varepsilon]\!]\ \Psi \to F \circ \mathcal{E}_C[\![\gamma]\!]\ \Psi,\ \Psi))\ \Psi$$

Declaration

$$\mathcal{E}_C[\![\text{BEGIN DEF } \xi = \varepsilon;\ \gamma \text{ END}]\!]\,\Psi$$

$$= \Psi;\ tail(\mathcal{E}_A[\![\text{BEGIN DEF } \xi = \varepsilon;\ \gamma \text{ END}]\!]\ last(\Psi))$$

$$= \Psi;\ tail(comp(last(\Psi),\ \text{BEGIN DEF } \xi = \varepsilon;\ \gamma \text{ END}))$$

$$= \Psi;\ tail(\text{LET } \xi' = New_I(last(\Psi))\ \text{IN}\ (comp(last(\Psi),\ \xi' := \varepsilon;\ \gamma(\xi'/\xi))))$$

$$= \text{LET } \xi' = New_I(last(\Psi))\ \text{IN}\ (\Psi;\ tail(\mathcal{E}_A[\![\xi' := \varepsilon;\ \gamma(\xi'/\xi)]\!]\ last\ \Psi)$$

$$= \text{LET } \xi' = New_C(\Psi)\ \text{IN}\ (\mathcal{E}_C[\![\xi' := \varepsilon;\ \gamma(\xi'/\xi)]\!]\ \Psi)$$

Again it is easy to see that this resembles a definition in "standard" mathematical semantics.

In fact there is a very strong similarity between the definitions of $\mathcal{E}_I$ and $\mathcal{E}_C$.

5.2.1 Equvalence of the two models

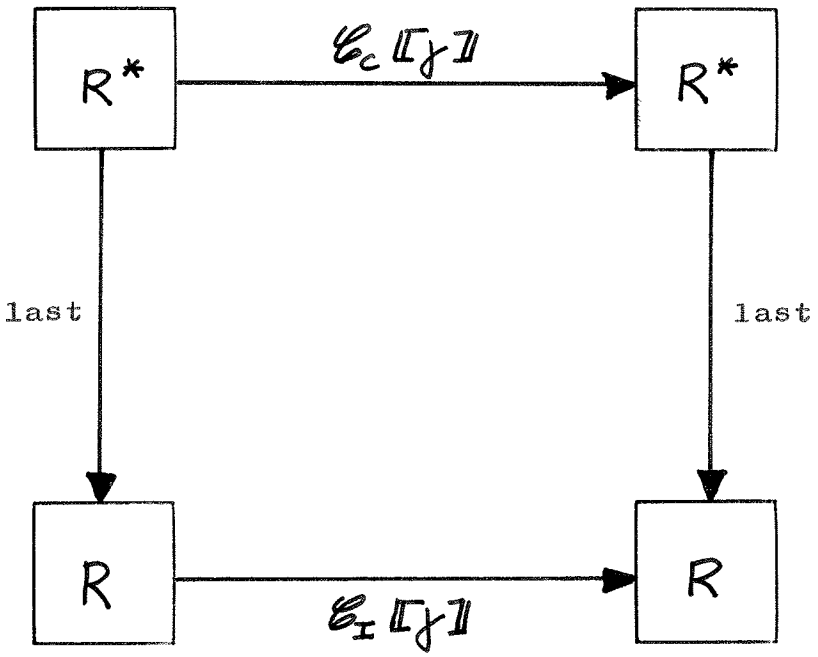
In the original formulation of Hoare and Lauer equivalence is expressed by

(Theorem 1) $\quad last \circ comp = result$.

In the new notation the corresponding theorem will be:

Theorem 4: $last(\mathcal{C}_c[\![f]\!]\Psi) = \mathcal{C}_I[\![f]\!]last(\Psi)$

which very informally can be said to illustrate, that mathematical semantics commute with the function last. It is the same as saying that the following diagram commutes for all $f \in COM$:



5.2.2 Proof for Theorem 4:

The proof is done using induction after the complexity of γ. This is called structural induction.

Command lists

$$
\begin{aligned}
& last(\mathcal{C}_c[\![\gamma_1;\gamma_2]\!]\Psi) \\
= \; & last(\mathcal{C}_c[\![\gamma_2]\!] \circ \mathcal{C}_c[\![\gamma_1]\!]\Psi) \\
= \; & last(\mathcal{C}_c[\![\gamma_2]\!](last(\mathcal{C}_c[\![\gamma_1]\!]\Psi)))
\end{aligned}
$$

where it is used that $\mathcal{C}_c$ only appends to its argument list and that the appended elements only depends on the last element of the argument list

$$
\begin{aligned}
= \; & last(\mathcal{C}_c[\![\gamma_2]\!](\mathcal{C}_I[\![\gamma_1]\!]last\,\Psi)) \\
= \; & \mathcal{C}_I[\![\gamma_2]\!](last(\mathcal{C}_I[\![\gamma_1]\!]last\,\Psi)) \\
= \; & \mathcal{C}_I[\![\gamma_2]\!](\mathcal{C}_I[\![\gamma_1]\!]last\,\Psi) \\
= \; & \mathcal{C}_I[\![\gamma_1;\gamma_2]\!]last(\Psi)
\end{aligned}
$$

Assignment

$$
\begin{aligned}
& last(\mathcal{C}_c[\![\xi := \varepsilon]\!]\Psi) \\
= \; & last(\Psi; assign_c(\xi, \mathcal{E}_c[\![\varepsilon]\!]\Psi)\Psi) \\
= \; & assign_c(\xi, \mathcal{E}_c[\![\varepsilon]\!]\Psi)\Psi \\
= \; & assign_I(\xi, \mathcal{E}_I[\![\varepsilon]\!]last\,\Psi)last\,\Psi \\
= \; & \mathcal{C}_I[\![\xi := \varepsilon]\!]last\,\Psi
\end{aligned}
$$

IF-command

$$last(\mathcal{C}_c[\![IF\ \varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n\ FI]\!]\Psi)$$

$$= last([\mathcal{E}_c[\![\varepsilon_1]\!]\Psi \to \mathcal{C}_c[\![\gamma_1]\!]\Psi, \ \vdots \ \mathcal{E}_c[\![\varepsilon_n]\!]\Psi \to \mathcal{C}_c[\![\gamma_n]\!]\Psi])$$

$$= [\mathcal{E}_c[\![\varepsilon_1]\!]\Psi \to last(\mathcal{C}_c[\![\gamma_1]\!]\Psi), \ \vdots \ \mathcal{E}_c[\![\varepsilon_n]\!]\Psi \to last(\mathcal{C}_c[\![\gamma_n]\!]\Psi)]$$

$$= [\mathcal{E}_I[\![\varepsilon_1]\!]last(\Psi) \to \mathcal{C}_I[\![\gamma_1]\!]last(\Psi), \ \vdots \ \mathcal{E}_I[\![\varepsilon_n]\!]last(\Psi) \to \mathcal{C}_I[\![\gamma_n]\!]last(\Psi)]$$

$$= \mathcal{C}_I[\![IF\ \varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n\ FI]\!]last(\Psi)$$

WHILE-command

$$last(\mathcal{C}_c[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\Psi)$$

$$= last((FIX\ F.(\lambda\Psi.\mathcal{E}_c[\![\varepsilon]\!]\Psi \to F\circ\mathcal{C}_c[\![\gamma]\!]\Psi, \Psi))\Psi)$$

$$= last((FIX\ F.(\lambda\Psi.\mathcal{E}_I[\![\varepsilon]\!](last(\Psi) \to F\circ\mathcal{C}_c[\![\gamma]\!]\Psi, \Psi))\Psi)$$

$$= last((FIX\ F.(\lambda\Psi.\mathcal{E}_I[\![\varepsilon]\!](last\ \Psi) \to F\circ(last(\mathcal{C}_c[\![\gamma]\!]\Psi), last\ \Psi))\Psi)$$

$$= last((FIX\ F.(\lambda\Psi.\mathcal{E}_I[\![\varepsilon]\!](last\ \Psi) \to F\circ\mathcal{C}_I[\![\gamma]\!](last\ \Psi), last\ \Psi))\Psi)$$

$$= last(FIX\ F.(\lambda\psi.\mathcal{E}_I[\![\varepsilon]\!]\psi \to F\circ\mathcal{C}_I[\![\gamma]\!]\psi, \psi))\Psi)$$

$$= (FIX\ F.(\lambda\psi.\mathcal{E}_I[\![\varepsilon]\!]\psi \to F\circ\mathcal{C}_I[\![\gamma]\!]\psi, \psi))last(\Psi)$$

$$= \mathcal{C}_I[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]last(\Psi)$$

with the same remark as for command lists.

Declarations

$$
\begin{aligned}
& last\,(\mathscr{C}_c[\![\,BEGIN\ DEF\ \xi = \varepsilon;\gamma\ END\,]\!]\,\Psi) \\
= \ & last\,(\,LET\ \xi' = New_c(\Psi)\ IN \\
& \quad (\mathscr{C}_c[\![\,\xi' := \varepsilon;\gamma(\xi'/\xi)]\!]\,\Psi)\,) \\
= \ & LET\ \xi' = New_c(\Psi)\ IN \\
& \quad (last\,(\mathscr{C}_c[\![\,\xi' := \varepsilon;\gamma(\xi'/\xi)]\!]\,\Psi)) \\
= \ & LET\ \xi' = New_I(last\,\Psi)\ IN \\
& \quad (\mathscr{C}_I[\![\,\xi' := \varepsilon;\gamma(\xi'/\xi)]\!]\,last(\Psi)) \\
= \ & LET\ \xi' = New_I(last\,\Psi)\ IN \\
& \quad (\mathscr{C}_I[\![\,\xi' := \varepsilon;\gamma(\xi'/\xi)]\!]\,last(\Psi)) \\
= \ & \mathscr{C}_I[\![\,BEGIN\ DEF\ \xi = \varepsilon;\gamma\ END\,]\!]\,last\,\Psi
\end{aligned}
$$

end of proof
for Theorem 4

We have the following connections between the two notations:

$$(*)\quad \begin{cases} \mathcal{C}_I[\![\gamma]\!]\,last(\Psi) \\ = result(last(\Psi), \gamma) \end{cases}$$

and

$$(**)\quad \begin{cases} last(\mathcal{C}_c[\![\gamma]\!]\,\Psi) \\ = last(\Psi;\ tail(\mathcal{C}_A[\![\gamma]\!]\,last(\Psi))) \\ = last(\Psi;\ tail(comp(last(\Psi), \gamma))) \\ = last \circ comp(last(\Psi), \gamma) \end{cases}$$

where it is used that:

1. comp(ψ,γ) has at least one element.
2. If comp(ψ,γ) has exactly one element, then this element is ψ.

It is now easy to verify:

Theorem 5: Theorem 1 and Theorem 4 are equivalent.

Proof: Assume Theorem 1- then Theorem 4 follows directly from (*) and (**).

Assume Theorem 4 - then Theorem follows from (*) and (**) since the set of all single-element lists is a subset of the range for the function $last \in [R^* \to R]$.

end of proof
for Theorem 5

This give an alternative proof for Theorem 4: Prove Theorem 1 - use Theorem 5.

An alternative proof for Theorem 1: Prove Theorem 4 -use Theorem 5.

5.3 Relational Theory

Hoare and Lauer interprets this theory in terms of the computational model:

$$\psi(R_\gamma)\psi' \equiv \psi' = \text{last} \circ \text{comp}(\psi, \gamma) \ .$$

Using that

$$\text{result} = \text{last} \circ \text{comp}$$

and

$$\mathcal{E}_I[\![\gamma]\!]\psi = \text{result}(\psi, \gamma)$$

we get an interpretational of the relational theory directly in terms of $\mathcal{E}_I$:

$$\psi(R_\gamma)\psi' \equiv \psi' = \mathcal{E}_I[\![\gamma]\!]\psi \ .$$

This is a very natural way to describe in mathematical semantics the relation connecting initial and final states for execution of a given command, γ.

We can now prove the theorem corresponding to Theorem 2:

Theorem 6: Relational theory is satisfied by the mathematical semantic model represented by $\mathcal{E}_I$.

5.3.1 Proof for Theorem 6:

A1 - Command lists

$$
\begin{aligned}
&\Updownarrow \quad \psi\,(\gamma_1;\gamma_2)\,\psi' \\
&\Updownarrow \quad \psi' = \mathcal{E}_I[\![\gamma_1;\gamma_2]\!]\,\psi \\
&\Updownarrow \quad \psi' = \mathcal{E}_I[\![\gamma_2]\!](\mathcal{E}_I[\![\gamma_1]\!]\,\psi) \\
&\Updownarrow \quad \exists\psi''\,[\,\psi'' = \mathcal{E}_I[\![\gamma_1]\!]\,\psi \wedge \psi' = \mathcal{E}_I[\![\gamma_2]\!]\,\psi''\,] \\
&\Updownarrow \quad \exists\psi''\,[\,\psi\,(\gamma_1)\,\psi'' \wedge \psi''\,(\gamma_2)\,\psi'\,]
\end{aligned}
$$

A2 - Assignment

$$
\begin{aligned}
&\Updownarrow \quad \psi\,(\xi := \varepsilon)\,\psi' \\
&\Updownarrow \quad \psi' = \mathcal{E}_I[\![\xi := \varepsilon]\!]\,\psi \\
&\Updownarrow \quad \psi' = assign_I(\xi, \mathcal{E}_I[\![\varepsilon]\!]\,\psi)\,\psi
\end{aligned}
$$

A4 - IF-command

$$
\begin{aligned}
&\Updownarrow \quad \psi\,(IF\ \varepsilon_1 \to \gamma_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to \gamma_n\ FI)\,\psi' \\
&\Updownarrow \quad \psi' = \mathcal{E}_I[\![IF\ \varepsilon_1 \to \gamma_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to \gamma_n\ FI]\!]\,\psi \\
&\Updownarrow \quad \psi' = [\,\mathcal{E}_I[\![\varepsilon_1]\!]\,\psi \to \mathcal{E}_I[\![\gamma_1]\!]\,\psi, \\
&\qquad\qquad \vdots \\
&\qquad\qquad \mathcal{E}_I[\![\varepsilon_n]\!]\,\psi \to \mathcal{E}_I[\![\gamma_n]\!]\,\psi\,] \\
&\Updownarrow \quad B_\psi \neq \emptyset \wedge \psi' = \mathcal{E}_I[\![\gamma_{min\,B_\psi}]\!]\,\psi \\
&\Updownarrow \quad B_\psi \neq \emptyset \wedge \psi\,(\gamma_{min\,B_\psi})\,\psi'
\end{aligned}
$$

A7 - A8 - WHILE-command

In both cases we start with the following rewriting:

$$
\begin{array}{ll}
 & \psi\ (\mathit{WHILE}\ \varepsilon\ \mathit{DO}\ \gamma\ \mathit{OD})\ \psi' \\
\Updownarrow & \\
 & \psi' = \mathcal{E}_I[\![\mathit{WHILE}\ \varepsilon\ \mathit{DO}\ \gamma\ \mathit{OD}]\!]\,\psi \\
\Updownarrow & \\
 & \psi' = (\mathit{FIX}\ F.\ (\lambda\psi.\ \mathcal{E}_I[\![\varepsilon]\!]\psi \to F \circ \mathcal{E}_I[\![\gamma]\!]\psi, \psi))\ \psi \\
\Updownarrow & \\
(*) & \psi' = (\mathcal{E}_I[\![\varepsilon]\!]\psi \to \mathcal{E}_I[\![\mathit{WHILE}\ \varepsilon\ \mathit{DO}\ \gamma\ \mathit{OD}]\!] \circ \mathcal{E}_I[\![\gamma]\!]\psi, \psi)
\end{array}
$$

A7:

The proof is now finished using induction after the number of times, n, the loop-command γ is executed.

n = 0: Then by (*) we have $\mathcal{E}_I[\![\varepsilon]\!]\psi$ = false and $\psi' = \psi$ which immediately gives $\neg\, \varepsilon_{\psi'}$.

n > 1: Let ψ be a fixed configuration.
Let $\psi^+ = \mathcal{E}_I[\![\gamma]\!]\psi$ and let ψ' be as above.

By (*) we deduce that $\psi' = \mathcal{E}_I[\![\mathit{WHILE}\ \varepsilon\ \mathit{DO}\ \gamma\ \mathit{OD}]\!]\,\psi^+$ and since $\mathcal{E}_I[\![\mathit{WHILE}\ \varepsilon\ \mathit{DO}\ \gamma\ \mathit{OD}]\!]\,\psi^+$ executes the loop-command, γ, one times less than $\mathcal{E}_I[\![\mathit{WHILE}\ \varepsilon\ \mathit{DO}\ \gamma\ \mathit{OD}]\!]\,\psi$ does, the inductional hypothesis gives the desired $\neg\, \varepsilon_{\psi'}$.

A8:

Again the proof is done using induction after the number of times, n, the loop-command γ is executed.

We first write A8 in our current interpretation:

$$
\begin{array}{l}
\forall\psi\psi'\,[\,I(\psi) \wedge \varepsilon_\psi \wedge \psi' = \mathcal{E}_I[\![\gamma]\!]\psi \Rightarrow I(\psi')\,] \Longrightarrow \\
\forall\psi\psi'\,[\,I(\psi) \wedge \psi' = \mathcal{E}_I[\![\mathit{WHILE}\ \varepsilon\ \mathit{DO}\ \gamma\ \mathit{OD}]\!]\,\psi \Rightarrow I(\psi')\,]
\end{array}
$$

Assume: 1. The antecedent of A8:

$$\forall \psi \psi^{+} [\, I(\psi) \wedge \mathcal{E}_{\psi} \wedge \psi^{+} = \mathcal{C}_I[\![\gamma]\!]\psi \Rightarrow I(\psi^{+}) \,]$$

2. $I(\psi) \wedge \psi' = \mathcal{C}_I[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\psi$.

n = 0: By (*) $\psi' = \psi$ which imediately gives $I(\psi')$.

n > 1: Let ψ be a fixed configuration.
Let $\psi^{+} = \mathcal{C}_I[\![\gamma]\!]\psi$
and $\psi' = \mathcal{C}_I[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\psi$.

From (*) and from assumption 2 follows that the antecedent of assumption 1 is satisfied (with $\psi = \psi$ and $\psi^{+} = \psi^{+}$).

Assumption 1 then gives $I(\psi^{+})$.

From (*) follows that $\psi' = \mathcal{C}_I[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\psi^{+}$ and since $\mathcal{C}_I[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\psi^{+}$ executes the loop-command, γ, one times less than $\mathcal{C}_I[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\psi$ does, the inductional hypothesis gives the desired $I(\psi')$.

A9 - Declaration

$$\Updownarrow\ \psi\ (\ BEGIN\ DEF\ \xi = \varepsilon ; \gamma\ END\ )\ \psi'$$

$$\Updownarrow\ \psi' = \mathcal{C}_I[\![BEGIN\ DEF\ \xi = \varepsilon ; \gamma\ END]\!]\psi$$

$$\Updownarrow\ \psi' = LET\ \xi' = New_I(\psi)\ IN\ (\ \mathcal{C}_I[\![\xi' := \varepsilon ; \gamma(\xi'/\xi)]\!]\psi\ )$$

$$\Updownarrow\ LET\ \xi' = New_I(\psi)\ IN\ (\ \psi' = \mathcal{C}_I[\![\xi' := \varepsilon ; \gamma(\xi'/\xi)]\!]\psi\ )$$

$$\Updownarrow\ LET\ \xi' = New_I(\psi)\ IN\ (\ \psi\ (\xi' := \varepsilon ; \gamma(\xi'/\xi))\ \psi'\ )$$

$$\Updownarrow\ \exists \xi' [\ \xi' = New_I(\psi) \wedge \psi\ (\xi' := \varepsilon ; \gamma(\xi'/\xi))\ \psi'\ ]$$

where it should be remembered that we have made the following identification $\psi = augment(\psi, \xi')$. (see page 50).

end of proof
for Theorem 6

5.4 Deductive Theory

At this stage we have two possible dirrections for progress. The first possibility is to repeat the interpretation of the dedeuctive theory in terms of the relational theory. Since this interpretation:

$$(*) \quad P\{\gamma\}Q \equiv \forall\psi\psi'[P(\psi) \wedge \psi(\gamma)\psi' \Rightarrow Q(\psi')]$$

does not involve result, next or comp it remains unaltered in the new notation of mathematical semantics and Theorem 3 can be repeated without any changes.

The other possibility is to interpret the deductive theory directly in terms of $\mathscr{E}_I$. From $(*)$ and from

$$(**) \quad \psi(\gamma)\psi' \equiv \psi' = \mathscr{E}_I[\![\gamma]\!]\psi$$

we get the interpretation

$$P\{\gamma\}Q \equiv \forall\psi[P(\psi) \wedge \mathscr{E}_I[\![\gamma]\!]\psi \text{ defined} \Rightarrow Q(\mathscr{E}_I[\![\gamma]\!]\psi)]$$

or

$$P\{\gamma\}Q \equiv \forall\psi[P(\psi) \Rightarrow Q(\mathscr{E}_I[\![\gamma]\!]\psi)]$$

where we use the convention, that $Q(\perp)$ = true for any predicate, Q.

We can now prove a theorem corresponding to the corollary of Theorem 3:

Theorem 7: Deductive theory is satisfied by the mathematical semantic model represented by $\mathscr{E}_I$.

5.4.1 Proof for Theorem 7:

D1 - Command lists

$$
\begin{array}{ll}
\Updownarrow & P\{f_1\}Q \wedge Q\{f_2\}R \\
\Updownarrow & \forall\psi\,[\,P(\psi) \Rightarrow Q(\mathcal{E}_I[\![f_1]\!]\psi)\,] \wedge \\
 & \forall\psi\,[\,Q(\psi) \Rightarrow R(\mathcal{E}_I[\![f_2]\!]\psi)\,] \\
\Downarrow & \\
 & \forall\psi\,[\,P(\psi) \Rightarrow R(\mathcal{E}_I[\![f_2]\!](\mathcal{E}_I[\![f_1]\!]\psi))\,] \\
\Updownarrow & \forall\psi\,[\,P(\psi) \Rightarrow R(\mathcal{E}_I[\![f_1;f_2]\!]\psi)\,] \\
\Updownarrow & P\{f_1;f_2\}R
\end{array}
$$

D2 - Assignment

$$
\begin{array}{ll}
\Updownarrow & P(\varepsilon/\xi)\{\xi := \varepsilon\}P \\
\Updownarrow & \forall\psi\,[\,P(\varepsilon/\xi)(\psi) \Rightarrow P(\mathcal{E}_I[\![\xi := \varepsilon]\!]\psi)\,] \\
\Updownarrow & \forall\psi\,[\,P(\varepsilon/\xi)(\psi) \Rightarrow P(assign_I(\xi, \mathcal{E}_I[\![\varepsilon]\!]\psi)\psi)\,] \\
\Updownarrow & \forall\psi\,[\,P(\varepsilon/\xi)(\psi) \Rightarrow P(\varepsilon/\xi)(\psi)\,] \\
\Updownarrow & \text{true}
\end{array}
$$

D4 - IF-command

$$
\begin{array}{ll}
\Updownarrow & \forall i \le n\,[\,(P \wedge \varepsilon_i)\{f_i\}Q\,] \\
\Updownarrow & \forall\psi\ \forall i \le n\,[\,P(\psi) \wedge (\varepsilon_i)_\psi \Rightarrow Q(\mathcal{E}_I[\![f_i]\!]\psi)\,] \\
\Updownarrow & \forall\psi\ \forall i \le n\,[\,P(\psi) \wedge \mathcal{E}_I[\![\varepsilon_i]\!]\psi \Rightarrow Q(\mathcal{E}_I[\![f_i]\!]\psi)\,] \\
\Downarrow & \forall\psi\,[\,P(\psi) \Rightarrow Q([\mathcal{E}_I[\![\varepsilon_1]\!]\psi \to \mathcal{E}_I[\![f_1]\!]\psi, \\
 & \qquad\qquad \vdots \\
 & \qquad\qquad \mathcal{E}_I[\![\varepsilon_n]\!]\psi \to \mathcal{E}_I[\![f_n]\!]\psi])\,] \\
\Updownarrow & \\
\Updownarrow & \forall\psi\,[\,P(\psi) \Rightarrow Q(\mathcal{E}_I[\![\text{IF } \varepsilon_1 \to f_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to f_n \text{ FI}]\!]\psi)\,] \\
 & P\{\text{IF } \varepsilon_1 \to f_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to f_n \text{ FI}\}Q
\end{array}
$$

D6 - WHILE-command

$$
\begin{array}{ll}
 & (P \wedge \varepsilon)\{\gamma\}P \\
\Updownarrow & \\
 & \forall\psi\,[\,(P\wedge\varepsilon)\psi \Rightarrow P(\mathcal{E}_I[\![\gamma]\!]\psi)\,] \\
(*)\ \Downarrow & \\
 & \forall\psi\,[\,P(\psi) \Rightarrow (P\wedge\neg\varepsilon)(\mathcal{E}_I[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\psi)\,] \\
\Updownarrow & \\
 & P\{\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}\}(P\wedge\neg\varepsilon)
\end{array}
$$

where (*) is proved by computational induction. This proof-technique is described in Manna and Vuillemin /M/ page 53o.

Proof for (*):

Assume: $\forall\psi\,[(P\wedge\varepsilon)\psi \Rightarrow P(\mathcal{E}_I[\![\gamma]\!]\psi)\,]$

Since $\mathcal{E}_I[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]$

$= FIX\ F.(\lambda\psi.\,\mathcal{E}_I[\![\varepsilon]\!]\psi \rightarrow F\circ\mathcal{E}_I[\![\gamma]\!]\psi, \psi)$

it is enough to prove

(I) $\forall\psi\,[\,P(\psi) \Rightarrow (P\wedge\neg\varepsilon)(\bot)$

and (II) $\forall\Theta\,[\,\forall\psi\,[\,P(\psi) \Rightarrow (P\wedge\neg\varepsilon)\Theta\psi\,]$

$\Rightarrow \forall\psi\,[\,P(\psi) \Rightarrow (P\wedge\neg\varepsilon)(\mathcal{E}_I[\![\varepsilon]\!]\psi \rightarrow \Theta\circ\mathcal{E}_I[\![\gamma]\!]\psi, \psi)]]$

(It should also be proved that the used predicate is "admissible" - see /M/ - but this is omitted).

(I):

This follows immediately from the convention $P(\bot)$=true for all predicates, P.

(II):

Let $\theta \in [R \to R]$ be given

Assume: 1. $\forall \psi' [(P \wedge \varepsilon) \psi' \Rightarrow P(\mathcal{E}_I [\![f]\!] \psi')]$

2. $\forall \psi' [P\psi' \Rightarrow (P \wedge \neg\varepsilon) \theta \psi']$

Prove: $\forall \psi [P\psi \Rightarrow (P \wedge \neg\varepsilon)(\mathcal{E}_I [\![\varepsilon]\!] \psi \to \theta \circ \mathcal{E}_I [\![f]\!] \psi, \psi)]$

Let $\psi \in R$ be given. There are 3 possibilities:

<u>case 1: $\mathcal{E}_I [\![\varepsilon]\!] \psi = \bot$</u>

$$\Updownarrow \quad \begin{array}{l} P(\psi) \Rightarrow (P \wedge \neg\varepsilon)(\mathcal{E}_I [\![\varepsilon]\!] \psi \to \theta \circ \mathcal{E}_I [\![f]\!] \psi, \psi) \\ P(\psi) \Rightarrow (P \wedge \neg\varepsilon)(\bot) \end{array}$$

which follows from the convention $P(\bot)$=true for all predicates, P.

<u>case 2: $\mathcal{E}_I [\![\varepsilon]\!] \psi$ = false</u>

$$\begin{array}{l} \Updownarrow \\ \Updownarrow \end{array} \quad \begin{array}{l} P(\psi) \Rightarrow (P \wedge \neg\varepsilon)(\mathcal{E}_I [\![\varepsilon]\!] \psi \to \theta \circ \mathcal{E}_I [\![f]\!] \psi, \psi) \\ P(\psi) \Rightarrow (P \wedge \neg\varepsilon)\, \psi \\ P(\psi) \Rightarrow P(\psi) \wedge \neg(\mathcal{E}_I [\![\varepsilon]\!] \psi) \end{array}$$

which is trivially true.

case 3: $\mathcal{E}_I[\![\varepsilon]\!]\psi = \text{true}$

$$\Updownarrow \quad P(\psi) \Rightarrow (P \wedge \neg\varepsilon)(\mathcal{E}_I[\![\varepsilon]\!]\psi \rightarrow \Theta \circ \mathcal{E}_I[\![\gamma]\!]\psi, \psi)$$
$$P(\psi) \Rightarrow (P \wedge \neg\varepsilon)(\Theta \circ \mathcal{E}_I[\![\gamma]\!]\psi)$$

Assuming $P(\psi)$ assumption 1 gives:

$$P(\mathcal{E}_I[\![\gamma]\!]\psi)$$

Assumption 2 then gives:

$$(P \wedge \neg\varepsilon)(\Theta \circ \mathcal{E}_I[\![\gamma]\!]\psi)$$

(use $\psi' = \mathcal{E}_I[\![\gamma]\!]\psi$).

end of proof
for (*)

D7 - Declaration

$$\Updownarrow \quad P\{\xi' := \varepsilon;\ \gamma(\xi'/\xi)\}Q \ \wedge\ \xi' \text{ new unused identifier}$$
$$\Updownarrow \quad \forall\psi\ \exists\xi'\ [\ \xi' = New_I(\psi) \wedge P(\psi) \Rightarrow Q(\mathcal{E}_I[\![\xi' := \varepsilon;\ \gamma(\xi'/\xi)]\!]\psi)\ ]$$
$$\Updownarrow \quad \forall\psi\ [\ P(\psi) \Rightarrow LET\ \xi' = New_I(\psi)\ IN\ (Q(\mathcal{E}_I[\![\xi' := \varepsilon;\ \gamma(\xi'/\xi)]\!]\psi))\ ]$$
$$\Updownarrow \quad \forall\psi\ [\ P(\psi) \Rightarrow Q\ (LET\ \xi' = New_I(\psi)\ IN\ (\mathcal{E}_I[\![\xi' := \varepsilon;\ \gamma(\xi'/\xi)]\!]\psi))]$$
$$\Updownarrow \quad \forall\psi\ [\ P(\psi) \Rightarrow Q\ (\mathcal{E}_I[\![BEGIN\ DEF\ \xi = \varepsilon;\ \gamma\ END]\!]\psi)\ ]$$
$$P\{BEGIN\ DEF\ \xi = \varepsilon;\ \gamma\ END\}Q$$

where it should be noticed, that P and Q cannot involve ξ' since they are defined outside the scope of ξ'.

end of proof
for Theorem 7.

On this place it is natural to make a short pause for doing some comments about the work done in /D3/ by Donahue and in /L2/ by Ligler.

They both define a deductive theory in the style suggested by Hoare. They introduce formally a mathematical model - interpret the theory in this model - and show that the theory is satisfied by the model. It should be stressed that none of them make the full use of command-continuations. Ligler explicitly removes continuations as far as possible (no jumps). Donahue uses an interpretation, where command-continuations are universally quantified (see /D3/ page 359). This amounts to the same as an (implicit) continuation-removal.

Therefore /D3/ and /L2/ can be viewed as analogous to the interpretation of the deductive theory in terms of the mathematical semantics represented by $\mathscr{E}_I$ (Theorem 7).

5.5 Predicate-transformer Theory

In /D/ Dijkstra presents a Theory called weakest predicate-transformers. Its ideas are very close to those represented by Hoares Deductive Theory, but there are 2 main differences:

1. Dijkstra demands total correctness while Hoare uses partial correctness.

2. Dijkstra wants a sufficient and nescessary precondition while Hoare only demands sufficiency.

Dijkstra's predicate-transformer called wp (weakest predicate) is a member of $[\text{COM} \times \text{PRED} \rightarrow \text{PRED}]$, where PRED is the domain of first order predicates with free variables in the domains of identifiers, inputs and outputs. The intuitive meaning of wp will be:

> wp(γ,Q) is the weakest precondition (predicate) which guarantees that execution of the command, γ, stops with the predicate Q satisfied.

Dijkstra imposes 4 restrictions on the possible wp-functions:

1. $wp(\gamma,false) = false$ (excluded miracle)
2. $(P \Rightarrow Q) \Rightarrow (wp(\gamma,P) \Rightarrow wp(\gamma,Q))$
3. $wp(\gamma,P \wedge Q) = wp(\gamma,P) \wedge wp(\gamma,Q)$
4. $wp(\gamma,P \vee Q) = wp(\gamma,P) \vee wp(\gamma,Q)$

A wp-function satisfying 1-4 is called "healthy". If nondeterministic commands are considered "=" must be replaced by "$\Leftarrow$" in 4.

It should be noticed that

$$wp(\gamma,true)$$

is the predicate telling iff γ stops.

5.5.1 Predicate-transformer Theory used on SNAIL

wp 1 $\quad wp(f_1; f_2, Q) \equiv wp(f_1, wp(f_2, Q))$

wp 2 $\quad wp(\xi := \varepsilon, Q) \equiv Q(\varepsilon/\xi)$

wp 3 $\quad wp(\text{TEST } \varepsilon \text{ DO } f_1 \text{ OR } f_2 \text{ TSET}, Q)$

$\equiv (\varepsilon \wedge wp(f_1, Q)) \vee (\neg\varepsilon \wedge wp(f_2, Q))$

wp 4 $\quad wp(\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}, Q) \equiv B \neq \varnothing \wedge wp(f_{\min B}, Q)$

where $\quad B = \{ i \leq n \mid \varepsilon_i \}$

wp 5 $\quad wp(\text{DO } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ OD}, Q) \equiv (\exists i \geq 0 : H_i(Q))$

where $\quad H_0(Q) \equiv (B = \varnothing) \wedge Q$

and $\quad H_i(Q) \equiv wp(\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}, H_{i-1}(Q)) \vee H_0(Q) \qquad (i \geq 1)$

wp6 $\quad wp(\text{WHILE } \varepsilon \text{ DO } f \text{ OD}, Q) \equiv (\exists i \geq 0 : K_i(Q))$

where $\quad K_0(Q) \equiv (\neg\varepsilon) \wedge Q$

and $\quad K_i(Q) \equiv wp(\text{TEST } \varepsilon \text{ DO } f \text{ OR } e \text{ TSET}, K_{i-1}(Q)) \qquad (i \geq 1)$

wp7 $\quad wp(\text{BEGIN DEF } \xi = \varepsilon; f \text{ END}, Q)$

$\equiv \exists k \in V \, [\, \xi = k \wedge wp(\xi := \varepsilon; f, Q(k/\xi)) \,]$

wp8 $\quad wp(\text{BEGIN } e; f \text{ END}, Q) \equiv wp(f, Q)$

wp 9 $\quad wp(\text{READ}(\xi), Q) \equiv Q((\mu_I \downarrow 1), (\mu_I + 2)/\xi, \mu_I)$

wp 10 $\quad wp(\text{WRITE}(\varepsilon), Q) \equiv Q((\mu_0 @ \varepsilon)/\mu_0)$

wp 11 $\quad wp(e, Q) \equiv Q$

Our choice of selection rules for guards evaluated to true influence wp4 and wp5. If we had specified a rule selecting at random between these true guards our weakest preconditions would have been a little stronger:

$$wp4' \quad wp(\mathrm{IF}\ \varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n\ \mathrm{FI}, Q) \equiv B \neq \varnothing \ \wedge (\forall i : \varepsilon_i \Rightarrow wp(\gamma_i, Q))$$

wp5 would be as before, but since it refers to wp4 its semantic meaning would be altered too.

These new rules for IF- and DO-commands corresponds with Dijkstra's suggestions in /D1/ and /D2/.

In wp9 and wp10 μ_I and μ_o refer to the input- and output-part of the configuration on which the predicate is evaluated. In wp9 the presence of the term $\mu_I \downarrow 1$ implies that the input-part of the configuration before execution must have dimension greater than zero. (If this was not the case $\mu_I \downarrow 1$ would be undefined and the whole predicate would then yield false by convention).

It is easy to verify that wp1 - wp11 constitutes a <u>healthy</u> predicate-transformer.

wp1 - wp11 constitutes a theory. This can be interpreted in terms of mathematical semantics:

$$wp(\gamma, Q)\psi \equiv Q\,(\mathscr{E}_I[\![\gamma]\!]\psi)$$

where $Q(\perp)$=false for all predicates, Q.

It can now be shown:

<u>Theorem 8</u>: The predicate-transformer Theory is satisfied by the mathematical semantic model represented by $\mathscr{E}_I$.

The proof for Theorem 8 will be omitted. Later we shall investigate in greater details the situation where this theory is interpreted in terms of a mathematical <u>continuation</u>-semantics. (see page 86-93).

5.6 Continuations

Let $C = [R \to A]$. Then a mathematical semantics without continuations

$$\mathscr{E}_1 \in [COM \to [R \to R]]$$

can be lifted to a mathematical semantics involving continuations

$$\mathscr{E}_1' \in [COM \to [C \to C]]$$

defined by

$$\mathscr{E}_1' [\![\gamma]\!] \theta = \theta \circ \mathscr{E}_1 [\![\gamma]\!] .$$

At the other hand given a mathematical semantics with continations

$$\mathscr{E}_2 \in [COM \to [C \to C]]$$

we can remove the continations by defining

$$\mathscr{E}_2' \in [COM \to [R \to R]]$$

where

$$\mathscr{E}_2' [\![\gamma]\!] = \mathscr{E}_2 [\![\gamma]\!] \theta_{ID}$$
$$\theta_{ID} = \lambda\psi.\psi .$$

For a further discussion of continuation-removal see Donahue /D3/ and Ligler /L2/.

I now want to investigate our different theories in terms of a a mathematical semantics involving continuations.

Let $\mathscr{E}_{IC} \in [COM \to [C \to C]]$ be defined by

$$\mathscr{E}_{IC} [\![\gamma]\!] \theta = \theta \circ \mathscr{E}_I [\![\gamma]\!] .$$

5.6.1 Relational Theory

The interpretation

$$\psi\,(R_\gamma)\,\psi' \equiv \quad \psi' = \mathscr{E}_I\,[\![\gamma]\!]\,\psi$$

now becomes

$$\psi\,(R_\gamma)\,\psi' \equiv \quad \psi' = \mathscr{E}_{IC}\,[\![\gamma]\!]\,\theta_{ID}\,\psi$$

which adds very little to the clarity and understanding of this theory.

5.6.2 Deductive Theory

Omitting continuations we used the interpretation

$$P\{f\}Q \equiv \forall\psi\,[P(\psi) \wedge \mathcal{E}_I[\![f]\!]\psi \text{ defined} \Rightarrow Q(\mathcal{E}_I[\![f]\!]\psi)] .$$

The predicates considered in this theory is contained in the domain

$$\text{PRED} = [\text{R} \to \text{TF}]$$

where the domain TF of boolean values is the set {true, false} equipped with the partial ordering false $\sqsubseteq$ true .

Assume that the answer domain, A, contains TF. Then

$$\text{PRED} \subseteq \text{C}$$

and we can use the interpretation:

$$P\{f\}Q \equiv \forall\psi\,[P(\psi) \wedge \mathcal{E}_{IC}[\![f]\!]\theta_{TT}\psi \text{ defined} \Rightarrow \mathcal{E}_{IC}[\![f]\!]Q\psi]$$

where $\theta_{TT} = \lambda\psi.\ true$

This can be rewritten to

$$\begin{aligned}
&P\{f\}Q \equiv \forall\psi\,[P(\psi) \Rightarrow (\neg(\mathcal{E}_{IC}[\![f]\!]\theta_{TT}\psi \text{ defined}) \vee \mathcal{E}_{IC}[\![f]\!]Q\psi]\\
\Updownarrow\quad &P\{f\}Q \equiv \forall\psi\,[P(\psi) \Rightarrow (\neg(\mathcal{E}_{IC}[\![f]\!]Q\psi \text{ defined}) \vee \mathcal{E}_{IC}[\![f]\!]Q\psi]\\
\Updownarrow\quad &P\{f\}Q \equiv \forall\psi\,[P(\psi) \Rightarrow (\neg(\mathcal{E}_{IC}[\![f]\!](\neg Q)\psi))]
\end{aligned}$$

since false is the bottom (undefined) element of domain TF).

At last we receive our final interpretation:

$$P\{f\}Q \equiv \forall\psi\,[\,\mathcal{E}_{IC}[\![f]\!](\neg Q)\psi \Rightarrow \neg(P(\psi))\,] .$$

5.6.3 Predicate-transformer Theory

The use of continuations becomes much more convenient, when we consider this theory.

Omitting continuations we used the interpretation

$$wp(\gamma, Q)\psi \equiv Q(\mathcal{C}_I[\![\gamma]\!]\psi)$$

with the convention $Q(\perp)$=false for all predicates, Q.

Using continuations we get

$$wp(\gamma, Q)\psi \equiv \mathcal{C}_{IC}[\![\gamma]\!]Q\psi$$

or

$$wp(\gamma, Q) = \mathcal{C}_{IC}[\![\gamma]\!]Q$$

where $Q(\perp)$=false merely reflects the fact that false is chosen as the bottom-element in TF.

Thus we have established a connection between the two functions:

$$\text{wp}: [[\text{COM} \times \text{PRED}] \to \text{PRED}]$$
$$\mathcal{C}_{IC}: [\text{COM} \to [\text{C} \to \text{C}]]$$

where it should be remembered that $\text{PRED} \subseteq \text{C}$.

This gives a much more elegant interpretation of DIJKSTRA's theory for predicate-transformers. Moreover it gives a connection between program-execution represented by $\mathcal{C}_{IC}$ and program-verification represented by wp.

6 PREDICATE-TRANSFORMERS AND CONTINUATION SEMANTICS

In this section I shall elaborate the ideas brought forward on page 79 .

I shall not use the semantic function $\mathcal{E}_{IC}$, but instead the semantic function $\mathcal{E}$ defined on page 15-22.

$\mathcal{E}$ and $\mathcal{E}_{IC}$ are very close related. The main difference is that $\mathcal{E}$ uses the notion of enviroment, while $\mathcal{E}_{IC}$ does not. As pointed out on page 23-24 this amounts to the same as identifying the domain of unique identifiers with the domain of locations.

It will we proved:

Theorem 9: $\mathcal{E}_{IC}[\![\gamma]\!]\theta = \mathcal{E}[\![\gamma]\!]\varsigma_{ID}\theta$
where ς_{ID} is the enviroment mapping all identifiers to itself.

It should be noticed that using ς_{ID} we start execution with unique identifiers identical to the corresponding SNAIL-identifiers. If one or more of these identifiers are redeclared in an inner block we choose new unique identifiers different from the actual SNAIL-identifier.

The proof is done using structural induction on the complexity of γ.

In fact we ought to use simultaneous induction on the complexity of γ and ε but it will be taken for given that:

(*) $\kappa(\mathcal{E}_I [\![\varepsilon]\!] \psi) = \mathcal{E} [\![\varepsilon]\!] \varrho_{1D} \kappa\psi$.

(This is nescessary since we have left a formal definition of $\mathcal{E}_I$ to the reader).

It should be noticed that (*) is nothing else than continuation-removal for expressions.

6.1 Proof for Theorem 9:

Command lists

$$
\begin{aligned}
& \mathcal{C}_{IC}[\![\gamma_1;\gamma_2]\!]\theta \\
= \; & \theta \circ \mathcal{C}_I[\![\gamma_1;\gamma_2]\!] \\
= \; & \theta \circ \mathcal{C}_I[\![\gamma_2]\!] \circ \mathcal{C}_I[\![\gamma_1]\!] \\
= \; & (\mathcal{C}_{IC}[\![\gamma_2]\!]\theta)\,\mathcal{C}_I[\![\gamma_1]\!] \\
= \; & \mathcal{C}_{IC}[\![\gamma_1]\!]\{\mathcal{C}_{IC}[\![\gamma_2]\!]\theta\} \\
= \; & \mathcal{C}[\![\gamma_1]\!]\sigma_{ID}\{\mathcal{C}[\![\gamma_2]\!]\sigma_{ID}\theta\} \\
= \; & \mathcal{C}[\![\gamma_1;\gamma_2]\!]\sigma_{ID}\theta
\end{aligned}
$$

where the last but one equaillitysign is achieved using the inductional hypothesis.

Assignment

$$
\begin{aligned}
& \mathcal{C}_{IC}[\![\xi := \varepsilon]\!]\theta\psi \\
= \; & \theta \circ \mathcal{C}_I[\![\xi := \varepsilon]\!]\psi \\
= \; & \theta\,(assign_I(\xi, \mathcal{E}_I[\![\varepsilon]\!]\psi))\,\psi \\
(*) = \; & Update(\sigma_{ID}[\![\xi]\!]l, \mathcal{E}_I[\![\varepsilon]\!]\psi)\,\theta\psi \\
= \; & (\mathcal{E}[\![\varepsilon]\!]\sigma_{ID}\{\lambda\beta.\, Update(\sigma_{ID}[\![\xi]\!]l, \beta)\theta\})\,\psi \\
= \; & \mathcal{C}[\![\xi := \varepsilon]\!]\sigma_{ID}\theta\psi
\end{aligned}
$$

where $(*)$ is reached using the definition of the auxilliary functions, $assign_I$, update and assign, plus the fact that $\sigma_{ID}[\![\xi]\!]l = \xi$ for all identifiers.

IF-command

$$\mathcal{C}_{IC}[\![\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}]\!]\theta\psi$$

$$= \theta \circ \mathcal{C}_I[\![\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}]\!]\psi$$

$$= \theta\,([\mathcal{E}_I[\![\varepsilon]\!]\psi \to \mathcal{C}_I[\![f_1]\!]\psi, \;\vdots\; \mathcal{E}_I[\![\varepsilon_n]\!]\psi \to \mathcal{C}_I[\![f_n]\!]\psi])$$

$$= [\mathcal{E}_I[\![\varepsilon_1]\!]\psi \to \theta \circ \mathcal{C}_I[\![f_1]\!]\psi, \;\vdots\; \mathcal{E}_I[\![\varepsilon_n]\!]\psi \to \theta \circ \mathcal{C}_I[\![f_n]\!]\psi]$$

$$= [\mathcal{E}[\![\varepsilon_1]\!]\varrho_{1D}\kappa_\beta\psi \to \mathcal{C}[\![f_1]\!]\varrho_{1D}\theta\psi, \;\vdots\; \mathcal{E}[\![\varepsilon_n]\!]\varrho_{1D}\kappa_\beta\psi \to \mathcal{C}[\![f_n]\!]\varrho_{1D}\theta\psi] \qquad (\kappa_\beta = \lambda\beta.\lambda\psi.\beta)$$

where the inductional hypothesis is used.

By the definition of Oracle and the []-notation we get:

$$= (\mathcal{G}[\![\varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n]\!]\varrho_{1D}\{\text{Oracle}(\theta, \perp)\})\psi$$

$$= \mathcal{C}[\![\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}]\!]\varrho_{1D}\theta\psi$$

WHILE-command

$$
\begin{aligned}
& \mathcal{C}_{IC}[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\,\theta\psi \\
= \;& \theta \circ \mathcal{C}_I[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\,\psi \\
= \;& \theta\,(\text{FIX } F.(\lambda\psi.\ \mathcal{E}_I[\![\varepsilon]\!]\psi \rightarrow F \circ \mathcal{C}_I[\![\gamma]\!]\psi, \psi))\,\psi \\
(*) = \;& (\text{FIX } F.(\lambda\psi.\ \mathcal{E}_I[\![\varepsilon]\!]\psi \rightarrow F \circ \mathcal{C}_I[\![\gamma]\!]\psi, \theta\psi))\,\psi \\
= \;& (\text{FIX } F.\ \mathcal{E}[\![\varepsilon]\!]\varsigma_{1D} \rightarrow \text{Cond}(\mathcal{C}[\![\gamma]\!]\varsigma_{1D} F, \theta))\,\psi \\
= \;& \mathcal{C}[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\,\varsigma_{1D}\,\theta\psi
\end{aligned}
$$

where $(*)$ is intuitive straightforward since every execution of a WHILE-command is nonterminating or executes the first alternative of the Cond-construction a number of times followed by exactly one execution of the second alternative.

A formal proof using computational induction is left to the reader.

Declaration

$$\mathcal{C}_{Ic}[\![\text{BEGIN DEF } \xi = \varepsilon; \gamma \text{ END}]\!]\,\theta\,\psi$$

$$= \theta \circ \mathcal{C}_{I}[\![\text{BEGIN DEF } \xi = \varepsilon; \gamma \text{ END}]\!]\,\psi$$

$$= \theta\,(\text{LET } \xi' = \text{New}_{I}(\psi) \text{ IN } (\mathcal{C}_{I}[\![\xi' := \varepsilon; \gamma(\xi'/\xi)]\!]\,\psi))$$

$$= \text{LET } \xi' = \text{New}_{I}(\psi) \text{ IN } (\theta \circ \mathcal{C}_{I}[\![\xi' := \varepsilon; \gamma(\xi'/\xi)]\!]\,\psi)$$

$$= \text{LET } \xi' = \text{New}_{I}(\psi) \text{ IN } (\mathcal{C}_{Ic}[\![\xi' := \varepsilon; \gamma(\xi'/\xi)]\!]\,\theta\psi)$$

$$= \text{LET } \xi' = \text{New}_{I}(\psi) \text{ IN } (\mathcal{C}[\![\xi' := \varepsilon; \gamma(\xi'/\xi)]\!]\,\varsigma_{ID}\,\theta\,\psi)$$

where the inductional hypothesis is used.

$$= \text{LET } \xi' = \text{New}_{I}(\psi) \text{ IN } (\mathcal{C}[\![\xi' := \varepsilon]\!]\,\varsigma_{ID} \dagger \mathcal{C}[\![\gamma(\xi'/\xi)]\!]\,\varsigma_{ID}\,\theta\,\psi)$$

$$= \text{LET } \xi' = \text{New}_{I}(\psi) \text{ IN}$$
$$(\mathcal{E}[\![\varepsilon]\!]\,\varsigma_{ID} \dagger \lambda\beta.\ \text{Update}(\varsigma_{ID}[\![\xi']\!]/L, \beta) \dagger \mathcal{C}[\![\gamma(\xi'/\xi)]\!]\,\varsigma_{ID}\,\theta\,\psi)$$

$$= \text{LET } \xi' = \text{New}_{I}(\psi) \text{ IN}$$
$$(\mathcal{E}[\![\varepsilon]\!]\,\varsigma_{ID} \dagger \lambda\beta.\ \text{Update}(\xi', \beta) \dagger \mathcal{C}[\![\gamma(\xi'/\xi)]\!]\,\varsigma_{ID}\,\theta\,\psi)$$

$$= \mathcal{E}[\![\varepsilon]\!]\,\varsigma_{ID} \dagger \lambda\beta.\ \text{New}(\varsigma_{ID}) \dagger \lambda\alpha.\ \text{Update}(\alpha, \beta)$$
$$\dagger \mathcal{C}[\![\gamma(\xi'/\xi)]\!]\,\varsigma_{ID}\,\theta\,\psi$$

assuming that New and New_{I} is implemented in analogous ways (i.e. they select the locations in the same order).

$$= \mathcal{E}[\![\varepsilon]\!]\,\varsigma_{ID} \dagger \lambda\beta.\ \text{New}(\varsigma_{ID}) \dagger \lambda\alpha.\ \text{Update}(\alpha, \beta)$$
$$\dagger (\lambda\varsigma.\ \mathcal{C}[\![\gamma]\!]\,\varsigma\,\theta)(\varsigma_{ID}[\alpha/\xi])$$

It should be noticed that ς_{ID} is updated so that ξ now is bound to a new unique identifier α.

$$= \mathcal{D}[\![\text{DEF } \xi = \varepsilon]\!]\,\varsigma_{ID} \dagger \lambda\varsigma.\ \mathcal{C}[\![\gamma]\!]\,\varsigma\,\theta\,\psi$$

$$= \mathcal{C}[\![\text{BEGIN DEF } \xi = \varepsilon; \gamma \text{ END}]\!]\,\varsigma_{ID}\,\theta\,\psi$$

end of proof
for Theorem 9

In fact the earlier description of the predicate-transformer theory is a little simplified since it does not involve enviroments. (see page 73-75).

Predicates uses identifiers and these should be bound to locations. Therefore a predicate is a member of the domain

$$PRED_1 = [ENV \rightarrow [R \rightarrow TF]]$$

Using the idea brought forward on page 76 where predicates are viewed as special kind of continuations we receive an interpretation of Dijkstra's theory in terms of the mathematical continuation-semantic, $\mathcal{E}$, defined on page 15-22.

$$wp(\gamma, Q(\varsigma)) \equiv \mathcal{E}[\![\gamma]\!]\varsigma\, Q(\varsigma)$$

In all cases except wp7 the involved enviroment, ς, is kept constant and to avoid a lot of redundant notation it will often be omitted writing Q in place of $Q(\varsigma)$.

In the rest of this section I shall again assume that expressions can be evaluated in SNAIL without sideeffects. I shall use $\mathcal{E}_\psi$ as a shorthand for the value of expression ε in configuration ψ ($\mathcal{E}_\psi = \mathcal{E}[\![\varepsilon]\!]\varsigma K_\beta \psi$ where $K_\beta = \lambda\beta.\lambda\psi.\beta$).

We can now show:

Theorem 10: The predicate-transformer theory is satisfied by the mathematical continuation-semantic model represented by $\mathcal{E}$.

The proof is done using structural induction.

6.2 Proof for Theorem 10:

wp1 - Command lists

$$
\begin{aligned}
& wp(\gamma_1;\gamma_2, Q) \\
\equiv\ & \mathcal{C}[\![\gamma_1;\gamma_2]\!]\sigma\, Q \\
\equiv\ & \mathcal{C}[\![\gamma_1]\!]\sigma\,\{\mathcal{C}[\![\gamma_2]\!]\sigma\, Q\} \\
\equiv\ & wp(\gamma_1, \mathcal{C}[\![\gamma_2]\!]\sigma\, Q) \\
\equiv\ & wp(\gamma_1, wp(\gamma_2, Q))
\end{aligned}
$$

wp2 - Assignment

$$
\begin{aligned}
& wp(\xi := \varepsilon, Q)\,\psi \\
\equiv\ & \mathcal{C}[\![\xi := \varepsilon]\!]\sigma\, Q\,\psi \\
\equiv\ & \mathcal{E}[\![\varepsilon]\!]\sigma\,\{\lambda\beta.\ Update(\sigma[\![\xi]\!]|L, \beta)\, Q\}\,\psi \\
\equiv\ & Update(\sigma[\![\xi]\!]|L, \varepsilon_\psi)\, Q\,\psi \\
\equiv\ & Q \circ Assign(\sigma[\![\xi]\!]|L, \varepsilon_\psi)\,\psi \\
\equiv\ & Q\,(\varepsilon_\psi / \xi)\,\psi
\end{aligned}
$$

by the definition of the auxilliary function Assign.

wp3 - TEST-command

$$
\begin{aligned}
& wp(\text{TEST } \varepsilon \text{ DO } \gamma_1 \text{ OR } \gamma_2 \text{ TSET}, Q)\,\psi \\
\equiv\ & \mathcal{C}[\![\text{TEST } \varepsilon \text{ DO } \gamma_1 \text{ OR } \gamma_2 \text{ TSET}]\!]\sigma\, Q\,\psi \\
\equiv\ & \mathcal{E}[\![\varepsilon]\!]\sigma\,\{Cond(\mathcal{C}[\![\gamma_1]\!], \mathcal{C}[\![\gamma_2]\!])\sigma\, Q\,\psi \\
\equiv\ & (\varepsilon_\psi \wedge \mathcal{C}[\![\gamma_1]\!]\sigma\, Q\,\psi) \vee (\neg\varepsilon_\psi \wedge \mathcal{C}[\![\gamma_2]\!]\sigma\, Q\,\psi) \\
\equiv\ & (\varepsilon_\psi \wedge wp(\gamma_1, Q)\,\psi) \vee (\neg\varepsilon_\psi \wedge wp(\gamma_2, Q)\,\psi) \\
\equiv\ & ((\varepsilon \wedge wp(\gamma_1, Q)) \vee (\neg\varepsilon \wedge wp(\gamma_2, Q)))\,\psi
\end{aligned}
$$

wp4 - IF-command

$$
\begin{aligned}
& wp(\,IF\ \varepsilon_1 \to f_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to f_n\ FI\,,\ Q)\ \psi \\
\equiv\ & \mathcal{E}[\![\,IF\ \varepsilon_1 \to f_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to f_n\ FI\,]\!]\,\sigma\ Q\ \psi \\
\equiv\ & \mathcal{E}[\![\,\varepsilon_1 \to f_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to f_n\,]\!]\,\sigma\ \{\mathit{Oracle}\,(Q,\perp)\}\ \psi \\
\equiv\ & \mathit{Oracle}\,(Q,\perp) \ll \mathcal{E}[\![\varepsilon_1]\!]\sigma,\ \mathcal{E}[\![f_1]\!]\sigma\rangle, \cdots\cdots, \langle \mathcal{E}[\![\varepsilon_n]\!]\sigma,\ \mathcal{E}[\![f_n]\!]\sigma \gg\ \psi \\
\equiv\ & (\exists i : (\varepsilon_i)_\psi)\ \wedge\ \mathcal{E}[\![\,f_{\min B_\psi}]\!]\,\sigma\ Q\ \psi \\
\equiv\ & (\ B_\psi \neq \varnothing\ )\ \wedge\ wp(f_{\min B_\psi},\ Q)\ \psi \\
\equiv\ & (\quad B \neq \varnothing\quad \wedge\ wp\,(f_{\min B},\ Q)\ )\ \psi
\end{aligned}
$$

where $B_\psi = \{\, i \le n \mid (\varepsilon_i)_\psi \,\}$

and $B = \{\, i \le n \mid \varepsilon_i \,\}$

wp5 - DO-command

The proof of this axiom is omitted. It follows the same lines as the proof for IF (wp4) and WHILE (wp6).

wp6 - WHILE-command

$$
\begin{aligned}
& wp(\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}, Q)\,\psi \\
\equiv\ & \mathcal{E}[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\,\varsigma\, Q\,\psi \\
\equiv\ & (\text{FIX } \Theta'.\ \mathcal{E}[\![\varepsilon]\!]\,\varsigma\,\{\,Cond(\mathcal{E}[\![\gamma]\!]\,\varsigma\,\Theta', Q)\,\})\,\psi \\
(+) \equiv\ & \mathcal{E}[\![\varepsilon]\!]\,\varsigma\,\{\,Cond(\mathcal{E}[\![\gamma]\!]\,\varsigma\,\{\mathcal{E}[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\,\varsigma\, Q\,\}, Q)\,\}\,\psi
\end{aligned}
$$

The proof is now finished using induction after the number of times, n, the loop-command γ is executed.

In this proof I shall make use of continuation-removals. For expressions this is the same as assuming that they can be evaluated without sideeffects or transfer of control (jumps). For commands it only excludes transfer of control.

Formally this can be expressed by:

$$(*)\quad \exists\, \varepsilon_\psi\ \forall\, \kappa :\ \mathcal{E}[\![\varepsilon]\!]\,\varsigma\,\kappa\,\psi = \kappa(\varepsilon_\psi)$$

$$(**)\quad \exists\, \psi^+\ \forall\, \Theta :\ \mathcal{C}[\![\gamma]\!]\,\varsigma\,\Theta\,\psi = \Theta(\psi^+)$$

where ε, γ, ς and ψ are universally quantified.

n = 0:

$$\equiv\ Q\psi \wedge \neg\, \varepsilon_\psi$$

where (*) is used.

$$\equiv\ K_0(Q)\,\psi$$ (see definition of K_i on page 74)

and since $\neg\varepsilon_\psi$ by a straightforward inductive argumentation gives $K_i(Q)\,\psi = K_0(Q)\psi$ for all $i \geq 1$ we have:

$$\equiv\ (\exists\, i \geq 0 :\ K_i(Q)\,\psi\ )$$

$$\equiv\ ((\exists\, i \geq 0 :\ K_i(Q))\ \psi$$

n > 1:

$\equiv \mathcal{E}[\![\gamma]\!]\sigma\{\mathcal{E}[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\sigma Q\}\psi$

$\equiv \mathcal{E}[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\sigma Q\psi^+$

where $\psi^+ = \mathcal{E}[\![\gamma]\!]\sigma\{\lambda\psi.\psi\}\psi$

(The existence of such an ψ^+ follows from (**)).

Since $\mathcal{E}[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\sigma Q\psi^+$ executes the loop-command, γ, one times less than $\mathcal{E}[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\psi$ does the inductive hypothesis gives:

$\equiv (\exists i \geq 0 : K_i(Q))\psi^+$

$\equiv (\exists i \geq 0 : K_i(Q)\psi^+)$

(+) and (*) then gives that:

$\psi^+ = \mathcal{E}[\![\gamma]\!]\sigma\{\lambda\psi.\psi\}\psi$

$= \mathcal{E}[\![TEST\ \varepsilon\ DO\ \gamma\ OR\ e\ TSET]\!]\sigma Q\{\lambda\psi.\psi\}\psi$

and we can conclude using (**):

$\equiv (\exists i \geq 0 : K_i(Q)(\mathcal{E}[\![TEST\ \varepsilon\ DO\ \gamma\ OR\ e\ TSET]\!]\sigma Q\{\lambda\psi.\psi\}\psi)$

$\equiv (\exists i \geq 0 : \mathcal{E}[\![TEST\ \varepsilon\ DO\ \gamma\ OR\ e\ TSET]\!]\sigma K_i(Q)\psi)$

$\equiv (\exists i \geq 0 : wp(TEST\ \varepsilon\ DO\ \gamma\ OR\ e\ TSET, K_i(Q))\psi)$

$\equiv (\exists i \geq 1 : K_i(Q)\psi)$

$\equiv (\exists i \geq 0 : K_i(Q)\psi)$

since $\mathcal{E}_\psi$ = true implies that $K_0(Q)\psi$ = false

$\equiv ((\exists i \geq 0 : K_i(Q))\ \psi$

wp7 - Declaration (not empty)

$$wp(\text{BEGIN DEF } \xi = \varepsilon; \gamma \text{ END}, Q(\varsigma))\,\psi$$
$$\equiv \mathcal{C}[\![\text{BEGIN DEF } \xi = \varepsilon; \gamma \text{ END}]\!]\varsigma\, Q(\varsigma)\,\psi$$
$$\equiv \mathcal{D}[\![\text{DEF } \xi = \varepsilon]\!]\varsigma\{\lambda\varsigma'.\mathcal{C}[\![\gamma]\!]\varsigma' Q(\varsigma)\}\psi$$
$$\equiv \mathcal{E}[\![\varepsilon]\!]\varsigma\{\lambda\beta.New(\varsigma)\{\lambda\alpha.Update(\alpha,\beta)$$
$$\{(\lambda\varsigma'.\mathcal{C}[\![\gamma]\!]\varsigma' Q(\varsigma))(\varsigma[\alpha/\xi])\}\}\}\psi$$

$$\equiv Update(\varsigma^+[\![\xi]\!]L, \varepsilon_\psi)\{\mathcal{C}[\![\gamma]\!]\varsigma^+ Q(\varsigma)\}\psi$$

where α^+ is a new unused location and $\varsigma^+ = \varsigma[\alpha/\xi]$.
If ξ is not declared in any block sourrounding this one the situation is rather simple. Then γ is unknown outside this block and neither Q or its precondition can have ξ as free variable. By this reason $Q(\varsigma) = Q(\varsigma^+)$ and we get

$$\equiv Update(\varsigma^+[\![\xi]\!]L, \varepsilon_\psi)\{\mathcal{C}[\![\gamma]\!]\varsigma^+ Q(\varsigma^+)\}\psi$$

which according to the proof for wp2, definition of our interpretation and wp1 can be rewritten as follows:

$$\equiv wp(\xi := \varepsilon, \mathcal{C}[\![\gamma]\!]\varsigma^+ Q(\varsigma^+))\,\psi$$
$$\equiv wp(\xi := \varepsilon, wp(\gamma, Q(\varsigma^+)))\,\psi$$
$$\equiv wp(\xi := \varepsilon, wp(\gamma, Q(\varsigma)))\,\psi$$
$$\equiv wp(\xi := \varepsilon; \gamma, Q(\varsigma))\,\psi$$

If ξ is declared in a sourrounding block the situation is a bit more complicated. The equation derived tells us that $\mathcal{C}[\![\gamma]\!]$ and and Update is performed in a new enviroment where ξ is bound to a new unused location; but after this the old enviroment ς where ξ is bound to the old location is reinstated.
Since Q refers to the old ξ which cannot be altered during execution of this block the value of this free variable must be the same before and after execution. This is expressed by:

$$\exists k \in V\ [\, \xi_\psi = k \wedge Update(\varsigma^+[\![\xi]\!]L, \varepsilon_\psi)\{\mathcal{E}[\![f]\!]\varsigma^+ Q(k/\xi)(\varsigma)\}\psi\,]$$

$$\equiv \exists k \in V\ [\, \xi_\psi = k \wedge Update(\varsigma^+[\![\xi]\!]L, \varepsilon_\psi)\{\mathcal{E}[\![f]\!]\varsigma^+ Q(k/\xi)(\varsigma^+)\psi\,]$$

since $Q(k/\xi)$ no longer depends on the binding for ξ.
Now we can proceed as before:

$$\equiv \exists k \in V\ [\, \xi_\psi = k \wedge wp(\xi := \varepsilon, \mathcal{E}[\![f]\!]\varsigma^+ Q(k/\xi)(\varsigma^+))\ \psi\,]$$

$$\equiv \exists k \in V\ [\, \xi_\psi = k \wedge wp(\xi := \varepsilon, wp(f, Q(k/\xi)\varsigma^+))\psi\,]$$

$$\equiv \exists k \in V\ [\, \xi_\psi = k \wedge wp(\xi := \varepsilon, wp\ f, Q(k/\xi)\varsigma))\ \psi\,]$$

$$\equiv \exists k \in V\ [\, \xi_\psi = k \wedge wp(\xi := \varepsilon; f\ ,\ Q(k/\xi)\varsigma)\ \psi\,]$$

$$\equiv (\exists k \in V\ [\, \xi_{\cdot} = k \wedge wp(\xi := \varepsilon; f\ ,\ Q(k/\xi)\varsigma)\,])\psi$$

(please notice that the equation derived on page 91 is a special case of this equation).

wp8 - Declaration (empty)

$$wp(\text{BEGIN } e; f \text{ END}, Q)$$

$$\equiv \mathcal{E}[\![\text{BEGIN } e; f \text{ END}]\!]\varsigma Q$$

$$\equiv \mathcal{D}[\![e]\!]\varsigma\{\lambda\varsigma.\ \mathcal{E}[\![f]\!]\varsigma Q\}$$

$$\equiv \mathcal{E}[\![f]\!]\varsigma Q$$

$$\equiv wp(f, Q)$$

wp9 - READ-command

$$
\begin{aligned}
& wp(READ(\xi), Q)\,\psi \\
\equiv\ & \mathcal{E}[\![READ(\xi)]\!]\varsigma\, Q\,\psi \\
\equiv\ & Read(\varsigma[\![\xi]\!]/L)\, Q\,\psi \\
\equiv\ & Q \circ Rd(\varsigma[\![\xi]\!]/L)\,\psi \\
\equiv\ & Q\,((\kappa_I \vee 1), (\kappa_I + 2) / \xi, \kappa_I)
\end{aligned}
$$

by the definition of the auxilliary function Rd and the remarks about wp9 on page 75.

wp10 - WRITE-command

$$
\begin{aligned}
& wp(WRITE(\varepsilon), Q)\,\psi \\
\equiv\ & \mathcal{E}[\![WRITE(\varepsilon)]\!]\varsigma\, Q\,\psi \\
\equiv\ & \mathcal{E}[\![\varepsilon]\!]\varsigma\, \{\lambda\beta.\ Write(\beta)\, Q\}\,\psi \\
\equiv\ & Write(\varepsilon_\psi)\, Q\,\psi \\
\equiv\ & Q \circ Wr(\varepsilon_\psi)\,\psi \\
\equiv\ & Q\,((\kappa_0 @ \varepsilon_\psi) / \kappa_0)\,\psi
\end{aligned}
$$

by the definition of the auxilliary function Wr.
($\psi = \langle z, \mu_I, \mu_0 \rangle$).

wp11 - Empty-command

$$
\begin{aligned}
& wp(e, Q) \\
\equiv\ & \mathcal{E}[\![e]\!]\varsigma\, Q \\
\equiv\ & Q
\end{aligned}
$$

end of proof
for Theorem 10

7 COMPILER CORRECTNESS - TOY LANGUAGES

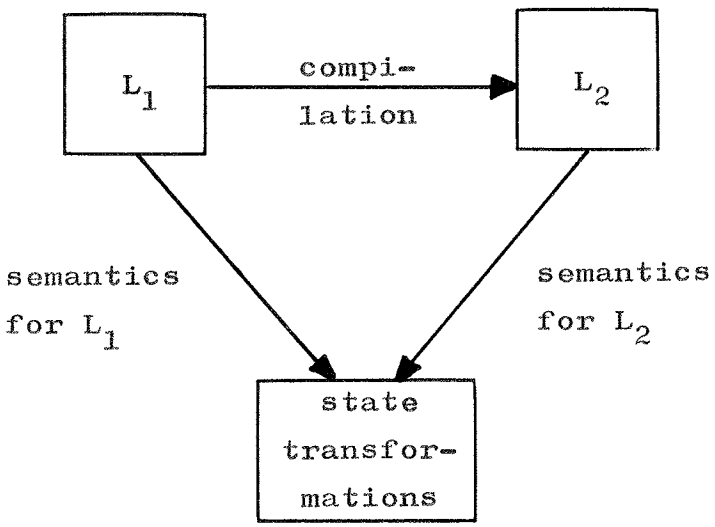
In the next pages of this paper I shall present a rather complicated proof for correctness of a given compiler. The languages involved in this proof are not toy-languages. Therefore the proof will be lengthy and it may be difficult to extract the main ideas from all the details.

By this reason I will now present a much simpler proof, where the main ideas and the used notation will be identical to those in the complicated one.

A proof of this kind consits of 4 parts:

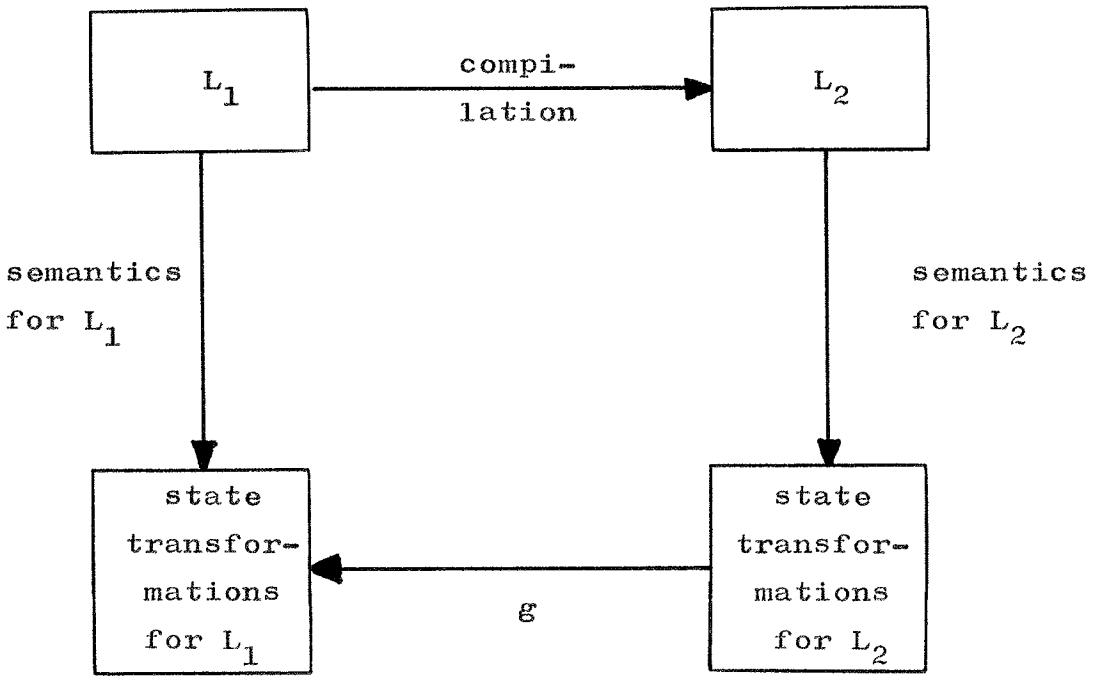
1. Definition of language L_1 (mathematical semantics)
2. Definition of language L_2 (mathematical semantics)
3. Definition of compilation from L_1 to L_2 and definition of compiler correctness.
4. The actual proof.

In very simple situations compiler correctness can be defined as commutation of the following diagram:



but in most cases the functionality of state transformations will differ for the two languages.

We then have a diagram of the type:



where the definition of the function g is what is meant with "definition of compiler correctness" in point 3 on the last page.

In some cases the simple diagram from the last page can be resumed by changing "state transformations" to "functions from inputs to outputs".

In the simple proof the situation will be as described by the last diagram. In the complicated proof we first show a diagram of the second type, then a diagram of the first type with "functions from inputs to outputs".

In /M2/ Milner and Weyhrauch presents a very elegant proof for compilercorrectness using algebraic methods. The languages are very simple and it is not proved that the method is appliable to more practical languages. The proofs are done using LCF, an interactive proof generator. LCF is described in Milner /M1/.

For a very simple proof of compiler correctness see Strachey /S3/.

7.1 Algorithmic language - L_1

L_1 is a very simple language. Commands are separated by ";" and executed in order of appearence. The only command-type is assignment.

Syntactis domains

$\gamma \in$	COM	commands
$\xi \in$	ID	identifiers

Value domains

$n \in$	N	natural numbers (0,1,2,3.....)
$z \in$	S = [ID → N]	states

Syntax

$$\gamma ::= \gamma_1 ; \gamma_2 \mid \xi := \varepsilon$$

$$\varepsilon ::= 0 \mid \xi \mid \underline{\text{SUC}}\ \varepsilon$$

Semantic functions

$$\mathcal{C}: \quad \text{COM} \to \text{S} \to \text{S}$$

$$\mathcal{E}: \quad \text{EXP} \to \text{S} \to \text{N}$$

Semantic equations

$$\mathcal{C}[\![\gamma_1 ; \gamma_2]\!] = \mathcal{C}[\![\gamma_2]\!] \circ \mathcal{C}[\![\gamma_1]\!]$$

$$\mathcal{C}[\![\xi := \varepsilon]\!]z = \text{assign}(\xi, \mathcal{E}[\![\varepsilon]\!]z)z$$

$$\mathcal{E}[\![0]\!]z = 0$$

$$\mathcal{E}[\![\xi]\!]z = z[\![\xi]\!]$$

$$\mathcal{E}[\![\underline{\text{SUC}}\ \varepsilon]\!]z = \mathcal{E}[\![\varepsilon]\!]z + 1$$

Auxilliary functions

assign: ID × N → S → S

$$\text{assign}\ (\xi, n)\ \sigma = \sigma'$$

$$\Updownarrow$$

$$\sigma'(\xi') = \begin{cases} \sigma(\xi') & \text{for } \xi' \neq \xi \\ n & \text{for } \xi' = \xi \end{cases}$$

7.2 Assembly language - L_2

L_2 is a simple assembly language. It is designed for a machine with one single accumulator. A program consists of a list of instructions. ";" acts as append-operator. Parentheses around single-element lists are omitted.

A configuration is a pair whose first element describes the value of the accumulator and whose second element describes the state.

Syntactic domains

$\pi \in$	INSTR	instructions
$\xi \in$	ID	identifiers

Value domains

$n \in$	N	natural numbers (0,1,2,3,....)
$\sigma \in$	S = [ID → N]	states
$\tau \in$	T = N × S	configurations

Syntax

$$\pi ::= \pi_1; \pi_2 \mid \text{ZERO} \mid \text{LOAD}\ \xi \mid \text{STORE}\ \xi \mid \text{SUCC}$$

Semantic functions

$$\mathcal{A}: \quad \text{INSTR} \rightarrow T \rightarrow T$$

Semantic equations

$$\mathcal{A}[\![\pi_1; \pi_2]\!] = \mathcal{A}[\![\pi_2]\!] \circ \mathcal{A}[\![\pi_1]\!]$$

$$\mathcal{A}[\![\text{ZERO}]\!]\langle n, \sigma\rangle = \langle 0, \sigma\rangle$$

$$\mathcal{A}[\![\text{LOAD}\ \xi]\!]\langle n, \sigma\rangle = \langle \sigma[\![\xi]\!], \sigma\rangle$$

$$\mathcal{A}[\![\text{STORE}\ \xi]\!]\langle n, \sigma\rangle = \langle n, \text{assign}(\xi, n)\sigma\rangle$$

$$\mathcal{A}[\![\text{SUCC}]\!]\langle n, \sigma\rangle = \langle n+1, \sigma\rangle$$

Auxilliary functions

As for L_1.

7.3 Definition of compiler and compiler correctness

I shall now define two functions

$$h : \mathrm{COM} \rightarrow \mathrm{INSTR}$$
$$k : \mathrm{EXP} \rightarrow \mathrm{INSTR}$$

and show the following:

(I) $$\exists m \; [\; \mathcal{A}[\![h[\![\gamma]\!]]\!] \langle n, \sigma \rangle = \langle m, \mathcal{C}[\![\gamma]\!] \sigma \rangle \;]$$

(II) $$\mathcal{A}[\![k[\![\varepsilon]\!]]\!] \langle n, \sigma \rangle = \langle \mathcal{E}[\![\varepsilon]\!] \sigma, \sigma \rangle$$

where γ, ε, n and σ are universaly quantified over COM, EXP, N and S respectively.

In other words:

> Starting in a specified state the same state will be reached by executing γ and by executing $h[\![\gamma]\!]$. Evaluation of $k[\![\varepsilon]\!]$ will place the value of ε in the accumulator, but not alter the state.

(I) in the above definition can be seen as a case of the diagram on page **95** if the function g is defined by

$$g : [\mathrm{T} \rightarrow \mathrm{T}] \rightarrow [\mathrm{S} \rightarrow \mathrm{S}]$$

$$g(f) = \lambda\sigma.\,((f \langle 0, \sigma \rangle) \downarrow 2) \quad .$$

7.4 Compiler Correctness

The proof of I and II will be done by induction after the complexity of γ and ε - structural induction.

In the proof every equalitysign will have a label indicating how this equality was established.

I	- inductional hypothesis I
II	- inductional hypothesis II
AL	- definition of L_1
AS	- definition of L_2
COM	- definition of compilation

Proof of I:

Command lists

$$
\begin{aligned}
& \mathcal{A}[\![h[\![\gamma_1;\gamma_2]\!]]\!]\langle n,z\rangle \\
\overset{COM}{=}\ & \mathcal{A}[\![h[\![\gamma_1]\!];h[\![\gamma_2]\!]]\!]\langle n,z\rangle \\
\overset{AS}{=}\ & (\mathcal{A}[\![h[\![\gamma_2]\!]]\!]\circ\mathcal{A}[\![h[\![\gamma_1]\!]]\!])\langle n,z\rangle \\
\overset{I}{=}\ & \mathcal{A}[\![h[\![\gamma_2]\!]]\!]\langle n',\mathcal{E}[\![\gamma_1]\!]z\rangle \\
\overset{I}{=}\ & \langle m,\mathcal{E}[\![\gamma_2]\!]\circ\mathcal{E}[\![\gamma_1]\!]z\rangle \\
\overset{AL}{=}\ & \langle m,\mathcal{E}[\![\gamma_1;\gamma_2]\!]z\rangle
\end{aligned}
$$

Assignment

$$\mathcal{A}[\![h[\![\xi := \varepsilon]\!]]\!] \langle n, z\rangle$$

$$\overset{COM}{=} \mathcal{A}[\![k[\![\varepsilon]\!]; \text{STORE } \xi]\!] \langle n, z\rangle$$

$$\overset{AS}{=} \mathcal{A}[\![\text{STORE } \xi]\!] \circ \mathcal{A}[\![k[\![\varepsilon]\!]]\!] \langle n, z\rangle$$

$$\overset{II}{=} \mathcal{A}[\![\text{STORE } \xi]\!] \langle \mathcal{E}[\![\varepsilon]\!] z, z\rangle$$

$$\overset{M}{=} \langle \mathcal{E}[\![\varepsilon]\!] z, \text{assign } (\xi, \mathcal{E}[\![\varepsilon]\!] z) z\rangle$$

$$\overset{AL}{=} \langle m, \mathcal{E}[\![\xi := \varepsilon]\!] z\rangle$$

Proof of II:

Zero

$$\mathcal{A}[\![k[\![0]\!]]\!] \langle n, z\rangle$$

$$\overset{COM}{=} \mathcal{A}[\![\text{ZERO}]\!] \langle n, z\rangle$$

$$\overset{AS}{=} \langle 0, z\rangle$$

$$\overset{AL}{=} \langle \mathcal{E}[\![0]\!] z, z\rangle$$

Identifier

$$\mathcal{A}[\![k[\![\xi]\!]]\!]\langle n,z\rangle$$

$$\overset{COM}{=} \mathcal{A}[\![LOAD\ \xi]\!]\langle n,z\rangle$$

$$\overset{AS}{=} \langle z[\![\xi]\!], z\rangle$$

$$\overset{AL}{=} \langle \mathcal{E}[\![\xi]\!]z, z\rangle$$

Successor

$$\mathcal{A}[\![k[\![\underline{SUC}\ \varepsilon]\!]]\!]\langle n,z\rangle$$

$$\overset{COM}{=} \mathcal{A}[\![k[\![\varepsilon]\!];\ SUCC]\!]\langle n,z\rangle$$

$$\overset{AS}{=} \mathcal{A}[\![SUCC]\!] \circ \mathcal{A}[\![k[\![\varepsilon]\!]]\!]\langle n,z\rangle$$

$$\overset{II}{=} \mathcal{A}[\![SUCC]\!]\langle \mathcal{E}[\![\varepsilon]\!]z, z\rangle$$

$$\overset{AS}{=} \langle \mathcal{E}[\![\varepsilon]\!]z + 1,\ z\rangle$$

$$\overset{AL}{=} \langle \mathcal{E}[\![\underline{SUC}\ \varepsilon]\!]z, z\rangle$$

end of proof for
compiler correctness

8 MATHEMATICAL SEMANTIC FOR ASSLA

In this section I shall define a language called ASSLA (assembly language). It is especially designed to fit as target-language for a SNAIL-compiler. It is a simple machine language involving symbolic names and made for a machine with one single pushdownlist, but no simple accumulators.

A program in ASSLA will consist of a list of instructions and ";" will act as appendoperator. Parentheses arround unary lists wille be omitted. NIL denotes the empty list.

Many comments in the semantic description of SNAIL on page 15-22 will apply equally well to this section. Such commands will not be repeated.

1. Syntactic domains

$\pi \in$	SEG	segments
$\pi \in$	INSTR	instructions
$\xi \in$	ID	identifiers
$k \in$	CONST	constants
$\oplus \in$	DOP	dyadic operators

2. Value domains

$\sigma \in T = P \times R$	total configurations
$\psi \in R = S \times In \times Out$	partial config.
$z \in S = [L \to V]$	states
$\kappa_I \in In = V^*$	inputs
$\kappa_O \in Out = V^*$	outputs
$\kappa_D \in P = V^*$	pushdownstacks
$\theta \in M = [T \to A]$	continuations
$\beta \in V$	values
$\alpha \in L$	locations

$\varrho \in ENV = [Id \to D]$ enviroments

$D = L + M$ denotations

R answers

A total configuration $\sigma = \langle \kappa_P, \psi \rangle$ describes

κ_P contents of stack

ψ same as for SNAIL

3. Syntax

$\pi ::=$ BLOCK π_1; DEF ξ; π_2 END |
BEGIN π_0; $\xi_1 : \pi_1$; $\xi_2 : \pi_2$; ....; $\xi_n : \pi_n$ END |
π_1; π_2 | STORE ξ | LOAD ξ | CONST β
READ | WRITE | J ξ | JT ξ | JF ξ
NEG | DO $\oplus$ | ABORT | e

$\oplus ::= + / - | * | / | \wedge | \vee | \supset | \equiv$

$n \geq 1$

4. Semantic functions

$\mathcal{S}$: SEG → IN → OUT

$\mathcal{A}$: INSTR → ENV → M → M

5. Semantic equations

$$\mathcal{S}[\![\pi]\!]\mu_I = (\mathcal{A}[\![\pi]\!]\sigma_{IN}\theta_{IN}\tau_{IN})\downarrow 2\downarrow 3$$

$$\sigma_{IN} = \lambda\xi.\perp \qquad \psi_{IN} = \langle \zeta_{IN}, \mu_I, \mu_{O_{IN}} \rangle$$
$$\theta_{IN} = \lambda 5.5 \qquad \mu_{P_{IN}} = NIL$$
$$\tau_{IN} = \langle \mu_{P_{IN}}, \psi_{IN} \rangle \qquad \zeta_{IN} = \lambda\kappa.\perp$$
$$\mu_{O_{IN}} = NIL$$

where the initial situation is as for SNAIL - the initial pushdownstack is empty.

In the definition of $\mathcal{A}$ we shall use the theory of fixpoints. This theory guarantees, under certain conditions which are satisfied here, that a set of fixpointequations has a minimal solution and moreover it gives a constructive way of finding this minimal solution. For further information about this theory see de Bakker /B/.
For the use of this theory in mathematical semantics see /S1/ and /S4/ which describe a situation analogous with ours.

$$\mathcal{A}[\![\text{BEGIN } \pi_0;\ \xi_1 : \pi_1;\ \ldots\ldots;\ \xi_n : \pi_n \text{ END}]\!]\,\varrho\,\theta = \theta_0$$

$$\begin{cases} \theta_0 = \mathcal{A}[\![\pi_0]\!]\varrho'\theta_1 \\ \theta_1 = \mathcal{A}[\![\pi_1]\!]\varrho'\theta_2 \\ \quad\vdots \\ \theta_{n-1} = \mathcal{A}[\![\pi_{n-1}]\!]\varrho'\theta_n \\ \theta_n = \mathcal{A}[\![\pi_n]\!]\varrho'\theta \\ \varrho' = \varrho[\theta_1,\ldots\theta_n / \xi_1,\ldots,\xi_n] \end{cases}$$

where the bracket indicate a set of fixpoint equations whose minimal solution should be used.

$$\mathcal{A}[\![\pi_1;\pi_2]\!]\varrho\theta = \mathcal{A}[\![\pi_1]\!]\varrho\,\{\mathcal{A}[\![\pi_2]\!]\varrho\theta\}$$

$$\mathcal{A}[\![\text{BLOCK } \pi_1;\text{DEF}\,\xi;\pi_2\ \text{END}]\!]\varrho\theta = \mathcal{A}[\![\pi_1]\!]\varrho\,\{New(\varrho)\,\{\lambda\kappa.\,\mathcal{A}[\![\pi_2]\!]\varrho[\kappa/\xi]\theta\}\}$$

$$\mathcal{A}[\![\text{STORE}\,\xi]\!]\varrho\theta = \lambda\langle\kappa_P,\langle z,\kappa_I,\kappa_0\rangle\rangle.\,\theta\langle\kappa_P+2,\langle Assign(\varrho[\![\xi]\!]L,\kappa_I\downarrow 1)z,\kappa_I,\kappa_0\rangle\rangle$$

$$\mathcal{A}[\![\text{LOAD}\,\xi]\!]\varrho\theta = \lambda\langle\kappa_P,\langle z,\kappa_I,\kappa_0\rangle\rangle.\,\theta\langle Contents(\varrho[\![\xi]\!]L)z@\kappa_P,\langle z,\kappa_I,\kappa_0\rangle\rangle$$

$$\mathcal{A}[\![\text{CONST}\,\beta]\!]\varrho\theta = \lambda\langle\kappa_P,\psi\rangle.\ \theta\langle(\beta)@\kappa_P,\psi\rangle$$

$$\mathcal{A}[\![\text{READ}]\!]\varrho\theta = \lambda\langle\kappa_P,\langle z,\kappa_I,\kappa_0\rangle\rangle.\,\theta(dom(\kappa_I)=0 \Rightarrow \langle\kappa_P,\langle\text{READERROR},\kappa_I,\kappa_0\rangle\rangle,$$
$$\langle(\kappa_I\downarrow 1)@\kappa_P,\langle z,\kappa_I+2,\kappa_0\rangle\rangle)$$

$$\mathcal{A}[\![\text{WRITE}]\!]\varrho\theta = \lambda\langle\kappa_P,\langle z,\kappa_I,\kappa_0\rangle\rangle.\,\theta\langle\kappa_P+2,\langle z,\kappa_I,\kappa_0@(\kappa_P\downarrow 1)\rangle\rangle$$

$$\mathcal{A}[\![\text{J}\,\xi]\!]\varrho\theta = \varrho[\![\xi]\!]\mid M$$

$$\mathcal{A}[\![\text{JT}\,\xi]\!]\varrho\theta = \lambda\langle\kappa_P,\psi\rangle.\,((\kappa_P\downarrow 1 \to(\varrho[\![\xi]\!]\mid M),\theta)\langle\kappa_P+2,\psi\rangle)$$

$$\mathcal{A}[\![\text{JF}\,\xi]\!]\varrho\theta = \lambda\langle\kappa_P,\psi\rangle.\,((\kappa_P\downarrow 1 \to(\theta,(\varrho[\![\xi]\!]\mid M))\langle\kappa_P+2,\psi\rangle)$$

$$\mathcal{A}[\![\text{NEG}]\!]\varrho\theta = \lambda\langle\kappa_P,\psi\rangle.\,\theta\langle(\neg(\kappa_P\downarrow 1))@(\kappa_P+2),\psi\rangle$$

$$\mathcal{A}[\![\text{DO}\,\oplus]\!]\varrho\theta = \lambda\langle\kappa_P,\psi\rangle.\,\theta\langle((\kappa_P\downarrow 2)\oplus(\kappa_P\downarrow 1))@(\kappa_P+3),\psi\rangle$$

$$\mathcal{A}[\![\text{ABORT}]\!]\varrho\theta = \perp$$

$$\mathcal{A}[\![e]\!]\varrho\theta = \theta$$

ASSLA has two different block-structures, and it is important to notice the difference between their use:

1. BLOCK...END

   This is used to simulate a SNAIL-declaration. It is executed as follows:

   a) π_1 is executed in the old enviroment
   b) A new enviroment ς' is formed from ς by binding identifier ξ to a new location
   c) π_2 is executed in enviroment ς'
   d) The enviroment is reset to ς.

2. BEGIN...END

   This defines by fixpointtheory a set of labels. The scope of these labels will be this block and all inner blocks without labels of the same name.

6. Auxilliary functions

Contents and Assign is exactly as in the semantic for SNAIL. New has another domain structure since ENV is different. Beyond this New is as in the definition of SNAIL.

We have now defined two different functions with the name New - context will show which one is meant.

9 COMPILATION FROM SNAIL TO ASSLA

I shall now define a compilation with sourcelanguage SNAIL and targetlanguage ASSLA. The compilation will be a function, h. To ease the logical structuring of the following proofs, h will be defined recursively in terms of itself and an auxilliary function called k.

h: PROG → SEG

k: EXP → SEG

$\mathcal{K}$: CONST → V

($\mathcal{K}$ is exactly as in the mathematical semantics for SNAIL)

$h\,[\![\varphi_1 ; \varphi_2]\!] = h\,[\![\varphi_1]\!] ; h\,[\![\varphi_2]\!]$

$h\,[\![\xi := \varepsilon]\!] = k\,[\![\varepsilon]\!] ;$ STORE ξ

$h\,[\![$ TEST ε DO φ_1 OR φ_2 TSET $]\!]$

$=$ BEGIN $k\,[\![\varepsilon]\!] ;$ JF $\xi_1 ; h\,[\![\varphi_1]\!] ;$ J $\xi_2 ; \xi_1 : h\,[\![\varphi_2]\!] ; \xi_2 : e$ END

$h\,[\![$ IF $\varepsilon_1 \to \varphi_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to \varphi_n]\!]$ $\quad (n \geq 1)$

$=$ BEGIN $k\,[\![\varepsilon_1]\!] ;$ JT $\xi_1 ; \cdots\cdots ; k\,[\![\varepsilon_n]\!] ;$ JT $\xi_n ;$ ABORT;

$\xi_1 : h\,[\![\varphi_1]\!] ;$ J $\xi_{n+1} ; \cdots\cdots ; \xi_n : h\,[\![\varphi_n]\!] ; \xi_{n+1} : e$ END

$h\,[\![$ DO $\varepsilon_1 \to \varphi_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to \varphi_n]\!]$ $\quad (n \geq 1)$

$=$ BEGIN $e ; \xi_0 : k\,[\![\varepsilon_1]\!] ;$ JT $\xi_1 ; \cdots\cdots ; k\,[\![\varepsilon_n]\!] ;$ JT $\xi_n ;$ J $\xi_{n+1} ;$

$\xi_1 : h\,[\![\varphi_1]\!] ;$ J $\xi_0 ; \cdots\cdots ; \xi_n : h\,[\![\varphi_n]\!] ;$ J $\xi_n ; \xi_{n+1} : e$ END

$h\,[\![$ WHILE ε DO φ OD $]\!]$

$=$ BEGIN $e ; \xi_1 : k\,[\![\varepsilon]\!] ;$ JF $\xi_2 ; h\,[\![\varphi]\!] ;$ J $\xi_1 ; \xi_2 : e$ END

$$h[\![\text{BEGIN DEF } \xi = \varepsilon\,;\, \gamma \text{ END}]\!] = \text{BLOCK } k[\![\varepsilon]\!];\ \text{DEF } \xi;\ \text{STORE } \xi;\ h[\![\gamma]\!]\ \text{END}$$

$$h[\![\text{BEGIN } e;\, \gamma \text{ END}]\!] = h[\![\gamma]\!]$$

$$h[\![\text{READ } (\xi)]\!] = \text{READ};\ \text{STORE } \xi$$

$$h[\![\text{WRITE}(\varepsilon)]\!] = k[\![\varepsilon]\!];\ \text{WRITE}$$

$$h[\![e]\!] = e$$

where all the labels $\xi_0, \ldots\ldots, \xi_{n+1}$ are supposed to be new and unused (generated by the compiler).

$$k[\![\text{TRUE}]\!] = \text{CONST TT}$$

$$k[\![\text{FALSE}]\!] = \text{CONST FF}$$

$$k[\![\xi]\!] = \text{LOAD } \xi$$

$$k[\![k]\!] = \text{CONST } \mathcal{K}[\![k]\!]$$

$$k[\![\neg\varepsilon]\!] = k[\![\varepsilon]\!];\ \text{NEG}$$

$$k[\![\varepsilon_1 \oplus \varepsilon_2]\!] = k[\![\varepsilon_1]\!];\ k[\![\varepsilon_2]\!];\ \text{DO } \oplus$$

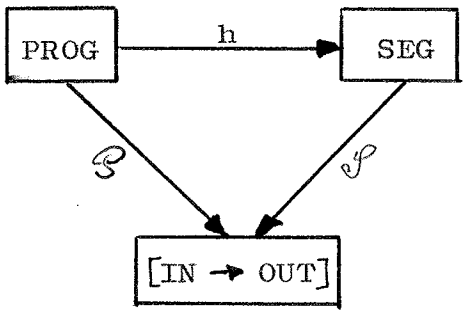
10 CORRECTNESS OF COMPILATION FROM SNAIL TO ASSLA

The main result in this section will be:

Theorem 11:

(I) $\mathcal{S}[\![h[\![\gamma]\!]]\!] = \mathcal{P}[\![\gamma]\!]$

The contents of this theorem can also be expressed as commutation of the following diagram:



Proof:

$$
\begin{aligned}
&\mathcal{S}[\![h[\![\gamma]\!]]\!]\,(\kappa_I)\\
&= (\mathcal{A}[\![h[\![\gamma]\!]]\!]\,\varsigma_{IN}\,\theta_{IN}\,\tau_{IN})\downarrow 2\downarrow 3\\
&\overset{(*)}{=} (\langle \kappa_{P_{IN}}, \mathcal{E}[\![\gamma]\!]\,\varsigma^*_{IN}\,\theta^*_{IN}\,\psi_{IN}\rangle)\downarrow 2\downarrow 3\\
&= (\mathcal{E}[\![\gamma]\!]\,\varsigma^*_{IN}\,\theta^*_{IN}\,\psi_{IN})\downarrow 3\\
&= \mathcal{P}[\![\gamma]\!]\,(\kappa_I)
\end{aligned}
$$

where $\varsigma_{IN}, \varsigma^*_{IN}, \theta_{IN}$ etc are as in the definition of $\mathcal{S}$ and $\mathcal{P}$ (see page 105 and page 17).

The equility marked $(*)$ follows from the following Theorem 12.

Theorem 12:

(II) $\mathcal{A}[\![h[\![f]\!]]\!]\varsigma\theta\langle\kappa_D,\psi\rangle = \langle\kappa_D, \mathcal{E}[\![f]\!]\varsigma^*\theta^*\psi\rangle$

(III) $\mathcal{A}[\![k[\![E]\!]]\!]\varsigma\theta\langle\kappa_D,\psi\rangle = \theta\langle(\mathcal{E}[\![E]\!]\varsigma^*\kappa_\beta\psi)@\kappa_D, \mathcal{E}[\![E]\!]\varsigma^*\kappa_\psi\psi\rangle$

where $\varsigma^* = \lambda\xi.\ (\varsigma[\![\xi]\!] \in M \rightarrow (\varsigma[\![\xi]\!])^*, \varsigma[\![\xi]\!])$

$(\varsigma[\![\xi]\!])^* = \lambda\psi.\ ((\varsigma[\![\xi]\!]\langle\kappa_D,\psi\rangle)\downarrow 2)$

$\theta^* = \lambda\psi.\ ((\theta\langle\kappa_D,\psi\rangle)\downarrow 2)$

$\kappa_\beta = \lambda\beta.\ \lambda\psi.\ \beta$

$\kappa_\psi = \lambda\beta.\ \lambda\psi.\ \psi$

In the following proof each equalitysign will have a label indicating how this equality was established.

COM	-	definition of compiler
AS	-	definition of ASSLA
AL	-	definition of SNAIL
II	-	inductional hypothesis II
III	-	inductional hypothesis III
IND	-	"local" inductional hypothesis
SIMP	-	simple algebraic simplification (it may refer to the definition of auxilliary functions)

Some notational difficulties arise since SNAIL and ASSLA have different functionality for their continuations.
The star-notation above should be viewed as an operator converting enviroments and continuations of ASSLA to corresponding enviroments and continuations for SNAIL. Please notice that the names given to ς_{IN}, θ_{IN}, ς^*_{IN} and θ^*_{IN} in the definition of $\mathcal{P}$ and $\mathcal{P}$ are consistent with this use of star.

10.1 Proof of Theorem 12:

This is done by using simultaneous induction after the complexity of γ and ε respectively.

It should however be noticed, that some of the proofs for (II) depend on the following assumption:

(*) All continuations used on the left hand side of (II) applied to any configurations do not alter the push-downstack. Or more precise: the stack is the same before and after execution.

This assumption can be justified using (II).

To avoid a circular argumentation (*) must be included in the inductional hypothesis together with (II) and (III).

To some extent this resembles the notion of realizability in Ligler /L2/. Ligler restricts himself to consider only those states and enviroments which can actually be reached from a given initial situation. I restrict myself to consider only those command-continuations which are "wellbehaved" in the sense described above.

10.1.1 Proof of (II):

Command lists

$$\mathcal{A}[\![h[\![f_1; f_2]\!]]\!]\varrho\theta <\kappa_P, \psi>$$

$$\overset{COM}{=} \mathcal{A}[\![h[\![f_1]\!]; h[\![f_2]\!]]\!]\varrho\theta <\kappa_P, \psi>$$

$$\overset{AS}{=} \mathcal{A}[\![h[\![f_1]\!]]\!]\varrho\{\mathcal{A}[\![h[\![f_2]\!]]\!]\varrho\theta\} <\kappa_P, \psi>$$

$$\overset{II}{=} <\kappa_P, \mathcal{E}[\![f_1]\!]\varrho^*\{\lambda\psi.((\mathcal{A}[\![h[\![f_2]\!]]\!]\varrho\theta <\kappa_P, \psi>)\downarrow 2)\}\psi>$$

$$\overset{II}{=} <\kappa_P, \mathcal{E}[\![f_1]\!]\varrho^*\{\lambda\psi.\ \mathcal{E}[\![f_2]\!]\varrho^*\theta^*\psi\}\psi>$$

$$\overset{SIMP}{=} <\kappa_P, \mathcal{E}[\![f_1]\!]\varrho^*\{\mathcal{E}[\![f_2]\!]\varrho^*\theta^*\}\psi>$$

$$\overset{AL}{=} <\kappa_P, \mathcal{E}[\![f_1; f_2]\!]\varrho^*\theta^*\psi>$$

Assignment

$$\mathcal{A}[\![h[\![\xi := \varepsilon]\!]]\!]\varrho\theta <\kappa_P, \psi>$$

$$\overset{COM}{=} \mathcal{A}[\![k[\![\varepsilon]\!]; \text{STORE}\ \xi]\!]\varrho\theta <\kappa_P, \psi>$$

$$\overset{AS}{=} \mathcal{A}[\![k[\![\varepsilon]\!]]\!]\varrho\{\mathcal{A}[\![\text{STORE}\ \xi]\!]\varrho\theta\} <\kappa_P, \psi>$$

$$\overset{III}{=} \mathcal{A}[\![\text{STORE}\ \xi]\!]\varrho\theta <(\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\beta\psi)@\kappa_P, \mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\psi\psi>$$

$$\overset{AS}{=} \theta <\kappa_P, <\text{assign}(\varrho[\![\xi]\!]/L, \mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\beta\psi)(\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_2\psi), \kappa_I/\kappa_o>>$$

$$\overset{AL}{=} \theta <\kappa_P, \mathcal{E}[\![\varepsilon]\!]\varrho^*\{\lambda\beta.\ Update(\varrho[\![\xi]\!]/L, \beta)\{\lambda\psi.\psi\}\}\psi>$$

$$\overset{SIMP}{=} <\kappa_P, \mathcal{E}[\![\varepsilon]\!]\varrho^*\{\lambda\beta.\ Update(\varrho[\![\xi]\!]/L, \beta)\theta^*\}\psi>$$

$$\overset{AL}{=} <\kappa_P, \mathcal{E}[\![\xi := \varepsilon]\!]\varrho^*\theta^*\psi>$$

where $\kappa_2 = \lambda\beta.\ \lambda\psi.\ (\psi\downarrow 1)$

TEST-command

$$\mathcal{A}[\![h[\![\text{TEST } \varepsilon \text{ DO } \gamma_1 \text{ OR } \gamma_2 \text{ TSET}]\!]\varsigma\theta\rangle\langle\kappa_p,\psi\rangle$$

$$\overset{COM}{=} \mathcal{A}[\![\text{BEGIN } k[\![\varepsilon]\!]; \text{JF } \xi_1; h[\![\gamma_1]\!]; \text{J } \xi_2; \xi_1: h[\![\gamma_2]\!]; \xi_2: e \text{ END}]\!]\varsigma\theta\langle\kappa_p,\psi\rangle$$

$$\overset{AS}{=} \theta_0\langle\kappa_p,\psi\rangle$$

where
$$\begin{cases} \theta_0 = \mathcal{A}[\![k[\![\varepsilon]\!]; \text{JF } \xi_1; h[\![\gamma_1]\!]; \text{J } \xi_2]\!]\varsigma'\theta_1 \\ \theta_1 = \mathcal{A}[\![h[\![\gamma_2]\!]]\!]\varsigma'\theta_2 \\ \theta_2 = \mathcal{A}[\![e]\!]\varsigma'\theta \\ \varsigma' = \varsigma[\theta_1,\theta_2/\xi_1,\xi_2] \end{cases}$$

(please notice that this is a simple set of equations - no fixpoints are involved since all jumps are forward).

$$\overset{SIMP}{=} \mathcal{A}[\![k[\![\varepsilon]\!]]\!]\varsigma\{\mathcal{A}[\![\text{JF } \xi_1; h[\![\gamma_1]\!]; \text{J } \xi_2]\!]\varsigma'\theta_1\}\langle\kappa_p,\psi\rangle$$

$$\overset{III}{=} \mathcal{A}[\![\text{JF } \xi_1]\!]\varsigma'\{\mathcal{A}[\![h[\![\gamma_1]\!]; \text{J } \xi_2]\!]\varsigma'\theta_1\}$$
$$\langle(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)@\kappa_p, \mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi\rangle$$

ς^* is used in place of $(\varsigma')^*$ since evaluation of ε, γ_1 and γ_2 do not depend on ξ_1 and ξ_2 since these are new unused labels.

case 1: $\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi = \bot$

$$\overset{AS}{=} \bot$$

$$\overset{AL}{=} \langle\kappa_p, \mathcal{E}[\![\text{TEST } \varepsilon \text{ DO } \gamma_1 \text{ OR } \gamma_2 \text{ TSET}]\!]\varsigma^*\theta^*\psi\rangle$$

case 2: $\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi$ = false

$$\overset{AS}{=} \theta_1\langle\kappa_p, \mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi\rangle$$

$$\overset{SIMP}{=} \mathcal{A}[\![h[\![\gamma_2]\!]]\!]\varsigma'\theta\langle\kappa_p, \mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi\rangle$$

$$\overset{II}{=} \langle\kappa_p, \mathcal{E}[\![\gamma_2]\!]\varsigma^*\theta^*(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi)\rangle$$

$\overset{AL}{=} \langle \kappa_P, \mathcal{E}[\![E]\!]\varsigma^* \{ Cond(\mathcal{C}[\![\gamma_1]\!], \mathcal{C}[\![\gamma_2]\!])\varsigma^*\theta^* \} \psi \rangle$

$\overset{AL}{=} \langle \kappa_P, \mathcal{C}[\![TEST\ E\ DO\ \gamma_1\ OR\ \gamma_2\ TSET]\!]\varsigma^*\theta^*\psi \rangle$

case 3: $\mathcal{E}[\![E]\!]\varsigma^*\kappa_\beta\psi$ = true

$\overset{AS}{=} \mathcal{A}[\![h[\![\gamma_1]\!]]\!]\varsigma' \{ \mathcal{A}[\![J\xi_2]\!]\varsigma'\theta_1 \} \langle \kappa_P, \mathcal{E}[\![E]\!]\varsigma^*\kappa_\psi\psi \rangle$

$\overset{II}{=} \langle \kappa_P, \mathcal{C}[\![\gamma_1]\!]\varsigma^* \{ \lambda\psi.((\mathcal{A}[\![J\xi_2]\!]\varsigma'\theta_1 \langle \kappa_P, \mathcal{E}[\![E]\!]\varsigma^*\kappa_\psi\psi \rangle)\downarrow 2) \}$
$(\mathcal{E}[\![E]\!]\varsigma^*\kappa_\psi\psi) \rangle$

$\overset{AS}{=} \langle \kappa_P, \mathcal{C}[\![\gamma_1]\!]\varsigma^* \{ \lambda\psi.((\theta \langle \kappa_P, \mathcal{E}[\![E]\!]\varsigma^*\kappa_\psi\psi \rangle)\downarrow 2) \}$
$(\mathcal{E}[\![E]\!]\varsigma^*\kappa_\psi\psi) \rangle$

$\overset{SIMP}{=} \langle \kappa_P, \mathcal{E}[\![E]\!]\varsigma^* \{ Cond(\mathcal{C}[\![\gamma_1]\!], \mathcal{C}[\![\gamma_2]\!]\varsigma^*\theta^* \} \psi \rangle$

$\overset{AL}{=} \langle \kappa_P, \mathcal{C}[\![TEST\ E\ DO\ \gamma_1\ OR\ \gamma_2\ TSET]\!]\varsigma^*\theta^*\psi \rangle$

IF-command

Proof omitted. It is quite analogous to the proof for TEST. Both commands only have forward jumps. The main difference is the increase in the number of labels.

DO-command

Proof omitted. It is quite analogous to the proof for WHILE. Both commands have forward and backward jumps and therefore involve fixpointequations. The main difference is the number of labels.

WHILE-command

$$\mathcal{A}[\![h[\![\text{WHILE } e \text{ DO } \gamma \text{ OD}]\!]]\!]\varsigma\theta\langle\kappa_P,\psi\rangle$$

$$\overset{COM}{=} \mathcal{A}[\![\text{BEGIN } e;\xi_1:k[\![e]\!];JF\xi_2;h[\![\gamma]\!];J\xi_1;\xi_2:\text{END}]\!]\varsigma\theta\langle\kappa_P,\psi\rangle$$

$$\overset{AS}{=} \theta\langle\kappa_P,\psi\rangle$$

where
$$\begin{cases} \theta_0 = \mathcal{A}[\![e]\!]\varsigma'\theta_1 \\ \theta_1 = \mathcal{A}[\![k[\![e]\!];JF\xi_2;h[\![\gamma]\!];J\xi_1]\!]\varsigma'\theta_2 \\ \theta_2 = \mathcal{A}[\![e]\!]\varsigma'\theta \\ \varsigma' = \varsigma[\theta_1,\theta_2/\xi_1,\xi_2] \end{cases}$$

The equations in the bracket is fixpointequations. The minimal solution should be chosen.

$$\overset{AS}{=} \theta_0\langle\kappa_P,\psi\rangle$$

where

$$(*)\quad \begin{cases} \theta_0 = \mathcal{A}[\![k[\![e]\!];JF\xi_2;h[\![\gamma]\!];J\xi_1]\!]\varsigma'\theta \\ \varsigma' = \varsigma[\theta_0,\theta/\xi_1,\xi_2] \end{cases}$$

(again the minimal solution should be chosen).

SEE NEXT PAGE
$$= \theta^+\langle\kappa_P,\psi\rangle$$

where θ^+ is a minimal solution to the following equation

$$(**)\quad \theta^+\langle\kappa_P,\psi\rangle = \langle\kappa_P,\mathcal{E}[\![e]\!]\varsigma^*\{\text{Cond}(\mathcal{E}[\![\gamma]\!]\varsigma^*\theta^+,\theta^*)\}\psi\rangle$$

$$\overset{SIMP}{=} (\text{FIX } \theta^+.(\lambda\langle\kappa_P,\psi\rangle.\langle\kappa_P,\mathcal{E}[\![e]\!]\varsigma^*\{\text{Cond}(\mathcal{E}[\![\gamma]\!]\varsigma^*\theta^+,\theta^*)\}\psi\rangle))\langle\kappa_P,\psi\rangle$$

$$\overset{SIMP}{=} \langle\kappa_P,(\text{FIX } \theta^{**}.\mathcal{E}[\![e]\!]\varsigma^*\{\text{Cond}(\mathcal{E}[\![\gamma]\!]\varsigma^*\theta^{**},\theta^*)\})\ \psi\rangle$$

$$\overset{AL}{=} \langle\kappa_P,\mathcal{E}[\![\text{WHILE } e \text{ DO } \gamma \text{ OD}]\!]\varsigma^*\theta^*\psi\rangle$$

Proof for equivalence of (*) and (**):

$$\mathcal{A}[\![k[\![\varepsilon]\!];\ JF\,\xi_2;\ h[\![\gamma]\!];\ J\xi_1]\!]\varrho'\theta\,\langle\kappa_p,\psi\rangle$$

$$\overset{SIMP}{=}\ \mathcal{A}[\![k[\![\varepsilon]\!]]\!]\varrho'\{\mathcal{A}[\![JF\xi_2;\ h[\![\gamma]\!];\ J\xi_1]\!]\varrho'\theta\}\langle\kappa_p,\psi\rangle$$

$$\overset{III}{=}\ \mathcal{A}[\![JF\xi_2]\!]\varrho'\{\mathcal{A}[\![h[\![\gamma]\!];\ J\xi_1]\!]\varrho'\theta\}$$
$$\langle(\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\beta\psi)@\mu_p,\ \mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\psi\psi\rangle$$

where we have used ϱ^* in place of $(\varrho')^*$ since evaluation of and ε cannot depend on the new labels ξ_1 and ξ_2.

case 1: $\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\beta\psi = \bot$

$$\overset{AS}{=}\ \bot$$

$$\overset{AL}{=}\ \langle\kappa_p,\ \mathcal{E}[\![\varepsilon]\!]\varrho^*\{Cond(\mathcal{E}[\![\gamma]\!]\varrho^*\theta_0^*,\theta^*)\}\psi\rangle$$

case 2: $\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\beta\psi$ = false

$$\overset{AS}{=}\ \theta\,\langle\kappa_p,\ \mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\psi\psi\rangle$$

$$\overset{SIMP}{=}\ \langle\kappa_p,\ \mathcal{E}[\![\varepsilon]\!]\varrho^*\{\lambda\psi.\ \theta^*\psi\}\psi\rangle$$

$$\overset{AL}{=}\ \langle\kappa_p,\ \mathcal{E}[\![\varepsilon]\!]\varrho^*\{Cond(\mathcal{E}[\![\gamma]\!]\varrho^*\theta_0^*,\theta^*)\}\psi\rangle$$

case 3: $\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\beta\psi$ = true

$$\overset{AS}{=}\ \mathcal{A}[\![h[\![\gamma]\!]]\!]\varrho'\{\mathcal{A}[\![J\xi_1]\!]\varrho'\theta\,\langle\kappa_p,\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\psi\psi\rangle$$

$$\overset{II}{=}\ \langle\kappa_p,\ \mathcal{E}[\![\gamma]\!]\varrho^*\{\lambda\psi.((\mathcal{A}[\![J\xi_1]\!]\varrho'\theta$$
$$\langle\kappa_p,\ \mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\psi\psi\rangle)\downarrow 2)\}(\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\psi\psi)\rangle$$

$$\overset{AS}{=}\ \langle\kappa_p,\ \mathcal{E}[\![\gamma]\!]\varrho^*\{\lambda\psi.((\theta_0\,\langle\kappa_p,\ \mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\psi\psi\rangle)\downarrow 2)\}(\mathcal{E}[\![\varepsilon]\!]\varrho^*\kappa_\psi\psi)\rangle$$

$$\overset{AL}{=}\ \langle\kappa_p,\ \mathcal{E}[\![\varepsilon]\!]\varrho^*\{Cond(\mathcal{E}[\![\gamma]\!]\varrho^*\theta_0^*,\theta^*)\}\psi\rangle$$

end of proof
for equivalence

Declaration (not empty)

$$\mathcal{A}[\![h[\![\text{BEGIN DEF } \xi=\varepsilon;\gamma \text{ END}]\!]\varsigma\theta\rangle\langle\kappa_P,\psi\rangle$$

$\overset{COM}{=}$ $\mathcal{A}[\![\text{BLOCK } k[\![\varepsilon]\!];\ \text{DEF } \xi;\ \text{STORE } \xi;\ h[\![\gamma]\!]\ \text{END}]\!]\varsigma\theta\langle\kappa_P,\psi\rangle$

$\overset{AS}{=}$ $\mathcal{A}[\![k[\![\varepsilon]\!]]\!]\varsigma\{\!|\,\text{New}(\varsigma)\{\!|\lambda\kappa.\,\mathcal{A}[\![\text{STORE } \xi;\ h[\![\gamma]\!]]\!]\varsigma[\kappa/\xi]\theta\,|\!\}\!\}\langle\kappa_P,\psi\rangle$

$\overset{III}{=}$ $(\text{New}(\varsigma)\{\!|\lambda\kappa.\,\mathcal{A}[\![\text{STORE } \xi]\!]\varsigma[\kappa/\xi]\{\!|\mathcal{A}[\![h[\![\gamma]\!]]\!]\varsigma[\kappa/\xi]\theta\,|\!\}\!\}$

$\langle(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)@\kappa_P,\ \mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi\rangle$

$\overset{AS}{=}$ $\mathcal{A}[\![\text{STORE } \xi]\!]\varsigma[\alpha_0/\xi]\{\!|\mathcal{A}[\![h[\![\gamma]\!]]\!]\varsigma[\alpha_0/\xi]\theta\,|\!\}\!\}$

$\langle(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)@\kappa_P,\ \mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi\rangle$

where α_0 is a new unused location in ς.

$\overset{AS}{=}$ $\mathcal{A}[\![h[\![\gamma]\!]]\!]\varsigma[\alpha_0/\xi]\theta$

$\langle\kappa_P,\langle\text{Assign}(\alpha_0,\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_2\psi),\kappa_I,\kappa_0\rangle\rangle$

$\overset{II}{=}$ $\langle\kappa_P,\mathcal{E}[\![\gamma]\!](\varsigma[\alpha_0/\xi])^*\theta^*$

$\langle\text{Assign}(\alpha_0,\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_2\psi),\kappa_I,\kappa_0\rangle\rangle$

$\overset{SIMP}{=}$ $\langle\kappa_P,\mathcal{E}[\![\varepsilon]\!]\varsigma^*\{\!|\lambda\beta.\,\text{Update}(\alpha_0,\beta)\{\!|\mathcal{E}[\![\gamma]\!](\varsigma[\alpha_0/\xi])^*\theta^*\,|\!\}\!\}\psi\rangle$

$\overset{SIMP}{=}$ $\langle\kappa_P,\mathcal{E}[\![\varepsilon]\!]\varsigma^*\{\!|\lambda\beta.\,\text{New}(\varsigma^*)\{\!|\lambda\kappa.\,\text{Update}(\alpha,\beta)\{\!|\mathcal{E}[\![\gamma]\!](\varsigma[\kappa/\xi])^*\theta^*\,|\!\}\!\}\!\}\psi\rangle$

$\overset{AL}{=}$ $\langle\kappa_P,\mathcal{D}[\![\text{DEF } \xi=\varepsilon]\!]\varsigma^*\{\!|\lambda\varsigma'.\,\mathcal{E}[\![\gamma]\!]\varsigma'^*\theta^*\,|\!\}\psi\rangle$

$\overset{AL}{=}$ $\langle\kappa_P,\mathcal{E}[\![\text{BEGIN DEF } \xi=\varepsilon;\gamma \text{ END}]\!]\varsigma^*\theta^*\psi\rangle$

Declaration (empty)

$$\mathcal{A}[\![h[\![\text{BEGIN } e;\gamma \text{ END}]\!]\varsigma\theta\rangle\langle\kappa_P,\psi\rangle$$

$\overset{COM}{=}$ $\mathcal{A}[\![h[\![\gamma]\!]]\!]\varsigma\theta\langle\kappa_P,\psi\rangle$

$\overset{II}{=}$ $\langle\kappa_P,\mathcal{E}[\![\gamma]\!]\varsigma^*\theta^*\psi\rangle$

$\overset{AL}{=}$ $\langle\kappa_P,\mathcal{D}[\![e]\!]\varsigma^*\{\!|\lambda\varsigma'.\,\mathcal{E}[\![\gamma]\!]\varsigma'\theta^*\,|\!\}\psi\rangle$

$\overset{AL}{=}$ $\langle\kappa_P,\mathcal{E}[\![\text{BEGIN } e;\gamma \text{ END}]\!]\varsigma^*\theta^*\psi\rangle$

READ-command

In the case where $\dim(\kappa_I) \neq 0$ we have:

$$\mathcal{A}[\![h[\![READ(\xi)]\!]]\!]\varsigma\theta\langle\kappa_P,\psi\rangle$$

$\overset{COM}{=}$ $\mathcal{A}[\![READ; STORE\ \xi]\!]\varsigma\theta\langle\kappa_P,\langle z,\kappa_I,\kappa_0\rangle\rangle$

$\overset{AS}{=}$ $\mathcal{A}[\![STORE\ \xi]\!]\varsigma\theta\langle(\kappa_I\downarrow 1)@\kappa_P,\langle z,\kappa_I+2,\kappa_0\rangle\rangle$

$\overset{AS}{=}$ $\theta\langle\kappa_P,\langle Assign(\varsigma[\![\xi]\!]/L,\kappa_I\downarrow 1)z,\kappa_I+2,\kappa_0\rangle\rangle$

$\overset{SIMP}{=}$ $\langle\kappa_P,\theta^*\langle Assign(\varsigma[\![\xi]\!]/L,\kappa_I\downarrow 1)z,\kappa_I+2,\kappa_0\rangle\rangle$

$\overset{SIMP}{=}$ $\langle\kappa_P,\theta^* Rd(\varsigma^*[\![\xi]\!]/L)\langle z,\kappa_I,\kappa_0\rangle\rangle$

$\overset{SIMP}{=}$ $\langle\kappa_P, Read(\varsigma^*[\![\xi]\!]/L)\theta^*\psi\rangle$

$\overset{AL}{=}$ $\langle\kappa_P, \mathcal{E}[\![READ(\xi)]\!]\varsigma^*\theta^*\psi\rangle$

If $\dim(\kappa_I) = 0$ both sides of (II) yields $\langle\kappa_P\langle Readerror,\kappa_I,\kappa_0\rangle\rangle$.

WRITE-command

$$\mathcal{A}[\![h[\![WRITE(\varepsilon)]\!]]\!]\varsigma\theta\langle\kappa_P,\psi\rangle$$

$\overset{COM}{=}$ $\mathcal{A}[\![k[\![\varepsilon]\!]; WRITE]\!]\varsigma\theta\langle\kappa_P,\psi\rangle$

$\overset{AS}{=}$ $\mathcal{A}[\![k[\![\varepsilon]\!]]\!]\varsigma\{\mathcal{A}[\![WRITE]\!]\varsigma\theta\}\langle\kappa_P,\psi\rangle$

$\overset{III}{=}$ $\mathcal{A}[\![WRITE]\!]\varsigma\theta\langle(\mathcal{E}[\![\varepsilon]\!]\varsigma\kappa_\beta\psi)@\kappa_P,\mathcal{E}[\![\varepsilon]\!]\varsigma\kappa_\psi\psi\rangle$

$\overset{AS}{=}$ $\theta\langle\kappa_P,\langle\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_2\psi,\mathcal{E}[\![\varepsilon]\!]\varsigma\kappa_{\kappa_I}\psi,(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_{\kappa_0}\psi)@(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)\rangle\rangle$

$\overset{SIMP}{=}$ $\langle\kappa_P,\theta^*\circ WR(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi)\rangle$

$\overset{SIMP}{=}$ $\langle\kappa_P, WRITE(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)\theta^*(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi)\rangle$

$\overset{SIMP}{=}$ $\langle\kappa_P,\mathcal{E}[\![\varepsilon]\!]\varsigma^*\{\lambda\beta.\ WRITE(\beta)\theta^*\}\psi\rangle$

$\overset{AL}{=}$ $\langle\kappa_P,\mathcal{E}[\![WRITE(\varepsilon)]\!]\varsigma^*\theta^*\psi\rangle$

where $\kappa_2 = \lambda\beta.\ \lambda\psi.\ (\psi\downarrow 1)$

$\kappa_{\kappa_I} = \lambda\beta.\ \lambda\psi.\ (\psi\downarrow 2)$

$\kappa_{\kappa_0} = \lambda\beta.\ \lambda\psi.\ (\psi\downarrow 3)$

Empty-command

$$\mathcal{A}[\![h[\![e]\!]]\!]\varsigma\theta\langle\kappa_p,\psi\rangle$$

$$\overset{COM}{=}\ \mathcal{A}[\![e]\!]\varsigma\theta\langle\kappa_p,\psi\rangle$$

$$\overset{AS}{=}\ \theta\langle\kappa_p,\psi\rangle$$

$$\overset{SIMP}{=}\ \langle\kappa_p,\theta^*\psi\rangle$$

$$\overset{AL}{=}\ \langle\kappa_p,\mathcal{B}[\![e]\!]\varsigma^*\theta^*\psi\rangle$$

end of proof
for (II)

10.1.2 Proof of (III):

TRUE-expression

$$\mathcal{A}[\![k[\![\text{TRUE}]\!]]\!]\varrho\theta\langle\kappa_P,\psi\rangle$$
$$\overset{COM}{=} \mathcal{A}[\![\text{CONST TT}]\!]\varrho\theta\langle\kappa_P,\psi\rangle$$
$$\overset{AS}{=} \theta\langle(TT)@\kappa_P,\psi\rangle$$
$$\overset{AL}{=} \theta\langle(\mathcal{E}[\![\text{TRUE}]\!]\varrho^*\kappa_\beta\psi)@\kappa_P,\ \mathcal{E}[\![E]\!]\varrho^*\kappa_\psi\psi\rangle$$

FALSE-expression

Identical to proof for TRUE-expression.

Identifier

$$\mathcal{A}[\![k[\![\xi]\!]]\!]\varrho\theta\langle\kappa_P,\psi\rangle$$
$$\overset{COM}{=} \mathcal{A}[\![\text{LOAD }\xi]\!]\varrho\theta\langle\kappa_P,\langle z,\kappa_I,\kappa_O\rangle\rangle$$
$$\overset{AS}{=} \theta\langle(\text{Contents}\,(\varrho[\![\xi]\!]/L)\,z)@\kappa_P,\langle z,\kappa_I,\kappa_O\rangle\rangle$$
$$\overset{AL}{=} \theta\langle(\mathcal{E}[\![E]\!]\varrho^*\kappa_\beta\psi)@\kappa_P,\ \mathcal{E}[\![\xi]\!]\varrho^*\kappa_\psi\psi\rangle$$

Constant

$$\mathcal{A}[\![k[\![k]\!]]\!]\varrho\theta\langle\kappa_P,\psi\rangle$$
$$\overset{COM}{=} \mathcal{A}[\![\text{CONST }\mathcal{K}[\![k]\!]]\!]\varrho\theta\langle\kappa_P,\psi\rangle$$
$$\overset{AS}{=} \theta\langle\mathcal{K}[\![k]\!]@\kappa_P,\psi\rangle$$
$$\overset{AL}{=} \theta\langle(\mathcal{E}[\![k]\!]\varrho^*\kappa_\beta\psi)@\kappa_P,\ \mathcal{E}[\![E]\!]\varrho^*\kappa_\psi\psi\rangle$$

Negation-expression

$$\mathcal{A}[\![k[\![\neg\varepsilon]\!]]\!]\varsigma\theta<\kappa_P,\psi>$$

$$\overset{COM}{=}\ \mathcal{A}[\![k[\![\varepsilon]\!];NEG]\!]\varsigma\theta<\kappa_P,\psi>$$

$$\overset{AS}{=}\ \mathcal{A}[\![k[\![\varepsilon]\!]]\!]\varsigma\{\mathcal{A}[\![NEG]\!]\varsigma\theta\}<\kappa_P,\psi>$$

$$\overset{III}{=}\ \mathcal{A}[\![NEG]\!]\varsigma\theta<(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi)@\kappa_P,\ \mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi>$$

$$\overset{AS}{=}\ \theta<(\neg(\mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\beta\psi))@\kappa_P,\ \mathcal{E}[\![\varepsilon]\!]\varsigma^*\kappa_\psi\psi>$$

$$\overset{AL}{=}\ \theta<(\mathcal{E}[\![\neg\varepsilon]\!]\varsigma^*\kappa_\beta\psi)@\kappa_P,\ \mathcal{E}[\![\neg\varepsilon]\!]\varsigma^*\kappa_\psi\psi>$$

Dyadic expressions

$$\mathcal{A}[\![k[\![\varepsilon_1\oplus\varepsilon_2]\!]]\!]\varsigma\theta<\kappa_P,\psi>$$

$$\overset{COM}{=}\ \mathcal{A}[\![k[\![\varepsilon_1]\!];k[\![\varepsilon_2]\!];DO\ \oplus]\!]\varsigma\theta<\kappa_P,\psi>$$

$$\overset{AS}{=}\ \mathcal{A}[\![k[\![\varepsilon_1]\!]]\!]\varsigma\{\mathcal{A}[\![k[\![\varepsilon_2]\!];DO\ \oplus]\!]\varsigma\theta\}<\kappa_P,\psi>$$

$$\overset{III}{=}\ \mathcal{A}[\![k[\![\varepsilon_2]\!]]\!]\varsigma\{\mathcal{A}[\![DO\ \oplus\ \varsigma\theta\}<(\mathcal{E}[\![\varepsilon_1]\!]\varsigma^*\kappa_\beta\psi)@\kappa_P,\mathcal{E}[\![\varepsilon_1]\!]\varsigma\kappa_\psi\psi>$$

$$\overset{III}{=}\ \mathcal{A}[\![DO\ \oplus]\!]\varsigma\theta<((\mathcal{E}[\![\varepsilon_2]\!]\varsigma^*\kappa_\beta)(\mathcal{E}[\![\varepsilon_1]\!]\varsigma^*\kappa_\psi\psi))$$
$$@((\mathcal{E}[\![\varepsilon_1]\!]\varsigma^*\kappa_\beta\psi)@\kappa_P),\ (\mathcal{E}[\![\varepsilon_2]\!]\varsigma^*\kappa_\psi)(\mathcal{E}[\![\varepsilon_1]\!]\varsigma^*\kappa_\psi\psi)>$$

$$\overset{AS}{=}\ \theta<((\mathcal{E}[\![\varepsilon_1]\!]\varsigma^*\kappa_\beta\psi)\oplus((\mathcal{E}[\![\varepsilon_2]\!]\varsigma^*\kappa_\beta)(\mathcal{E}[\![\varepsilon_1]\!]\varsigma^*\kappa_\psi\psi)))@\kappa_P,$$
$$(\mathcal{E}[\![\varepsilon_2]\!]\varsigma^*\kappa_\psi)(\mathcal{E}[\![\varepsilon_1]\!]\varsigma^*\kappa_\psi\psi)>$$

$$\overset{AL}{=}\ \theta<(\mathcal{E}[\![\varepsilon_1\oplus\varepsilon_2]\!]\varsigma^*\kappa_\beta\psi)@\kappa_P,\ \mathcal{E}[\![\varepsilon_1\oplus\varepsilon_2]\!]\varsigma^*\kappa_\psi\psi>$$

end of proof
for (III)

end of proof
for Theorem 12

11 PREDICATE-TRANSFORMER THEORY - VIA ASSLA

In an earlier section of this paper (page 73 - 75) I have defined a theory wp1 - wp11 for SNAIL. I then used

$$wp(f, Q(s)) \equiv \mathcal{E}[\![f]\!]s\,Q(s)$$

and showed that the theory was satisfied under this interpretation.

Since this approach I have defined a compilation from SNAIL to ASSLA and proved that this compilation is correct in a certain specified sense. This opens another possibility for proving Theorem 10:

We gives an interpretation of wp1 - wp11 in terms of ASSLA:

$$wp(f, Q(s)) \equiv \mathcal{A}[\![h[\![f]\!]]\!]s\,Q(s)$$

where $Q \in PRED_2 = [ENV \to [T \to TF]]$.

Now theorem 10 can be proved using theorem 12 and the following:

Theorem 13: The predicate-transformer theory is satisfied by the mathematical semantic model represented by the compilation, h, and the semantic function for ASSLA, $\mathcal{A}$.

11.1 Proof for Theorem 13:

wp1 - Command lists

$$
\begin{aligned}
& wp(\gamma_1;\gamma_2, Q) \\
\equiv\ & \mathcal{A}[\![h[\![\gamma_1;\gamma_2]\!]]\!]_\varsigma Q \\
\equiv\ & \mathcal{A}[\![h[\![\gamma_1]\!]; h[\![\gamma_2]\!]]\!]_\varsigma Q \\
\equiv\ & \mathcal{A}[\![h[\![\gamma_1]\!]]\!]_\varsigma \circ \mathcal{A}[\![h[\![\gamma_2]\!]]\!]_\varsigma Q \} \\
\equiv\ & wp(\gamma_1, wp(\gamma_2, Q))
\end{aligned}
$$

wp2 - Assignment

$$
\begin{aligned}
& wp(\xi := \varepsilon, Q)\ \sigma \\
\equiv\ & \mathcal{A}[\![h[\![\xi := \varepsilon]\!]]\!]_\varsigma Q\ \sigma \\
\equiv\ & \mathcal{A}[\![k[\![\varepsilon]\!]; STORE]\!]_\varsigma Q\ \sigma \qquad (\sigma = \langle\langle \kappa_D, \langle z, \kappa_I, \kappa_O\rangle\rangle) \\
\equiv\ & \mathcal{A}[\![k[\![\varepsilon]\!]]\!]_\varsigma \circ \mathcal{A}[\![STORE]\!]_\varsigma Q \}\ \langle\langle \kappa_D, \langle z, \kappa_I, \kappa_O\rangle\rangle \\
(*) \equiv\ & \mathcal{A}[\![STORE]\!]_\varsigma Q\ \langle\langle \varepsilon_\psi @ \kappa_D, \langle z, \kappa_I, \kappa_O\rangle\rangle \\
\equiv\ & Q\ \langle\langle \kappa_D, \langle Assign(\varsigma[\![\xi]\!]L, \varepsilon_\psi) z, \kappa_I, \kappa_O\rangle\rangle \\
(**) \equiv\ & Q\ (\varepsilon/\xi)\ \sigma
\end{aligned}
$$

where (*) follows from (III) page
and (**) from the definition of the auxilliary function assign.

wp3 - TEST-command

$$wp(\text{TEST } \varepsilon \text{ DO } f_1 \text{ OR } f_2 \text{ TSET}, Q)\ \sigma$$

$$\equiv \mathcal{A}[\![\, h[\![\text{TEST } \varepsilon \text{ DO } f_1 \text{ OR } f_2 \text{ TSET}]\!]\, \varrho\, Q\, \sigma$$

$$\equiv \mathcal{A}[\![\text{BEGIN } k[\![\varepsilon]\!];\ \text{JF } \xi_1;\ h[\![f_1]\!];\ \text{J}\xi_2;\ \xi_1 : h[\![f_2]\!];\ \xi_2 : e \text{ END}]\!]\, \varrho\, Q\, \sigma$$

$$\equiv \theta_0\, \sigma$$

where

$$\theta_0 \equiv \mathcal{A}[\![\, k[\![\varepsilon]\!];\ \text{JF } \xi_1;\ h[\![f_1]\!];\ \text{J}\xi_2\,]\!]\, \varrho'\, Q_1$$

$$\theta_1 \equiv \mathcal{A}[\![\, h[\![f_2]\!]\,]\!]\, \varrho'\, Q_2$$

$$Q_2 \equiv Q$$

$$\varrho' = \varrho\,[Q_1, Q_2 \,/\, \xi_1, \xi_2\,]$$

It should be noticed that this is a simple set of equations not involving fixpoints.

$$\equiv \mathcal{A}[\![\, k[\![\varepsilon]\!];\ \text{JF } \xi_1;\ h[\![f_1]\!];\ \text{J}\xi_2\,]\!]\, \varrho'\, Q_1\, \langle \mu_P, \psi \rangle$$

$$\equiv \mathcal{A}[\![\, \text{JF } \xi_1\,]\!]\, \varrho'\, \{\mathcal{A}[\![\, h[\![f_1]\!];\ \text{J}\xi_2\,]\!]\, \varrho' Q_1\}\, \langle \varepsilon_\psi \,@\, \mu_P, \psi \rangle$$

$$\equiv (\varepsilon_\psi \wedge \mathcal{A}[\![\, h[\![f_1]\!]\,]\!]\, \varrho'\, \{\mathcal{A}[\![\, \text{J}\xi_2\,]\!]\, \varrho' Q_1\}\, \sigma) \vee (\neg \varepsilon_\psi \wedge Q_1\, \sigma)$$

$$\equiv (\varepsilon_\psi \wedge wp(f_1, \mathcal{A}[\![\, \text{J}\xi_2\,]\!]\, \varrho' Q_1\, \sigma)) \vee (\neg \varepsilon_\psi \wedge \mathcal{A}[\![\, h[\![f_2]\!]\,]\!]\, \varrho' Q_2\, \sigma)$$

$$\equiv (\varepsilon_\psi \wedge wp(f_1, Q)\, \sigma) \vee (\neg \varepsilon_\psi \wedge wp(f_2, Q)\, \sigma)$$

$$\equiv ((\varepsilon \wedge wp(f_1, Q)) \vee (\neg \varepsilon \wedge wp(f_2, Q)))\ \sigma$$

where it is used that Q cannot depend on ξ_1 or ξ_2 since these are new unused identifiers.

wp4 - IF-command

$$wp(\text{IF } \varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n \text{ FI}, Q)\,\sigma$$

$$\equiv \mathcal{A}[\![h[\![\text{IF } \varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n \text{ FI}]\!]\varrho Q\sigma$$

$$\equiv \mathcal{A}[\![\text{BEGIN } k[\![\varepsilon_1]\!]; JT\,\xi_1; \ldots; k[\![\varepsilon_n]\!]; JT\,\xi_n; \text{ABORT}$$

$$\xi_1 : h[\![\gamma_1]\!]; J\,\xi_{n+1}; \ldots; \xi_n : h[\![\gamma_n]\!]; \xi_{n+1} : e \text{ END}]\!]\varrho Q\sigma$$

$$\equiv (\exists i\ (\varepsilon_i)_\psi) \wedge (\mathcal{A}[\![h[\![\gamma_{\min B_\psi}]\!]]\!]\varrho Q\sigma)$$

since execution of Abort would yield $\perp$ = false .

$$\equiv (B_\psi \neq \varnothing) \wedge wp(\gamma_{\min B_\psi}, Q)\,\sigma$$

$$\equiv (B \neq \varnothing \wedge wp(\gamma_{\min B}, Q))\,\sigma$$

(many details analogous with those in the proof for TEST are omitted).

wp5 - DO-command

The proof will be omitted. It is rather difficult. But the problems are of a technical kind such as indexing etc.

There are no intrinsic problems different from those represented by IF and WHILE.

wp6 - WHILE-command

$$
\begin{aligned}
& wp\,(WHILE\ \varepsilon\ DO\ \gamma\ OD,\ Q)\ \sigma \\
\equiv\ & \mathcal{A}[\![h[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]]\!]\varsigma\, Q\, \sigma \\
\equiv\ & \mathcal{A}[\![BEGIN\ e;\ \xi_1 := k[\![\varepsilon]\!];\ JF\,\xi_2;\ h[\![\gamma]\!];\ J\,\xi_1;\ \xi_2 := e\ END]\!]\varsigma\, Q\, \sigma \\
\equiv\ & (FIX\ \theta'.\ \mathcal{A}[\![k[\![\varepsilon]\!];\ JF\,\xi_2;\ h[\![\gamma]\!]]\!]\varsigma\,\theta',\ Q)\ \langle \mu_P, \psi \rangle \\
\equiv\ & \mathcal{A}[\![\ JF\,\xi_2;\ h[\![\gamma]\!]]\!]\varsigma\ \{\mathcal{A}[\![h[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]]\!]\varsigma\, Q\} \\
& \qquad \langle (\mathcal{E}[\![\varepsilon]\!]\varsigma^* \kappa_\beta\, \psi)@\mu_P,\ \mathcal{E}[\![\varepsilon]\!]\varsigma^* \kappa_\psi\, \psi \rangle \\
\equiv\ & \mathcal{E}[\![\varepsilon]\!]\varsigma^* \kappa_\beta\, \psi \Rightarrow \mathcal{A}[\![h[\![\gamma]\!]]\!]\varsigma\ \{\mathcal{A}[\![WHILE\ \varepsilon\ DO\ \gamma\ OD]\!]\varsigma Q\sigma,\ Q\sigma
\end{aligned}
$$

where many details analogous with those in the proof for TEST are omitted.

Now the proof can be finished in a way analogous to the proof on page **89** (proof for wp6 when the predicate-transformer theory is interpreted directly in terms of the mathematical semantics for SNAIL).

wp7 - Declaration (not empty)

$$wp\,(\text{BEGIN DEF } \xi = \varepsilon;\ \gamma\ \text{END},\ Q(\varsigma))\ \sigma$$

$$\equiv \mathcal{A}[\![\, h[\![\text{BEGIN DEF } \xi = \varepsilon;\ \gamma\ \text{END}]\!]\varsigma\, Q(\varsigma)\ \sigma$$

$$\equiv \mathcal{A}[\![\, \text{BLOCK } k[\![\varepsilon]\!];\ \text{DEF } \xi;\ \text{STORE } \xi;\ h[\![\gamma]\!]\ \text{END}]\!]\varsigma\, Q(\varsigma)\ \langle \mu_P, \psi\rangle$$

$$\equiv \text{New}(\varsigma)\,\{\lambda\alpha.\ \mathcal{A}[\![\text{STORE } \xi;\ h[\![\gamma]\!]]\!]\varsigma[\alpha/\xi]\ Q(\varsigma)\}\ \langle \varepsilon_\psi @ \mu_P, \psi\rangle$$

$$\equiv \mathcal{A}[\![\text{STORE } \xi]\!]\varsigma^+\{\mathcal{A}[\![\, h[\![\gamma]\!]]\!]\varsigma^+ Q(\varsigma)\}\ \langle \varepsilon_\psi @ \mu_P, \psi\rangle$$

where α^+ is a new unused location and $\varsigma^+ = \varsigma[\alpha^+/\xi]$.

$$\equiv \mathcal{A}[\![\, h[\![\gamma]\!]]\!]\varsigma^+ Q(\varsigma)\ \langle \mu_P, \langle \text{Assign}(\varsigma^+[\![\xi]\!]/L, \varepsilon_\psi), \mu_I, \mu_O\rangle$$

using an argument analogous as that on page 91 we can now finish the proof:

$$\equiv \exists k \in V\ [\, \xi_\psi = k \wedge wp(\xi := \varepsilon;\ \gamma, Q(k/\xi)\varsigma\ \sigma\,]$$

$$\equiv (\exists k \in V\ [\, \xi = k \wedge wp(\xi := \varepsilon;\ \gamma, Q(k/\xi)\varsigma\,])\ \sigma$$

wp8 - Declaration (empty)

$$wp(\text{BEGIN } e;\ \gamma\ \text{END}, Q)$$

$$\equiv \mathcal{A}[\![\, h[\![\text{BEGIN } e;\ \gamma\ \text{END}]\!]]\!]\varsigma\, Q$$

$$\equiv \mathcal{A}[\![\, h[\![\gamma]\!]]\!]\varsigma\, Q$$

$$\equiv wp(\gamma, Q)$$

wp9 - READ-command

$$
\begin{aligned}
& wp(READ(\xi), Q)\,\sigma \\
\equiv\ & \mathcal{A}[\![h[\![READ(\xi)]\!]]\!]\varrho Q\,\sigma \\
\equiv\ & \mathcal{A}[\![READ; STORE\ \xi]\!]\varrho Q\langle\mu_P, \langle z, \mu_I, \mu_O\rangle\rangle \\
\equiv\ & \mathcal{A}[\![STORE\ \xi]\!]\varrho\theta\langle(\mu_I \downarrow 1)@\mu_P, \langle z, \mu_I+2, \mu_O\rangle\rangle \\
\equiv\ & Q\langle\mu_P, \langle Assign(\varrho[\![\xi]\!]L, (\mu_I \downarrow 1))z, \mu_I, \mu_O\rangle\rangle \\
\equiv\ & Q\,((\mu_I \downarrow 1), (\mu_I+2)/\xi, \mu_I)\,\sigma
\end{aligned}
$$

by the definition of the auxilliary function Assign and the remarks about wp9 on page 75.

wp10 - WRITE-command

$$
\begin{aligned}
& wp(WRITE(\varepsilon), Q)\,\sigma \\
\equiv\ & \mathcal{A}[\![h[\![WRITE(\varepsilon)]\!]]\!]\varrho Q\,\sigma \\
\equiv\ & \mathcal{A}[\![k[\![\varepsilon]\!]; WRITE]\!]\varrho Q\langle\mu_P, \langle z, \mu_I, \mu_O\rangle\rangle \\
\equiv\ & \mathcal{A}[\![WRITE(\varepsilon)]\!]\varrho\theta\langle\varepsilon_\psi @ \mu_P, \langle z, \mu_I, \mu_O\rangle\rangle \\
\equiv\ & Q\langle\mu_P, \langle z, \mu_I, \mu_O @ \varepsilon_\psi\rangle\rangle \\
\equiv\ & Q\,((\mu_O @ \varepsilon_\psi)/\mu_O)\,\sigma
\end{aligned}
$$

wp11 - Empty-command

$$
\begin{aligned}
& wp(e, Q) \\
\equiv\ & \mathcal{A}[\![h[\![e]\!]]\!]\varrho Q \\
\equiv\ & \mathcal{A}[\![e]\!]\varrho Q \\
\equiv\ & Q
\end{aligned}
$$

end of proof
for Theorem 13

It is interesting to compare the proof for Theorem 13 to the proof for Theorem 10 . There seems to be very little difference in proof-methods and proof-complexity.

In most cases the proofs using ASSLA are a little longer than the corresponding proofs using an interpretation directly in terms of SNAIL. At the other hand it sometimes becomes easier to explain the different phases of a proof when the ASSLA-interpretation is used. This is due to the fact, that the compilation from SNAIL to ASSLA in a very natural way splits the execution of a SNAIL-command into a number of small welldefined steps.

12 CONCLUSION

In this paper I have investigated different semantic approaches. The example-language, SNAIL, is representative for most problems met in practical programming languages. Yet SNAIL is a rather small language. This is due to the fact that each problem as far as possible only is represented by one single command, while a normal programming language usually has many analogous commands. Most features met in SNAIL are very well known and standard ingredients in all common programming languages. An exception from this is the two nondeterministic commands proposed by Dijkstra. They can be seen as generalizations of the usual TEST-command and WHILE-command. It should be stressed that I give a deterministic treatment of these commands. However a pure nondeterministic approach could be done in the used notation without much additional effort.

The notation used to describe mathematical semantics are developed in Oxford by Strachey, Scott and others. The particular division into syntactic domains, value domains, syntax, semantic functions, semantic equations and auxilliary functions is taken from Ligler.

Hoare and Lauer define 4 different semantic description-methods. The first two of these can be classified as constructive approaches while the last two are implicit approaches. In terms of logic the first two are models while the last two are (satisfied) theories. In /H1/ Hoare and Lauer describes the use of these 4 types of semantics on a very simple language consisting of assignment, WHILE-command and normal sequential command-sequencing. In /L/ Lauer uses the same 4 semantics on a much more complicated language (same complexity as SNAIL). This language involves nondeterministic commands. This is handled by using sets instead of single elements. However this demands the use of a number of non-standard set-operators which in some degree obscures the readability and clarity. In chapter 4 I used the 4 different approaches on SNAIL. The formulation was pretty close to that used by Hoare and Lauer though some change of notation had been done to ease a later transformation to a formulation similar to that used in the mathematical semantics developed in Oxford.

Chapter 5 was separated into two distinct parts. The first 4 sections was a reiteration of the definitions and theorems of chapter 4, but now the formulation and the used notation is following the lines given by mathematical semantics. It was shown that the two constructive models were nothing else than "standard" mathematical semantics. The second part of chapter 5 was dedicated to a further development of some ideas connecting to Hoares and Lauers work. First of all a third theory was introduced. This was Dijkstra's weakest predicate-transformer theory. In many ways this theory is analogous to one of the original ones - the deductive theory given by Hoare. The main difference is that Dijkstra demands total correctness and both sufficiency and nescessarity while Hoare only demands partial correctness and sufficiency. Moreover it was investigated which gains could be received by introducing the notion of command-continuations. It turned out that this gives a very nice and smooth interpretation of Dijkstra's theory. A direct connection was established between command-continuations representing program-execution and predicate-transformers representing program-verification.

In the next chapter this connection was further elaborated. It was shown that the original mathematical continuation-semantics given for SNAIL in chapter 3 is equivalent to the first of Hoares and Lauers constructive models when this is equipped with command-continuations. Next Dijkstra's theory was interpreted directly in terms of the given mathematical continuation-semantics, and it was proved that the theory is satisfied by this model.

The last five chapters of this paper dealt with compiler correctness. It was first discussed what should be meant by the word "compiler correctness". It turned out, that this is nothing else than comutation of certain semantic functions and that most theorems into this field can be given an algebraic formulation using the notions taken from the theory dealing with homomorphisms. First a very simple correctness proof was given to explain the used notation. Then an assembly language, ASSLA, was

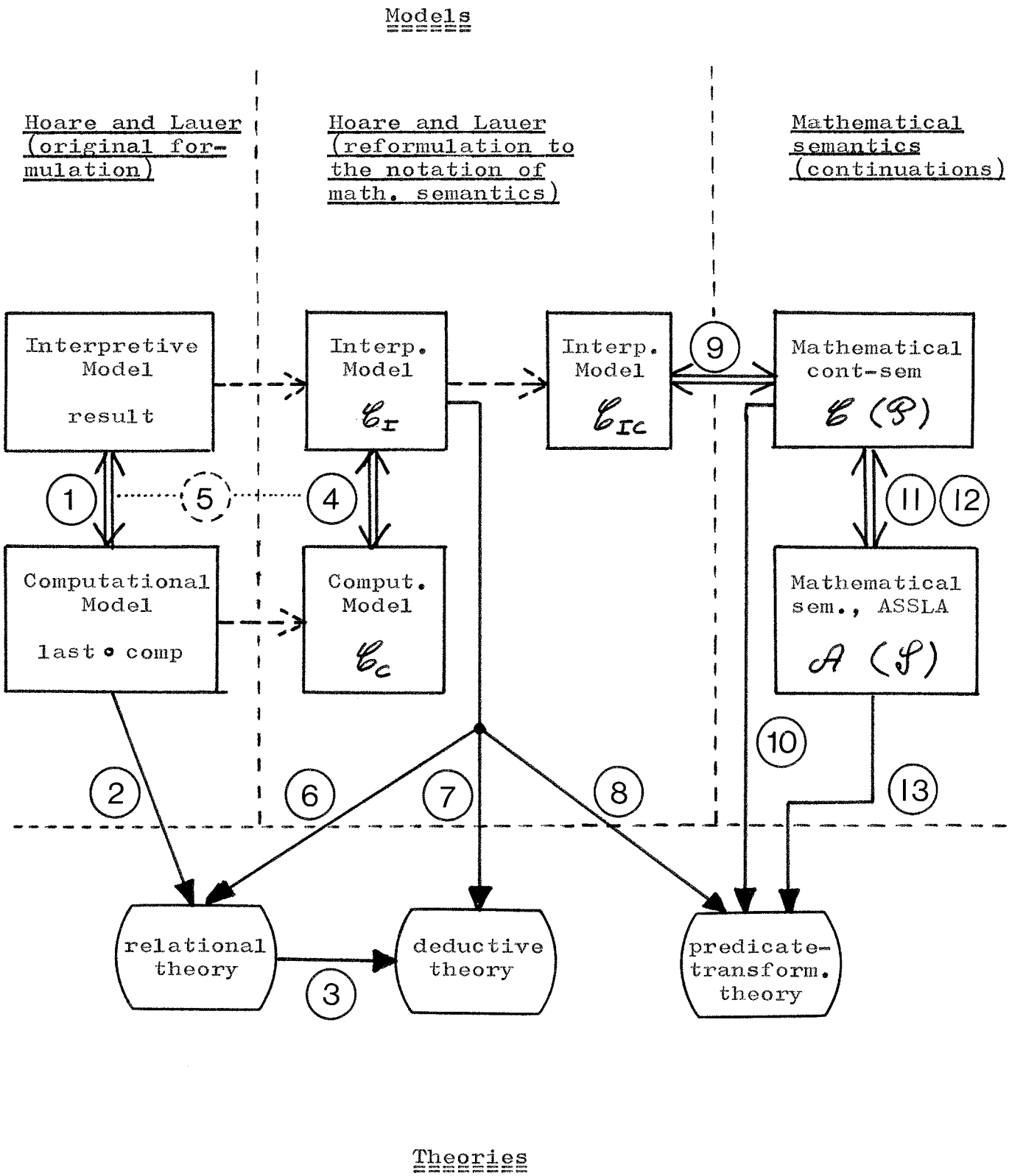
defined. This language is especially fit to act as target-language for a SNAIL-compiler. Such a compilation was defined and a nontrivial proof was given for the correctness. At last it was shown that this compilation can be used to give an interpretation of Dijkstra's theory via ASSLA. It was investigated whether this gave any gain in the clarity of the used proofs, but this could not be seen.

As a summary it could be said that this paper has pointed out a strong congruence between three apparently very different semantic approaches: the mathematical semantics developed in Oxford, the ideas described in /H1/ by Hoare and Lauer and the weakest predicate-transformer theory introduced by Dijkstra. Besides this a non-trivial compiler has been defined and proved correct.

One of the main purposes with this paper has been to compare different semantic approaches. However the reader should not only see such different semantics as competitors. It is very fruitfull to see them as (quoting from /H1/)"consistent and complementary" description-methods.

SNAIL is a rather strong language. It omits procedures, transfer of control and data-structures. It is my believe that these features can be incorporated without too much additional effort. An exception is transfer of control which would be difficult to include in those description-methods not involving command-continuations.

12.1 Diagram showing the connection between the different theorems



Explanation of used symbols:

Model

Theory

Tranformation of model to another notation

Indicates that the theory pointed to is satisfied by the pointing model.

Indicates equivalence or compiler correctness

Number of used Theorem

appendix A:

EVALUATION OF GUARDS FOR DIJKSTRA'S GUARDED COMMANDS

In /D1/ and /D2/ Dijkstra describes two nondeterministic commands. We have the following syntax for these new commands (quoting from the semantic definition of SNAIL):

$\gamma ::=$ ..... | IF ω FI | DO ω OD | .....

$\omega ::= \varepsilon_1 \rightarrow \gamma_1 \# \ldots\ldots \# \varepsilon_n \rightarrow \gamma_n \quad (n \geq 1)$

$\varepsilon ::=$ .....

The idea proposed by Dijkstra is that each (boolean) expression $\varepsilon_1, \varepsilon_2, \ldots\ldots, \varepsilon_n$ shall act as a guard for the corresponding commands $\gamma_1, \gamma_2 \ldots, \gamma_n$. By this I mean that γ_i can only be executed if ε_i is true. If more than one guard is true at the same time we make a nondeterministic choice between those.

There are two questions to be raised:

1. Can guards have sideeffects?
2. Can evaluation of guards lead to program abortion or infinite loop?

In /D1/ and /D2/ Dijkstra has no comments about this. Nevertheless these two questions are essential for an unambigious definition of the semantics for IF- and DO-commands.

There are (at least) 6 different ways to evaluate a set of guards:

1. All guards are evaluated. Evaluation is done in a prespecified order. The first true is chosen.

2. As 1, but among the true guards one is chosen at random.

3. All guards are evaluated. Evaluation is done in random order. The first guard evaluated to true is chosen.

4. As 3, but among the true guards one is chosen at random.

5. As 1, but evaluation of guards is stopped as soon as the first true is found. This is selected.

6. As 3, but evaluation of guards is stopped as soon as the first true is found. This is selected.

It is easy to verify that in the general case all these 6 possibilities are semantical different. That means that there exist programs for which different results are yielded for the very same inputs.

If the questions raised on the last page can be answered:

1. Guards evaluates without sideeffects.
2. Guards cannot evaluate to $\perp$.
   (that means no abortion and no infinite loops).

then the above mentioned 6 possibilities group into 2 classes. 1,3,5 and 6 becomes semantical equivalent. So does 2 and 4.

Note: The difference between the two claases 1,3,5,6 and 2,4 is conceptually exactly the same as the difference between the evaluation of $\vee$ (or) in SNAIL and LISP respectively. In the first case all operands/expressions are evluated. In the second case evaluation is stopped as soon as "we know enough".

	T	F	$\perp$
T	T	T	$\perp$
F	T	F	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$

SNAIL

	T	F	$\perp$
T	T	T	T
F	T	F	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$

LISP

For an explanation of used notation please see page 11-12.

Example:

A very common reason for sideeffects is the use of input- and output routines.

It will be supposed that the procedure READ does the following:

1. Reads an element from a sequential inpustream. This stops when such an element is read or the given syntax for inputs is violated.

2. If a legal input was encountered this is assigned to the variable INCHAR and READ returns the value true.

3. If a nonlegal input was encountered the value of INCHAR becomes undefined and READ returns the value false.

We want to write a program which alternately reads a legal input into INCHAR (after possibly having skipped a number of bad inputs) and calls ACTION to perform some work on this input . The program should run for ever.

Please note that READ has sideeffects. It alters the variable INCHAR (and the inputstream).

A first proposal for such a program might be:

```
P 1 :      DO
              READ  →  ACTION( INCHAR )  #
           ¬ READ  →  DUMMY
           OD
```

It is rather easy to verify that this program does not do what we wanted it to do in either of the 6 possibilities.

There seems to be two problems:

1. Exution can stop. This happens if neither of the two guards are evaluated to true.

2. We may skip a legal input. This can happen because evaluation of READ and ¬ READ both alters the value of INCHAR.

We make a new proposal:

```
P 2 :    DO
            READ   →  ACTION( INCHAR )   #
          ¬ READ   →  DUMMY              #
            TRUE   →  ACTION( INCHAR )
         OD
```

It can be verified that P 2 will work as intended if method 5 is used for evaluation of guards. In all other possibilities it will be incorrect.

This might tempt us to argue that 5 is the only usable evaluation method. This would be wrong. The program P 2 is not a representative for good and readable programming. It requires quite a lot of thought to persuade oneself that the inclusion of the last guarded line works correct. Since one of Dijkstra's ideas in proposing the use of these nondeterministic commands was to ease understandable programming a more reasonable conclusion is:

1. Evaluation of guards must not have sideeffects, program abortion or infinite loops.

2. The nice user should not write programs whose correctness depends on which of the evaluation methods is used. Or in other words - the user should not make any assumptions about the order of guardevaluation.

A program satisfying these claims:

```
P 3 :      BOO := READ
           DO
              BOO → ACTION( INCHAR ); BOO := READ #
            ¬ BOO → BOO := READ
           OD
```

P 3 will work correct for all 6 different evaluationmethods. (BOO is supposed to be a simple boolean variable).

In /D4/ Dijkstra states that the IF- and DO-command yields undefined if one or more guards is undefined (program abortion or infinite loop). Nothing is said about sideeffects.

COLLECTION OF DEFINITIONS

This appendix is merely a collection of the given definitions. It is intended to be used for reference purposes.

The included definitions are:

1. Mathematical semantics for SNAIL (definition of the functions $\mathscr{E}$, $\mathscr{E}$, $\mathscr{D}$ and $\mathscr{F}$)
2. Interpretive model used on SNAIL
3. Computational model used on SNAIL
4. Relational theory used on SNAIL
5. Deductive theory used on SNAIL
6. $\mathscr{E}_I$ used on MINI-SNAIL
7. $\mathscr{E}_C$ used on MINI-SNAIL
8. Predicate-transformer theory used on SNAIL
9. Mathematical semantics for ASSLA (definition of the function $\mathscr{A}$)
10. Compilation from SNAIL to ASSLA

$$\mathcal{E}[\![\gamma_1;\gamma_2]\!]\varsigma\theta = \mathcal{E}[\![\gamma_1]\!]\varsigma\,\{\mathcal{E}[\![\gamma_2]\!]\varsigma\theta\}$$

$$\mathcal{E}[\![\xi := \varepsilon]\!]\varsigma\theta = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\lambda\beta.\ Update(\varsigma[\![\xi]\!]\,L,\beta)\theta\}$$

$$\mathcal{E}[\![\text{TEST } \varepsilon \text{ DO } \gamma_1 \text{ OR } \gamma_2 \text{ TSET}]\!]\varsigma\theta = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{Cond(\mathcal{E}[\![\gamma_1]\!], \mathcal{E}[\![\gamma_2]\!])\varsigma\theta\}$$

$$\mathcal{E}[\![\text{IF } \omega \text{ FI}]\!]\varsigma\theta = \mathcal{G}[\![\omega]\!]\varsigma\,\{Oracle(\theta,\bot)\}$$

$$\mathcal{E}[\![\text{DO } \omega \text{ OD}]\!]\varsigma\theta = FIX\ \theta'.\ \mathcal{G}[\![\omega]\!]\varsigma\,\{Oracle(\theta',\theta)\})$$

$$\mathcal{E}[\![\text{BEGIN } \delta;\gamma \text{ END}]\!]\varsigma\theta = \mathcal{D}[\![\delta]\!]\varsigma\,\{\lambda\varsigma.\ \mathcal{E}[\![\gamma]\!]\varsigma\theta\}$$

$$\mathcal{E}[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]\varsigma\theta = FIX\ \theta'.\ \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{Cond(\mathcal{E}[\![\gamma]\!]\varsigma\theta', \theta)\}$$

$$\mathcal{E}[\![\text{READ } (\xi)]\!]\varsigma\theta = Read(\varsigma[\![\xi]\!]\,L)\,\theta$$

$$\mathcal{E}[\![\text{WRITE } (\varepsilon)]\!]\varsigma\theta = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\lambda\beta.\ Write(\beta)\theta\}$$

$$\mathcal{E}[\![e]\!]\varsigma\theta = \theta$$

$$\mathcal{E}[\![\text{TRUE}]\!]\varsigma\kappa = \kappa(TT)$$

$$\mathcal{E}[\![\text{FALSE}]\!]\varsigma\kappa = \kappa(FF)$$

$$\mathcal{E}[\![\xi]\!]\varsigma\kappa = \lambda\psi.\ \kappa(Contents(\varsigma[\![\xi]\!]\,L)(\psi\downarrow 1))$$

$$\mathcal{E}[\![k]\!]\varsigma\kappa = \kappa(\mathcal{K}[\![k]\!])$$

$$\mathcal{E}[\![\neg\varepsilon]\!]\varsigma\kappa = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\lambda\beta.\ \kappa(\neg\beta)\}$$

$$\mathcal{E}[\![\varepsilon_1 \oplus \varepsilon_2]\!]\varsigma\kappa = \mathcal{E}[\![\varepsilon_1]\!]\varsigma\,\{\lambda\beta_1.\ \mathcal{E}[\![\varepsilon_2]\!]\varsigma\,\{\lambda\beta_2.\ \kappa(\beta_1\oplus\beta_2)\}\}$$

$$\mathcal{D}[\![\text{DEF } \xi = \varepsilon]\!]\varsigma\chi = \mathcal{E}[\![\varepsilon]\!]\varsigma\,\{\lambda\beta.\ New(\varsigma)\,\{\lambda\alpha.\ Update(\alpha,\beta)\{\chi(\varsigma[\alpha/\xi])\}\}\}$$

$$\mathcal{D}[\![e]\!]\varsigma\chi = \chi(\varsigma)$$

$$\mathcal{G}[\![\varepsilon_1 \to \gamma_1 \,\#\, \cdots\cdots \,\#\, \varepsilon_n \to \gamma_n]\!]\varsigma\eta$$
$$= \eta\langle\langle \mathcal{E}[\![\varepsilon_1]\!]\varsigma, \mathcal{E}[\![\gamma_1]\!]\varsigma\rangle, \ldots\ldots, \langle \mathcal{E}[\![\varepsilon_n]\!]\varsigma, \mathcal{E}[\![\gamma_n]\!]\varsigma\rangle\rangle$$

2 Interpretive model used on SNAIL

$next(\psi, \gamma_0) =$

CASES γ_0 IN

$\xi := \varepsilon; \gamma' \rightarrow (assign_I(\xi, \varepsilon_\psi)\psi, \gamma')$,

TEST ε DO γ_1 OR γ_2 TSET$;\gamma' \rightarrow$ $[\varepsilon_\psi \rightarrow (\psi, \gamma_1;\gamma'),$
$\neg\varepsilon_\psi \rightarrow (\psi, \gamma_2;\gamma')]$,

IF $\varepsilon_1 \rightarrow \gamma_1 \# \cdots \# \varepsilon_n \rightarrow \gamma_n$ FI$;\gamma' \rightarrow$ $[(\varepsilon_1)_\psi \rightarrow (\psi, \gamma_1;\gamma'),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow (\psi, \gamma_n;\gamma')]$,

DO $\varepsilon_1 \rightarrow \gamma_1 \# \cdots \# \varepsilon_n \rightarrow \gamma_n$ OD$;\gamma' \rightarrow$ $[(\varepsilon_1)_\psi \rightarrow (\psi, \gamma_1;\gamma_0),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow (\psi, \gamma_n;\gamma_0),$
$TT \rightarrow (\psi, \gamma')]$,

WHILE ε DO γ OD$;\gamma' \rightarrow$ $[\varepsilon_\psi \rightarrow (\psi, \gamma;\gamma_0),$
$\neg\varepsilon_\psi \rightarrow (\psi, \gamma')]$,

BEGIN DEF $\xi=\varepsilon;\gamma$ END$;\gamma' \rightarrow$ LET $\xi' = New_I(\psi)$ IN
$(augment_I(\psi, \xi'), \xi'=\varepsilon;\gamma(\xi'/\xi);\gamma')$,

BEGIN $e;\gamma$ END$;\gamma' \rightarrow (\psi, \gamma;\gamma')$,

READ $(\xi);\gamma' \rightarrow (Rd_I(\xi)\psi, \gamma')$,

WRITE $(\varepsilon);\gamma' \rightarrow (Wr_I(\varepsilon_\psi)\psi, \gamma')$,

$e;\gamma' \rightarrow (\psi, \gamma')$

ENDCASE

where $\gamma(\xi'/\xi)$ denotes substitution of ξ' for all free ξ .
In this formulation γ' is assumed to be a command or the empty list NIL. Please note that next is undefined if the control state is empty.

3 Computational model used on SNAIL

$comp(\psi, f_0)$ =

CASES f_0 IN

$NIL \rightarrow (\psi)$,

$f_1 ; f_2 \rightarrow comp(\psi, f_1) ; tail \circ comp(last \circ comp(\psi, f_1), f_2)$,

$e \rightarrow (\psi, \psi)$,

$\xi := \varepsilon \rightarrow (\psi, assign_I(\xi, \varepsilon_\psi)\psi)$,

TEST ε DO f_1 OR f_2 TSET $\rightarrow$ $[\ \varepsilon_\psi \rightarrow comp(\psi, f_1),$
$\neg \varepsilon_\psi \rightarrow comp(\psi, f_2)\ ]$,

IF $\varepsilon_1 \rightarrow f_1 \# \cdots \# \varepsilon_n \rightarrow f_n$ FI $\rightarrow$ $[\ (\varepsilon_1)_\psi \rightarrow comp(\psi, f_1),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow comp(\psi, f_n)\ ]$,

DO $\varepsilon_1 \rightarrow f_1 \# \cdots \# \varepsilon_n \rightarrow f_n$ OD $\rightarrow$ $[\ (\varepsilon_1)_\psi \rightarrow comp(\psi, f_1 ; f_0),$
$\vdots$
$(\varepsilon_n)_\psi \rightarrow comp(\psi, f_n ; f_0),$
$TT \rightarrow (\psi)\ ]$,

WHILE ε DO f OD $\rightarrow$ $[\ \varepsilon_\psi \rightarrow comp(\psi, f ; f_0),$
$\neg \varepsilon_\psi \rightarrow (\psi)\ ]$

BEGIN DEF $\xi = \varepsilon$; f END $\rightarrow$ LET $\xi' = New_I(\psi)$ IN
$comp(augment_I(\psi, \xi'), \xi' := \varepsilon ; f(\xi'/\xi))$

BEGIN e ; f END $\rightarrow$ $comp(\psi, f)$,

READ(ξ) $\rightarrow$ $(\psi, Rd_I(\xi)\psi)$,

WRITE(ε) $\rightarrow$ $(\psi, Wr_I(\varepsilon_\psi)\psi)$

ENDCASE

4 Relational theory used on SNAIL

A1. $\psi\,(\mathcal{S}_1;\mathcal{S}_2)\,\psi' \equiv \exists\psi''\,[\psi(\mathcal{S}_1)\,\psi'' \wedge \psi''(\mathcal{S}_2)\,\psi']$

A2. $\psi\,(\xi := \varepsilon)\,\psi' \equiv \psi' = assign_I(\xi, \varepsilon_\psi)\,\psi$

A3. $\psi\,(TEST\ \varepsilon\ DO\ \mathcal{S}_1\ OR\ \mathcal{S}_2\ TSET)\,\psi'$

$\equiv (\varepsilon_\psi \wedge \psi(\mathcal{S}_1)\psi') \vee (\neg\varepsilon_\psi \wedge \psi(\mathcal{S}_2)\psi')$

A4 $\psi\,(IF\ \varepsilon_1 \to \mathcal{S}_1 \#\cdots\# \varepsilon_n \to \mathcal{S}_n\ FI)\,\psi'$

$\equiv B_\psi \neq \emptyset \wedge \psi(\mathcal{S}_{min\,B_\psi})\,\psi'$

A5 $\psi\,(DO\ \varepsilon_1 \to \mathcal{S}_1 \#\cdots\# \varepsilon_n \to \mathcal{S}_n\ OD)\,\psi' \Rightarrow \mathcal{B}_{\psi'} = \emptyset$

A6 $\forall\psi\psi'\,[I(\psi) \wedge \psi(\mathcal{S}_{min\,B_\psi})\,\psi' \Rightarrow I(\psi')] \Rightarrow$

$\forall\psi\psi'\,[I(\psi) \wedge \psi(DO\ \varepsilon_1 \to \mathcal{S}_1 \#\cdots\# \varepsilon_n \to \mathcal{S}_n\ OD)\,\psi' \Rightarrow I(\psi')]$

A7 $\psi\,(WHILE\ \varepsilon\ DO\ \mathcal{S}\ OD)\,\psi' \Rightarrow \neg\varepsilon_{\psi'}$

A8 $\forall\psi\psi'\,[I(\psi) \wedge \varepsilon_\psi \wedge \psi(\mathcal{S})\,\psi' \Rightarrow I(\psi')] \Rightarrow$

$\forall\psi\psi'\,[I(\psi) \wedge \psi(WHILE\ \varepsilon\ DO\ \mathcal{S}\ OD)\,\psi' \Rightarrow I(\psi')]$

A9 $\psi\,(BEGIN\ DEF\ \xi = \varepsilon;\ \mathcal{S}\ END)\,\psi'$

$\equiv \exists\xi'\,[\xi' = New_I(\psi) \wedge augment_I(\psi,\xi')(\xi' := \varepsilon;\ \mathcal{S}(\xi'/\xi))\,\psi']$

A10 $\psi\,(BEGIN\ e;\ \mathcal{S}\ END)\,\psi' \equiv \psi(\mathcal{S})\,\psi'$

A11 $\psi\,(READ\,(\xi))\,\psi' \equiv \psi' = Rd_I(\xi)\,\psi$

A12 $\psi\,(WRITE(\varepsilon))\,\psi' \equiv \psi' = Wr_I(\varepsilon_\psi)\,\psi$

A13 $\psi\,(e)\,\psi' \equiv \psi' = \psi$

where $B_\psi = \{i \mid (\varepsilon_i)_\psi\}$

and $\mathcal{B}_\psi = \emptyset \Rightarrow \mathcal{S}_{min\,B_\psi} = e$.

$I(\psi)$ is any first order logical formula possibly depending on ψ.

D1 $$\frac{P\{f_1\}Q \wedge Q\{f_2\}R}{P\{f_1;f_2\}R}$$

D2 $$P(E/\xi)\{\xi := E\}P$$

D3 $$\frac{(P\wedge E)\{f_1\}Q \wedge (P\wedge\neg E)\{f_2\}Q}{P\{\text{TEST } E \text{ DO } f_1 \text{ OR } f_2 \text{ TSET}\}Q}$$

D4 $$\frac{\forall i\le n\,[(P\wedge E_i)\{f_i\}Q]}{P\{\text{IF } E_1\to f_1 \# \cdots \# E_n \to f_n \text{ FI}\}Q}$$

D5 $$\frac{\forall i\le n\,[(P\wedge E_i)\{f_i\}P]}{P\{\text{DO } E_1\to f_1 \# \cdots \# E_n\to f_n \text{ OD}\}(P\wedge\neg(\exists i: E_i))}$$

D6 $$\frac{(P\wedge E)\{f\}P}{P\{\text{WHILE } E \text{ DO } f \text{ OD}\}(P\wedge(\neg E))}$$

D7 $$\frac{P\{\xi' := E;\ f(\xi'/\xi)\}Q}{P\{\text{BEGIN DEF } \xi = E;\ f \text{ END}\}Q}$$ where ξ' is a new unused identifier.

D8 $$\frac{P\{f\}Q}{P\{\text{BEGIN } e;\ f \text{ END}\}Q}$$

D9 $$P((\kappa_I\downarrow 1),(\kappa_I\uparrow 2)/\xi,\kappa_I)\{\text{READ}(\xi)\}P$$

D10 $$P((\kappa_0@E)/\kappa_0)\{\text{WRITE}(E)\}P$$

D11 $$P\{e\}P$$

6 $\mathcal{E}_I$ used on MINI-SNAIL.

$$\mathcal{E}_I[\![\gamma_1;\gamma_2]\!] = \mathcal{E}_I[\![\gamma_2]\!] \circ \mathcal{E}_I[\![\gamma_1]\!]$$

$$\mathcal{E}_I[\![\xi := \varepsilon]\!]\psi = assign_I(\xi, \mathcal{E}_I[\![\varepsilon]\!]\psi)\,\psi$$

$$\mathcal{E}_I[\![\text{IF } \varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n \text{ FI}]\!]\psi$$

$$= [\,\mathcal{E}_I[\![\varepsilon_1]\!]\psi \to \mathcal{E}_I[\![\gamma_1]\!]\psi,\ \vdots\ \mathcal{E}_I[\![\varepsilon_n]\!]\psi \to \mathcal{E}_I[\![\gamma_n]\!]\psi\,]$$

$$\mathcal{E}_I[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!]$$

$$= FIX\ F.\ (\lambda\psi.\ \mathcal{E}_I[\![\varepsilon]\!]\psi \to F \circ \mathcal{E}_I[\![\gamma]\!]\psi, \psi)$$

$$\mathcal{E}_I[\![\text{BEGIN DEF } \xi = \varepsilon;\ \gamma \text{ END}]\!]\psi$$

$$= \text{LET } \xi' = New_I(\psi) \text{ IN } (\mathcal{E}_I[\![\xi' := \varepsilon;\ \gamma(\xi'/\xi)]\!]\psi)$$

7 $\mathcal{E}_c$ used on MINI-SNAIL

$$\mathcal{E}_c[\![\gamma_1;\gamma_2]\!] = \mathcal{E}_c[\![\gamma_2]\!] \circ \mathcal{E}_c[\![\gamma_1]\!]$$

$$\mathcal{E}_c[\![\xi := \varepsilon]\!]\Psi = \Psi;\ assign_c(\xi, \mathcal{E}_c[\![\varepsilon]\!]\Psi)\Psi$$

$$\mathcal{E}_c[\![\text{IF } \varepsilon_1 \to \gamma_1 \# \cdots \# \varepsilon_n \to \gamma_n \text{ FI}]\!]\Psi$$

$$= [\mathcal{E}_c[\![\varepsilon_1]\!]\Psi \to \mathcal{E}_c[\![\gamma_1]\!]\Psi,\ \vdots\ \mathcal{E}_c[\![\varepsilon_n]\!]\Psi \to \mathcal{E}_c[\![\gamma_n]\!]\Psi]$$

$$\mathcal{E}_c[\![\text{WHILE } \varepsilon \text{ DO } \gamma \text{ OD}]\!] =$$

$$= \text{FIX } F.\ (\lambda\Psi.\ \mathcal{E}_c[\![\varepsilon]\!]\Psi \to F \circ \mathcal{E}_c[\![\gamma]\!]\Psi, \Psi)$$

$$\mathcal{E}_c[\![\text{BEGIN DEF } \xi = \varepsilon;\ \gamma \text{ END}]\!]\Psi$$

$$= \text{LET } \xi' = New_c(\Psi) \text{ IN } (\mathcal{E}_c[\![\xi' := \varepsilon;\ \gamma(\xi'/\xi)]\!]\Psi)$$

8 Predicate-transformer Theory used on SNAIL

wp 1 $\quad wp(f_1; f_2, Q) \equiv wp(f_1, wp(f_2, Q))$

wp 2 $\quad wp(\xi := \varepsilon, Q) \equiv Q(\varepsilon/\xi)$

wp 3 $\quad wp(\text{TEST } \varepsilon \text{ DO } f_1 \text{ OR } f_2 \text{ TSET}, Q)$

$\equiv (\varepsilon \wedge wp(f_1, Q)) \vee (\neg\varepsilon \wedge wp(f_2, Q))$

wp 4 $\quad wp(\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}, Q) \equiv B \neq \varnothing \wedge wp(f_{\min B}, Q)$

where $\quad B = \{ i \leq n \mid \varepsilon_i \}$

wp 5 $\quad wp(\text{DO } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ OD}, Q) \equiv (\exists i \geq 0 : H_i(Q))$

where $\quad H_0(Q) \equiv (B = \varnothing) \wedge Q$

and $\quad H_i(Q) \equiv wp(\text{IF } \varepsilon_1 \to f_1 \# \cdots \# \varepsilon_n \to f_n \text{ FI}, H_{i-1}(Q)) \vee H_0(Q) \qquad (i \geq 1)$

wp6 $\quad wp(\text{WHILE } \varepsilon \text{ DO } f \text{ OD}, Q) \equiv (\exists i \geq 0 : K_i(Q))$

where $\quad K_0(Q) \equiv (\neg\varepsilon) \wedge Q$

and $\quad K_i(Q) \equiv wp(\text{TEST } \varepsilon \text{ DO } f \text{ OR } e \text{ TSET}, K_{i-1}(Q)) \qquad (i \geq 1)$

wp7 $\quad wp(\text{BEGIN DEF } \xi = \varepsilon; f \text{ END}, Q)$

$\equiv \exists k \in V [\, \xi = k \wedge wp(\xi := \varepsilon; f, Q(k/\xi)) \,]$

wp8 $\quad wp(\text{BEGIN } e; f \text{ END}, Q) \equiv wp(f, Q)$

wp 9 $\quad wp(\text{READ}(\xi), Q) \equiv Q((\mu_I \downarrow 1), (\mu_I + 2)/\xi, \mu_I)$

wp 10 $\quad wp(\text{WRITE}(\varepsilon), Q) \equiv Q((\mu_0 @ \varepsilon)/\mu_0)$

wp 11 $\quad wp(e, Q) \equiv Q$

9 Mathematical semantics for ASSLA

$$\mathcal{A}[\![\text{BEGIN } \pi_0;\ \xi_1 : \pi_1;\ \ldots\ldots;\ \xi_n : \pi_n \text{ END}]\!]\,\varsigma\,\theta = \theta_0$$

$$\left\{\begin{array}{l}
\theta_0 = \mathcal{A}[\![\pi_0]\!]\,\varsigma'\,\theta_1 \\
\theta_1 = \mathcal{A}[\![\pi_1]\!]\,\varsigma'\,\theta_2 \\
\quad\vdots \\
\theta_{n-1} = \mathcal{A}[\![\pi_{n-1}]\!]\,\varsigma'\,\theta_n \\
\theta_n = \mathcal{A}[\![\pi_n]\!]\,\varsigma'\,\theta \\
\varsigma' = \varsigma[\theta_1, \ldots \theta_n / \xi_1, \ldots, \xi_n]
\end{array}\right.$$

where the bracket indicate a set of fixpoint equations whose minimal solution should be used.

$$\mathcal{A}[\![\pi_1;\ \pi_2]\!]\,\varsigma\,\theta = \mathcal{A}[\![\pi_1]\!]\,\varsigma\,\{\mathcal{A}[\![\pi_2]\!]\,\varsigma\,\theta\}$$

$$\mathcal{A}[\![\text{BLOCK } \pi_1;\ \text{DEF } \xi;\ \pi_2 \text{ END}]\!]\,\varsigma\,\theta = \mathcal{A}[\![\pi_1]\!]\,\varsigma\,\{\text{New}(\varsigma)\,\{\lambda\alpha.\ \mathcal{A}[\![\pi_2]\!]\,\varsigma[\alpha/\xi]\,\theta\}\}$$

$$\mathcal{A}[\![\text{STORE } \xi]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \langle z, \mu_I, \mu_0\rangle\rangle.\ \theta\langle\mu_P+2, \langle \text{Assign}(\varsigma[\![\xi]\!]L, \mu_I\downarrow 1)\,z, \mu_I, \mu_0\rangle\rangle$$

$$\mathcal{A}[\![\text{LOAD } \xi]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \langle z, \mu_I, \mu_0\rangle\rangle.\ \theta\langle\text{Contents}(\varsigma[\![\xi]\!]L)z\,@\,\mu_P, \langle z, \mu_I, \mu_0\rangle\rangle$$

$$\mathcal{A}[\![\text{CONST } \beta]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \psi\rangle.\ \theta\langle(\beta)\,@\,\mu_P, \psi\rangle$$

$$\mathcal{A}[\![\text{READ}]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \langle z, \mu_I, \mu_0\rangle\rangle.\ \theta(\text{dom}(\mu_I) = 0 \rightarrow \langle\mu_P, \langle\text{READERROR}, \mu_I, \mu_0\rangle\rangle,\ \langle(\mu_I\downarrow 1)\,@\,\mu_P, \langle z, \mu_I+2, \mu_0\rangle\rangle)$$

$$\mathcal{A}[\![\text{WRITE}]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \langle z, \mu_I, \mu_0\rangle\rangle.\ \theta\langle\mu_P+2, \langle z, \mu_I, \mu_0\,@\,(\mu_P\downarrow 1)\rangle\rangle$$

$$\mathcal{A}[\![\text{J } \xi]\!]\,\varsigma\,\theta = \varsigma[\![\xi]\!] \mid M$$

$$\mathcal{A}[\![\text{JT } \xi]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \psi\rangle.\ ((\mu_P\downarrow 1 \rightarrow (\varsigma[\![\xi]\!] \mid M), \theta)\langle\mu_P+2, \psi\rangle)$$

$$\mathcal{A}[\![\text{JF } \xi]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \psi\rangle.\ ((\mu_P\downarrow 1 \rightarrow (\theta, (\varsigma[\![\xi]\!] \mid M))\langle\mu_P+2, \psi\rangle)$$

$$\mathcal{A}[\![\text{NEG}]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \psi\rangle.\ \theta\langle(\neg(\mu_P\downarrow 1))\,@\,(\mu_P+2), \psi\rangle$$

$$\mathcal{A}[\![\text{DO } \oplus]\!]\,\varsigma\,\theta = \lambda\langle\mu_P, \psi\rangle.\ \theta\langle((\mu_P\downarrow 2) \oplus (\mu_P\downarrow 1))\,@\,(\mu_P+3), \psi\rangle$$

$$\mathcal{A}[\![\text{ABORT}]\!]\,\varsigma\,\theta = \perp$$

$$\mathcal{A}[\![e]\!]\,\varsigma\,\theta = \theta$$

$h[\![\mathcal{J}_1; \mathcal{J}_2]\!] = h[\![\mathcal{J}_1]\!]; h[\![\mathcal{J}_2]\!]$

$h[\![\xi := \varepsilon]\!] = k[\![\varepsilon]\!]; \text{STORE } \xi$

$h[\![\text{TEST } \varepsilon \text{ DO } \mathcal{J}_1 \text{ OR } \mathcal{J}_2 \text{ TSET}]\!]$

$= \text{BEGIN } k[\![\varepsilon]\!]; \text{JF } \xi_1; h[\![\mathcal{J}_1]\!]; \text{J } \xi_2; \xi_1 : h[\![\mathcal{J}_2]\!]; \xi_2 : e \text{ END}$

$h[\![\text{IF } \varepsilon_1 \to \mathcal{J}_1 \# \cdots\cdots \# \varepsilon_n \to \mathcal{J}_n]\!] \qquad (n \geq 1)$

$= \text{BEGIN } k[\![\varepsilon_1]\!]; \text{JT } \xi_1; \cdots\cdots; k[\![\varepsilon_n]\!]; \text{JT } \xi_n; \text{ABORT};$

$\xi_1 : h[\![\mathcal{J}_1]\!]; \text{J } \xi_{n+1}; \cdots\cdots; \xi_n : h[\![\mathcal{J}_n]\!]; \xi_{n+1} : e \text{ END}$

$h[\![\text{DO } \varepsilon_1 \to \mathcal{J}_1 \# \cdots\cdots \# \varepsilon_n \to \mathcal{J}_n]\!] \qquad (n \geq 1)$

$= \text{BEGIN } e; \xi_0 : k[\![\varepsilon_1]\!]; \text{JT } \xi_1; \cdots\cdots; k[\![\varepsilon_n]\!]; \text{JT } \xi_n; \text{J } \xi_{n+1};$

$\xi_1 : h[\![\mathcal{J}_1]\!]; \text{J } \xi_0; \cdots\cdots; \xi_n : h[\![\mathcal{J}_n]\!]; \text{J } \xi_n; \xi_{n+1} : e \text{ END}$

$h[\![\text{WHILE } \varepsilon \text{ DO } \mathcal{J} \text{ OD}]\!]$

$= \text{BEGIN } e; \xi_1 : k[\![\varepsilon]\!]; \text{JF } \xi_2; h[\![\mathcal{J}]\!]; \text{J } \xi_1; \xi_2 : e \text{ END}$

$h[\![\text{BEGIN DEF } \xi = \varepsilon; \mathcal{J} \text{ END}]\!]$

$= \text{BLOCK } k[\![\varepsilon]\!]; \text{DEF } \xi; \text{STORE } \xi; h[\![\mathcal{J}]\!] \text{ END}$

$h[\![\text{BEGIN } e; \mathcal{J} \text{ END}]\!] = h[\![\mathcal{J}]\!]$

$h[\![\text{READ } (\xi)]\!] = \text{READ}; \text{STORE } \xi$

$h[\![\text{WRITE } (\varepsilon)]\!] = k[\![\varepsilon]\!]; \text{WRITE}$

$h[\![e]\!] = e$

where all the labels $\xi_0, \cdots\cdots, \xi_{n+1}$ are supposed to be new and unused (generated by the compiler).

$$
\begin{aligned}
k[\![\text{TRUE}]\!] &= \text{CONST}\ TT \\
k[\![\text{FALSE}]\!] &= \text{CONST}\ FF \\
k[\![\xi]\!] &= \text{LOAD}\ \xi \\
k[\![k]\!] &= \text{CONST}\ \mathcal{K}[\![k]\!] \\
k[\![\neg \varepsilon]\!] &= k[\![\varepsilon]\!];\ \text{NEG} \\
k[\![\varepsilon_1 \oplus \varepsilon_2]\!] &= k[\![\varepsilon_1]\!];\ k[\![\varepsilon_2]\!];\ \text{DO}\ \oplus
\end{aligned}
$$

appendix C:

AN ALTERNATIVE DEFINITION OF IF-COMMANDS

In a letter to Dijkstra (January 1976) Dana Scott proposes the use of the "ungaurded IF-command". This is exactly the same as Dijkstra's "guarded IF-command" except that all guards are identical to the boolean constant true.

Let an extended syntax for SNAIL be:

$$\gamma ::= \ldots\ldots \mid \text{IF } \omega \text{ FI} \mid \text{IF } \omega' \text{ FI} \mid \text{DO } \omega \text{ OD} \mid \ldots\ldots$$

$$\omega ::= \varepsilon_1 \rightarrow \gamma_1 \# \ldots\ldots \# \varepsilon_n \rightarrow \gamma_n \qquad n \geq 1$$

$$\omega' ::= \gamma_1 \,[]\, \ldots\ldots \,[]\, \gamma_n \qquad n \geq 1$$

(we cannot make an ungaurded DO-command since this would never stop).

Furthermore I will assume that guards are evaluated without sideeffects. The discussion in appendix A shows that this is a reasonable assumption. Contrary it will not be nescessary to assume that guards are evaluated without program-abortion and infinite loops.

Let us first see what the wp-transformer should be for the unguarded IF-command. Intuitively we must have:

$$wp(\text{IF } \gamma_1 \,[]\, \ldots\ldots \,[]\, \gamma_n \text{ FI}, Q) \equiv \bigcap_{i=1}^{n} wp(\gamma_i, Q)$$

where $\cap$ is the normal boolean conjunction (generalized to n arguments) and extended for use on functions with boolean values by the following definition

$$\left(\bigcap_{i=1}^{n} g_i\right)(x) = \bigcap_{i=1}^{n} (g_i(x))$$

This together with Theorem 10 on page 86 inspires us to define

$$\mathcal{E}[\![\text{IF } \gamma_1 \,[]\, \ldots\ldots \,[]\, \gamma_n \text{ FI}]\!]\sigma\theta = \bigcap_{i=1}^{n} \mathcal{E}[\![\gamma_i]\!]\sigma\theta$$

It is now possible to make a (iterative) definition of the "guarded-IF" in terms of the "unguarded-IF" and TEST.

The definition is: (n number of guards)

n = 1: IF $\varepsilon_1 \Rightarrow \gamma_1$ FI ≡ TEST ε_1 DO γ_1 OR ABORT TSET

where ABORT (which is not part of the original SNAIL-language) could be replaced by a nonterminating WHILE-command as WHILE TRUE DO DUMMY OD.

n = 2: IF $\varepsilon_1 \Rightarrow \gamma_1 \# \varepsilon_2 \Rightarrow \gamma_2$ FI ≡

```
TEST (ε1 ∨ ε2)
DO
  IF
      TEST ε1 DO γ1 OR γ2 TSET
      TEST ε2 DO γ2 OR γ1 TSET
  FI
OR
  ABORT
TSET
```

n > 2: IF $\varepsilon_1 \Rightarrow \gamma_1 \# \cdots \# \varepsilon_n \Rightarrow \gamma_n$ FI ≡

```
IF
    ε1 ∨ ···· ∨ εn-1 → IF ε1 ⇒ γ1 # ····· # εn-1 ⇒ γn-1 FI #
    εn → γn
FI
```

The "guarded-DO" can be defined by

DO $\varepsilon_1 \Rightarrow \gamma_1 \# \cdots \# \varepsilon_n \Rightarrow \gamma_n$ OD ≡

```
TSET (ε1 ∨ ···· ∨ εn)
DO
  IF ε1 ⇒ γ1 # ··· # εn ⇒ γn FI ;
  DO ε1 ⇒ γ1 # ··· # εn ⇒ γn OD
OR
  DUMMY
TSET
```

Compared with

WHILE ε DO γ OD ≡ TEST ε DO (γ; WHILE ε DO γ OD) OR DUMMY TSET

this shows that the interest lies in IF, whereas DO is formed from IF by the standard method of recursive repetition.

This way of progress differs from that used in the actual SNAIL definition page 15-22 in that it gives a true nondeterministic approach whereas the Oracle-function had to be implemented in a deterministic way.

REFERENCES

A AIELLO,L, AIELLO,M and WEYHRAUCH,R.W: The semantics of Pascal in LCF, Stanford - Computer Science department, Stanford University, (1974), Memo AIM-221.

B de BAKKER,J.W: The fixed point approach in semantics: Theory and applications, Foundations of computer science, Mathematical Centre Tracts 63, (1975), 3-53.

D DIJKSTRA,E.W: A simple axiomatic basis for programming language constructs, Indagationes Mathematicae, 36 (1974) 1-15 (EWD 372).

D1 DIJKSTRA,E.W: Guarded commands, non determinacy and calculus for the derivation of programs, Comm. ACM, 18, 453-457 (aug. 1975).

D2 DIJKSTRA,E.W: Sequencing primitives revisited, Technical University Eindhoven, The Netherlands (1973) (EWD 398 + EWD 399).

D3 DONAHUE,J.E: The mathematical semantics of axiomatically defined programming language constructs, Colloques IRIA, Arc et Senans 1-3 juillet 1975, 353-367.

D4 DIJKSTRA,E.W: A discipline of programming, Prentice Hall Inc. Englewood Cliffs, New Jersey, (1976).

H HOARE,C.A.R: An axiomatic basis for computer programming, Comm. ACM, 12 576-580 (1967)

H1 HOARE,C.A.R and LAUER,P: Consistent and complementary formal theories of the semantics of programming languages, Acta Inform. 3, 135-153 (1974) by Springer-Verlag

L LAUER,P: Consistent formal theories of the semantics of programming languages, IBM Laboratory Vienna, TR 25.121 (Nov. 1971)

L1 LEVIN,M: Mathematical logic for computer scientists, Massachusetts Institute of Technology, MAC TR-131, (June 1974)

L2 LIGLER,G: Surface properties of programming language constructs, Colloques IRIA, Arc et Senans 1-3 juillet 1975, 299-323.

M MANNA,Z and VUILLEMIN,J: Fixpoint approach to the theory of computation, Comm. ACM, 15 528-536 (1972).

M1 MILNER,R: Implementation and applications of Scott's logic for computable functions, Proceedings of an ACM conference on proving assertions about programs, 1-6 (Jan. 1972)

M2 MILNER,R and WEYHRAUCH,R: Proving compiler correctness in a mechanized logic, Machine Intelligence 7, 51-70, Edinburgh University Press, (1972).

M3 MOSSES,P: The mathematical semantics of Algol 60, Technical Monograph PRG-12, Oxford University Computing Laboratory,(Jan. 1974).

M4 MOSSES,P: Compound domain descriptions - a proposal, (17. march 1976 - unpublished)

R REYNOLDS,J.C: Definitional interpreters for high-order programming languages, Proc. ACM 1972, Annual Conf., ACM, New York, 717-740.

S SCOTT,D: Outline of a mathematical theory of computation, Proc. of the fourth anual Princeton conference on information science and systems, 169-176,and Technical Monograph PRG-2, Oxford University Computing Laboratory, (Nov. 1970).

S1 SCOTT,D and STRACHEY,C: Toward a mathematical semantics for computer languages, Proc. of the symposium on computers and automata, Polytechnic Institute of Brooklyn, and Technical Monograph PRG-6, Oxford University Computing Laboratory, (Aug. 1971).

S2 STRACHEY,C: The varieties of programming language, Proc. of the international computing symposium 222-233, Cini Foundations Venice, and Technical Monograph PRG-10, Oxford University Computing Laboratory, (March 1973).

S3 STRACHEY,C: The semantic bridge or the purpose of a formal theory of semantics for programming languages and their implementations, (unpublished, 1o. feb. 1975).

S4 STRACHEY,C and WADSWORTH,C.P: Continuations. A mathematical semantics for handling full jumps, Technical Monograph PRG-11, Oxford University Computing Laboratory, (Jan 1974).