stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 74/79 JUNI

A. DE BRUIN

GOTO STATEMENTS: SEMANTICS AND DEDUCTION SYSTEMS

Preprint

2e boerhaavestraat 49 amsterdam

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

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

AMS(MOS) subject classification scheme (1970): 68A05

ACM-Computing Reviews-category: 5.24

Goto statements: semantics and deduction systems *)

by

A. de Bruin

ABSTRACT

A simple language containing goto statements is presented, together with a denotational and operational semantics for it. Equivalence of these semantical descriptions is proven.

Furthermore, soundness and completeness of a Hoare-like proof system for the language is shown. This is done in two steps. Firstly, a proof system is given and validity is defined using (a variant of) direct semantics. In this case soundness and completeness proofs are relatively easy. After that, a proof system is given which is more in the style of the one by CLINT & HOARE [5], and validity in this system is defined using continuation semantics. This validity definition is then related to validity in the first system and, using this correspondence, soundness and completeness for the second system is proven.

KEY WORDS & PHRASES: *goto statements, denotational semantics, operational semantics, partial correctness, Hoare-like deduction systems, soundness, completeness, continuation semantics.*

---

*) This report will be submitted for publication elsewhere.

1. INTRODUCTION

In this report we present several ways of looking at the meaning of goto statements. We define a simple language containing goto statements, and present an operational definition of its semantics in the sense of COOK [6]. We also give a denotational semantics, using the concept of continuations (STRACHEY & WADSWORTH [11]). Furthermore we prove that these definitions are equivalent.

After that we turn our attention towards a Hoare-like deduction system, as proposed by CLINT & HOARE [5], for proving partial correctness of programs of our language. It appears to be surprisingly complicated to justify this system. The essential rule in the deduction system is (for programs with one label only)

$$\frac{\{p\}\ \underline{\text{goto}}\ L\ \{\underline{\text{false}}\} \vdash \{p_1\}A_1\{p\} \qquad \{p\}\ \underline{\text{goto}}\ L\ \{\underline{\text{false}}\} \vdash \{p\}A_2\{p_2\}}{\vdash \{p_1\}A_1;L{:}A_2\{p_2\}}$$

and the unusual assumption {p} goto L {false} already gives an indication of possible complications. The main problem is how validity of the construct {p}A{q} has to be defined.

If we investigate how the inference rule given above will be used in correctness proofs, we observe that the assumption {p} goto L {false} is used as a trick to indicate that p always holds before execution of goto L. Or, stated another way, the assertion p in the assumption serves as a so called *label invariant*: if we want to prove partial correctness of a program S which contains a label L, then we can use the assumption {p} goto L {false} in the proof to describe that p holds whenever label L is encountered during evaluation of S. Thus the introduction of an assumption like {p} goto L {false} in a proof only serves the purpose to indicate what the label invariant at L will be.

This report gives two variants of the deduction system and the above observations are used in the first one. Here there are no assumptions, the label invariants needed are stated explicitly within the formulae of the

system. These formulae will have the form

$$\langle L_1:p_1,\ldots,L_n:p_n \mid \{p\}A\{q\}\rangle,$$

where p_i is the invariant corresponding to label L_i. Validity of a formula like this one has to be defined in terms of the meaning of statement A occurring in it. Things become too complicated if we use the customary denotational definition with continuations and environments. Techniques in the spirit of "continuation removal" (MILNE & STRACHEY [8]) are used to define the meaning of statements such that a definition of validity is possible which is both perspicuous and useful. After that, soundness and completeness of the deduction system will be proved.

Once this result has been established we investigate a deduction system like the one given by Clint and Hoare. We give a definition of validity of formulae like {p}A{q} using the ordinary continuation semantics. This definition resembles closely the one given in MILNE & STRACHEY [8]. Furthermore this definition of validity is such that

$$\{p_1\}\ \underline{\text{goto}}\ L_1\ \{\underline{\text{false}}\},\ldots,\{p_n\}\ \underline{\text{goto}}\ L_n\ \{\underline{\text{false}}\} \models \{p\}A\{q\}$$

holds, if and only if in the other system the formula

$$\langle L_1:p_1,\ldots,L_n:p_n \mid \{p\}A\{q\}\rangle$$

is valid. This result will then be used to prove soundness and completeness for the second deduction system.

This two level approach has the following advantages. In the first variant of the system we take only those elements of the Clint-Hoare system into account that are really necessary. This has as a consequence that the definition of validity and the arguments in the soundness and completeness proofs are as perspicuous as possible. Though straightforward proofs of these properties for the second system must essentially be the same as the ones for the first variant, they are bound to be obscured through all additional details which we have to deal with. The way we handle this problem is to separate the "essential proof" from the "additional details".

The rules and axioms in the second system are just like the ones in other Hoare-like systems, and we can combine these into one system quite easily. Using the validity definition for the second variant of the system, as given in this report, it must be possible to combine the results stated here with analogous results concerning other language constructs (such as while statements, recursive procedures and the like; cf. APT [1], APT & DE BAKKER [2], DE BAKKER [4]).

2. SYNTAX

We use the following classes of symbols:

Var, the (infinite) class of *variables* with typical elements x,y,z. We assume this class to be ordered

Lvar, the class of *label variables* with typical element L

Fsym = $\{fu_1,\ldots,fu_m\}$, the class of *function symbols*. We denote the arity of fu_i by arf_i

Rsym = $\{re_1,\ldots,re_n\}$, the class of *relation symbols*. The arity of re_i is denoted by arr_i.

Next, using a self-explanatory variant of BNF, we define the classes *Bexp* (*boolean expressions*) with typical element b, *Exp* (*expressions*) with typical elements s,t, *Stat* (*statements*) with typical element A, and *Prog* (*programs*) with typical element S:

Bexp b::= $\underline{\text{true}} \mid b_1 \vee b_2 \mid \neg b \mid re_1(s_1,\ldots,s_{arr_1}) \mid \ldots \mid re_n(s_1,\ldots,s_{arr_n})$

Exp s::= $x \mid fu_1(s_1,\ldots,s_{arf_1}) \mid \ldots \mid fu_m(s_1,\ldots,s_{arf_m})$

Stat A::= $x:=s \mid (A_1;A_2) \mid \underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}} \mid \underline{\text{goto}}\ L$

Prog S::= $L:A \mid L:A;S$

with the additional requirement: if $L_1:A_1;\ldots;L_n:A_n$ is a program, then all labels L_i are different.

The symbols fu_i and re_i are the primitive function and relation symbols. We did not specify them further, because we do not wish to go into details concerning the basic calculations our programs S can perform. Rather do we want to describe the way programs specify more complex calculations using these primitives as building blocks.

The clause $(A_1;A_2)$ in the definition of *Stat* deserves some comment. It is usual to omit parentheses in cases like this one, thus admitting the grammar to be ambiguous. In general there is no problem there, because the meaning of, say $(A_1;A_2);A_3$ and $A_1;(A_2;A_3)$ will be the same. Of course this holds in our case too. However, complications show up in our definition of the operational semantics. For instance, for the auxiliary semantic function *Comp* the equality

$$Comp((A_1;A_2);A_3) = Comp(A_1;(A_2;A_3))$$

does not hold.

Putting parentheses all over the place is tedious though. We therefore use the convention that the operator " ; " associates to the right, which means that $A_1;A_2;\ldots;A_n$ should be read as $(A_1;(A_2;(\ldots;A_n)..))$.

Anticipating the deduction systems of chapters 6 and 8 we give the definition of the syntactical class *Assn* (the *assertions*) with typical elements p,q.

$$Assn \quad p::= \underline{true} \mid p_1 \vee p_2 \mid \neg p \mid re_1(s_1,\ldots,s_{arr_1}) \mid \ldots \mid re_n(s_1,\ldots,s_{arr_n}) \mid \exists x[p]$$

It turns out that *Assn* is just a language *L* for the first order predicate calculus, based on the classes *Fsym* and *Rsym*. Furthermore we see that *Exp* is exactly the class of the terms of *L*, and that *Bexp* is the set of all quantifier free formulae of *L*.

The rest of this paragraph gives some notational conventions and useful definitions. We use the symbol $\equiv$ to refer to *syntactical identity*, i.e., $B \equiv C$ means that B and C are the same sequence of symbols. The following abbreviations will be used:

$b_1 \wedge b_2 \equiv \neg(\neg b_1 \vee \neg b_2)$
$b_1 \supset b_2 \equiv \neg b_1 \vee b_2$
$\underline{if}\ b_1\ \underline{then}\ b_2\ \underline{else}\ b_3\ \underline{fi} \equiv (b_1 \wedge b_2) \vee (\neg b_1 \wedge b_3)$
$\underline{false} \equiv \neg \underline{true}$
$[L_i:A_i]_{i=1}^{n} \equiv L_1:A_1;\ldots;L_n:A_n.$

We define the property that *a label* L *occurs in* A inductively by

a) no label occurs in a statement of the form x:=s

b) L occurs in $(A_1;A_2)$ and in $\underline{\text{if}}$ b $\underline{\text{then}}$ A_1 $\underline{\text{else}}$ A_2 $\underline{\text{fi}}$, if either L occurs in A_1 or L occurs in A_2

c) the only label occurring in $\underline{\text{goto}}$ L is L.

Let $S \equiv [L_i:A_i]_{i=1}^{n}$. We say that

+) L *is declared in* S if $L \equiv L_i$ for some i $(1 \leq i \leq n)$

+) L *occurs in* S if either L is declared in S or L occurs in some A_i $(1 \leq i \leq n)$

+) S *is normal* if all labels occurring in S are declared in S.

3. OPERATIONAL SEMANTICS

In our semantics functions will be used abundantly. Often these functions will be of higher order, which means that they have functions as arguments and/or values. In order to keep our notation as clear as possible we first state some conventions on this point.

a) The class of all functions with domain A and range B will be denoted by $(A \to B)$

b) The class of all partial functions with domain A and range B will be denoted by $(A \xrightarrow{\text{part}} B)$

c) The convention will be used that " $\to$ " associates to the right. For example, $A \to B \to C$ must be read as $A \to (B \to C)$

d) We will in general omit parentheses around arguments, using the convention that function application associates to the left. Thus, assuming $f \in (A \to B \to C \to D)$, $a \in A$, $b \in B$, $c \in C$, for some A,B,C and D, the entity $((f(a))(b))(c)$ can be written as fabc

e) The above convention has the following exception: every syntactic entity used as an argument will be enclosed in $[\![\cdot]\!]$-type brackets. This is done to provide a clear distinction between the object language of chapter 2 and the language used to denote the semantic objects.

As a starting point of our semantical considerations we first discuss the meaning given to the symbols in $\mathcal{F}sym$ and $\mathcal{R}sym$. An *interpretation I of the primitive symbols* is an (m+n+1)-tuple $\langle D, \underline{fu}_1, \ldots, \underline{fu}_m, \underline{re}_1, \ldots, \underline{re}_n \rangle$, where

D is a non-empty domain,
$\underline{fu}_i$ is a function in $(D^{arf_i} \to D)$, for $i = 1,\ldots,m$, and
$\underline{re}_i$ is a relation $\subset D^{arr_i}$ $(i = 1,\ldots,n)$.

All semantic functions to be defined will depend on an underlying interpretation of the primitive symbols, though the notation we use won't show this dependence. For instance, the function giving the meaning of the expressions will be denoted by $\mathcal{V}$, instead of $\mathcal{V}_{\mathcal{I}}$ or something like that.

We now choose an arbitrary interpretation $\mathcal{I}$ and assume this interpretation to remain fixed for the rest of this paper (unless we explicitly state otherwise).

A *state* is a valuation of the variables from $\mathcal{Var}$ in our domain of interpretation D. The set of all states is denoted by Σ, with typical element σ. In principle the meaning of a statement will be a partial function from states to states. The function is partial, due to the possibility of nonterminating computations. We consider it useful not to allow partial functions and therefore include the undefined state $\perp$ in Σ. This leads to the following definition:

$$\Sigma = (\mathcal{Var} \to D) \cup \{\perp\}.$$

We denote the set of all defined states Σ_0, i.e. $\Sigma_0 = (\mathcal{Var} \to D)$.

Let $d \in D$. A *variant* $\sigma\{d/x\}$ *of a state* σ is a state σ' differing from σ only in the variable x, or explicitly

$$\sigma\{d/x\} = \begin{cases} \perp, \text{ if } \sigma = \perp \text{ and otherwise} \\ \sigma' \in \Sigma_0 \text{ such that } \sigma'[\![y]\!] = \begin{cases} \sigma[\![y]\!] & \text{if } x \not\equiv y \\ d & \text{if } x \equiv y. \end{cases} \end{cases}$$

The next syntactic classes to be handled are $\mathcal{Bexp}$ and $\mathcal{Exp}$. We will define inductively the semantic functions

$$\mathcal{V}: \mathcal{Exp} \to \Sigma_0 \to D$$
$$\mathcal{W}: \mathcal{Bexp} \to \Sigma_0 \to \{ff,tt\}.$$

Note that $\mathcal{V}[\![s]\!]\sigma$, and $\mathcal{W}[\![b]\!]\sigma$ are not defined for $\sigma=\perp$.

DEFINITION OF $\mathcal{V}$.

$\mathcal{V}[\![x]\!]\sigma = \sigma[\![x]\!]$

$\mathcal{V}[\![fu_i(s_1,\ldots,s_{arf_i})]\!]\sigma = \underline{fu}_i(\mathcal{V}[\![s_1]\!]\sigma,\ldots,\mathcal{V}[\![s_{arf_i}]\!]\sigma)$.

DEFINITION OF $\mathcal{W}$.

$\mathcal{W}[\![\underline{true}]\!]\sigma = tt$

$\mathcal{W}[\![b_1 \vee b_2]\!]\sigma = tt$, if $\mathcal{W}[\![b_1]\!]\sigma = tt$ or $\mathcal{W}[\![b_2]\!]\sigma = tt$, and ff otherwise

$\mathcal{W}[\![\neg b]\!]\sigma = tt$, if $\mathcal{W}[\![b]\!]\sigma = ff$, and ff otherwise

$$\mathcal{W}[\![re_i(s_1,\ldots,s_{arr_i})]\!]\sigma = \begin{cases} tt, & \text{if } \langle \mathcal{V}[\![s_1]\!]\sigma,\ldots,\mathcal{V}[\![s_{arr_i}]\!]\sigma\rangle \in \underline{re}_i \\ ff, & \text{otherwise.} \end{cases}$$

The semantic definitions given above are basic in the sense that they will be the same for the denotational semantics. We now turn to the operational semantics proper.

We want to define the meaning of a statement A as a function that, given an initial state σ as an argument, yields a so called *computation sequence* τ. Such a computation sequence is a possibly infinite row of states from Σ, the elements of which can be viewed as the successive intermediate states produced by evaluation of the statement A starting in initial state σ. The semantic function that maps statements on their meaning in the above sense will be called *Comp*.

In order to be able to handle these computation sequences, we present the following definitions:

a) (*computation sequences*)

Σ^+ is the class of all non-empty finite sequences $\langle\sigma_0,\ldots,\sigma_n\rangle$ for some $n \geq 0$, such that $\sigma_i \in \Sigma$ for $i = 0,1,\ldots,n$

Σ^ω is the class of all infinite sequences $\langle\sigma_0,\sigma_1,\ldots\rangle$, such that $\sigma_i \in \Sigma$ for all $i \in \mathbb{N}$

Σ^∞, with typical element τ, is defined through $\Sigma^\infty = \Sigma^+ \cup \Sigma^\omega$.

b) (*concatenation*)

Let $\tau_1,\tau_2 \in \Sigma^\infty$. The concatenation of τ_1 and τ_2, notation $\tau_1{}^\frown\tau_2$, is defined by

1) if $\tau_1 = \langle\sigma_0,\ldots,\sigma_n\rangle \in \Sigma^+$ and $\tau_2 = \langle\sigma'_0,\ldots,\sigma'_m\rangle \in \Sigma^+$ then
$\tau_1{}^\frown\tau_2 = \langle\sigma_0,\ldots,\sigma_n,\sigma'_0,\ldots,\sigma'_m\rangle \in \Sigma^+$

2) if $\tau_1 = \langle\sigma_0,\ldots,\sigma_n\rangle \in \Sigma^+$ and $\tau_2 = \langle\sigma'_0,\sigma'_1,\ldots\rangle \in \Sigma^\omega$ then
$\tau_1{}^\frown\tau_2 = \langle\sigma_0,\sigma_1,\ldots,\sigma_n,\sigma'_0,\sigma'_1,\ldots\rangle \in \Sigma^\omega$

3) if $\tau_1 \in \Sigma^\omega$ then $\tau_1{}^\frown\tau_2 = \tau_1$.

c) (*κ-function*)

The function $\kappa \in (\Sigma^\infty \to \Sigma)$ is defined by

$$\kappa(\tau) = \begin{cases} \bot, & \text{if } \tau \in \Sigma^\omega \\ \sigma_n, & \text{if } \tau = \langle\sigma_0,\ldots,\sigma_n\rangle \in \Sigma^+. \end{cases}$$

There is a last remark to be made before we give an exact definition of *Comp*. We must be aware of the fact that A can contain substatements of the form goto L, and we should have a way to get to know how evaluation of A proceeds once such a substatement is reached. We therefore supply the function *Comp* with an extra argument, namely an element of *Prog*, meant to provide the "declaration" of the labels occurring in A. *Comp* will then have the following functionality:

$$Comp\colon Prog \times Stat \longrightarrow \Sigma \xrightarrow{\text{part}} \Sigma^\infty$$

and the computation sequence $Comp[\![\langle S,A\rangle]\!]\sigma$ is meant to be the row of intermediate states appearing during evaluation of A starting in state σ, where the labels are defined by the program S.

DEFINITION (*Comp*).

A. $Comp[\![\langle S,A\rangle]\!]\sigma = \langle\bot\rangle$ if $\sigma=\bot$

B. $Comp[\![\langle S,A\rangle]\!]\sigma$ for $\sigma \in \Sigma_0$ is defined recursively by

1. $Comp[\![\langle S,x:=s\rangle]\!]\sigma = \langle\sigma\{V[\![s]\!]\sigma/x\}\rangle$

2. $Comp[\![\langle S,$ if b then A_1 else A_2 fi$\rangle]\!]\sigma = \begin{cases} \langle\sigma\rangle^\frown Comp[\![\langle S,A_1\rangle]\!]\sigma, & \text{if } W[\![b]\!]\sigma = tt \\ \langle\sigma\rangle^\frown Comp[\![\langle S,A_2\rangle]\!]\sigma, & \text{if } W[\![b]\!]\sigma = ff \end{cases}$

3. $Comp[\![\langle S,$ goto L$\rangle]\!]\sigma = \begin{cases} \langle\sigma\rangle^\frown Comp[\![\langle S,A_i;A_{i+1};\ldots A_n\rangle]\!]\sigma, & \text{if } S \equiv [L_1:A_i]_{i=1}^n, \\ & \quad \text{and } L \equiv L_i \text{ for some } i,\ 1 \le i \le n \\ \text{undefined, otherwise} \end{cases}$

4. $Comp[\![\langle S,(x:=s;A')\rangle]\!]\sigma = \langle\sigma\{V[\![s]\!]\sigma/x\}\rangle^\frown Comp[\![\langle S,A'\rangle]\!](\sigma\{V[\![s]\!]\sigma/x\})$

5. $Comp[\![\langle S,((A";A''');A')\rangle]\!]\sigma = \langle\sigma\rangle^\frown Comp[\![\langle S,(A";(A''';A'))\rangle]\!]\sigma$

6. $Comp[\![\langle S, (\underline{\text{if}}\ b\ \underline{\text{then}}\ A''\ \underline{\text{else}}\ A'''\ \underline{\text{fi}}; A') \rangle]\!]\sigma =$

$$\begin{cases} \langle\sigma\rangle^{\frown} Comp[\![\langle S, (A''; A') \rangle]\!]\sigma, & \text{if } \mathcal{W}[\![b]\!]\sigma = tt \\ \langle\sigma\rangle^{\frown} Comp[\![\langle S, (A'''; A') \rangle]\!]\sigma, & \text{if } \mathcal{W}[\![b]\!]\sigma = ff \end{cases}$$

7. $Comp[\![\langle S, (\underline{\text{goto}}\ L; A') \rangle]\!]\sigma = \langle\sigma\rangle^{\frown} Comp[\![\langle S, \underline{\text{goto}}\ L \rangle]\!]\sigma.$

Some remarks on this definition will be useful. This style of defining is taken from COOK [6]. The definition should be viewed as a method for stepwise generating computation sequences. Each step will consist of replacing an occurrence of some expression $Comp[\![\langle S,A \rangle]\!]\sigma$ using a rule from the definition. The rule to be applied depends on the form of A and is in fact uniquely determined by A. It is possible that this process won't terminate. In that case an element τ of Σ^{ω} will be generated. However this τ is well defined in the sense that every member of it is precisely determined.

The difficulties that arise by allowing goto-statements in the language are reflected in clauses 4 to 7 of the definition. The problem is that $Comp[\![\langle S, (A_1; A_2) \rangle]\!]\sigma$ cannot be defined easily in terms of $Comp[\![\langle S,A_1 \rangle]\!]$ and $Comp[\![\langle S,A_2 \rangle]\!]$, because evaluation of A_1 may terminate through execution of a substatement which is a jump out of A_1. The solution given here is to decompose a statement $(A_1;A_2)$, using rule 5 or 6, as long as it remains unclear whether an assignment or a jump has to be executed first. When this has become known, rule 4 or 7 can be applied. The extra states $\langle\sigma\rangle$ which are added in the right-hand sides of clauses 2,5,6 and 7 are strictly speaking superfluous. They are introduced in order to be able to use induction in the proof of lemma 5.2 in a more elegant way. Note however that the $\langle\sigma\rangle$ added in rule 3 is necessary, because we want $Comp[\![\langle L:\underline{\text{goto}}\ L,\ \underline{\text{goto}}\ L \rangle]\!]\sigma$ to be equal to $\langle\sigma,\sigma,\ldots\rangle \in \Sigma^{\omega}$, not to $\langle\sigma\rangle \in \Sigma^{+}$.

Finally, from the definition it can be seen that the following holds: if all labels in A and S are declared in S, then $Comp[\![\langle S,A \rangle]\!]\sigma$ is defined for all σ.

We close this chapter by defining the operational meaning $\mathcal{O}[\![S]\!]$ for each program S in *Prog*. This meaning will be a state transformation, i.e. an element of $(\Sigma \to \Sigma)$. The state $\mathcal{O}[\![S]\!]\sigma$ is meant to be the last element of the computation sequence τ, generated by evaluation of S starting in state σ. More precisely:

DEFINITION ($\mathcal{O}$). The function $\mathcal{O}$ has functionality

$$\mathcal{O}: Prog \to \Sigma \xrightarrow{part} \Sigma$$

and is defined by

$$\mathcal{O}[\![S]\!]\sigma = \kappa(Comp[\![<S,A_1;\ldots A_n>]\!]\sigma),$$

if $S \equiv [L_i:A_i]_{i=1}^{n}$.

4. DENOTATIONAL SEMANTICS

We now give semantical definitions in the style of SCOTT & STRACHEY [9], with additions (due to STRACHEY & WADSWORTH [11] among others) to accomodate the peculiarities that goto-statements entail. The mathematical concepts used in these definitions are summarized below, so that we will be able to refer to them later on. Furthermore it can serve as a very concise introduction for those who are not yet acquainted with it. More details can be found in STOY [10] for instance.

1. A pair $<C,\sqsubseteq>$ is called a *complete partial order* (or a *cpo*) iff C is a non-empty set and $\sqsubseteq$ a partial order (i.e. a relation that is reflexive, transitive and anti-symmetric) such that
   a) C contains a smallest element, called *bottom* and written as $\perp_C$ or just $\perp$, i.e. $\forall c \in C: \perp \sqsubseteq c$
   b) Every sequence $c_1 \sqsubseteq c_2 \sqsubseteq \ldots$ of elements from C (called a *chain*, notation $<c_i>_{i=1}^{\infty}$ or $<c_i>_i$) has a least upper bound $\sqcup_i c_i$, satisfying
      α) $\forall c_i: c_i \sqsubseteq \sqcup_j c_j$ (upper bound)
      β) $\forall d \in C: [(\forall c_i: c_i \sqsubseteq d) \Rightarrow \sqcup_i c_i \sqsubseteq d]$ (the least one).
2. Σ as defined in the previous chapter, supplied with partial order $\sqsubseteq$, defined by

   $$\sigma \sqsubseteq \sigma \Leftrightarrow (\sigma=\sigma' \vee \sigma=\perp)$$

   is a cpo. A cpo with partial order defined this way is called *discrete*.

3. Let A and B be cpo's, and $f \in (A \to B)$
   a) f is called *monotonic* iff $\forall a,b \in A: a \sqsubseteq b \Rightarrow fa \sqsubseteq fb$
   b) f is called *strict* iff $f\bot = \bot$. The class of all strict functions from A to B will be denoted by $(A \to_s B)$
   c) f is called *continuous* iff f is monotonic and for every chain $\langle a_i \rangle_i$ in A, we have $f(\bigsqcup_i a_i) = \bigsqcup_i (fa_i)$. The class of all continuous functions in $(A \to B)$ will be denoted by $[A \to B]$.

4. Let A and B be cpo's. Then $[A \to B]$ is a cpo, if order, bottom and lub are defined by
   a) $f \sqsubseteq g \iff \forall a \in A: fa \sqsubseteq ga$
   b) $\bot_{[A \to B]} = \lambda a \cdot \bot_B$
   c) if $\langle f_i \rangle_i$ is a chain then $\bigsqcup_i f_i = \lambda a \cdot \bigsqcup_i (f_i a)$.

5. Let A_i be a cpo for $i = 1,\ldots,n$. Then $A_1 \times \ldots \times A_n$ is a cpo, if order, bottom and lub are defined by
   a) $\langle a_1,\ldots,a_n \rangle \sqsubseteq \langle a_1',\ldots,a_n' \rangle$ iff $a_i \sqsubseteq a_i'$ for $i = 1,\ldots,n$
   b) $\bot_{A_1 \times \ldots \times A_n} = \langle \bot,\ldots,\bot \rangle$
   c) if $\langle a_1^{(i)},\ldots,a_n^{(i)} \rangle_i$ is a chain, then $\bigsqcup_i(\langle a_1^{(i)},\ldots,a_n^{(i)} \rangle) = \langle \bigsqcup_i a_1^{(i)},\ldots,\bigsqcup_i a_n^{(i)} \rangle$.

6. Let A be a cpo. Every continuous function $f \in [A \to A]$ has a *least fixed point*, written μf, with properties
   a) $f(\mu f) = \mu f$ *fixed point property*, notation *fpp*
   b) $\forall x \in A[f(x) \sqsubseteq x \Rightarrow \mu f \sqsubseteq x]$ *least fixed point property* (*lfp*)
   c) $\mu f = \bigsqcup_i f^i(\bot)$, with $f^i(\bot)$ defined by $f^0(\bot) = \bot$, $f^{i+1}(\bot) = f(f^i(\bot))$.

We now discuss the denotational semantics for statements from *Stat*. Again we are faced with problems about what to do with substatements of the form goto L. In the operational semantics this was solved by giving *Comp* an extra argument $S \equiv [L_i:A_i]_{i=1}^n$, which was used in essence to associate with each label L_i the statement $A_i;\ldots;A_n$. The meaning of goto L_i was practically the same as the meaning of $A_i;\ldots;A_n$, which could be reduced to a state transformation (i.e. $\kappa \circ \mathit{Comp}[\![\langle S, A_i;\ldots;A_n \rangle]\!]$, always a strict function). This function, applied to a state σ yields a final state σ', which is the result of evaluation of the statement $A_i;\ldots;A_n$. In other words, σ' is the result of evaluation of the rest of the program which will

be executed after goto L_i has been evaluated.

The denotational semantics uses the same approach but in a more abstract way. Instead of giving for each label a program text that specifies a state transformation, we now provide this transformation directly. This is organized as follows: the semantic function N is given an extra argument γ, called an *environment*, which is a function from *Lvar* to $(\Sigma \rightarrow_s \Sigma)$. In the definition of N we then will have a clause like $N[\![\underline{\text{goto}}\ L]\!]\gamma = \gamma[\![L]\!]$. (How this $\gamma[\![L]\!]$ is obtained from the declaration of L in a program S will be discussed later when we come to define the meaning of programs.)

Thus we see that the meaning $\gamma[\![L]\!]$ of the statement goto L in an environment γ is a state transformation that doesn't describe the evaluation of goto L only, but also of the rest of the program to be evaluated once goto L has been executed. But then the same must be true for an arbitrary statement A as well. In the operational semantics care was taken of this, because the text of the rest of the program to be evaluated remained available (see clauses 4-6 in the definition of *Comp*). Here we will use an abstraction of this idea resembling the approach of the goto statement. Instead of keeping track of a text defining a state transformation, we supply this transformation as an extra argument of N. Such a transformation ϕ is called a *continuation*, and it is meant to describe the effect of evaluation of the "rest of the program", textually following the statement being defined. Summarizing: if ϕ specifies how evaluation of the program proceeds once the right-hand end of A has been reached, and if γ specifies for every label L how evaluation of the program proceeds once we have reached L, then we want $N[\![A]\!]\gamma\phi$ to specify the evaluation of the program starting from the left-hand end of A.

This approach also solves the problem how to define the meaning of $(A_1;A_2)$ in terms of the meanings of A_1 and A_2. The meaning $N[\![(A_1;A_2)]\!]\gamma\phi$ of $(A_1;A_2)$ in environment γ with continuation ϕ, will be equal to the meaning of A_1 in environment γ, but now with a new continuation ϕ'. For if evaluation of A_1 terminates normally (i.e. not through execution of a goto statement), then afterwards the statement A_2 has to be evaluated. Thus the continuation ϕ' must be equal to the meaning of A_2 in environment γ with continuation ϕ.

The exact definition of N will use some new domains which will be defined now:

a) $M = (\Sigma \to_s \Sigma)$ with typical elements ϕ,ψ, is the domain of the *continuations*. We use the convention that continuations appearing as an argument of N will be enclosed in curly brackets if that improves readability.

b) $\Gamma = (Lvar \to M)$ with typical element γ, is the domain of the *environments*. We define a *variant* $\gamma\{\phi/L\}$ *of an environment* the same way as we did in the case of states: $(\gamma\{\phi/L\})[\![L']\!] = \gamma[\![L]\!]$ if $L' \neq L$, and ϕ if $L' \equiv L$. We also use a simultaneous version: $(\gamma\{\phi_i/L_i\}_{i=1}^n)[\![L]\!] = \gamma[\![L]\!]$ if $L \neq L_i$ for $i = 1,\ldots,n$, and ϕ_i if $L \equiv L_i$ (when we use such a construct, all L_i will be different).

DEFINITION (N). The semantic function N with functionality

$$N: Stat \to \Gamma \to M \to M$$

is defined inductively by

$$N[\![x:=s]\!]\gamma\phi\sigma = \begin{cases} \perp & , \text{ if } \sigma=\perp \\ \phi(\sigma\{V[\![s]\!]\sigma/x\}), & \text{otherwise,} \end{cases}$$

$$N[\![(A_1;A_2)]\!]\gamma\phi\sigma = N[\![A_1]\!]\gamma\{N[\![A_2]\!]\gamma\phi\}\sigma,$$

$$N[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}]\!]\gamma\phi\sigma = \begin{cases} \perp & , \text{ if } \sigma=\perp \\ N[\![A_1]\!]\gamma\phi\sigma, & \text{if } \sigma\neq\perp \text{ and } W[\![b]\!]\sigma = tt \\ N[\![A_2]\!]\gamma\phi\sigma, & \text{if } \sigma\neq\perp \text{ and } W[\![b]\!]\sigma = ff, \end{cases}$$

$$N[\![\underline{\text{goto}}\ L]\!]\gamma\phi\sigma = \gamma[\![L]\!]\sigma.$$

The claim on the functionality of N in the above definition must be justified. It is though easy to show that $\forall A \in Stat\ \ \forall\gamma \in \Gamma\ \ \forall\phi \in M$: $N[\![A]\!]\gamma\phi \in M$.

The following lemma holds:

LEMMA 4.1. *For all* $A \in Stat$, *and all* $\gamma \in \Gamma$ *we have*

$$\lambda\langle\phi_1,\ldots,\phi_{n+1}\rangle\cdot N[\![A]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\phi_{n+1} \in [M^{n+1} \to M].$$

PROOF. Straightforward by induction on the structure of A. □

We now turn to the definition of the meaning of programs. The semantic function M: $Prog \to \Gamma \to M$ will be used for this purpose.

A program $S \equiv [L_i:A_i]_{i=1}^n$ can be considered as a combination of a statement $A_1;\ldots;A_n$ and a definition of the labels $L_1,\ldots,L_n$. The state $M[\![[L_i:A_i]_{i=1}^n]\!]\gamma\sigma$ is meant to be the final state reached by evaluation of $A_1;\ldots;A_n$ starting in initial state σ, where the labels L_i are defined by S (for $i = 1,\ldots,n$) and all other labels by γ.

DEFINITON (M). $M[\![[L_i:A_i]_{i=1}^n]\!]\gamma = N[\![A_1;\ldots;A_n]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\}$, where

$$<\phi_1,\ldots,\phi_n> = \mu[\lambda<\psi_1,\ldots,\psi_n>\cdot<N[\![A_i]\!](\gamma\{\psi_j/L_j\}_{j=1}^n)\psi_{i+1}>_{i=1}^n],$$

$$\text{where} \quad \psi_{n+1} = \lambda\sigma\cdot\sigma.$$

REMARKS.

a) There is an assumption in the above definition that has to be justified. We have to show that the operator of which $<\phi_1,\ldots,\phi_n>$ should be the least fixed point is a continuous one, i.e. a member of $[M^n \to M^n]$. In that case this least fixed point exists. The fact that this transformation is continuous can be proved using lemma 4.1.

b) The function ϕ_i can intuitively be seen as the state transformation defined by evaluation of $A_i;\ldots;A_n$ where the labels are defined by S and γ. This might be clarified as follows. By *fpp* we have

$$\phi_n = N[\![A_n]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\}$$

and

$$\phi_{n-1} = N[\![A_{n-1}]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\phi_n =$$

$$= N[\![A_{n-1}]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{N[\![A_n]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\}\} =$$

$$= N[\![A_{n-1};A_n]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\}.$$

Repeating this argument we get

$$\phi_i = N[\![A_i;\ldots;A_n]\!](\gamma\{\phi_j/L_j\}_{j=1}^n)\{\lambda\sigma\cdot\sigma\} \quad (i = 1,\ldots,n).$$

Moreover, these ϕ_i are precisely the values which we would expect to be associated with the labels L_i.

For later reference we state the following definitions and results.

LEMMA 4.2. *Let* $S \equiv [L_i:A_i]_{i=1}^n \in Prog$, *let* $\gamma \in \Gamma$ *and let* ϕ_i *be derived from* S *and* γ *as in the definition of* M. *Also, let* $\phi_i^{(k)}$ *and* $\gamma^{(k)}$ *be defined inductively by:*

$$\phi_i^{(0)} = \lambda\sigma\cdot\perp \qquad \textit{for } i = 1,\ldots,n$$

$$\phi_{n+1}^{(k)} = \lambda\sigma\cdot\sigma \qquad \textit{for } k = 0,1,\ldots$$

$$\gamma^{(k)} = \gamma\{\phi_j^{(k)}/L_j\}_{j=1}^n \qquad \textit{for } k = 0,1,\ldots$$

$$\phi_i^{(k+1)} = M[\![A_i]\!]\gamma^{(k)}\phi_{i+1}^{(k)} \qquad \textit{for } i = 1,\ldots,n$$

4.2.1. $$\phi_i = \bigsqcup_k \phi_i^{(k)} \qquad (i = 1,\ldots,n).$$

PROOF. This is a straightforward consequence of facts 5 and 6c from the theoretical remarks in the beginning of this chapter. □

4.2.2. $$\phi_i = M[\![A_i;\ldots;A_n]\!](\gamma\{\phi_j/L_j\}_{j=1}^n)\{\lambda\sigma\cdot\sigma\}.$$

PROOF. See remark b) above. □

4.2.3. $$\phi_i^{(k)} \sqsubseteq M[\![A_i;\ldots;A_n]\!]\gamma^{(k-1)}\{\lambda\sigma\cdot\sigma\} \qquad (1 \le i \le n,\ k = 1,2,\ldots).$$

PROOF. Induction on k. The basic step (k = 1) can be proved using the fact that $\lambda\phi\cdot M[\![A]\!]\gamma\phi$ is monotonic and that $M[\![A_i;\ldots A_n]\!]\gamma\phi = M[\![A_i]\!]\gamma\{M[\![A_{i+1};\ldots;A_n]\!]\gamma\phi\}$. The induction step is proved as follows:

$$\phi_i^{(k)} = M[\![A_i]\!]\gamma^{(k-1)}\phi_{i+1}^{(k-1)} =$$

$$= M[\![A_i]\!]\gamma^{(k-1)}\{M[\![A_{i+1}]\!]\gamma^{(k-2)}\phi_{i+2}^{(k-2)}\} = (\#).$$

Now we use lemma 4.1, and the fact that continuity implies monotonicity to

show that $N[\![A_{i+1}]\!]\gamma^{(k-2)}\phi_{i+2}^{(k-2)} \sqsubseteq N[\![A_{i+1}]\!]\gamma^{(k-1)}\phi_{i+2}^{(k-1)}$ and thus, using 4.1 again:

$$(\#) \sqsubseteq N[\![A_i]\!]\gamma^{(k-1)}\{N[\![A_{i+1}]\!]\gamma^{(k-1)}\phi_{i+2}^{(k-1)}\} =$$

$$= N[\![A_i;A_{i+1}]\!]\gamma^{(k-1)}\phi_{i+2}^{(k-1)}.$$

Repeating the argument we get

$$\phi_i^{(k)} \sqsubseteq N[\![(..(A_i;A_{i+1});...);A_n)]\!]\gamma^{(k-1)}\{\lambda\sigma\cdot\sigma\} =$$

$$= N[\![A_i;A_{i+1};...;A_n]\!]\gamma^{(k-1)}\{\lambda\sigma\cdot\sigma\},$$

where the last identity is easy to prove from the definition of N. □

We will close this chapter by taking another look at the meaning of statements A. We saw that the function $N[\![A]\!]$ essentially yields a continuation as a result. This result depends on a number of continuations, which are supplied to $N[\![A]\!]$ either directly as an argument (the ϕ in $N[\![A]\!]\gamma\phi$) or implicitly through γ, as meaning of the labels occurring in A. In the literature (MILNE & STRACHEY [8]) a method called "continuation removal" is described to dispose of the ϕ in the above formula, yielding a more direct approach: the meaning of a statement is a state transformation instead of a continuation transformation. This has only been done for statements A which didn't contain goto statements as substatements.

We now take one further step: we show how to deal with goto-substatements. We will define a function A giving the meaning of a statement A as a (total) function from Σ_0 to $\Sigma_0 \cup (\Sigma_0 \times Lvar)$, such that

$$A[\![A]\!]\sigma = \sigma'$$

means that evaluation of A terminates normally in state σ' (i.e. not as the result of an execution of a goto statement), and

$$A[\![A]\!]\sigma = \langle\sigma', L\rangle$$

means that evaluation of A terminates by execution of a substatement <u>goto</u> L

in state σ'.

Put another way, a statement A containing goto-substatements can be viewed as a statement with one *entry point* (where evaluation of A starts), but with several *exit points*, namely the *normal exit point* (the right-hand end of the statement) and the *special exit points* (viz. the substatements goto L). We call an exit point determined by a substatement goto L an L-*exit point*. The function $\dot{A}$ then specifies for every initial state σ the kind of exit point which will be reached and the final state in which this exit point will be reached. This is a formalization of the considerations by ARBIB & ALAGIĆ [3].

The function $\dot{A}$, applied to a statement A and an initial state σ, thus yields a final state σ' which is the result of evaluation of A, and not of evaluation of A followed by some continuation (as was the case in $N[\![A]\!]\gamma\phi\sigma$). Since in the deduction systems to be discussed later we deal with formulae $\{p\}A\{q\}$, where q is a predicate on the final state at the normal exit point, we can expect that the function $\dot{A}$ will be more useful than N (see chapter 6).

We now give the definition of $\dot{A}$.

DEFINITION ($\dot{A}$). The function $\dot{A}$ with functionality $\dot{A}$: $Stat \to \Sigma_0 \to \Sigma_0 \cup (\Sigma_0 \times Lvar)$ is inductively defined by

$$\dot{A}[\![x:=s]\!]\sigma = \sigma\{V[\![s]\!]\sigma/x\}$$

$$\dot{A}[\![(A_1;A_2)]\!]\sigma = \begin{cases} \dot{A}[\![A_2]\!](\dot{A}[\![A_1]\!]\sigma), & \text{if } \dot{A}[\![A_1]\!]\sigma \in \Sigma_0 \\ \dot{A}[\![A_1]\!]\sigma, & \text{otherwise} \end{cases}$$

$$\dot{A}[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}]\!]\sigma = \begin{cases} \dot{A}[\![A_1]\!]\sigma, & \text{if } W[\![b]\!]\sigma = tt \\ \dot{A}[\![A_2]\!]\sigma, & \text{if } W[\![b]\!]\sigma = ff \end{cases}$$

$$\dot{A}[\![\underline{\text{goto}}\ L]\!]\sigma = \langle\sigma, L\rangle.$$

We have the following lemma on the relation between $\dot{A}$ and N.

LEMMA 4.3.

1°. $\dot{A}[\![A]\!]\sigma = \sigma' \iff \forall\gamma \in \Gamma\ \ \forall\phi \in M\colon N[\![A]\!]\gamma\phi\sigma = \phi\sigma'$

2°. $\dot{A}[\![A]\!]\sigma = \langle\sigma', L\rangle \iff \forall\gamma \in \Gamma\ \ \forall\phi \in M\colon N[\![A]\!]\gamma\phi\sigma = \gamma[\![L]\!]\sigma'$.

PROOF. The $\Rightarrow$-parts of 1° and 2° are straightforward by structural induction. The $\Leftarrow$-parts can be proven by contradiction. For instance, proving 2° "$\Leftarrow$", suppose $\forall\gamma \in \Gamma \quad \forall\phi \in M$: $N[\![A]\!]\gamma\phi\sigma = \gamma[\![L]\!]\sigma'$, and $A[\![A]\!]\sigma \neq \langle\sigma',L\rangle$. Then we have two possibilities.

The first one is $A[\![A]\!]\sigma = \langle\sigma'',L'\rangle$ (where $\sigma' \neq \sigma''$ or $L \neq L'$) and thus, using 2° "$\Rightarrow$", $\forall\gamma \in \Gamma \quad \forall\phi \in M$: $N[\![A]\!]\gamma\phi\sigma = \gamma[\![L']\!]\sigma''$. Now choose γ such that $\gamma[\![L]\!]\sigma' \neq \gamma[\![L']\!]\sigma''$ and we have a contradiction.

The other possibility is $A[\![A]\!]\sigma = \sigma''$. Then we have ($1^{\circ}$ "$\Rightarrow$") $\forall\gamma \in \Gamma \quad \forall\phi \in M$: $N[\![A]\!]\gamma\phi\sigma = \phi\sigma''$, and we reach a contradiction by choosing γ and ϕ such that $\gamma[\![L]\!]\sigma' \neq \phi\sigma''$. □

5. OPERATIONAL AND DENOTATIONAL SEMANTICS ARE EQUIVALENT

Our aim in this chapter is to prove the following

THEOREM 5.1. *Let* $S \equiv [L_i:A_i]_{i=1}^n$ *be a program. If* S *is normal (i.e. all labels in* S *are declared) then*

$$\forall\gamma \in \Gamma: O[\![S]\!] = M[\![S]\!]\gamma.$$

PROOF. We first prove $O[\![S]\!] \sqsubseteq M[\![S]\!]\gamma$, using the following

LEMMA 5.2. *Let* $S \equiv [L_i:A_i]_{i=1}^n \in Prog$, $\gamma \in \Gamma$, *and let* ϕ_i *be derived from* S *and* γ *as in the definition of* M. *Let* $A \in Stat$ *and let all labels occurring in* A *and* S *be declared in* S. *Then*

$$\forall\sigma \in \Sigma: \kappa(Comp[\![\langle S,A\rangle]\!]\sigma) \sqsubseteq N[\![A]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\}\sigma \qquad \ldots(*)$$

PROOF of the lemma. Because all labels are declared in S, it is impossible that $Comp[\![\langle S,A\rangle]\!]\sigma$ be undefined. If $\sigma=\perp$, or if $\sigma\neq\perp$ and $Comp[\![\langle S,A\rangle]\!]\sigma \in \Sigma^{\omega}$, then the left-hand side of (*) is equal to $\perp$, and the inequality holds. So let us assume that $\sigma\neq\perp$ and $Comp[\![\langle S,A\rangle]\!]\sigma \in \Sigma^{+}$. We prove (*) using induction on the length of the computation sequence $Comp[\![\langle S,A\rangle]\!]\sigma$ (in fact, assuming that this computation sequence is finite, we can prove equality in (*)). We distinguish several cases, depending on the structure of A. We shall abbreviate $\gamma\{\phi_i/L_i\}_{i=1}^n$ to $\bar{\gamma}$.

a) $A \equiv x:=s$. Then the left-hand side of (*) is equal to $\sigma\{\mathcal{V}[\![s]\!]\sigma/x\}$, and so is the right-hand side.

b) $A \equiv$ *if* b *then* A_1 *else* A_2 *fi*. Assume $\mathcal{W}[\![b]\!]\sigma = tt$ (the other case can be proved analogously). Then $\kappa(Comp[\![<S,A>]\!]\sigma) = \kappa(<\sigma>^{\frown}Comp[\![<S,A_1>]\!]\sigma)$. Now the length of $Comp[\![<S,A_1>]\!]\sigma$ is clearly one less than the length of $Comp[\![<S,A>]\!]\sigma$, so we can apply the induction hypothesis, yielding

$$\kappa(Comp[\![<S,A>]\!]\sigma) = \kappa(Comp[\![<S,A_1>]\!]\sigma) \sqsubseteq$$

$$\sqsubseteq \mathcal{N}[\![A_1]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma = \mathcal{N}[\![A]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma.$$

c) $A \equiv$ *goto* L. Because all labels in A are defined in S, we have $L \equiv L_j$ for some j. Thus

$$\kappa(Comp[\![<[L_i:A_i]_{i=1}^{n},A>]\!]\sigma) =$$

$$= \kappa(<\sigma>^{\frown}Comp[\![<S,A_j;\ldots;A_n>]\!]\sigma) \sqsubseteq \quad \text{(ind. hyp.)}$$

$$\sqsubseteq \mathcal{N}[\![A_j;\ldots;A_n]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma = \quad \text{(4.2.2)}$$

$$= \phi_j\sigma = \bar{\gamma}[\![L_j]\!]\sigma =$$

$$= \mathcal{N}[\![\textit{goto } L_j]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma.$$

d) $A \equiv (x:=s;A')$.

$$\kappa(Comp[\![<S,A>]\!]\sigma) = \kappa(Comp[\![<S,A'>]\!](\sigma\{\mathcal{V}[\![s]\!]\sigma/x\})) \sqsubseteq \quad \text{(ind.)}$$

$$\sqsubseteq \mathcal{N}[\![A']\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}(\sigma\{\mathcal{V}[\![s]\!]\sigma/x\}) = \quad \text{(def. } \mathcal{N}\text{)}$$

$$= \mathcal{N}[\![x:=s]\!]\bar{\gamma}\{\mathcal{N}[\![A']\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\}\sigma = \quad \text{(def. } \mathcal{N}\text{)}$$

$$= \mathcal{N}[\![(x:=s;A')]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma.$$

e) $A \equiv ((A_1;A_2);A')$

$$\kappa(Comp[\![<S,A>]\!]\sigma) = \kappa(Comp[\![<S,(A_1;(A_2;A'))>]\!]\sigma) \sqsubseteq \quad \text{(ind.)}$$

$$\sqsubseteq \mathcal{N}[\![(A_1;(A_2;A'))]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma =$$

$$= \mathcal{N}[\![((A_1;A_2);A')]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma.$$

f) $A \equiv$ (*if* b *then* A_1 *else* A_2 *fi*;A'). We suppose, without loss of generality, that $\mathcal{W}[\![b]\!]\sigma = tt$. Then

$$\kappa(\mathit{Comp}[\![\langle S,A\rangle]\!]\sigma) = \kappa(\mathit{Comp}[\![\langle S,A_1;A'\rangle]\!]\sigma) \sqsubseteq \quad \text{(ind. hyp.)}$$

$$\sqsubseteq N[\![A_1;A']\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma = \quad \text{(def. } N\text{)}$$

$$= N[\![A_1]\!]\bar{\gamma}\{N[\![A']\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\}\sigma = \quad \text{(def. } N\text{)}$$

$$= N[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}]\!]\bar{\gamma}\{N[\![A']\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\}\sigma =$$

$$= N[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}};A')]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma.$$

g) $A \equiv (\underline{\text{goto}}\ L;A')$.

$$\kappa(\mathit{Comp}[\![\langle S,A\rangle]\!]\sigma) = \kappa(\mathit{Comp}[\![\langle S,\ \underline{\text{goto}}\ L\rangle]\!]\sigma) \sqsubseteq \quad \text{(ind. hyp.)}$$

$$\sqsubseteq N[\![\underline{\text{goto}}\ L]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma = \quad \text{(def. } N\text{)}$$

$$= N[\![\underline{\text{goto}}\ L]\!]\bar{\gamma}\{N[\![A']\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\}\sigma = \quad \text{(def. } N\text{)}$$

$$= N[\![(\underline{\text{goto}}\ L;A')]\!]\bar{\gamma}\{\lambda\sigma\cdot\sigma\}\sigma.$$

This ends the proof of lemma 5.2. □

We now use the lemma to prove $O[\![S]\!] \sqsubseteq M[\![S]\!]\gamma$ in the following way. Choose $\gamma \in \Gamma$ and $\sigma \in \Sigma$. By definition of O we have

$$O[\![S]\!]\sigma = \kappa(\mathit{Comp}[\![\langle S,A_1;\ldots;A_n\rangle]\!]\sigma).$$

Now all labels in S are declared and thus the same holds for $A_1;\ldots;A_n$. The lemma then gives us

$$\kappa(\mathit{Comp}[\![\langle S,A_1;\ldots;A_n\rangle]\!]\sigma) \sqsubseteq N[\![A_1;\ldots;A_n]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\}\sigma,$$

where the ϕ_i are obtained from S and γ as in the definition of M. But, for those ϕ_i, the definition of M gives us

$$N[\![A_1;\ldots;A_n]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\}\sigma = M[\![S]\!]\gamma\sigma$$

which gives us the desired result.

For the proof of $M[\![S]\!]\gamma \sqsubseteq O[\![S]\!]$, we again use a lemma:

LEMMA 5.3. *Let* $S \equiv [L_i:A_i]_{i=1}^n \in \mathit{Prog}$ *be normal. Let* $\gamma \in \Gamma$ *and* $k \in \mathbb{N}$. *Let* $\phi_i, \phi_i^{(k)}$ *and* $\gamma^{(k)}$ *be derived from* S *and* γ *as in lemma* 4.2. *Then, for all* $A \in \mathit{Stat}$ *such that all labels in* A *are declared in* S, *and for all* $\sigma \in \Sigma$,

we have

$$N[\![A]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma \sqsubseteq \kappa(Comp[\![<S,A>]\!]\sigma) \qquad \ldots(+)$$

PROOF of the lemma. We use induction on the entity $<k,c[\![A]\!]>$ with lexicographic ordering $\prec$. We don't take $c[\![A]\!]$ to be the obvious complexity of A. This wouldn't work because of the form of the definition of *Comp*. For instance, in rule 5 the statement (A";(A"';A')) occurring in the right-hand side of the rule would be as complex, according to the usual complexity measure, as ((A";A");A') in the left-hand side.

We define $c[\![A]\!]$ inductively by: $c[\![x:=s]\!] = c[\![\underline{goto}\ L]\!]=1$; $c[\![A_1;A_2]\!] = 2c[\![A_1]\!]+c[\![A_2]\!]$; $c[\![\underline{if}\ b\ \underline{then}\ A_1\ \underline{else}\ A_2\ \underline{fi}]\!] = c[\![A_1]\!]+c[\![A_2]\!]$.

a) $A \equiv x:=s$. Then $N[\![x:=s]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma = \kappa(Comp[\![<S,A>]\!]\sigma) = \sigma\{V[\![s]\!]\sigma/x\}$.

b) $A \equiv \underline{if}\ b\ \underline{then}\ A_1\ \underline{else}\ A_2\ \underline{fi}$. Without loss of generality, we assume $W[\![b]\!]\sigma = tt$. Then the left-hand side of (+) equals $N[\![A_1]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma$, and the right-hand side equals $\kappa(Comp[\![<S,A_1>]\!]\sigma)$. Now, because $<k,c[\![A_1]\!]> \prec <k,c[\![A]\!]>$, the desired result follows from the induction hypothesis.

c) $A \equiv \underline{goto}\ L$. From the assumptions of the lemma, we infer that $L \equiv L_i$ for some i. Now:

$$N[\![\underline{goto}\ L_i]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma = (\gamma^{(k)})[\![L_i]\!]\sigma = \phi_i^{(k)}\phi \sqsubseteq \qquad (4.2.3)$$

$$\sqsubseteq N[\![A_i;\ldots;A_n]\!]\gamma^{(k-1)}\{\lambda\sigma\cdot\sigma\}\sigma.$$

Also $<k-1,c[\![A_i;\ldots A_n]\!]> \prec <k,c[\![\underline{goto}\ L_i]\!]>$, so we can apply the induction hypothesis

$$N[\![A_i;\ldots;A_n]\!]\gamma^{(k-1)}\{\lambda\sigma\cdot\sigma\}\sigma \sqsubseteq$$

$$\sqsubseteq \kappa(Comp[\![<S,A_i;\ldots;A_n>]\!]\sigma) = \qquad \text{(def. } Comp\text{)}$$

$$= \kappa(Comp[\![<S,\underline{goto}\ L_i>]\!]\sigma).$$

d) $A \equiv (x:=s;A')$.

$$N[\![(x:=s;A')]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma =$$

$$= N[\![A']\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}(\sigma\{V[\![s]\!]\sigma/x\}) \sqsubseteq \qquad (c[\![A']\!] \prec c[\![x:=s;A']\!])$$

$$\sqsubseteq \kappa(Comp[\![<S,A'>]\!](\sigma\{V[\![s]\!]\sigma/x\})) =$$

$$= \kappa(Comp[\![<S,(x:=s;A')>]\!]\sigma).$$

e) $A \equiv ((A_1;A_2);A')$. We have $N[\![((A_1;A_2);A')]\!] = N[\![(A_1;(A_2;A'))]\!]$ and $c[\![(A_1;(A_2;A'))]\!] \prec c[\![((A_1;A_2);A')]\!]$. The induction hypothesis thus yields

$$N[\![(A_1;(A_2;A'))]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma \sqsubseteq$$
$$\sqsubseteq \kappa(Comp[\![<S,(A_1;(A_2;A'))>]\!]\sigma) =$$
$$= \kappa(Comp[\![<S,((A_1;A_2);A')>]\!]\sigma).$$

f) $A \equiv (\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}};A')$. Without loss of generality, we assume that $\mathcal{W}[\![b]\!]\sigma = tt$. We then have

$$N[\![A]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma = N[\![A_1;A']\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma \sqsubseteq \qquad \text{(ind.hyp.)}$$
$$\sqsubseteq \kappa(Comp[\![<S,A_1;A'>]\!]\sigma) = \kappa(Comp[\![<S,A>]\!]\sigma).$$

g) $A \equiv (\underline{\text{goto}}\ L;A')$.

$$N[\![(\underline{\text{goto}}\ L;A')]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma = N[\![\underline{\text{goto}}\ L]\!]\gamma^{(k)}\{\lambda\sigma\cdot\sigma\}\sigma \sqsubseteq \quad \text{(ind. hyp.)}$$
$$\sqsubseteq \kappa(Comp[\![<S,\underline{\text{goto}}\ L>]\!]\sigma) = \kappa(Comp[\![<S,\underline{\text{goto}}\ L;A'>]\!]\sigma).$$

This concludes the proof of lemma 5.3. □

We now are able to prove $M[\![S]\!]\gamma \sqsubseteq O[\![S]\!]$.

$$M[\![S]\!]\gamma = N[\![A_1;\ldots;A_n]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\} =$$
$$= N[\![A_1;\ldots;A_n]\!](\gamma\{\bigsqcup_k \phi_i^{(k)}/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\} = \qquad \text{(lemma 4.1)}$$
$$= \bigsqcup_k (N[\![A_1;\ldots;A_n]\!](\gamma\{\phi_i^{(k)}/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\}).$$

Now, taking $A \equiv A_1;\ldots;A_n$, the assumptions of lemma 5.3 are satisfied. Thus we can conclude

$$\forall k \in \mathbb{N}:\ N[\![A_1;\ldots;A_n]\!](\gamma\{\phi_i^{(k)}/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\} \sqsubseteq \kappa \circ Comp[\![<S,A_1;\ldots;A_n>]\!],$$

and thus

$$\bigsqcup_k N[\![A_1;\ldots;A_n]\!](\{\phi_i^{(k)}/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\} \sqsubseteq \kappa \circ Comp[\![<S,A_1;\ldots;A_n>]\!].$$

But also, by definition of O:

$$\kappa \circ Comp[\![<S,A_1;\ldots;A_n>]\!] = O[\![S]\!],$$

which completes the proof. □

Note that theorem 5.1 is independent of the interpretation of the primitive relation and function symbols chosen, in the sense that the theorem holds for all underlying interpretations I.

6. DEDUCTION SYSTEM: FIRST VARIANT

In [7] HOARE proposed to attach meanings to programs by means of a proof system which can be used to derive properties of programs. These properties are described by (partial) *correctness formulae*, essentially having the form $\{p\}S\{q\}$. Such a construct has informally the following meaning: if evaluation of S terminates, starting from an initial state in which p (the *precondition*) holds, then in the final state q (the *postcondition*) holds.

We start with a discussion of these conditions p, which in the sequel will be called *assertions*. The class of all assertions is $Assn$, with typical elements p,q. We define:

$$p ::= \underline{\text{true}} \mid p_1 \vee p_2 \mid \neg p \mid re_1(s_1,\ldots,s_{arr_1}) \mid \ldots \mid re_n(s_1,\ldots,s_{arr_n}) \mid \exists x[p].$$

We define $\underline{\text{false}}$, $p_1 \wedge p_2$, $p_1 \supset p_2$ and $\underline{\text{if}}$ b $\underline{\text{then}}$ p_1 $\underline{\text{else}}$ p_2 $\underline{\text{fi}}$ as in chapter 2.

The assertions are meant to describe predicates on states. The semantic function giving the meaning of assertions is T and has functionality

$$T: Assn \to \Sigma_0 \to \{ff,tt\}.$$

T is defined inductively by:

a) $T[\![\underline{\text{true}}]\!]\sigma = tt$

b) $T[\![p_1 \vee p_2]\!]\sigma = tt$ if $T[\![p_1]\!]\sigma = tt$ or $T[\![p_2]\!]\sigma = tt$, and ff otherwise

c) $T[\![\neg p]\!]\sigma = tt$ if $T[\![p]\!]\sigma = ff$, and ff otherwise

d) $T[\![re_i(s_1,\ldots,s_{arr_i})]\!]\sigma = tt$ if $\langle V[\![s_1]\!]\sigma,\ldots,V[\![s_{arr_i}]\!]\sigma\rangle \in \underline{re_i}$, and ff otherwise

e) $T[\![\exists x[p]]\!]\sigma = tt$ if there exists an element d in our domain of interpretation D such that $T[\![p]\!](\sigma\{d/x\}) = tt$, and ff otherwise.

Note that T depends on the underlying interpretation I, because V does (d)), and also through clause e) of the definition.

Next some definitions and results on substitution. We say that an occurrence of a variable x in an assertion p is *bound*, if this occurrence is within a sub-assertion of the form $\exists x[p']$. An occurrence of x in an assertion p is called *free* if it is not bound.

The result of substituting all (free) occurrences of x in s and p by t, will be denoted by $s[t/x]$ and $p[t/x]$ respectively. The definition of $s[t/x]$ is

a) $y[t/x] \equiv t$ if $y \equiv x$ and y otherwise

b) $(fu_i(s_1,\ldots,s_{arf_i}))[t/x] \equiv fu_i(s_1[t/x],\ldots,s_{arf_i}[t/x])$.

Using this definition $p[t/x]$ can be defined by

a) $\underline{\text{true}}[t/x] \equiv \underline{\text{true}}$

b) $(p_1 \vee p_2)[t/x] \equiv p_1[t/x] \vee p_2[t/x]$

c) $(\neg p)[t/x] \equiv \neg(p[t/x])$

d) $(re_i(s_1,\ldots,s_{arr_i}))[t/x] \equiv re_i(s_1[t/x],\ldots,s_{arr_i}[t/x])$

e) $(\exists y[p])[t/x] \equiv$
$\begin{cases} \exists y[p], \text{ if } x \equiv y \\ \exists y[p[t/x]], \text{ if } x \not\equiv y \text{ and } y \text{ does not occur in } t \\ \exists z[p[z/y][t/x]] \text{ if } x \not\equiv y \text{ and } y \text{ occurs in } t, \text{ where } z \\ \qquad \text{is the first variable in } \mathit{Var} \text{ such that } z \not\equiv x, \\ \qquad z \text{ doesn't occur in } t, \ z \text{ doesn't occur free in } p. \end{cases}$

The following results on substitution will be useful.

LEMMA 6.1.

a) *If* x *doesn't occur in* s *then* $\forall d \in D: \mathcal{V}[\![s]\!]\sigma = \mathcal{V}[\![s]\!](\sigma\{d/x\})$

b) *if* x *doesn't occur free in* p *then* $\forall d \in D: \mathcal{T}[\![p]\!]\sigma = \mathcal{T}[\![p]\!](\sigma\{d/x\})$

c) $\mathcal{V}[\![s[t/x]]\!]\sigma = \mathcal{V}[\![s]\!](\sigma\{\mathcal{V}[\![t]\!]\sigma/x\})$

d) $\mathcal{T}[\![p[t/x]]\!]\sigma = \mathcal{T}[\![p]\!](\sigma\{\mathcal{V}[\![t]\!]\sigma/x\})$.

PROOF. Straightforward by induction. We prove the hardest case of d), i.e. where the assertion has the form $\exists y[p]$. There are three cases.

1) $y \equiv x$. $\mathcal{T}[\![\exists x[p][t/x]]\!]\sigma = \mathcal{T}[\![\exists x[p]]\!]\sigma = tt$ iff $\exists d \in D: \mathcal{T}[\![p]\!](\sigma\{d/x\}) = tt$.
   Now $\mathcal{T}[\![p]\!](\sigma\{d/x\}) = \mathcal{T}[\![p]\!](\sigma\{\mathcal{V}[\![t]\!]\sigma/x\}\{d/x\})$, and therefore $\mathcal{T}[\![\exists x[p]]\!]\sigma = tt$, iff $\exists d \in D: \mathcal{T}[\![p]\!](\sigma\{\mathcal{V}[\![t]\!]\sigma/x\}\{d/x\}) = tt$, and this is true whenever $\mathcal{T}[\![\exists x[p]]\!](\sigma\{\mathcal{V}[\![t]\!]\sigma/x\}) = tt$.

2) $y \not\equiv x$ and y doesn't occur in t. $\mathcal{T}[\![(\exists y[p])[t/x]]\!]\sigma = \mathcal{T}[\![\exists y[p[t/x]]]\!]\sigma = tt$ iff $\exists d \in D: \mathcal{T}[\![p[t/x]]\!](\sigma\{d/y\}) = tt$ (ind. hyp.)

iff $\exists d \in D$: $T[\![p]\!](\sigma\{d/y\}\{V[\![t]\!](\sigma\{d/y\})/x\}) = tt$ (a))

iff $\exists d \in D$: $T[\![p]\!](\sigma\{V[\![t]\!]\sigma/x\}\{d/y\}) = tt$

iff $T[\![p]\!](\sigma\{V[\![t]\!]\sigma/x\}) = tt$.

3) $y \neq x$ and y occurs in t. $T[\![\exists y[p][t/x]]\!]\sigma = T[\![\exists z[p[z/y][t/x]]]\!]\sigma$ = (#)
where $z \neq x$, z doesn't occur in t, z doesn't occur free in p.
Now (#) = tt iff $\exists d \in D$: $T[\![p[z/y][t/x]]\!](\sigma\{d/z\}) = tt$ (ind. hyp.)

iff $\exists d \in D$: $T[\![p]\!](\sigma\{d/z\}\{V[\![t]\!]\sigma'/x\}\{\sigma"[\![x]\!]/y\}) = tt$,

where $\sigma' = \sigma\{d/z\}$ and $\sigma" = \sigma'\{V[\![t]\!]\sigma'/x\}$. Now $x \neq z$, so $\sigma"[\![z]\!] = d$.
Furthermore z doesn't occur in t, so $V[\![t]\!]\sigma' = V[\![t]\!](\sigma\{d/z\}) = V[\![t]\!]\sigma$.
Thus we get

(#) = tt iff $\exists d \in D$: $T[\![p]\!](\sigma\{d/z\}\{V[\![t]\!]\sigma/x\}\{d/y\}) = tt$.

Because z doesn't occur in t, and y does, we have $z \neq y$. Also we have $z \neq x$ and $y \neq x$, so

(#) = tt iff $\exists d \in D$: $T[\![p]\!](\sigma\{V[\![t]\!]\sigma/x\}\{d/y\}\{d/z\}) = tt$.

Because z doesn't occur free in p, we can use result b) of the lemma, to get

(#) = tt iff $\exists d \in D$: $T[\![p]\!](\sigma\{V[\![t]\!]\sigma/x\}\{d/y\}) = tt$
iff $T[\![\exists y[p]]\!](\sigma\{V[\![t]\!]\sigma/x\}) = tt$. □

Having defined assertions and substitution, we now proceed to describe how these notions are to be used in correctness formulae. A typical axiom of our proof system will be the *assignment axiom*, roughly of the form

$\{p[s/x]\}x:=s\{p\}$.

This axiom can be justified by the following considerations. The statement x:=s transforms an initial state σ to a final state $\sigma' = \sigma\{V[\![s]\!]\sigma/x\}$. Now suppose $p[s/x]$ is true in σ, that is, $T[\![p[s/x]]\!]\sigma = tt$, or (lemma 6.1d) $T[\![p]\!](\sigma\{V[\![s]\!]\sigma/x\}) = tt$. But $\sigma\{V[\![s]\!]\sigma/x\}$ is equal to the final state σ', so we have that p is true in σ', which is what we wanted.

A rule of inference in the system will be the *rule of composition*, stated informally

from $\{p_1\}A_1\{p_2\}$ and $\{p_2\}A_2\{p_3\}$ infer $\{p_1\}A_1;A_2\{p_3\}$.

The justification of this rule goes somewhat like this. Say we start

evaluating $A_1;A_2$ in state σ where p_1 is true. Now, after evaluation of A_1, we have reached an intermediate state σ' where (due to $\{p_1\}A_1\{p_2\}$) the assertion p_2 holds. Evaluating A_2 in state σ' delivers a final state σ'' where p_3 holds, for $\{p_2\}A_2\{p_3\}$ is true. Thus we have the desired result.

Another rule of inference is the *rule of consequence*:

$$\text{from } p_1 \supset p_2,\ p_3 \supset p_4 \text{ and } \{p_2\}A\{p_3\} \text{ infer } \{p_1\}A\{p_4\},$$

which is obviously valid.

The fact that we allow goto statements in our language complicates things. The problem becomes apparent if we take another look at the rule of composition. For instance, if the first statement A_1 in $A \equiv A_1;A_2$ is identical to goto L, then the justification of the rule as given above doesn't apply anymore. After evaluation of A_1, the next statement to be executed is not A_2, as was assumed there. Complications are caused by the fact that a statement A can have more than one exit point, namely the normal exit point and the special L-exit points (cf. the discussion after lemma 4.2).

We can maintain the rule of composition though, if we formulate the meaning of the formula $\{p_1\}A\{p_2\}$ somewhat differently, namely as follows: if A is evaluated beginning in a state where p_1 holds, and evaluation of A terminates *at the normal exit point of* A, in state σ', then p_2 holds in σ'.

Now according to this informal validity definition the formula

$$\{p\}\ \underline{\text{goto}}\ L\ \{q\} \qquad \ldots(\#)$$

would be valid for every assertion p and q, for evaluation of goto L always terminates by "jumping away". However this brings up new problems. For example, the formula

$$\{\underline{\text{true}}\}\ L_1{:}x{:=}1;\ \underline{\text{goto}}\ L_2;L_2{:}x{:=}x\ \{x=0\}$$

would now be derivable, by the following steps

1. $\{\underline{\text{true}}\}\ L_1{:}x{:=}1\ \{x=1\}$ (assignment)
2. $\{x=1\}\ \underline{\text{goto}}\ L_2\ \{x=0\}$ (#)
3. $\{x=0\}\ L_2{:}x{:=}x\ \{x=0\}$ (assignment)
4. $\{\underline{\text{true}}\}\ L_1{:}x{:=}1;\ \underline{\text{goto}}\ L_2;\ L_2{:}x{:=}x\ \{x=0\}$ (composition).

But clearly, after evaluation of L_1:x:=1; goto L_2; L_2:x:=x the postcondition x=1 holds.

These difficulties have been solved by CLINT & HOARE [5]. Their solution is in essence to put a restriction on the preconditions p allowed in (#), and amounts to the following. Suppose we want to prove $\{p\}S\{q\}$, where $S \equiv L_1:A_1;\ldots;L_n:A_n$. Now assume we can find a list of *label invariants* $p_1,\ldots,p_n$. These p_i are assertions which we assume to be true every time label L_i is reached during execution of S, starting in initial state satisfying p. We now refine our notion of validity once more, and define validity (with respect to the invariants p_i at L_i for i = 1,...,n) informally as follows:

(*) The formula $\{p\}A\{q\}$ is called valid, iff for every evaluation of A the following holds: if p holds for the initial state, then *either* evaluation terminates at the normal exit point of A and q holds, *or* evaluation terminates at an L_i-exit point of A and p_i holds (for some i, $1 \leq i \leq n$).

One can see that, according to (*), the formulae $\{p\}$ goto L_i $\{q\}$ are no longer valid for all p. Validity holds however for all preconditions p such that $p \supset p_i$. In particular $\{p_i\}$ goto L_i {false} is valid (i = 1,...,n). Notice also that the inference rules and the assignment axiom given earlier remain valid according to (*).

Now if we can *derive* $\{p_i\}A_i\{p_{i+1}\}$ using these rules and axioms, and also the formulae $\{p_j\}$ goto L_j {false}, then we know that $\{p_i\}A_i\{p_{i+1}\}$ must be valid according to (*). This means the following: if we consider evaluation of A_i as a sub-statement of $S \equiv L_1:A_1;\ldots;L_n:A_n$, starting at an initial state for which p_i holds, then we can infer from the validity of $\{p_i\}A_i\{p_{i+1}\}$ that at the normal exit point p_{i+1} holds, and that at every L_j-exit point p_j holds. In other words: when evaluation of A_i terminates because label L_j has been reached then the corresponding invariant p_j must hold ($1 \leq j \leq n$).

But from this we can infer that $\{p_1\}S\{p_{n+1}\}$ holds. For, consider an evaluation of S with initial state satisfying p_1, and suppose that this evaluation terminates. Then this evaluation can be split up in a finite number of subsequent evaluations of sub-statements A_i, and since by the

above considerations we are assured that at all "links" labelled L_j the corresponding invariant p_j holds we can infer that p_{n+1} is true when the last evaluation of sub-statement A_n terminates (necessarily at the normal exit point).

The above considerations suggest the following inference rule [5]:

> if we can derive $\{p_i\}A_i\{p_{i+1}\}$ $(i = 1,\ldots,n)$ from the assumptions $\{p_j\}$ goto L_j {false} $(j = 1,\ldots,n)$, then we may infer $\{p_1\}L_1:A_1;\ldots;L_n:A_n\{p_{n+1}\}$.

Now the formula {true}S{x=0}, where $S \equiv L_1:x:=1;$ goto $L_2;$ $L_2:x:=x$ (sc. the above incorrect derivation) cannot be derived anymore, but a derivation of {true}S{x=1} can be made straightforwardly (take $p_1 \equiv$ true, $p_2 \equiv x=1$).

The inference rule given above leads to compact proofs but, as it stands, is not so suitable for proof-theoretical considerations. Accordingly, we shall now give a more tractable variant of the proof system. In chapter 8 we shall give a formal justification of the above rule.

It can easily be seen that the assumptions $\{p_j\}$ goto L_j {false} $(j = 1,\ldots,n)$ are introduced in the above inference rule only because our proof system must be able to contain information on the label invariants p_i which are used in the proofs. The method that we apply is to take these invariants up in the formulae occurring in the proofs. Our correctness formulae will look like

$$<L_1:p_1,\ldots,L_n:p_n|\{p\}A\{q\}>,$$

so the invariants p_i corresponding to L_i are supplied explicitly in our formulae, instead of implicitly in the assumptions used in a proof. The informal meaning of the above formula is the one as given by (*).

After this introduction the following definitions must be clear.

DEFINITION (Syntax of correctness formulae).
The class *Invℓ* (*list of label invariants*) with typical element D is defined by

$$D::= L:p|L:p,D$$

where it is required that if $D \equiv L_1:p_1,\ldots,L_n:p_n$, then $L_i \neq L_j$ for $i \neq j$.

We write $[L_i:p_i]_{i=1}^n$ instead of $L_1:p_1,\ldots,L_n:p_n$.

We say (L:p) *occurs in* D (notation: (L:p) in D) iff $L \equiv L_j$, $p \equiv p_j$ and $D \equiv [L_i:p_i]_{i=1}^n$ for some j $(1 \leq j \leq n)$.

The class *Corr* (*correctness formulae*) with typical element f is defined by

$$f ::= p \mid \langle D;\{p\}A\{q\}\rangle \mid \{p\}S\{q\}.$$

We write $\langle D \mid \{p\}A\{q\}\rangle$ instead of $\langle D;\{p\}A\{q\}\rangle$.

DEFINITION (proof system *H*).

The axioms of *H* are given by the following schemes:

(A1) $\langle D \mid \{p[s/x]\}x:=s\{p\}\rangle$

(A2) $\langle D \mid \{p\}$ goto L $\{$false$\}\rangle$,

where $D \equiv [L_i:p_i]_{i=1}^n$, $L \equiv L_j$, $p \equiv p_j$ for some j $(1 \leq j \leq n)$.

(A3) p,

where p is a valid assertion (i.e. $\forall \sigma \in \Sigma_0$: $T[\![p]\!]\sigma = tt$).

The rules of inference have the form

$$\frac{f_1,\ldots,f_n}{f_{n+1}}$$

("from $f_1,\ldots,$ and f_n, infer f_{n+1}") and are given by the following schemes:

(R1) $$\frac{p_1 \supset p_2, p_3 \supset p_4, \langle D \mid \{p_2\}A\{p_3\}\rangle}{\langle D \mid \{p_1\}A\{p_4\}\rangle}$$

(R2) $$\frac{p_1 \supset p_2, p_3 \supset p_4, \{p_2\}S\{p_3\}}{\{p_1\}S\{p_4\}}$$

(R3) $$\frac{\langle D \mid \{p_1\}A\{p_2\}\rangle, \langle D \mid \{p_2\}A'\{p_3\}\rangle}{\langle D \mid \{p_1\}A;A'\{p_3\}\rangle}$$

(R4) $$\frac{\langle D \mid \{p \wedge b\}A\{q\}\rangle \quad \langle D \mid \{p \wedge \neg b\}A'\{q\}\rangle}{\langle D \mid \{p\}$ if b then A else A' fi $\{q\}\rangle}$$

(R5) $$\frac{\langle D|\{p_1\}A_1\{p_2\}\rangle,\ldots,\langle D|\{p_n\}A_n\{p_{n+1}\}}{\{p_1\}[L_i:A_i]_{i=1}^{n}\{p_{n+1}\}}$$

where $D \equiv [L_i:p_i]_{i=1}^{n}$.

DEFINITION (normal pair, normal correctness formula, normal fragment of H).

A pair <D,A> is called *normal* if all labels in A occur in D.

A correctness formula f is called *normal* if either f is an assertion, or $f \equiv \langle D|\{p\}A\{q\}\rangle$ and <D,A> is anormal pair, or $f \equiv \{p\}S\{q\}$ and S is a normal program (i.e. all labels in S are declared).

The *normal fragment* of the proof system H, denoted by H_N, is the system H restricted to normal formulae only.

DEFINITION (formal proof).

Let $f \in Corr$. A sequence $f_1,\ldots,f_n$ with $f_i \in Corr$ $(i = 1,\ldots,n)$ is called *a formal proof of* f *in* H if

a) $f \equiv f_n$

b) for all f_i with $1 \leq i \leq n$ the following holds:

either 1) f_i is (an instance of) an axiom

or 2) there exist $f_{i_1},\ldots,f_{i_k} \in Corr$ with $1 \leq i_j < i$ for $1 \leq j \leq k$, such that

$$\frac{f_{i_1},\ldots,f_{i_k}}{f_i}$$

is (an instance of) a rule of inference.

We say that f is *provable*, notation $\vdash f$, if there exists a formal proof of f.

The system defined above is dependent on the interpretation I of the primitive relation and function symbols, because the axioms of (A3) are determined by T, which function depends on I. We include all true assertions as axioms because we don't want to pay attention to deduction systems for the assertions only. We want to focus on the rules which can be used to prove properties of statements and programs. Also, in the proof of completeness of our system ("every valid formula is provable") we don't want to be hindered by deduction systems for the assertions which are possibly incomplete.

We now turn to the question of validity of correctness formulae (again with respect to an interpretation I). We use the notation $\models f$ to denote that f is valid. An informal definition of the concept has been given in the remarks preceding the definition of the deduction system. We will now formalize the ideas developed there. By now it must be clear that in the validity definition the semantic function A will be much easier in use than the function N (see the remarks preceding the definition of A at the end of chapter 4).

DEFINITON (validity).

Validity of a correctness formula f, notation $\models f$, is defined by

a) $\models p$ iff $\forall \sigma \in \Sigma_0$: $T[\![p]\!]\sigma = tt$

b) $\models \langle D | \{p\}A\{q\} \rangle$ iff

$$\forall \sigma \in \Sigma_0: T[\![p]\!]\sigma = tt \Rightarrow \left[\begin{array}{l} (\exists \sigma' \in \Sigma_0: A[\![A]\!]\sigma = \sigma' \wedge T[\![q]\!]\sigma' = tt) \vee \\ (\exists \sigma' \in \Sigma_0\ \exists (L:p')\ \underline{in}\ D: A[\![A]\!]\sigma = \langle \sigma', L \rangle \wedge T[\![p']\!]\sigma' = tt) \end{array}\right]$$

c) $\models \{p\}S\{q\}$ iff $\forall \gamma \in \Gamma\ \ \forall \sigma, \sigma' \in \Sigma_0$: $[(T[\![p]\!]\sigma = tt \wedge \sigma' = M[\![S]\!]\gamma\sigma) \Rightarrow T[\![q]\!]\sigma' = tt]$.

In words this amounts to the following. An assertion p is valid if it is true in all (defined) states. A formula {p}S{q} is valid, if evaluation of S with initial state σ satisfying p, either doesn't terminate or terminates in final state σ' for which q holds. The most complicated case is $f \equiv \langle D | \{p\}A\{q\} \rangle$. This f is valid if for every state σ satisfying p the following holds: if evaluation of A terminates normally in σ' then we want q to be true in σ'; if evaluation terminates by a jump to some L in state σ', we want this L to be an L_j in $D \equiv [L_i : p_i]_{i=1}^n$, and the corresponding assertion p_j must be true in σ'.

7. SOUNDNESS AND COMPLETENESS OF H_N

In this chapter we will show that the deduction system is sound ("$\vdash f \Rightarrow \models f$"), and complete ("$\models f \Rightarrow \vdash f$"). Now the definition of provability as well as that of validity shows that both notions are dependent on

the interpretation I chosen. In this chapter we will prove that $\vdash f \Rightarrow \models f$ holds for all correctness formulae. The converse is not true in general. Following COOK [6], we have to put a restriction on the interpretations allowed: only those interpretations are taken into account which make the class *Assn expressive with respect to the language Prog*. Only if *Assn* is expressive we can be assured that it is possible to find suitable label invariants $p_1,\ldots,p_n \in Assn$ for every program $S \equiv [L_i:A_i]_{i=1}^n$. The completeness theorem to be proved will then be that under every interpretation I such that *Assn* is expressive with respect to *Prog* we have that $\models f \Rightarrow \vdash f$ for every normal correctness formula f.

DEFINITION (validity of rules of inference).
A rule of inference

$$\frac{f_1,\ldots,f_n}{f_{n+1}}$$

is called *valid* if $(\models f_1,\ldots, \models f_n) \Rightarrow \models f_{n+1}$.

Note that validity of an inference rule again depends on the underlying interpretation I just like the validity of a correctness formula.

LEMMA 7.1. *Every axiom and every rule of inference in H is valid.*

PROOF.

(A1) We have to prove $\models \langle D | \{p[s/x]\}x:=s\{p\}\rangle$. We have $A[\![x:=s]\!]\sigma = \sigma\{V[\![s]\!]\sigma/x\}$ for all $\sigma \in \Sigma_0$. Furthermore $T[\![p[s/x]]\!]\sigma = tt$ implies $T[\![p]\!](\sigma\{V[\![s]\!]\sigma/x\}) = tt$ by lemma 6.1d. Thus we have that for all $\sigma \in \Sigma_0$ with $T[\![p[s/x]]\!]\sigma = tt$ there is a σ', namely $\sigma\{V[\![s]\!]\sigma/x\}$, such that $A[\![x:=s]\!]\sigma = \sigma'$ and $T[\![p]\!]\sigma' = tt$.

(A2) We have to prove $\models \langle D \mid \{p_j\}\underline{goto}\ L_j\{\underline{false}\}\rangle$ for $D \equiv [L_i:p_i]_{i=1}^n$. Choose $\sigma \in \Sigma_0$ such that $T[\![p_j]\!]\sigma = tt$. We have $A[\![\underline{goto}\ L_j]\!]\sigma = \langle\sigma, L_j\rangle$. Thus there is a σ', namely σ itself and a pair (L:p") in D, namely $(L_j:p_j)$, such that $A[\![\underline{goto}\ L_j]\!]\sigma = \langle\sigma', L\rangle$ and $T[\![p"]\!]\sigma' = tt$.

(A3) Evident.

(R1) Suppose $\models p_1 \supset p_2$, $\models p_3 \supset p_4$ and $\models \langle D|\{p_2\}A\{p_3\}\rangle$. We want to prove $\models \langle D|\{p_1\}A\{p_4\}\rangle$. Choose a $\sigma \in \Sigma_0$, and assume $T[\![p_1]\!]\sigma = tt$. From $p_1 \supset p_2$ we infer $T[\![p_2]\!]\sigma = tt$. The fact that $\models \langle D|\{p_2\}A\{p_3\}$ holds yields

either $\exists\sigma' \in \Sigma_0$: $A[\![A]\!]\sigma=\sigma' \wedge T[\![p_3]\!]\sigma'=tt$. But in this case we can use $\models p_3 \supset p_4$ to infer $\exists\sigma' \in \Sigma_0$: $A[\![A]\!]\sigma=\sigma' \wedge T[\![p_4]\!]\sigma'=tt$ ...(*)

or $\exists\sigma' \in \Sigma_0$: $\exists(L:p")$ in D: $A[\![A]\!]\sigma = <\sigma',L> \wedge T[\![p"]\!]\sigma'=tt$ ...(**)

But now we have proved $T[\![p_1]\!]\sigma=tt \Rightarrow$ (*) $\vee$ (**), and we conclude that $\models <D \mid \{p_1\}A\{p_4\}>$ holds.

(R2) Analogously.

(R3) Suppose $\models <D \mid \{p_1\}A\{p_2\}>$ and $\models <D \mid \{p_2\}A'\{p_3\}>$. We have to prove $\models <D \mid \{p_1\}A;A'\{p_3\}>$. Choose a $\sigma \in \Sigma_0$ such that $T[\![p_1]\!]\sigma = tt$. From $\models <D \mid \{p_1\}A\{p_2\}>$ we infer that

either $(A[\![A]\!]\sigma=\sigma' \wedge T[\![p_2]\!]\sigma'=tt)$ for some $\sigma' \in \Sigma_0$ ...(1)

or $(A[\![A]\!]\sigma=<\sigma',L> \wedge T[\![p"]\!]\sigma'=tt)$ for some $\sigma' \in \Sigma_0$, $(L:p")$ in D ...(2)

ad (1). $\models <D \mid \{p_2\}A'\{p_3\}>$ and $T[\![p_2]\!]\sigma'=tt$ for some $\sigma' \in \Sigma_0$ give us:

either $(A[\![A']\!]\sigma'=\sigma" \wedge T[\![p_3]\!]\sigma"=tt)$ for some $\sigma" \in \Sigma_0$. From $A[\![A]\!]\sigma=\sigma'$ and $A[\![A']\!]\sigma'=\sigma"$ we infer $A[\![A;A']\!]\sigma=\sigma"$. Furthermore we have $T[\![p_3]\!]\sigma"=tt$,

or $(A[\![A']\!]\sigma'= <\sigma",L> \wedge T[\![p"]\!]\sigma"=tt)$ for some $\sigma" \in \Sigma_0$ and some pair $(L:p")$ in D. But then $A[\![A]\!]\sigma=\sigma'$ and $A[\![A']\!]\sigma' = <\sigma",L>$ give us $A[\![A;A']\!]\sigma = <\sigma",L>$ and we have also $T[\![p"]\!]\sigma"=tt$.

ad (2). From $A[\![A]\!]\sigma = <\sigma',L>$ we have $A[\![A;A']\!]\sigma = <\sigma',L>$. Furthermore we have $T[\![p"]\!]\sigma'=tt$.

The conclusion is that for every choice of σ the conditions imposed by the definition of $\models <D \mid \{p_1\}A;A'\{p_3\}>$ are satisfied.

(R4) can be proved analogously, using results like

$(A[\![A]\!]\sigma=\sigma' \wedge W[\![b]\!]\sigma=tt) \Rightarrow A[\![$ if b then A else A' fi$]\!]\sigma = \sigma'$.

(R5) Suppose $\models <D \mid \{p_1\}A_i\{p_{i+1}\}>$ $(i = 1,\ldots,n)$, where $D \equiv [L_i:p_i]_{i=1}^n$. We have to prove $\models \{p_1\}[L_i:A_i]_{i=1}^n\{p_{n+1}\}$, or equivalently

$\forall\gamma \in \Gamma \quad \forall\sigma,\sigma' \in \Sigma_0[(T[\![p_1]\!]\sigma=tt \wedge M[\![[L_i:A_i]_{i=1}^n]\!]\gamma\sigma=\sigma') \Rightarrow T[\![p_{n+1}]\!]\sigma'=tt]$.

So, choose $\gamma \in \Gamma$, and let ϕ_i, $\phi_i^{(k)}$ and $\gamma^{(k)}$ be derived from $[L_i:A_i]_{i=1}^n$ and γ as in Lemma 4.2. We now prove the following lemma.

LEMMA. $\forall k \in \mathbb{N}[\forall\sigma,\sigma' \in \Sigma: (T[\![p_i]\!]\sigma=tt \wedge \sigma'=\phi_i^{(k)}\sigma) \Rightarrow T[\![p_{n+1}]\!]\sigma'=tt$, *for* $i = 1,\ldots,n+1]$.

PROOF (induction on k).

Basis (k=0). This is easy, because (i) $\phi_i^{(0)} = \lambda\sigma\cdot\perp$ for $i = 1,\ldots,n$ and therefore there is no $\sigma' \in \Sigma_0$ such that $\sigma' = \phi_i^{(0)}\sigma$; (ii) $\phi_{n+1}^{(i)} = \lambda\sigma\cdot\sigma$,

but then the assumption reduces to $T[\![p_{n+1}]\!]\sigma=tt \wedge \sigma'=(\lambda\sigma\cdot\sigma)\sigma=\sigma$, and thus the conclusion $T[\![p_{n+1}]\!]\sigma' = tt$ holds.

Induction step. Choose an i ($1 \leq i \leq n$; the case $i = n+1$ is again trivial) and choose $\sigma,\sigma' \in \Sigma_0$ such that $T[\![p_i]\!]\sigma=tt$ and $\sigma' = \phi_i^{(k+1)}\sigma$. Now $\phi_i^{(k+1)}\sigma = M[\![A_i]\!]\gamma^{(k)}\phi_i^{(k)}\sigma$. From $\models$ <D | $\{p_i\}A_i\{p_{i+1}\}$> we know that

either $A[\![A_i]\!]\sigma=\sigma''$ and $T[\![p_{i+1}]\!]\sigma''=tt$, for some $\sigma'' \in \Sigma_0$. But then we have $\sigma'=\phi_i^{(k+1)}\sigma = \phi_{i+1}^{(k)}\sigma''$ (using 4.3.1°). Induction hypothesis, and $T[\![p_{i+1}]\!]\sigma''=tt$ yield $T[\![p_{n+1}]\!]\sigma'=tt$,

or $A[\![A_i]\!]\sigma = <\sigma'',L_j>$ and $T[\![p_j]\!]\sigma''=tt$, for some $\sigma'' \in \Sigma_0$ and some j ($1 \leq j \leq n$). Now $\sigma' = \phi_i^{(k+1)}\sigma = M[\![A_i]\!]\gamma^{(k)}\phi_{i+1}^{(k)}\sigma = \gamma^{(k)}[\![L_j]\!]\sigma''=\phi_j^{(k)}\sigma''$ (using 4.3.2°). Induction hypothesis and $T[\![p_j]\!]\sigma''=tt$ yield $T[\![p_{n+1}]\!]\sigma'=tt$.

This proves the lemma. □

Now, returning to the proof of $\models \{p_1\}[L_i:A_i]_{i=1}^n\{p_{n+1}\}$, we have by definition that $M[\![[L_i:A_i]_{i=1}^n]\!]\gamma = M[\![A_1;\ldots;A_n]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\{\lambda\sigma\cdot\sigma\} = \phi_1$ by Lemma 4.2.2. And $\phi_1 = \bigsqcup_k \phi_1^{(k)}$, by Lemma 4.2.1.

Choose $\sigma,\sigma' \in \Sigma_0$ such that $T[\![p_1]\!]\sigma=tt$ and $\sigma' = \phi_1\sigma$. Then (because $\phi_1\sigma = \bigsqcup_k (\phi_1^{(k)}\sigma)$) there is a $\bar{k}$ such that $\sigma' = \phi_1^{(\bar{k})}\sigma$, and the lemma gives us $T[\![p_{n+1}]\!]\sigma'=tt$.

This proves $\models \{p_1\}[L_i:A_i]_{i=1}^n\{p_{n+1}\}$, which was the last clause in the proof of 7.1. □

THEOREM 7.2. *The proof system H is sound, i.e. for every interpretation I and every correctness formula* f *we have* $\vdash f \Rightarrow \models f$.

PROOF. Induction on the length of the proof of f, using Lemma 7.1. □

We now turn our attention to the question of completeness of the proof system, i.e., $\models f \Rightarrow \vdash f$. If f is an assertion p, then we can simply use axiom scheme (A3), so there is no problem here.

The next possibility is f ≡ <D | $\{p\}A\{q\}$>. Suppose this f is valid. Now we have to construct a formal proof of this formula. This will be done using the concept of *weakest precondition:* we will show that for all $D \in \mathit{Invl}$, $A \in \mathit{Stat}$ and $p \in \mathit{Assn}$ (such that <D,A> is normal), we can construct

a q ∈ *Assn* which is the weakest formula that makes <D | {q}A{p}> valid (by "weakest" we mean true in as many states as possible). This is part of Lemma 7.4.

In the same lemma we show that for this assertion q the formula <D | {q}A{p}> is provable. Once we have reached this result, the rest is easy. We use the property that, if q expresses the weakest precondition of A with respect to p and D, and if ⊨ <D | {p'}A{p}> for some p' ∈ *Assn*, then we have ⊨ p' ⊃ q (otherwise the precondition q wouldn't be the weakest one). Thus in this case we can derive <D | {p'}A{p}> using (R1).

DEFINITION (weakest precondition).
Let A ∈ *Stat*, p ∈ *Assn*, D ∈ *Invl*. We say that q *expresses the weakest precondition of* A *with respect to postcondition* p *and invariant list* D iff

$$\forall \sigma \in \Sigma_0 : T[\![q]\!]\sigma = tt \iff \left[\begin{array}{l} (\exists \sigma' \in \Sigma_0 : [A[\![A]\!]\sigma = \sigma' \wedge T[\![p]\!]\sigma' = tt]) \vee \\ (\exists \sigma' \in \Sigma_0\ \exists (L:p')\ \underline{\text{in}}\ D : [A[\![A]\!]\sigma = \langle \sigma', L\rangle \wedge T[\![p']\!]\sigma' = tt]) \end{array}\right].$$

We write p ≃ wp⟦A,p,D⟧ to express this.

LEMMA 7.3. *Let* A ∈ *Stat*, p,q ∈ *Assn*, D ∈ *Invl*. *If* q ≃ wp⟦A,p,D⟧, *then*
a) ⊨ <D | {q}A{p}> (*i.e., p is a precondition*)
b) ∀p' ∈ *Assn*: [⊨ <D | {p'}A{p}> ⇒ ⊨ p' ⊃ q] (q *is the weakest*).

PROOF. Immediate from the definitions. □

LEMMA 7.4. *For all* A ∈ *Stat*, p ∈ *Assn*, D ∈ *Invl such that* <D,A> *is normal, we can find* q ∈ *Assn for which* q ≃ wp⟦A,p,D⟧. *Moreover for this* q *we also have* ⊢ <D | {q}A{p}> *in* H_N.

PROOF. By induction on the structure of A. We distinguish four cases.
1°. A ≡ x:=s. Choose p ∈ *Assn* and D ∈ *Invl*. Then p[s/x] ≃ wp⟦x:=s,p,D⟧. For, choose σ ∈ Σ_0. We have to show

$$T[\![p[s/x]]\!]\sigma = tt \iff (\exists \sigma' \in \Sigma_0 [(A[\![x:=s]\!]\sigma = \sigma') \wedge T[\![p]\!]\sigma' = tt]) \vee (\exists \sigma' \in \Sigma_0\ \exists (L:p'')\ \underline{\text{in}}\ D[(A[\![A]\!]\sigma = \langle \sigma, L\rangle) \wedge T[\![p'']\!]\sigma' = tt]).$$

Now $A[\![x:=s]\!]\sigma = \sigma\{V[\![s]\!]\sigma/x\} \in \Sigma_0$, so the above equivalence comes down to

$$T[\![p[s/x]]\!]\sigma = tt \iff \exists \sigma' \in \Sigma_0 [(A[\![x:=s]\!]\sigma = \sigma') \wedge T[\![p]\!]\sigma' = tt] \iff T[\![p]\!](\sigma\{V[\![s]\!]\sigma/x\}) = tt,$$

and this is 6.2d.

Furthermore, we have $\vdash$ <D | {p[s/x]}x:=s{p}> by (A1).

2°. A ≡ $A_1;A_2$. Choose p ∈ *Assn* and D ∈ *Invl* such that <D,A> is normal. By induction there is a q' ∈ *Assn* with q' ≃ wp⟦A_2,p,D⟧ and $\vdash$ <D | {q'}A_2{p}> in H_N. Again by induction we have q ∈ *Assn* such that q ≃ wp⟦A_1,q';D⟧ and $\vdash$ <D | {q}A_1{q'}> in H_N.

First, we will show that for this q we have q ≃ wp⟦$A_1;A_2$,p,D⟧. Choose a σ ∈ Σ_0. We have to prove

$T⟦q⟧\sigma=tt \iff (\exists\sigma'\in\Sigma_0[A⟦A_1;A_2⟧\sigma=\sigma' \wedge T⟦p⟧\sigma'=tt]) \vee$

$(\exists\sigma'\in\Sigma_0\ \exists(L:p')\ \underline{in}\ D[A⟦A_1;A_2⟧\sigma = <\sigma',L> \wedge T⟦p'⟧\sigma'=tt])$.

We distinguish two cases:

a) $A⟦A_1⟧\sigma=\sigma''$. We then have $A⟦A_1;A_2⟧\sigma = A⟦A_2⟧\sigma''$ by definition of A. Using these facts the above equivalence reduces to

$T⟦q⟧\sigma=tt \iff (\exists\sigma'\in\Sigma_0[A⟦A_2⟧\sigma''=\sigma' \wedge T⟦p⟧\sigma'=tt]) \vee$

$(\exists\sigma'\in\Sigma_0\ \exists(L:p')\ \underline{in}\ D[A⟦A_2⟧\sigma'' = <\sigma',L> \wedge T⟦p'⟧\sigma'=tt])$,

and by q' ≃ wp⟦A_2,p,D⟧ this is equivalent to $T⟦q⟧\sigma=tt \iff T⟦q'⟧\sigma''=tt$.

Now we have by q ≃ wp⟦A_1,q',D⟧

$T⟦q⟧\sigma=tt \iff (\exists\sigma'\in\Sigma_0[A⟦A_1⟧\sigma=\sigma' \wedge T⟦q'⟧\sigma'=tt]) \vee$

$(\exists\sigma'\in\Sigma_0\ \exists(L:p')\ \underline{in}\ D[A⟦A_1⟧\sigma = <\sigma',L> \wedge T⟦p'⟧\sigma'=tt])$.

Substituting σ'' for $A⟦A_1⟧\sigma$ (that is the assumption) the right-hand side of the equivalence reduces to $T⟦q'⟧\sigma''=tt$, and we are ready.

b) $A⟦A_1⟧\sigma = <\sigma'',L''>$. We then also have that $A⟦A_1;A_2⟧\sigma = <\sigma'',L''>$ and the definition of q ≃ wp⟦$A_1;A_2$,p,D⟧ reduces to

$T⟦q⟧\sigma=tt \iff \exists p''\in Assn[(L'':p'')\ \underline{in}\ D \wedge T⟦p''⟧\sigma''=tt]$.

But this equivalence is immediate from q ≃ wp⟦A_1,q',D⟧ and $A⟦A_1⟧\sigma = <\sigma'',L''>$. So, we have proved that q ≃ wp⟦$A_1;A_2$,p,D⟧.

The proof that <D | {q}$A_1;A_2${p}> can be derived in H_N is easy by the assumptions on q and q' (using rule (R3) of composition), and the fact that the pair <D,$A_1;A_2$> is normal (which means that <D | {p_1}A_1{p_2}> and <D | {p_2}A_2{p_3}> are normal formulae).

3°. A ≡ *if* b *then* A_1 *else* A_2 *fi*. Choose p ∈ *Assn* and D ∈ *Invl*, such that <D,A> is normal. By induction we have q_1,q_2 ∈ *Assn* such that

q_1 ≃ wp⟦A_1,p,D⟧ and $\vdash$ <D | {q_1}A_1{p}> in H_N

q_2 ≃ wp⟦A_2,p,D⟧ and $\vdash$ <D | {q_2}A_2{p}> in H_N.

We will show that for q ≡ *if* b *then* q_1 *else* q_2 *fi* we have

a) $q \simeq wp[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}, p, D]\!]$

b) $\langle D \mid \{q\}\ \underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}\ \{p\}\rangle$ in H_N.

a) Choose a $\sigma \in \Sigma_0$. Without loss of generality $W[\![b]\!]\sigma = tt$. Then
$T[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ q_1\ \underline{\text{else}}\ q_2\ \underline{\text{fi}}]\!]\sigma = tt \iff T[\![q_1]\!]\sigma = tt$.
Also $A[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}]\!]\sigma = A[\![A_1]\!]\sigma$. Now $q_1 \simeq wp[\![A_1, p, D]\!]$ is equivalent to

$$T[\![q_1]\!]\sigma = tt \iff (\exists \sigma' \in \Sigma_0\ [A[\![A_1]\!]\sigma = \sigma' \wedge T[\![p]\!]\sigma' = tt]) \vee (\exists \sigma' \in \Sigma_0\ \exists (L{:}p')\ \underline{\text{in}}\ D\ [A[\![A_1]\!]\sigma = \langle \sigma', L\rangle \wedge T[\![p']\!]\sigma' = tt]).$$

Combining these results, we get

$$T[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ q_1\ \underline{\text{else}}\ q_2\ \underline{\text{fi}}]\!]\sigma = tt \iff (\exists \sigma' \in \Sigma_0 [A[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}]\!]\sigma = \sigma' \wedge T[\![p]\!]\sigma' = tt]) \vee (\exists \sigma' \in \Sigma_0\ \exists (L{:}p')\ \underline{\text{in}}\ D[A[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}]\!]\sigma = \langle \sigma', L\rangle \wedge T[\![p']\!]\sigma' = tt])$$

and this is the result we were aiming at.

b) We have $q \wedge b \equiv (\underline{\text{if}}\ b\ \underline{\text{then}}\ q_1\ \underline{\text{else}}\ q_2\ \underline{\text{fi}}) \wedge b$, and thus $\models q \wedge b \supset q_1$. Also, using (A3) and (R1), and $\vdash \langle D \mid \{q_1\}A\{p\}\rangle$ (in H_N) we get $\vdash \langle D \mid \{q \wedge b\}A_1\{p\}\rangle$ in H_N. Analogously $\vdash \langle D \mid \{q \wedge \neg b\}A_2\{p\}\rangle$ in H_N. So, by inference rule (R4): $\vdash \langle D \mid \{q\}\ \underline{\text{if}}\ b\ \underline{\text{then}}\ A_1\ \underline{\text{else}}\ A_2\ \underline{\text{fi}}\ \{p\}\rangle$ in H_N.

4°. $A \equiv \underline{\text{goto}}\ L$. Choose $p \in Assn$ and $D \in Invl$, such that $\langle D, A\rangle$ is normal. This means that we have a $q \in Assn$ such that $(L{:}q)\ \underline{\text{in}}\ D$. We prove $q \simeq wp[\![\underline{\text{goto}}\ L, p, D]\!]$. We have to prove

$$T[\![q]\!]\sigma = tt \iff (\exists \sigma' \in \Sigma_0 [A[\![\underline{\text{goto}}\ L]\!]\sigma = \sigma' \wedge T[\![p]\!]\sigma' = tt]) \vee (\exists \sigma' \in \Sigma_0\ \exists (L'{:}p')\ \underline{\text{in}}\ D[A[\![\underline{\text{goto}}\ L]\!]\sigma = \langle \sigma', L'\rangle \wedge T[\![p']\!]\sigma' = tt]).$$

Because $A[\![\underline{\text{goto}}\ L]\!]\sigma = \langle \sigma, L\rangle \in \Sigma_0 \times Lvar$, the equivalence reduces to
$T[\![q]\!]\sigma = tt \iff \exists p' \in Assn[(L{:}p')\ \underline{\text{in}}\ D \wedge T[\![p']\!]\sigma = tt]$.
Now we have $(L{:}q)\ \underline{\text{in}}\ D$ and thus the right-hand side is equivalent to $T[\![q]\!]\sigma = tt$.
Furthermore, in order to show that $\vdash \langle D \mid \{q\}\ \underline{\text{goto}}\ L\ \{p\}\rangle$ in H_N, we have by (A2) $\vdash \langle D \mid \{q\}\ \underline{\text{goto}}\ L\ \{\underline{\text{false}}\}\rangle$ and by (A3) $\vdash \underline{\text{false}} \supset p$. So we can use (R1) to derive $\vdash \langle D \mid \{q\}\ \underline{\text{goto}}\ L\ \{p\}\rangle$ in H_N.

This completes the proof of Lemma 7.4. □

Observe that in this lemma it is not merely proved that there exists a formula q expressing the weakest precondition for any A, p and D, such that $\langle D, A\rangle$ is normal. The proof also provides a purely syntactical method

to derive such a formula. This shows that for every A, p and D with the above restriction, we can *construct* an assertion q expressing the weakest precondition. Thus this q is independent of the interpretation I of the primitive relation and function symbols.

Note also that there are many assertions expressing the weakest precondition of A with respect to p and D. For instance, if $q \simeq wp[\![A,p,D]\!]$ then the same holds for $q \wedge \underline{true}$ (to give a trivial example).

We now state and prove the completeness result for correctness formulae having the form $<D \mid \{p\}A\{q\}>$.

LEMMA 7.5. *For all* $A \in Stat$, $p,q \in Assn$ *and* $D \in Invl$ *such that* $<D,A>$ *is normal we have*

$$\models <D \mid \{p\}A\{q\}> \Rightarrow \vdash <D \mid \{p\}A\{q\}> \text{ in } H_N.$$

PROOF. Choose A,p,q and D such that the assumptions are true. Suppose also $\models <D \mid \{p\}A\{q\}>$. Now by Lemma 7.4 there is a $p' \simeq wp[\![A,q,D]\!]$ for which $\vdash <D \mid \{p'\}A\{q\}>$ in H_N. By Lemma 7.3 and $\models <D \mid \{p\}A\{q\}>$ we have $\models p \supset p'$. So, using (A3) and (R1) we get $\vdash <D \mid \{p\}A\{q\}>$ in H_N. □

Note that up till now no claims have been made on the interpretation I. The only additional condition was that $<D,A>$ should be normal. It can be seen that this is necessary from the fact that

$$\models <L{:}p \mid \{\underline{true}\}\ \underline{if}\ \underline{true}\ \underline{then}\ x{:=}0\ \underline{else}\ \underline{goto}\ L'\ \underline{fi}\ \{x=0\}>$$

holds, even if $L \neq L'$. However, there is no way to derive this correctness formula in H.

We now turn our attention to the discussion of completeness with respect to correctness formulae of the form $\{p\}S\{q\}$. If we want to formally prove such a formula, we have to find suitable label invariants for all labels declared in S. It is at this point that we have to put the restriction on I which we mentioned in the introduction of this chapter.

The question arises whether in our case such a restriction is necessary. WAND [12] proved that such a restriction is needed for programs without goto statements, but containing while statements. He constructed an interpretation I and a correctness formula which was valid under I but not derivable,

because there was no assertion available which could express a suitable invariant. His counterexample can be transferred to our language in such a way that the same arguments he uses can be applied in our situation. Therefore we must make a restriction on $\mathcal{I}$.

Before we can give an exact definition of this restriction (*expressiveness*), we have to make a few preparations.

LEMMA 7.6. *Let* $S \equiv [L_i:A_i]_{i=1}^n \in \mathcal{Prog}$ *and* $\gamma \in \Gamma$. *Let* ϕ_i *be derived from* S *and* γ *as in the definition of* M. *If* S *is normal, then all* ϕ_i *are independent of* γ.

PROOF. The following holds: Let $A \in \mathcal{Stat}$ and $\{L_1,\ldots,L_n\}$ be the set of all labels occurring in A. Let $\phi,\psi_1,\ldots,\psi_n \in M$ and $\gamma \in \Gamma$. Then $N[\![A]\!](\gamma\{\psi_i/L_i\}_{i=1}^n)\phi$ is independent of γ. This fact can be proved by induction on the structure of A.

From this result we can infer that the operator

$$\Phi = \lambda\langle\psi_1,\ldots,\psi_n\rangle.\langle N[\![A_i]\!](\gamma\{\psi_j/L_j\}_{j=1}^n)\psi_{i+1}\rangle_{j=1}^n$$

is independent of γ, and thus the same must be true of $\langle\phi_1,\ldots,\phi_n\rangle$, being the least fixed point $\mu\Phi$ of Φ. □

DEFINITION (transformations derived from S).
Let S be a normal program. Then the ϕ_i defined as in Lemma 7.6, are called the *transformations derived from* S.

DEFINITION (weakest precondition of a transformation from M).
Let $\phi \in M$, $p \in \mathcal{Assn}$. We say that q *expresses the weakest precondition of* ϕ *with respect to* p iff $\forall\sigma\in\Sigma_0: [T[\![q]\!]\sigma=tt \iff (\phi\sigma\neq\perp \Rightarrow T[\![p]\!](\phi\sigma)=tt)]$.

DEFINITION (expressiveness).
Let $\mathcal{I}$ be an interpretation of the primitive relation and function symbols. We say that $\mathcal{Assn}$ *is expressive relative to* $\mathcal{Prog}$ *and* $\mathcal{I}$, iff for all assertions p and for all normal programs S the following holds: there are assertions $p_1,\ldots,p_n$ such that p_i expresses the weakest precondition of ϕ_i with respect to p, where ϕ_i are the transformations derived from S $(i = 1,\ldots,n)$.

If the primitive relation and function symbols of our language *Prog* are such that *Assn* is a language for Peano arithmetic, and if I_0 is the standard interpretation of this language in the natural numbers then, using recursion theory, we can show that the transformations ϕ_i derived from a normal program S are partial recursive functions in the free variables of S. A result of recursion theory is that for every partial recursive function ϕ: $\mathbb{N}^k \to \mathbb{N}^k$, there is a formula p in *Assn* with free variables $x_1,\ldots,x_k$, $y_1,\ldots,y_k$, which expresses this function, i.e. $\models p(\bar{\alpha}_1,\ldots,\bar{\alpha}_k,\bar{\beta}_1,\ldots,\bar{\beta}_k)$ iff $\phi(\alpha_1,\ldots,\alpha_k)$ is defined and equal to $<\beta_1,\ldots,\beta_k>$ (where $\bar{\alpha}_i,\bar{\beta}_i$ are numerals denoting the natural numbers α_i,β_i). From this we can infer that *Assn* is expressive relative to *Prog* and the standard interpretation I_0.

Now we have enough tools to state the main lemma needed to prove completeness for formulae of the form {p}S{q}. In essence this theorem states that the p_i from the definition of expressiveness are the label invariants which we are looking for.

LEMMA 7.7. *Let I be an interpretation such that Assn is expressive relative to Prog and I. Let* $S \equiv [L_i:A_i]_{i=1}^n \in Prog$ *be normal. Let* $p \in Assn$, *and let* ϕ_i *be the transformations derived from* S (i = 1,...,n). *Let* p_i *be the assertions expressing the weakest preconditions of* ϕ_i *with respect to* p (i = 1,...,n), *and let* $p_{n+1} \equiv p$. *Then*

$$\models <[L_i:p_i]_{i=1}^n \mid \{p_j\}A_j\{p_{j+1}\}>$$

for j = 1,...,n.

PROOF. Choose j ($1 \le j \le n$) and $\sigma \in \Sigma_0$ such that $T[\![p_j]\!]\sigma$=tt. There are two cases:

a) $A[\![A_j]\!]\sigma = \sigma' \in \Sigma_0$. According to the definition of $\models <D \mid \{p_j\}A_j\{p_{j+1}\}>$, in this case we have to prove that $T[\![p_{j+1}]\!]\sigma'$=tt. Now by the assumption on p_j and by definition of weakest precondition we have $T[\![p_j]\!]\sigma$=tt $\iff$ ($\phi_j\sigma \neq \perp \Rightarrow T[\![p]\!](\phi_j\sigma)$=tt). Also, $\phi_j\sigma = \phi_{j+1}\sigma'$ (by 4.3.1°, $A[\![A_j]\!]\sigma=\sigma'$ and $\phi_j\sigma = N[\![A_j]\!](\gamma\{\phi_i/L_i\}_{i=1}^n)\phi_{j+1}\sigma$). Combining these results, we get $T[\![p_j]\!]\sigma$=tt $\iff$ ($\phi_{j+1}\sigma' \neq \perp \Rightarrow T[\![p]\!](\phi_{j+1}\sigma')$=tt). But the right-hand side of this equivalence is (by definition of weakest precondition, and

by the assumption on p_{j+1}) equivalent to $T[\![p_{j+1}]\!]\sigma'=tt$.

b) $A[\![A_j]\!]\sigma = \langle\sigma',L\rangle \in \Sigma_0 \times Lvar$. From the fact that S is normal, we infer that L must be some L_k $(1 \leq k \leq n)$. We have to prove (by definition of $\models \langle D \mid \{p\}A\{q\}\rangle$) that $T[\![p_k]\!]\sigma'=tt$. Again we have:
$T[\![p_j]\!]\sigma=tt \iff (\phi_j\sigma\neq\perp \Rightarrow T[\![p]\!](\phi_j\sigma)=tt)$, and now we have $\phi_j\sigma = \phi_k\sigma'$ by 4.3.2° (analogously to a)). Combining the results, we get
$T[\![p_j]\!]\sigma=tt \iff (\phi_k\sigma'\neq\perp \Rightarrow T[\![p]\!](\phi_k\sigma')=tt)$
$\iff T[\![p_k]\!]\sigma'=tt.$ □

We now can collect our results in the completeness theorem 7.8.

THEOREM 7.8. *The deduction system H_N is complete in the sense of Cook, i.e., for every interpretation I such that Assn is expressive relative to Prog and I, and for every normal correctness formula f, we have $\models f \Rightarrow \vdash f$ in H_N.*

PROOF.

a) If $f \equiv p \in Assn$, then $\models p \Rightarrow \vdash p$ by (A3)

b) If $f \equiv \langle D \mid \{p\}A\{q\}\rangle$ then we can apply Lemma 7.5.

c) $f \equiv \{p\}S\{q\}$. Say $S \equiv [L_i:A_i]_{i=1}^n$. Let ϕ_i be the transformations derived from S. By Lemma 7.7 there are assertions p_j, expressing the weakest preconditions of ϕ_j with respect to q such that

$$\models \langle [L_i:p_i]_{i=1}^n \mid \{p_j\}A_j\{p_{j+1}\}\rangle$$

for $j = 1,\ldots,n$, where $p_{n+1} \equiv p$.
Note that these correctness formulae are normal by the fact that $\{p\}S\{q\}$ and thus S is normal. Lemma 7.5 then gives us

$$\vdash \langle [L_i:p_i]_{i=1}^n \mid \{p_j\}A_j\{p_{j+1}\}\rangle \text{ in } H_N$$

for $j = 1,\ldots,n$. Now we can apply rule (R5) to get

$$\vdash \{p_1\}S\{p_{n+1}\} \text{ in } H_N.$$

Now $p_{n+1} \equiv q$. Moreover $\models p \supset p_1$. For, assume $T[\![p]\!]\sigma=tt$ for some $\sigma \in \Sigma_0$. Then, by $\models \{p\}S\{q\}$, we have $\forall\gamma \in \Gamma \ \forall\sigma' \in \Sigma_0$: $\sigma' = M[\![S]\!]\gamma\sigma \Rightarrow T[\![q]\!]\sigma'=tt$. But $M[\![S]\!]\gamma\sigma = \phi_1\sigma$ (Lemma 4.2.2). Thus: $\sigma' = \phi_1\sigma \in \Sigma_0 \Rightarrow T[\![q]\!]\sigma'=tt$. But this is equivalent to $T[\![p_1]\!] =tt$, using the definition of weakest precondition.

We had $\vdash \{p_1\}S\{q\}$. Also $\models p \supset p_1$, and thus $\vdash p \supset p_1$ by (A3). Finally, using (R2), we conclude $\vdash \{p\}S\{q\}$ in H_N. □

8. DEDUCTION SYSTEM: SECOND VARIANT

The validity definition of Chapter 6 makes explicit use of the label invariants p_i, which therefore had to be provided by the formulae of the deduction system. The purpose of this chapter is to show that it is possible to define validity in such a way that the label invariants are not explicitly needed. We will, using continuation semantics, associate a truth value with a formula $\{p\}A\{q\}$. This truth value will be dependent on the meaning of the labels occurring within A, i.e. the value depends on the environment γ. Consequently, we will establish a semantical function G such that for every A, p and q we have $G[\![\{p\}A\{q\}]\!]: \Gamma \to \{ff,tt\}$.

This leads to a definition of validity which turns out to be equivalent to the one of Chapter 6 in the following sense: $\langle [L_i:p_i]_{i=1}^n \mid \{p\}A\{q\}\rangle$ is valid according to the definition in Chapter 6, if and only if $\{p\}A\{q\}$ is valid (using function G) in every environment γ for which all formulae $\{p_i\}\underline{\text{goto}}\ L_i\ \{\underline{\text{false}}\}$ are valid (i = 1,...,n). Or more formally,

$$\models \langle [L_i:p_i]_{i=1}^n \mid \{p\}A\{q\}\rangle \iff$$

$$\forall\gamma \in \Gamma[\bigwedge_{i=1}^n (G[\![\{p_i\}\ \underline{\text{goto}}\ L_i\ \{\underline{\text{false}}\}]\!]\gamma=tt) \Rightarrow G[\![\{p\}A\{q\}]\!]\gamma=tt].$$

Using this new approach we can define validity for the system as given in [5]. But, before we do that, we change this system somewhat. The system in [5] is presented as a natural deduction system, which means that the notion "proof from assumptions" is used. A line in a formal proof can be a formula which is introduced as an assumption. The system also has an inference rule in which assumptions are discharged, namely

$$\frac{\{p'\}\ \underline{\text{goto}}\ L\ \{\underline{\text{false}}\} \vdash \{p\}A_1\{p'\} \qquad \{p'\}\ \underline{\text{goto}}\ L\ \{\underline{\text{false}}\} \vdash \{p'\}A_2\{q\}}{\{p\}A_1;L:A_2\{q\}}$$

which discharges the assumption $\{p'\}\ \underline{\text{goto}}\ L\ \{\underline{\text{false}}\}$, needed in the derivation of $\{p\}A_1\{p'\}$ and $\{p'\}A_2\{q\}$. Thus, every derived formula f in the system of [5] will have a finite set Δ of assumptions attached by it, namely those assumptions which were used to derive f.

We transform this natural deduction system into a sequent calculus having formulae of the form

$$\Delta \to \{p\}A\{q\},$$

where Δ is meant to be the finite set of assumptions associated with the derivation of $\{p\}A\{q\}$. The advantage of this system over the natural deduction system is that validity of a formula can be defined more directly, now that every formula incorporates the relevant assumptions.

We now define the deduction system

DEFINITION (atomic correctness formula, correctness formula).
An *atomic correctness formula* is a formula of the form $\{p\}A\{q\}$. The class of all atomic correctness formulae will be denoted by $Afor$, and has g as a typical element.

A *correctness formula* is either an assertion, or a formula of the form $\Delta \to \{p\}A\{q\}$, or a formula of the form $\Delta \to \{p\}S\{q\}$, where Δ (the set of *assumptions*) is a finite set of atomic correctness formulae. The class of all correctness formulae will be denoted by $Corr$, and has f as a typical element.

The correctness formulae $\emptyset \to \{p\}A\{q\}$ and $\emptyset \to \{p\}S\{q\}$ will be abbreviated to $\{p\}A\{q\}$ and $\{p\}S\{q\}$ respectively.

DEFINITION (deduction system H'). The axioms are

(A1) $\Delta \to \{p[s/x]\}x:=s\{p\}$

(A2) $\Delta \to g$, where $g \in \Delta$

(A3) p,

where p is a valid assertion (i.e. $\forall \sigma \in \Sigma_0$: $T[\![p]\!]\sigma = tt$).

The rules of inference are

(R1) $$\frac{p_1 \supset p_2,\ p_3 \supset p_4,\ \Delta \to \{p_2\}A\{p_3\}}{\Delta \to \{p_1\}A\{p_4\}}$$

(R2) $$\frac{p_1 \supset p_2,\ p_3 \supset p_4,\ \Delta \to \{p_2\}S\{p_3\}}{\Delta \to \{p_1\}S\{p_4\}}$$

(R3) $$\frac{\Delta \to \{p_1\}A\{p_2\},\ \Delta \to \{p_2\}A'\{p_3\}}{\Delta \to \{p_1\}A;A'\{p_3\}}$$

(R4) $$\frac{\Delta \to \{p \wedge b\}A\{q\},\ \Delta \to \{p \wedge \neg b\}A'\{q\}}{\Delta \to \{p\}\ \underline{\text{if}}\ b\ \underline{\text{then}}\ A\ \underline{\text{else}}\ A'\ \underline{\text{fi}}\ \{q\}}$$

(R5) $$\frac{\Delta \cup \Delta' \to \{p_1\}A_1\{p_2\},\ldots,\Delta \cup \Delta' \to \{p_n\}A_n\{p_{n+1}\}}{\Delta' \to \{p_1\}[L_i:A_i]_{i=1}^{n}\{p_{n+1}\}}$$

where $\Delta = \{\ \{p_i\}\ \underline{\text{goto}}\ L_i\ \{\underline{\text{false}}\}\ \mid\ i = 1,\ldots,n\}$,
all L_i are different, and no L_i occurs in any assumption
in Δ' $(i = 1,\ldots,n)$.

The restriction on Δ' in (R5) is imposed to circumvent possibilities like the following. Suppose $\Delta = \{\ \{\underline{\text{true}}\}\ \underline{\text{goto}}\ L_1\ \{\underline{\text{false}}\},\{x=0\}\ \underline{\text{goto}}\ L_2\ \{\underline{\text{false}}\}\ \}$ and Δ' is the singleton $\{\ \{\underline{\text{true}}\}\ \underline{\text{goto}}\ L_2\ \{\underline{\text{false}}\}\ \}$. Then we can derive

$$\Delta \cup \Delta' \to \{\underline{\text{true}}\}\ x:=1;\underline{\text{goto}}\ L_2\ \{x=0\} \qquad \ldots(\#)$$

using the assumption in Δ', and furthermore

$$\Delta \cup \Delta' \to \{x=0\}x:=x\{x=0\}.$$

Thus, discharging Δ, using "(R5)"

$$\Delta' \to \{\underline{\text{true}}\}\ L_1:x:=1;\underline{\text{goto}}\ L_2;L_2:x:=x\ \{x=0\} \qquad \ldots(b)$$

and this formula is not valid (the assumption $\{\underline{\text{true}}\}\ \underline{\text{goto}}\ L_2\ \{\underline{\text{false}}\}$ in Δ' is not relevant for the validity of (b), because the meaning of $L_1:x:=1;\underline{\text{goto}}\ L_2;L_2:x:=x$ in any γ, given by M, doesn't depend on the meaning $\gamma[\![L_2]\!]$ of L_2 anymore). Difficulties stem from the fact that the assumption in Δ' was used in the derivation of (#), and not discharged.

We now come to the definition of the function G which we shall need to define validity of correctness formulae. The main problem in defining the value of $G[\![\{p\}A\{q\}]\!]$ in some environment γ is that $N[\![A]\!]\gamma\phi$ is not a function that transforms states just before evaluation of A into states immediately after this evaluation, while q is an assertion describing the latter states.

We can however say something about the states at the normal exit point of A in the following indirect way. Consider the formula $\{p\}A\{q\}$ and choose a predicate π (a function in $(\Sigma_0 \to \{ff,tt\})$ which we want to be true in every final state $\sigma' = N[\![A]\!]\gamma\phi\sigma$ corresponding to an initial state σ satisfying

$T[\![p]\!]\sigma=tt$. That is, we want

$$\forall\sigma,\sigma'\in\Sigma_0: [(T[\![p]\!]\sigma=tt \wedge \sigma'=N[\![A]\!]\gamma\phi\sigma) \Rightarrow \pi\sigma'=tt].$$

We will abbreviate this partial correctness condition to

$$\{T[\![p]\!]\}N[\![A]\!]\gamma\phi\{\pi\}.$$

Now, as this formula must correspond to $\{p\}A\{q\}$, we are looking for a relation between q, the continuation ϕ chosen, and the predicate π. It is reasonable to demand that $\pi(\phi\sigma")=tt$ for every (intermediate) state $\sigma"$ satisfying $T[\![q]\!]\sigma"=tt$ (provided $\phi\sigma"\neq\perp$). For, the continuation ϕ is the state transformation describing what happens after evaluation of A has terminated at the normal exit point. So we want q, ϕ and π to be related through $\{T[\![q]\!]\}\phi\{\pi\}$. It turns out that this constraint on ϕ and π is sufficient to lead to a satisfying validity definition.

DEFINITION (predicates; partial correctness, semantical level).
The class of *predicates* Π, with typical element π, is defined by $\Pi = \Sigma_0 \to \{ff,tt\}$. For any $\pi,\pi' \in \Pi$ and $\phi \in M$, we define

$$\{\pi\}\phi\{\pi'\} \iff \forall\sigma,\sigma' \in \Sigma_0: [(\pi\sigma=tt \wedge \sigma'=\phi\sigma) \Rightarrow \pi'\sigma'=tt].$$

DEFINITION (G). The function G with functionality G: $Afor \to \Gamma \to \{ff,tt\}$ is defined by

$$G[\![\{p\}A\{q\}]\!]\gamma=tt \iff \forall\pi\in\Pi\ \forall\phi\in M[\{T[\![q]\!]\}\phi\{\pi\} \Rightarrow \{T[\![p]\!]\}M[\![A]\!]\gamma\phi\{\pi\}].$$

We extend the domain of G to subsets Δ of $Afor$ as follows

$$G[\![\Delta]\!]\gamma=tt \iff \forall f\in\Delta: G[\![f]\!]\gamma=tt.$$

DEFINITION (validity). A correctness formula f is *valid* (written $\models$ f) is
1^o. $f \equiv p$ and $\forall\sigma\in\Sigma_0$: $T[\![p]\!]\sigma=tt$, or
2^o. $f \equiv \Delta \to \{p\}A\{q\}$ and $\forall\gamma\in\Gamma$: $G[\![\Delta]\!]\gamma=tt \Rightarrow G[\![\{p\}A\{q\}]\!]\gamma=tt$, or
3^o. $f \equiv \Delta \to \{p\}S\{q\}$ and $\forall\gamma\in\Gamma$: $G[\![\Delta]\!]\gamma=tt \Rightarrow \{T[\![p]\!]\}M[\![S]\!]\gamma\{T[\![q]\!]\}$.

We now investigate whether the system as it stands now is sound and complete. It will be proven at the end of this chapter that the system is sound. However, the system is not complete. For instance, a formula like

$$\{\{p\}x:=x;\underline{\text{goto}}\ L\ \{q\}\} \to \{p\}\ \underline{\text{goto}}\ L\ \{q\}$$

is valid but not derivable. Therefore we first prove soundness and completeness of a restriction of the system, namely the system consisting of normal correctness formulae only.

DEFINITION (normal correctness formulae). A correctness formula f is called *normal* if

1^{o}. $f \equiv p$, or

2^{o}. $f \equiv \Delta \to \{p\}A\{q\}$, where $\Delta = \{\{p_i\}\ \underline{\text{goto}}\ L_i\ \{\underline{\text{false}}\} \mid i = 1,\ldots,n\}$ such that all L_i are different and that the labels in A are all L_i's, or

3^{o}. $f \equiv \{p\}S\{q\}$, where S is a normal program.

The system H', restricted to the normal formulae, is called *the normal fragment of* H', and denoted by H'_N.

There is an obvious one to one correspondence between the normal correctness formulae as defined here and the normal correctness formulae from Chapter 6, given by the function Φ, defined by

$$\Phi[\![p]\!] = p$$

$$\Phi[\![\{\{p_i\}\ \underline{\text{goto}}\ L_i\ \{\underline{\text{false}}\} \mid i = 1,\ldots,n\} \to \{p\}A\{q\}]\!] = \langle [L_i : p_i]_{i=1}^{n} \mid \{p\}A\{q\}\rangle$$

$$\Phi[\![\{p\}S\{q\}]\!] = \{p\}S\{q\}.$$

If we compare the axioms and inference rules of H with the ones of H' we come to the following lemma:

LEMMA 8.1. *For every normal correctness formula* f *we have*

$$\vdash f \text{ (in } H') \iff \vdash \Phi[\![f]\!] \text{ (in } H).$$

PROOF. The $\Leftarrow$ - direction is obvious. The proof of " $\Rightarrow$ " essentially amounts to showing that H' is *conservative over* H'_N, i.e. if a normal formula is derivable in H' then it has a proof in H'_N. This can be shown using the fact that every inference rule has normal premisses if its conclusion is a normal formula. □

If we can prove the same result for validity instead of deducibility then we can infer from the results in Chapter 7 that H'_N is sound and complete.

To achieve this, we first prove some lemmas, relating the definition of validity of $f \equiv \Delta \to \{p\}A\{q\}$ with validity of $\Phi[\![f]\!]$.

LEMMA 8.2. *Suppose* $f \equiv \{\{p_i\}$ goto L_i $\{$false$\} \mid i = 1,\ldots,n\} \to \{p\}A\{q\}$ *is a correctness formula that is normal and valid. Then the following holds:*

a) $\forall \sigma,\sigma' \in \Sigma_0$: $(A[\![A]\!]\sigma=\sigma' \wedge T[\![p]\!]\sigma=tt) \Rightarrow T[\![q]\!]\sigma'=tt$

b) $\forall \sigma,\sigma' \in \Sigma_0$: $(A[\![A]\!]\sigma=<\sigma',L_i> \wedge T[\![p]\!]\sigma=tt) \Rightarrow T[\![p_i]\!]\sigma'=tt$ $(i = 1,\ldots,n)$.

PROOF.

a) Choose $\sigma,\sigma' \in \Sigma_0$ such that $T[\![p]\!]\sigma=tt$ and $A[\![A]\!]\sigma=\sigma'$. Choose γ_0 such that $\gamma_0[\![L_i]\!]=\lambda\sigma\cdot\perp$ for $i = 1,\ldots,n$. Then we can check that $G[\![\{p_i\}$ goto L_i $\{$false$\}]\!]\gamma_0=tt$ for $i = 1,\ldots,n$ and thus from validity of f we get $G[\![\{p\}A\{q\}]\!]\gamma_0=tt$. From the definition of G we then have

$$\forall \pi \in \Pi\ \forall \phi \in M[\{T[\![q]\!]\}\phi\{\pi\} \Rightarrow \{T[\![p]\!]\}(N[\![A]\!]\gamma_0\phi)\{\pi\}].$$

If we choose $\pi = T[\![q]\!]$ and $\phi = \lambda\sigma\cdot\sigma$, we can deduce from this

$$\{T[\![p]\!]\}(N[\![A]\!]\gamma_0\{\lambda\sigma\cdot\sigma\})\{T[\![q]\!]\}.$$

Combining this with $N[\![A]\!]\gamma_0\{\lambda\sigma\cdot\sigma\}\sigma = \sigma'$ (Lemma 4.3.1°) and $T[\![p]\!]\sigma = tt$, we get $T[\![q]\!]\sigma'=tt$.

b) Choose $\sigma,\sigma' \in \Sigma_0$ and i (where $1 \le i \le n$) such that $T[\![p]\!]\sigma = tt$ and $A[\![A]\!]\sigma = <\sigma',L_i>$. Now, if we take γ_0 such that

$$\gamma_0[\![L_j]\!]\sigma = \begin{cases} \sigma & \text{if } \sigma \neq \perp \text{ and } T[\![p_j]\!]\sigma = ff \\ \perp & \text{otherwise} \end{cases}$$

for $j = 1,\ldots,n$, we again have that $G[\![\{p_j\}$ goto L_j $\{$false$\}]\!]\gamma_0 = tt$ $(j = 1,\ldots,n)$. Arguing the same way as in the proof of a) we come to

$$\forall \pi \in \Pi\ \forall \phi \in M[\{T[\![q]\!]\}\phi\{\pi\} \Rightarrow \{T[\![p]\!]\}(N[\![A]\!]\gamma_0\phi)\{\pi\}].$$

Now we choose $\phi = \lambda\sigma\cdot\perp$ and $\pi = \lambda\sigma\cdot ff$. We then derive

$$\{T[\![p]\!]\}(N[\![A]\!]\gamma_0\{\lambda\sigma\cdot\perp\})\{\lambda\sigma\cdot ff\}.$$

Combining this with $N[\![A]\!]\gamma_0\{\lambda\sigma\cdot\perp\}\sigma = \gamma_0[\![L_i]\!]\sigma'$ (Lemma 4.3.2°) and with $T[\![p]\!]\sigma = tt$ we have that

$$\gamma_0[\![L_i]\!]\sigma' \neq \perp \Rightarrow (\lambda\sigma\cdot ff)(\gamma_0[\![L_i]\!]\sigma') = tt.$$

So we must have $\gamma_0[\![L_i]\!]\sigma' = \perp$, but this is (by definition of γ_0 and the fact that $\sigma' \neq \perp$) equivalent to $T[\![p_i]\!]\sigma' = tt$. □

LEMMA 8.3. *Suppose* $f \equiv \{\{p_i\}\ \underline{\text{goto}}\ L_i\ \{\underline{\text{false}}\} \mid i = 1,\ldots,n\} \to \{p\}A\{q\}$ *is a normal correctness formula. Then*

$$\models f \iff \forall\sigma\in\Sigma_0\colon\ T[\![p]\!]\sigma=tt \Rightarrow \left[\begin{array}{l}(\exists\sigma'\in\Sigma_0[A[\![A]\!]\sigma=\sigma' \wedge T[\![q]\!]\sigma'=tt]) \vee \\ (\exists\sigma'\in\Sigma_0[A[\![A]\!]\sigma=<\sigma',L_i> \wedge T[\![p_i]\!]\sigma'=tt])\end{array}\right].$$

PROOF. " $\Rightarrow$ ". Suppose $\models f$ and $T[\![p]\!]\sigma=tt$. There are two possibilities (by definition of A)

a) $A[\![A]\!]\sigma = \sigma'\in\Sigma_0$, and Lemma 8.2a yields $T[\![q]\!]\sigma'=tt$

b) $A[\![A]\!]\sigma = <\sigma',L>$. Since f is normal, which means that all labels in A are an L_i, we have that L is an L_i for some i ($1 \le i \le n$). We then can apply Lemma 8.2b to obtain $T[\![p_i]\!]\sigma' = tt$.

" $\Leftarrow$ ". Choose $\gamma \in \Gamma$ such that $G[\![\{p_i\}\ \underline{\text{goto}}\ L_i\ \{\underline{\text{false}}\}]\!]\gamma = tt$ for $i = 1,\ldots,n$. Then we must derive $G[\![\{p\}A\{q\}]\!]\gamma = tt$, or equivalently

$$\forall\pi\in\Pi\ \forall\phi\in M[\{T[\![q]\!]\}\phi\{\pi\} \Rightarrow \{T[\![p]\!]\}(N[\![A]\!]\gamma\phi)\{\pi\}].$$

So choose π_0 and ϕ_0 such that $\{T[\![q]\!]\}\phi_0\{\pi_0\}$ holds, and choose σ such that $T[\![p]\!]\sigma=tt$. We have to prove $\sigma'' = N[\![A]\!]\gamma\phi_0\sigma\neq\perp \Rightarrow \pi_0\sigma''=tt$. Again we have two possibilities:

a) $A[\![A]\!]\sigma=\sigma'$. Then by assumption $T[\![q]\!]\sigma'=tt$, and by Lemma 4.3.1°: $\sigma'' = N[\![A]\!]\gamma\phi_0\sigma = \phi_0\sigma'$. From $\{T[\![q]\!]\}\phi_0\{\pi_0\}$ we then have $\sigma''\neq\perp \Rightarrow \pi_0\sigma''=tt$.

b) $A[\![A]\!]\sigma = <\sigma',L_i>$. By assumption $T[\![p_i]\!]\sigma' = tt$, and by Lemma 4.3.2°: $\sigma'' = N[\![A]\!]\gamma\phi_0\sigma = \gamma[\![L_i]\!]\sigma'$. Now we use the fact that $G[\![\{p_i\}\ \underline{\text{goto}}\ L_i\{\underline{\text{false}}\}]\!]\gamma=tt$, or $\forall\pi\in\Pi\ \forall\phi\in M[T[\![\underline{\text{false}}]\!]\phi\{\pi\} \Rightarrow \{T[\![p_i]\!]\}(\gamma[\![L_i]\!])\{\pi\}]$. Taking $\pi = \pi_0$ and $\phi = \lambda\sigma\cdot\perp$, we get $\{T[\![p_i]\!]\}(\gamma[\![L_i]\!])\{\pi_0\}$, and from this we prove $\sigma''\neq\perp \Rightarrow \pi_0\sigma''=tt$. □

COROLLARY. *For all normal correctness formulae* f *we have*

$\models f$ *(according to the validity definition of this chapter)* $\iff$
$\models \Phi[\![f]\!]$ *(according to the definition of Chapter 6).*

PROOF. This is the lemma for $f \equiv \Delta \to \{p\}A\{q\}$. For all other cases for f we have that the respective validity definitions are the same. □

This corollary and the results of Chapter 7 now lead to

THEOREM 8.4. *The system* H'_N *is sound and complete in the sense of Cook.*

To conclude this chapter we show that system H' is sound (although, as we have seen before) not complete.

THEOREM 8.5. *For all correctness formulae* f, *we have* $\vdash f \Rightarrow \models f$.

PROOF. The proof is analogous to the proof of 7.1. We prove here the more interesting cases, viz. validity of (A1), (R1) and (R5).

(A1) Validity is proven if we can show that
$\forall\gamma\in\Gamma\ \forall\phi\in M\ \forall\pi\in\Pi[\{T[\![p]\!]\}\phi\{\pi\} \Rightarrow \{T[\![p[s/x]]\!]\}(N[\![x:=s]\!]\gamma\phi)\{\pi\}$.
So, choose γ, ϕ, π and σ such that $\{T[\![p]\!]\}\phi\{\pi\}$ and $T[\![p[s/x]]\!]\sigma=tt$ hold. Lemma 6.1d then gives $T[\![p]\!]\sigma'=tt$, where $\sigma' = \sigma\{V[\![s]\!]\sigma/x\}$. Now from $N[\![x:=s]\!]\gamma\phi\sigma = \phi\sigma'$ and $\{T[\![p]\!]\}\phi\{\pi\}$ we have $\sigma'' = N[\![x:=s]\!]\gamma\phi\sigma=\phi\sigma'\neq\perp\Rightarrow\pi\sigma''=tt$.

(R1) Suppose $\models p_1\supset p_2$, $\models p_3\supset p_4$ and $\models \Delta \to \{p_2\}A\{p_3\}$. We then have to prove $\models \Delta \to \{p_1\}A\{p_4\}$. Suppose that we have a $\gamma \in \Gamma$ such that $G[\![\Delta]\!]\gamma=tt$ (if there is no such γ then $\Delta \to \{p_1\}A\{p_4\}$ is vacuously valid). We then must prove $G[\![\{p_1\}A\{p_4\}]\!]\gamma=tt$, or
$\forall\phi\in M\ \forall\pi\in\Pi[\{T[\![p_4]\!]\}\phi\{\pi\} \Rightarrow \{T[\![p_1]\!]\}N[\![A]\!]\gamma\phi\{\pi\}$. We will prove this using the following fact:

$$\text{if } \forall\sigma\in\Sigma_0\text{: } \pi\sigma=tt \Rightarrow \pi'\sigma=tt, \text{ then } \{\pi'\}\phi\{\bar{\pi}\} \Rightarrow \{\pi\}\phi\{\bar{\pi}\} \qquad \ldots(*)$$

which can easily be verified.
Now suppose $\{T[\![p_4]\!]\}\phi\{\pi\}$. From $\models p_3\supset p_4$ and (*) we get $\{T[\![p_3]\!]\}\phi\{\pi\}$. From this, $\models \Delta \to \{p_2\}A\{p_3\}$ and $G[\![\Delta]\!]\gamma=tt$, we have $\{T[\![p_2]\!]\}(N[\![A]\!]\gamma\phi)\{\pi\}$. Then we use $\models p_1\supset p_2$ and (*) again to derive $\{T[\![p_1]\!]\}(N[\![A]\!]\gamma\phi)\{\pi\}$.

(R5) Let $\Delta = \{\{p_i\}\ \underline{\text{goto}}\ L_i\ \{\underline{\text{false}}\} \mid i = 1,\ldots,n\}$ where all L_i are different, and let Δ' be such that no L_i occurs in any formula in Δ'. Suppose furthermore that we have $\models \Delta\cup\Delta' \to \{p_i\}A_i\{p_{i+1}\}$ for $i = 1,\ldots,n$. We have to prove $\models \Delta' \to \{p_1\}[L_i:A_i]_{i=1}^n\{p_{n+1}\}$.
So, choose γ such that $G[\![\Delta']\!]\gamma=tt$. Let $\phi_i,\phi_i^{(k)}$ and $\gamma^{(k)}$ $(i = 1,\ldots,n;$ $k = 0,1,\ldots)$ be derived from γ and $[L_i:A_i]_{i=1}^n$ as in Lemma 4.2. We then have to prove $\{T[\![p_1]\!]\}\phi_1\{T[\![p_{n+1}]\!]\}$ (take i = 1 in Lemma 4.2.2).
Now if we can prove

$$\forall k\{T[\![p_i]\!]\}\phi_i^{(k)}\{T[\![p_{n+1}]\!]\} \quad (i = 1,\ldots,n+1) \qquad \ldots(\#)$$

then we are ready. For suppose we have $\sigma,\sigma' \in \Sigma_0$ such that $T[\![p_1]\!]\sigma$=tt and $\sigma' = \phi_1\sigma$. Then (since $\phi_1 = \bigsqcup_k (\phi_1^{(k)})$ we have $\sigma' = (\bigsqcup_k \phi_i^{(k)})\sigma = \bigsqcup_k (\phi_1^{(k)}\sigma)$, and because Σ is a discrete cpo there must be a $\bar{k}$ such that $\phi_1\sigma = \sigma'$. But then we can infer that $T[\![p_{n+1}]\!]\sigma'$=tt by applying (#) with i = 1 and k = $\bar{k}$.

We now prove (#) by induction on k. The basis (k = 0) is trivial, so we now perform the induction step. Choose i ($1 \le i \le n$, the case i = n+1 being trivial). We have to prove $\{T[\![p_i]\!]\}\phi_i^{(k+1)}\{T[\![p_{n+1}]\!]\}$ or $\{T[\![p_i]\!]\}(N[\![A_i]\!]\gamma^{(k)}\phi_{i+1}^{(k)})\{T[\![p_{n+1}]\!]\}$.

Choose $\sigma,\sigma'' \in \Sigma_0$ such that $T[\![p_i]\!]\sigma$=tt and $\sigma'' = N[\![A_i]\!]\gamma^{(k)}\phi_{i+1}^{(k)}\sigma$. We have to show $T[\![p_{n+1}]\!]\sigma''$=tt. We do this in a way that is analogous to the proof of Lemma 8.2, i.e. by choosing a suitable environment γ_0. We distinguish three cases: $A[\![A_i]\!]\sigma = \sigma'$, $A[\![A_i]\!]\sigma = \langle\sigma',L_j\rangle$ and $A[\![A_i]\!]\sigma = \langle\sigma',L\rangle$, where L is not an L_j.

1°. $A[\![A_i]\!]\sigma=\sigma'$ and thus (4.3.1°) $\sigma'' = N[\![A_i]\!]\gamma^{(k)}\phi_{i+1}^{(k)}\sigma = \phi_{i+1}^{(k)}\sigma'$. So, if we prove $T[\![p_{i+1}]\!]\sigma'$=tt, we can use the induction hypothesis to infer that $T[\![p_{n+1}]\!]\sigma''$=tt.

We have $\models \Delta\cup\Delta' \to \{p_i\}A_i\{p_{i+1}\}$; we also have $G[\![\Delta']\!]\gamma$=tt. Now, taking $\gamma_0 = \gamma\{\lambda\sigma\cdot\perp/L_i\}_{i=1}^n$, we can prove that (due to the fact that no L_i occurs in Δ') $G[\![\Delta']\!]\gamma_0$=tt. Also, $G[\![\{p_i\}$ goto L_i $\{\underline{\text{false}}\}]\!]\gamma_0$=tt. Thus we have $G[\![\Delta\cup\Delta']\!]\gamma_0$=tt and thus $G[\![\{p_i\}A_i\{p_{i+1}\}]\!]\gamma_0$=tt. The same way as in the proof of 8.2a we now get that $T[\![p_{i+1}]\!]\sigma'$=tt from $A[\![A_i]\!]\sigma=\sigma'$ and $T[\![p_i]\!]\sigma$=tt.

2°. $A[\![A_i]\!]\sigma = \langle\sigma',L_j\rangle$ and thus (4.3.2°) $\sigma'' = N[\![A_i]\!]\gamma^{(k)}\phi_{i+1}^{(k)}\sigma = \gamma^{(k)}[\![L_j]\!]\sigma' = \phi_j^{(k)}\sigma'$. So, if we can prove $T[\![p_j]\!]\sigma'$=tt, then we can use the induction hypothesis to infer that $T[\![p_{n+1}]\!]\sigma''$=tt. We do this by choosing $\gamma_0 = \gamma\{\bar{\phi}_t/L_t\}_{t=1}^n$, where $\bar{\phi}_t$ is defined by a) $\bar{\phi}_t\sigma = \sigma$, if $\sigma\neq\perp$ and $T[\![p_t]\!]\sigma$ = ff; b) $\bar{\phi}_t\sigma = \perp$ otherwise. We again can check that $G[\![\Delta\cup\Delta']\!]\gamma_0$=tt and thus $G[\![\{p_i\}A_i\{p_{i+1}\}]\!]\gamma_0$ = tt. In a way, analogous to the proof of 8.2b we then can deduce from $T[\![p_i]\!]\sigma$=tt and $A[\![A_i]\!]\sigma = \langle\sigma',L_j\rangle$ that $T[\![p_j]\!]\sigma'$=tt.

3°. $A[\![A_i]\!]\sigma = \langle\sigma',L\rangle$ where L is not an L_j. Lemma 4.3.2° yields $\sigma'' = N[\![A_i]\!]\gamma^{(k)}\phi_{i+1}^{(k)}\sigma' = \gamma^{(k)}[\![L]\!]\sigma' = \gamma[\![L]\!]\sigma'$ (for $\gamma^{(k)}$ differs from γ only in the arguments $L_1,\ldots,L_n$). Now taking $\gamma_0 = \gamma\{\lambda\sigma\cdot\perp/L_j\}_{j=1}^n$ we have that $\sigma'' = \gamma_0[\![L]\!]\sigma'$, but also that $G[\![\Delta\cup\Delta']\!]\gamma_0$=tt, so that $G[\![\{p_i\}A_i\{p_{i+1}\}]\!]\gamma_0$ = tt, or equivalently

$\forall\pi\in\Pi\ \forall\phi\in M[\{T[\![p_{i+1}]\!]\}\phi\{\pi\} \Rightarrow \{T[\![p_i]\!]\}N[\![A_i]\!]\gamma_0\phi\{\pi\}]$. Now choose $\phi = \lambda\sigma\cdot\perp$ and $\pi = T[\![p_{n+1}]\!]$. We then have $\{T[\![p_i]\!]\}(N[\![A_i]\!]\gamma_0\{\lambda\sigma\cdot\perp\})\{T[\![p_{n+1}]\!]\}$, which combined with $A[\![A_i]\!]\sigma = \langle\sigma',L\rangle$, $\sigma'' = \gamma_0[\![L]\!]\sigma'$ and $T[\![p_i]\!]\sigma=tt$ yields $T[\![p_{n+1}]\!]\sigma'' = tt$. □

ACKNOWLEDGEMENTS. *I am grateful to Jaco de Bakker for several remarks that led to improvements. I owe much to Jeff Zucker. He read the manuscript carefully and patiently, and his stimulating remarks have had a substantial influence on the final version.*

REFERENCES

[1] APT, K.R., *A sound and complete Hoare-like system for a fragment of PASCAL* (preprint), Report IW 97/78, Mathematisch Centrum (1978).

[2] APT, K.R. & J.W. DE BAKKER, *Semantics and proof theory of PASCAL procedures,* in: Proc. of the fourth colloquium on automata, languages and programming, Lecture Notes in Computer Science 52, pp.30-44, Springer (1977).

[3] ARBIB, M.A. & S. ALAGIĆ, *Proof rules for gotos,* Acta Informatica 11, pp.139-148 (1979).

[4] BAKKER, J.W. DE, *Correctness proofs for assignment statements,* Report IW 55/76, Mathematisch Centrum (1976).

[5] CLINT, M. & C.A.R. HOARE, *Program proving: jumps and functions,* Acta Informatica 1, pp.214-224 (1972).

[6] COOK, S.A., *Soundness and completeness of an axiom system for program verification,* SIAM J. on Computing, Vol. 7, nr. 1, pp.70-90 (1978).

[7] HOARE, C.A.R., *An axiomatic basis for computer programming,* Comm. of the ACM, Vol. 12, nr. 10, pp.576-580, 583 (1969).

[8] MILNE, R. & C. STRACHEY, *A theory of programming language semantics,* Chapman and Hall, London and Wiley, New York, 2 vols. (1976).

[9] SCOTT, D. & C. STRACHEY, *Towards a mathematical semantics for computer languages,* Proc. Symp. on computers and automata, Polytechnic

Institute of Brooklyn, pp.19-46 (1971); also: Tech. Mon. PRG-6, Oxford Univ. Computing Lab.

[10] STOY, J.E., *Denotational semantics - the Scott-Strachey approach to programming language theory*, M.I.T. Press, Cambridge, Mass. (1977).

[11] STRACHEY, C. & C. WADSWORTH, *Continuations, a mathematical semantics for handling full jumps*, Tech. Mon. PRG-11, Oxford Univ. Computing Lab., Programming research group (1974).

[12] WAND, M., *A new incompleteness result for Hoare's system*, J. of the ACM, Vol. 25, nr. 1, pp.168-175 (1978).