stichting
mathematisch
centrum

MC

DEPARTMENT OF COMPUTER SCIENCE IW 55/76 MARCH

J.W. DE BAKKER

CORRECTNESS PROOFS FOR ASSIGNMENT STATEMENTS

Prepublication

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), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

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

ACM-Computing Reviews-categories: 5.24

Correctness proofs for assignment statements [*)]

by

J.W. de Bakker

ABSTRACT

Correctness proofs for assignment statements are usually based on Hoare's or Floyd's assignment axiom. We observe that these axioms do not apply (directly) to assignment to subscripted variables. A refined definition of substitution for subscripted variables is proposed which preserves the validity of Hoare's axiom, and which is used in the formulation of an extension of Floyd's axiom. For both axioms, a validity proof is given within the framework of denotational semantics of Scott and Strachey. Moreover, it is shown that they yield the weakest precondition and strongest postcondition, respectively.

KEY WORDS & PHRASES: *assignment statement, program correctness, subscripted variables, Hoare's assignment axiom, Floyd's assignment axiom, substitution, denotational semantics, weakest precondition, strongest postcondition*

[*)] This paper is not for review; it is meant for publication elsewhere.

CONTENTS

1. Introduction 1

2. Syntax and substitution 4

3. Semantics of assignment and the validity of Hoare's axiom 8

4. An extended version of Floyd's axiom 15

References 20

1. INTRODUCTION

Consider the assignment statement $x := 1$. Using either Floyd's forward assignment axiom, or Hoare's backward assignment axiom, one may infer that after its execution the assertion $x = 1$ holds, as might be expected. Somewhat unexpectedly, however, after executing $a[s] := 1$, where s is some reasonably innocent subscript expression, we cannot at all be sure that the assertion $a[s] = 1$ holds. E.g., let $s \equiv a[2]$, and let $a[1]$ and $a[2]$ both have the value 2 before the assignment. Then, according to the usual semantics of assignment as defined e.g. for ALGOL 60 or PASCAL, we have that after $a[s] := 1$, the assertion $a[s] = 2$ holds! Related is the following observation: It is widely assumed that the program $x := v; v := w; w := x$ interchanges the values of v and w. However, this is not true in general. By way of counter example take $v \equiv a[a[1]]$, $w \equiv a[a[2]]$, and let $a[1]$ and $a[2]$ have initial values 2 and 1, respectively.

As far as we know, there is not yet a formal treatment available of the problems raised by assignment to subscripted variables. The present paper proposes a method which has as its main feature a refined definition of the notion of substitution for subscripted variables. Before saying more about this, let us recall the Floyd and Hoare axioms for assignment. Remember that, for any statement S and assertions p,q, the formula

(1.1) $\{p\}\ S\ \{q\}$

has the intended meaning: For each initial state σ satisfying p, if S transforms σ to final state $\sigma' = S(\sigma)$, then σ' satisfies q. Let us call p and q in (1.1) the precondition and postcondition, respectively. Now let S be the assignment statement $x := t$, with x a simple variable and t an integer expression. Moreover, for any assertion p, let $p[t/x]$ denote the result of substituting t for x in p, i.e., of replacing all occurrences of x in p by t. Similarly, we define $s[t/x]$, for s any integer expression. We have

Floyd's forward assignment axiom [1]

(F) $\{p\}\ x := t\ \{\exists y[p[y/x] \wedge x = t[y/x]]\}$
(where y is some variable not occurring in p or t)

Hoare's backward assignment axiom [2]

(*H*) $\{p[t/x]\}\ x := t\ \{p\}$

Observe that Floyd's axiom assumes the precondition given and tells us how to obtain the (strongest) postcondition, whereas Hoare's axiom assumes the postcondition given, and enables us to obtain the (weakest) precondition. (For the terms "strongest" and "weakest" see section 4.)

Examples.

By Floyd's axiom:

1.1. $\{x = 0\}\ x := x + 1\{\exists y[(x = 0)[y/x] \wedge x = (x+1)[y/x]]\}$
or, after simplification
$\{x = 0\}\ x := x + 1\{\exists y[y = 0 \wedge x = y + 1]\}$,
from which
$\{x = 0\}\ x := x + 1\{x = 1\}$
follows.
Observe that the variable y in the postcondition of (*F*) is used to store the value of x before the assignment.

1.2. $\{\underline{true}\}\ x := 1\{\exists y[(\underline{true})[y/x] \wedge x = 1[y/x]]\}$
or
$\{\underline{true}\}\ x := 1\{x = 1\}$.
Thus, (*F*) allows us to infer that, whatever precondition we have, after $x := 1$ the postcondition $x = 1$ always holds.

By Hoare's axiom:

2.1. $\{(x=1)[x + 1/x]\}\ x := x + 1\{x = 1\}$
which reduces to
$\{x + 1 = 1\}\ x := x + 1\{x = 1\}$

2.2. $\{(x=1)[1/x]\}\ x := 1\{x = 1\}$
which simplifies to
$\{1 = 1\}\ x := 1\{x = 1\}$,
again allowing us to infer that after $x := 1$ we necessarily have that $x = 1$.

Now let us consider the assignment statement $a[s] := 1$. One might be tempted to expect (*F*) or (*H*) to cover also assignment to subscripted variables. However, this would lead to the undesirable conclusion that, e.g., after $a[s] := 1$ we necessarily have that $a[s] = 1$. In fact, if one takes $s \equiv a[2]$, and treats $a[a[2]]$ as a variable of the same nature as the simple variable x, one could, e.g., apply (*H*) exactly as above, obtaining

$$\{(a[a[2]]=1)[1/a[a[2]]]\} \quad a[a[2]] := 1\{a[a[2]] = 1\}$$

which reduces to

$$\{1 = 1\} \quad a[a[2]] := 1\{a[a[2]] = 1\},$$

and we therefore see that a direct application of (*H*) to subscripted variables leads to an invalid result. Thus, (*F*) and (*H*) cannot be simply carried over to assignment in general, and we can state the main goal of our paper: Find suitable versions of (*F*) and (*H*) which *are* valid for assignment to subscripted variables as well.

The main tool for the solution of this problem is a careful analysis of substitution for subscripted variables, leading to a new definition which will be presented in section 2. It will turn out that (*H*) remains valid in the form as given, provided that the new definition of substitution is used. For (*F*) we need a somewhat more complex form which is introduced in section 4. Essentially, in order to deal with $a[s] := t$, we use existential quantification over two variables y and z - not occurring in p,s or t - such that z is used to store the old value of s and y to store the old value of $a[s]$.

A precise definition of the language we use is to be found in section 2, and in section 3 we present its semantics - which amounts to a simple definition in the style of the *denotational semantics* of SCOTT & STRACHEY [5] - together with a proof of the validity of (*H*). In section 4 we also prove that the precondition and postcondition as determined by (*H*) and (the extended version of) (*F*) are indeed "weakest" and "strongest", i.e., that they are the best one can expect, in a sense to be made precise below.

As remarked above, we have not encountered in the literature a satisfactory formal treatment of assignment to subscripted variables. Some partial treatment of the problem is to be found e.g. in IGARASHI, LONDON & LUCKHAM [4],

- and presumably elsewhere - where a rule is given which amounts to a definition of substitution of the following type: Let x,y be simple variables. Then

$$a[x][s/a[y]] \equiv (\underline{\text{if}}\ x = y\ \underline{\text{then}}\ s\ \underline{\text{else}}\ a[x]\underline{\text{fi}})$$

This rule is indeed obtainable as a special case of ours, but it does not extend to the general case where x and y are replaced by arbitrary subscript expressions.

ACKNOWLEDGEMENT: *I am indebted to K.R. Apt and P. van Emde Boas for a number of helpful discussions.*

2. SYNTAX AND SUBSTITUTION

Our analysis of assignment will be presented using a language with only a very modest syntax: We have *statements* (assignment statements and sequences thereof) and *expressions* (of two types: *integer* and *boolean*) made up from *constants* and *variables* (either *simple* or *subscripted*) by simple operations like addition and multiplication for integer expressions, and conjunction and implication for boolean expressions. For integer expressions we moreover have the possibility of forming *conditionals*. The full description is given in definition 2.1, in which we use a variant of BNF which should be self-explanatory.

DEFINITION 2.1 (Syntax of simple programming language).
The following syntactical classes are introduced:

a. A set *SV* of *simple variables*, with elements denoted by x,y,... . *SV* is an arbitrary set of symbols.

b. A set *AV* of *array variables*, with elements denoted by a,b,... . *AV* is an arbitrary set of symbols disjoint from *SV*.

c. A set *C* of *integer constants*, with elements denoted by n,m,... . *C* is a set of integer denotations, i.e., of certain symbolic entities suggesting corresponding integers as values.

d. A set V of *variables*, with elements denoted by v,w,..., and defined as $v ::= x \mid a[s]$

e. A set E of *integer expressions*, with elements denoted by s,t,..., and defined as $s ::= v \mid n \mid s_1 + s_2 \mid s_1 * s_2 \mid \underline{\text{if}}\ p\ \underline{\text{then}}\ s_1\ \underline{\text{else}}\ s_2\ \underline{\text{fi}}$

f. A set P of *boolean expressions*, with elements denoted by p,q,..., and defined as $p ::= \underline{\text{true}} \mid \underline{\text{false}} \mid p_1 \wedge p_2 \mid p_1 \supset p_2 \mid s = t \mid s > t$

A set S of *statements*, with elements S_1, S_2,..., and defined as $S ::= S_1 ;\ S_2 \mid v := t$

We assume that parentheses may be used freely in our language to enhance readability and to avoid ambiguities. We shall not bother to give any formal specification of this.

The next definition introduces the form in which correctness of statements is stated:

DEFINITION 2.2 (Correctness formulae).
A *correctness formula* is a construct of the form $\{p\}\ S\ \{q\}$, with $p,q \in P$ and $S \in S$.

We also need a definition of *syntactic equality:*

DEFINITION 2.3 (Syntactic equality).
For $s_1, s_2 \in E$, $s_1 \equiv s_2$ iff s_1 and s_2 are the same sequence of symbols. Similarly for $p_1 \equiv p_2$.

Example: Assuming that $C = \{0,1,2,...\}$, we have that $0 \equiv 0$ and $1 + 1 \not\equiv 2$. Also, $a[s] \equiv b[t]$ iff $a \equiv b$ and $s \equiv t$.

As counterpart of the notion of syntactic equality, we have that of *semantic equality*. Anticipating the definition given in section 3, we mention already that two integer expressions s and t are semantically equal (written as $s = t$ in the present section only, in section 3 we remedy this unsatisfactory notation) iff they result in the same value whatever initial values are given to the variables occurring in them. A similar definition applies to boolean expressions.

Example: $0 = 0$, $a[1+1] = a[2]$, $x + y = y + x$, $\underline{\text{if}}\ 1 = 1\ \underline{\text{then}}\ x\ \underline{\text{else}}\ y\ \underline{\text{fi}} = x$, $(\underline{\text{true}} \supset \underline{\text{false}}) = \underline{\text{false}}$.

We now introduce our definition of *substition*. By way of preparation consider a correctness formula $\{q\}\ S\ \{p\}$. Let us write - in an ad-hoc notation used only until the end of our informal explanation of the definition of substitution - $S{:}p$ for the boolean expression which is true for a given state σ iff p is true for the state $S(\sigma)$. Then, clearly, we have that $\{S{:}p\}\ S\ \{p\}$ is valid. (A precise definition of the notion of validity follows in section 3.) Now taking for S the assignment statement $v := t$, we obtain $\{(v{:=}t){:}\ p\}\ v := t\ \{p\}$, and we want to define $p[t/v]$ in such a way that $(v{:=}t){:}\ p$ holds iff $p[t/v]$ holds. Having established this, we conclude that $\{p[t/v]\}\ v := t\ \{p\}$ is valid, as desired. So let us determine the effect of $v := t$ upon the boolean expression p, in order to determine $(v{:=}t){:}\,p$. It is natural to proceed by induction on the structure of p, using, e.g., $(v{:=}t){:}\ \underline{\text{true}} = \underline{\text{true}},\ldots,(v{:=}t){:}\ p_1 \wedge p_2 = ((v{:=}t){:}\ p_1) \wedge ((v{:=}t){:}\ p_2),\ldots,$ $(v{:=}t){:}\ (s_1{=}s_2) = (((v{:=}t){:}\ s_1) = ((v{:=}t){:}\ s_2)),\ldots,(v{:=}t){:}\ (s_1{+}s_2) =$ $((v{:=}t){:}\ s_1) + (v{:=}t){:}\ s_2,\ldots,$etc. In this way, we decompose the expression until we encounter its variables, and accordingly we have to determine the value of $(v{:=}t){:}\ w$, for arbitrary variables v,w. This is done in the following case analysis:

1. Let $v \equiv x$, for some simple variable $x \in \mathit{SV}$.

   $(x{:=}t){:}\ x \quad = t$

   $(x{:=}t){:}\ y \quad = y \ (x{\neq}y)$

   $(x{:=}t){:}\ a[s] \quad = a[(x{:=}t){:}\ s]$

2. let $v \equiv a[s_1]$, for some $a \in \mathit{AV}$ and $s_1 \in E$.

   $(a[s_1] := t){:}\ x \quad = x$

   $(a[s_1] := t){:}\ b[s_2] = b[(a[s_1] := t){:}\ s_2] \quad (a{\neq}b)$

   $(a[s_1] := t){:}\ a[s_2]$. This is the most interesting case. We distinguish two subcases:

2.1. $(a[s_1] := t){:}\ s_2 \quad = s_1$. Then
$(a[s_1] := t){:}\ a[s_2] = t.$

2.2. $(a[s_1] := t){:}\ s_2 \quad \neq s_1$. Then
$(a[s_1] := t){:}\ a[s_2] = a[(a[s_1] := t){:}\ s_2]$

This explanation may help to understand the definition of substitution which now follows:

DEFINITION 2.4 (Substition).

For any $v \in V$ and $t \in E$.

1. $p[t/v]$ is defined by induction on the structure of p:

   $\underline{\text{true}}\ [t/v] \equiv \underline{\text{true}}$, $\underline{\text{false}}\ [t/v] \equiv \underline{\text{false}}$,

   $(p_1 \wedge p_2)\ [t/v] \equiv p_1[t/v] \wedge p_2\ [t/v]$, and similarly for $\supset$,

   $(s_1 = s_2)\ [t/v] \equiv (s_1[t/v] = s_2[t/v])$, and similarly for $>$.

2. $s[t/v]$ is defined by induction on the structure of s:

   $n[t/v] \equiv n$,

   $(s_1 + s_2)[t/v] \equiv s_1[t/v] + s_2[t/v]$, and similarly for $*$,

   $(\underline{\text{if}}\ p\ \underline{\text{then}}\ s_1\ \underline{\text{else}}\ s_2\ \underline{\text{fi}})\ [t/v] \equiv \underline{\text{if}}\ p[t/v]\ \underline{\text{then}}\ s_1[t/v]\ \underline{\text{else}}\ s_2[t/v]\ \underline{\text{fi}}$

3. $w[t/v]$ is defined by the following case analysis:

3.1. $v \equiv x$, for some $x \in SV$.

$x[t/x] \equiv t$

$y[t/x] \equiv y\ (x \neq y)$

$a[s][t/x] \equiv a[s[t/x]]$

3.2. $v \equiv a[s_1]$, for some $a \in AV$ and $s_1 \in E$.

$x[t/a[s_1]] \equiv x$

$b[s_2][t/a[s_1]] \equiv b[s_2[t/a[s_1]]] \qquad (a \neq b)$

$a[s_2][t/a[s_1]] \equiv \underline{\text{if}}\ s_2[t/a[s_1]] = s_1\ \underline{\text{then}}\ t\ \underline{\text{else}}\ a[s_2[t/a[s_1]]]\ \underline{\text{fi}}$.

Examples.

1. Let $a \neq b$.

   $b[a[2]][1/a[2]] \equiv$

   $b[a[2][1/a[2]]] \equiv$

   $b[\underline{\text{if}}\ 2[1/a[2]] = 2\ \underline{\text{then}}\ 1\ \underline{\text{else}}\ a[2[1/a[2]]]\ \underline{\text{fi}}] \equiv$

   $b[\underline{\text{if}}\ 2 = 2\ \underline{\text{then}}\ 1\ \underline{\text{else}}\ a[2]\ \underline{\text{fi}}]$.

   Semantically (though not syntactically) this last expression equals $b[1]$.

2. $a[a[2]][1/a[a[2]]] \equiv$

   $\underline{\text{if}}\ a[2][1/a[a[2]]] = a[2]\ \underline{\text{then}}\ 1\ \underline{\text{else}}\ a[a[2][1/a[a[2]]]]\ \underline{\text{fi}} \equiv$

   $\underline{\text{if}}(\underline{\text{if}}\ 2 = a[2]\ \underline{\text{then}}\ 1\ \underline{\text{else}}\ a[2]\ \underline{\text{fi}}) = a[2]\ \underline{\text{then}}\ 1$

   $\underline{\text{else}}\ a[\underline{\text{if}}\ 2 = a[2]\ \underline{\text{then}}\ 1\ \underline{\text{else}}\ a[2]\underline{\text{fi}}]\ \underline{\text{fi}}$.

   Observe that the last expression is semantically equal to $\underline{\text{if}}\ a[2] = 2$ $\underline{\text{then}}\ a[1]\ \underline{\text{else}}\ 1\ \underline{\text{fi}}$.

Using example 2, and anticipating the theorem stating the validity of (H) with the new definition of substitution, let us see what happens to the first example of section 1. Suppose we want to find out under which precondition we can be sure that the assignment a[a[2]] := 1 yields postcondition a[a[2]] = 1. (Above we noted that the identically true predicate is not sufficient.) We obtain the following instance of (H):

$$\{(a[a[2]] = 1)[1/a[a[2]]]\}\ a[a[2]] := 1\quad \{a[a[2]] = 1\}$$

or, using example 2 just given,

$$\{\underline{\text{if}}\ a[2] = 2\ \underline{\text{then}}\ a[1]\ \underline{\text{else}}\ 1\ \underline{\text{fi}} = 1\}$$
$$a[a[2]] := 1$$
$$\{a[a[2]] = 1\}$$

and it should be clear that the precondition if a[2] = 2 then a[1] = 1 else true fi does indeed imply that, after a[a[2]] := 1, the postcondition a[a[2]] = 1 holds.

Observe that Hoare's axiom does not enable us to find the (strongest) postcondition for precondition true, i.e., we are not yet in the position to give a formal derivation of the situation after a[a[2]] := 1 in this case. This will have to wait till section 4 which brings the extended version of Floyd's forward axiom.

3. SEMANTICS OF ASSIGNMENT AND THE VALIDITY OF HOARE'S AXIOM

In this section we define the meaning of the various programming constructs as introduced in section 2. Moreover, we define the notion of *validity* of a correctness formula, and we show that Hoare's axiom is valid.

The method used for our semantical definitions is in the spirit of the so-called "denotational semantics" as advocated by SCOTT & STRACHEY - e.g. in [5] -, which is best characterized as a systematic process of associating various mathematical objects - sets, functions and the like - with the linguistic constructs in our language. More specifically, we define a number of mappings which prescribe a value for the elements of each major syntactic class: integer expressions including variables, boolean expressions, and statements.

As our starting point we take three sets of values:

- A: a set of *addresses*, with elements denoted by α, $\alpha_1, \ldots$.
- I: a set of *integers*, with elements denoted by μ, $\nu, \ldots$.
- $\{T, F\}$: a set of truth-values, with the two elements T and F.

Furthermore, we give a name to the mappings from A to I:

- $\Sigma = (A \to I)$, is called the set of *stores* with elements denoted by σ, $\sigma_1, \ldots$.

Meaning will be provided to integer expressions in E by mapping them to I, to boolean expressions in P by mapping them to $\{T, F\}$ and to statements in S by mapping them to Σ. First, however, we have to clarify the role of the set A. Roughly, the idea is that we map each variable in V to an address in A - which mapping we call the *environment* ε - and then use the store σ to find the value ν stored in α. For simple variables this gives no problem, and we have the picture

$$x \xrightarrow{\varepsilon} \alpha \xrightarrow{\sigma} \nu \qquad .$$

where $x \in SV$, $\alpha \in A$, $\nu \in I$, $\varepsilon \in Env$ (the set of all environments) and $\sigma \in \Sigma$. For subscripted variables a⌈s] the situation is more complicated, as we shall discuss presently. Before doing this, we already mention that below we shall model the assignment x := t as an action which changes the store σ into σ' in the following way: For each $\alpha' \neq \alpha = \varepsilon(x)$, σ' remains the same as σ, but $\sigma'(x)$ is set to μ, where μ is the integer resulting from evaluating t in the (old) store σ.

Now what about ε for subscripted variables a[s]? We cannot directly define ε for such arguments. Instead, we first have to evaluate s, yielding an integer ν, and then we apply ε to the pair (a,ν). Moreover, in evaluating s we need the current environment and store in order to evaluate the variables occurring in s. Altogether, we obtain the situation defined below, in which we use the terminology of *left-hand-values* and *right-hand-values* as introduced originally in STRACHEY [6]. Let

- $Env = ((SV \cup (AV \times I)) \to A)$

i.e., each $\varepsilon \in Env$ is a mapping defined either on simple variables $x \in SV$, or on pairs (a,ν) with $a \in AV$ and $\nu \in I$. As a result, ε produces an address α. Furthermore, ε is required to satisfy the following uniqueness condition: $\varepsilon(x) = \varepsilon(y)$ iff $x \equiv y$, $\varepsilon(x) \neq \varepsilon(a,\nu)$ for all x, a and ν, and $\varepsilon(a,\nu) = \varepsilon(b,\mu)$ iff $a \equiv b$ and $\mu = \nu$.

- R: $(E \to (Env \times \Sigma \to I))$

  For each integer expression $s \in E$, R determines its *right-hand-value*, which is an integer and which depends on the current environment and store: $R(s)(\varepsilon,\sigma) = \nu$, for some $\nu \in I$.

- L: $(V \to (Env \times \Sigma \to A))$

  For each variable $v \in V$, L determines its *left-hand-value*, which is an address and which also depends on ε and σ: $L(v)(\varepsilon,\sigma) = \alpha$, for some $\alpha \in A$.

- T: $(P \to (Env \times \Sigma \to \{T, F\}))$

  For each boolean expression $p \in P$, T determines its value which is a truth-value, depending on ε and σ: $T(p)(\varepsilon,\sigma) \in \{T, F\}$.

The definitions of L, R and T are given in

DEFINITION 3.1 (Semantics of expressions).

1. $L(x)(\varepsilon,\sigma) = \varepsilon(x)$

   $L(a[s])(\varepsilon,\sigma) = \varepsilon(a,R(s)(\varepsilon,\sigma))$

2. $R(v)(\varepsilon,\sigma) = \sigma(L(v)(\varepsilon,\sigma))$

   $R(n)(\varepsilon,\sigma) = \nu$, where ν is the integer "suggested" by the integer constant n

   $R(s_1+s_2)(\varepsilon,\sigma) = plus\ (R(s_1)(\varepsilon,\sigma), R(s_2)(\varepsilon,\sigma))$, where *plus* has the usual mathematical meaning

   $R(s_1*s_2)(\varepsilon,\sigma)$ is defined similarly

   $$R(\underline{\text{if}}\ p\ \underline{\text{then}}\ s_1\ \underline{\text{else}}\ s_2\ \underline{\text{fi}}) = \begin{cases} R(s_1)(\varepsilon,\sigma), & \text{if } T(p)(\varepsilon,\sigma) = T \\ R(s_2)(\varepsilon,\sigma), & \text{if } T(p)(\varepsilon,\sigma) = F \end{cases}$$

3. $T(\underline{\text{true}})(\varepsilon,\sigma) = T$

   $T(\underline{\text{false}})(\varepsilon,\sigma) = F$

   $T(p_1 \wedge p_2)(\varepsilon,\sigma) = (T(p_1)(\varepsilon,\sigma)\ \&\ T(p_2)(\varepsilon,\sigma))$,
   where "&" has the usual meaning of conjunction of truth-values

   $T(p_1 \supset p_2)(\varepsilon,\sigma) = (T(p_1)(\varepsilon,\sigma) \Rightarrow T(p_2)(\varepsilon,\sigma))$,
   where "$\Rightarrow$" has the usual meaning of implication between truth-values

   $T(s_1 = s_2)(\varepsilon,\sigma) = (R(s_1)(\varepsilon,\sigma) = R(s_2)(\varepsilon,\sigma))$, where the right-most equality sign has the usual meaning of equality between integers

$T(s_1 > s_2)(\varepsilon,\sigma)$ is defined similarly.

Examples.

1. Let $\varepsilon(a,1) = \alpha_1$, $\varepsilon(a,2) = \alpha_2$, $\sigma(\alpha_1) = 2$, $\sigma(\alpha_2) = 1$. Let us, for convenience's sake, assume that 1,2,... are elements both of C and I. Then
   $R(a[a[1]])(\varepsilon,\sigma) = \sigma(L(a[a[1]])(\varepsilon,\sigma)) =$
   $\sigma(\varepsilon(a,R(a[1])(\varepsilon,\sigma))) = \sigma(\varepsilon(a,\sigma(L(a[1])(\varepsilon,\sigma)))) =$
   $\sigma(\varepsilon(a,\sigma(\varepsilon(a,R(1)(\varepsilon,\sigma))))) = \sigma(\varepsilon(a,\sigma(\varepsilon(a,1)))) =$
   $\sigma(\varepsilon(a,\sigma(\alpha_1))) = \sigma(\varepsilon(a,2)) = \sigma(\alpha_2) = 1.$
2. Let $\varepsilon(x) = \alpha_1$, $\varepsilon(y) = \alpha_2$, $\varepsilon(z) = \alpha_3$, $\sigma(\alpha_1) = 3$, $\sigma(\alpha_2) = 2$, $\sigma(\alpha_3) = 1$. Then
   $T(x+y=z)(\varepsilon,\sigma) = (R(x+y)(\varepsilon,\sigma) = R(z)(\varepsilon,\sigma)) =$
   $(\mathit{plus}(R(x)(\varepsilon,\sigma), R(y)(\varepsilon,\sigma)) = R(z)(\varepsilon,\sigma)) =$
   $(\mathit{plus}(\sigma(\varepsilon(x)), \sigma(\varepsilon(y))) = \sigma(\varepsilon(z))) =$
   $(\mathit{plus}(3,2)=1) = (5=1) = F.$

For the definition of the meaning of a statement, we need a new piece of notation:

DEFINITION 3.2 (Variants of the store).
Let $\sigma \in \Sigma$, $\alpha \in A$, $\nu \in I$. Then $\sigma\{\nu/\alpha\}$ is a new store defined as follows:

$$\sigma\{\nu/\alpha\}(\alpha') = \begin{cases} \nu, & \text{if } \alpha = \alpha' \\ \sigma(\alpha), & \text{if } \alpha \neq \alpha' \end{cases}.$$

Thus, $\sigma' = \sigma\{\nu/\alpha\}$ is like σ, but for the fact that $\sigma'(\alpha)$ is set to ν. In the sequel we shall use the easy lemma:

LEMMA 3.3.

a. $\sigma\{\nu_1/\alpha\}\{\nu_2/\alpha\} = \sigma\{\nu_2/\alpha\}$

b. $\sigma\{\nu_1/\alpha_1\}\{\nu_2/\alpha_2\} = \sigma\{\nu_2/\alpha_2\}\{\nu_1/\alpha_1\}$, *for* $\alpha_1 \neq \alpha_2$.

Proof. Obvious. □

We now define a mapping $M:(S\to(Env\times\Sigma\to\Sigma))$ which gives a meaning to statements.

DEFINITION 3.4 (Semantics of statements).

1. $M(v{:=}t)(\varepsilon,\sigma) = \sigma\{R(t)(\varepsilon,\sigma)/L(v)(\varepsilon,\sigma)\}$
2. $M(S_1;S_2)(\varepsilon,\sigma) = M(S_2)(\varepsilon,M(S_1)(\varepsilon,\sigma))$.

In words, an assignment statement v := t changes the store by updating its value in the left-hand-value of v with the right-hand-value of t.

Example. Let $\varepsilon(x) = \alpha_1$, $\varepsilon(y) = \alpha_2$, $\varepsilon(z) = \alpha_3$, $\sigma(\alpha_2) = 3$, $\sigma(\alpha_3) = 2$. We calculate $M(x{:=}y;\ y{:=}z;\ z{:=}x)(\varepsilon,\sigma)$ as follows:

$M(x{:=}y;\ y{:=}z;\ z{:=}x)(\varepsilon,\sigma) =$

$M(z{:=}x)(\varepsilon,\ M(x{:=}y;\ y{:=}z)(\varepsilon,\sigma)) =$

$M(z{:=}x)(\varepsilon,\ M(y{:=}z)(\varepsilon,\ M(x{:=}y)(\varepsilon,\sigma))) =$

$M(z{:=}x)(\varepsilon,\ M(y{:=}z)(\varepsilon,\sigma\{R(y)(\varepsilon,\sigma)/(L(x)(\varepsilon,\sigma)\})) =$

$M(z{:=}x)(\varepsilon,\ M(y{:=}z)(\varepsilon,\sigma\{3/\alpha_1\})) =$

$M(z{:=}x)(\varepsilon,\sigma\{3/\alpha_1\}\{2/\alpha_2\}) =$

$\sigma\{3/\alpha_1\}\{2/\alpha_2\}\{R(x)(\varepsilon,\sigma\{3/\alpha_1\}\{2/\alpha_2\}/$
$\qquad L(z)(\varepsilon,\sigma\{3/\alpha_1\}\{2/\alpha_2\})\} =$

$\sigma\{3/\alpha_1\}\{2/\alpha_2\}\{3/\alpha_3\}$,

and we see that we have indeed effectuated an interchange of the values of the simple variables y and z. (We leave to the reader the calculations for the case that y and z are replaced by a[a[1]] and a[a[2]].)

Finally, we define the notion of *validity*, which applies to boolean expressions p and to correctness formulae {p} S {q}.

DEFINITION 3.5 (Validity).

1. A boolean expression p is *satisfied* by a pair (ε,σ) iff $T(p)(\varepsilon,\sigma) = T$. A correctness formula {p} S {q} is satisfied by (ε,σ) iff $[T(p)(\varepsilon,\sigma) \Rightarrow T(q)(\varepsilon,\ M(S)(\varepsilon,\sigma))] = T$.
2. A boolean expression p is *valid* - denoted by $\models p$ - iff p is satisfied by all (ε,σ). A correctness formula {p} S {q} is valid - denoted by $\models$ {p} S {q} - iff {p} S {q} is satisfied by all (ε,σ).

Examples

1. The informal notion of semantic equality between integer expressions s and t, and between boolean expressions p and q - as used in section 2 - can now be stated formally as well: s = t (informally) iff $\models s = t$, and p = q (informally) iff $\models (p \supset q) \wedge (q \supset p)$.

2. Consider the following special case of ($\mathcal{H}$): $\models \{x = 0\}\ x := x + 1\ \{x = 1\}$. We show its validity, i.e., we show that $\forall\varepsilon,\sigma[\mathcal{T}(x=0)(\varepsilon,\sigma) \Rightarrow \mathcal{T}(x=1)(\varepsilon, \mathcal{M}(x:=x+1)(\varepsilon,\sigma))]$, or $\forall\varepsilon,\sigma[\mathcal{T}(x=0)(\varepsilon,\sigma)$ =) $\mathcal{T}(x=1)(\varepsilon,\sigma\{\mathcal{R}(x+1)(\varepsilon,\sigma)/\mathcal{L}(x)(\varepsilon,\sigma)\}]$. We distinguish two cases: (i) $\sigma(\varepsilon(x)) \neq 0$. Then $\mathcal{T}(x=0)(\varepsilon,\sigma) = \mathrm{F}$, and the implication is trivially true. (ii) $\sigma(\varepsilon(x)) = 0$. Then $\mathcal{T}(x=0)(\varepsilon,\sigma) = \mathrm{T}$, and we have $\mathcal{T}(x=1)(\varepsilon,\sigma\{\mathcal{R}(x+1)(\varepsilon,\sigma)/\mathcal{L}(x)(\varepsilon,\sigma)\}) = \mathcal{T}(x=1)(\varepsilon,\sigma\{1/\varepsilon(x)\}) = (\mathcal{R}(x)(\varepsilon,\sigma\{1/\varepsilon(x)\}) = \mathcal{R}(1)(\varepsilon,\sigma\{1/\varepsilon(x)\})) = (\sigma\{1/\varepsilon(x)\}(\varepsilon(x)) = 1) = (1=1) = \mathrm{T}$.

This closes our list of semantic definitions, and we are sufficiently prepared for the proof of our first theorem:

THEOREM 3.6 (*Validity of Hoare's axiom*). *For each* $p \in \mathcal{P}$, $v \in \mathcal{V}$ and $t \in \mathcal{E}$:

$$\models \{p[t/v]\}\ v := t\{p\}.$$

Proof. We have to show: $\forall\varepsilon,\sigma[\mathcal{T}(p[t/v])(\varepsilon,\sigma) \Rightarrow \mathcal{T}(p)(\varepsilon,\sigma\{\mathcal{R}(t)(\varepsilon,\sigma)/\mathcal{L}(v)(\varepsilon,\sigma)\})]$. Let us put $\alpha = \mathcal{L}(v)(\varepsilon,\sigma)$ and $\nu = \mathcal{R}(t)(\varepsilon,\sigma)$.It is sufficient to prove that $\forall\varepsilon,\sigma[\mathcal{T}(p[t/v])(\varepsilon,\sigma) = \mathcal{T}(p)(\varepsilon,\sigma\{\nu/\alpha\})]$. By a straightforward argument by induction on the structure of p we reduce this to the proof of $\forall\varepsilon,\sigma[\mathcal{R}(s[t/v])(\varepsilon,\sigma) = \mathcal{R}(s)(\varepsilon,\sigma\{\nu/\alpha\})]$. Again proceeding by induction, now on the structure of s, we have to show that $\forall\varepsilon,\sigma[\mathcal{R}(w[t/v])(\varepsilon,\sigma) = \mathcal{R}(w)(\varepsilon,\sigma\{\nu/\alpha\})]$. We consider only the case that $w \equiv a[s_1]$, $v \equiv a[s_2]$, for some $a \in \mathcal{AV}$ and $s_1, s_2 \in \varepsilon$, the various other cases being even simpler. So we have to show, by the definition of substitution:

$$\begin{aligned}&\forall\varepsilon,\sigma[\mathcal{R}(\underline{\text{if}}\ s_1[t/a[s_2]] = s_2\ \underline{\text{then}}\ t\ \underline{\text{else}}\ a[s_1[t/a[s_2]]]\ \underline{\text{fi}})(\varepsilon,\sigma)\\ &\quad=\\ &\quad\mathcal{R}(a[s_1])(\varepsilon,\sigma\{\nu/\alpha\})]\end{aligned} \tag{3.1}$$

or, by the semantic definitions, both

$$\begin{aligned}&\forall\varepsilon,\sigma[(\ \mathcal{R}(s_1[t/a[s_2]])(\varepsilon,\sigma) = \mathcal{R}(s_2)(\varepsilon,\sigma)) \Rightarrow\\ &\quad(\nu = \mathcal{R}(a[s_1])(\varepsilon,\sigma\{\nu/\alpha\}))]\end{aligned} \tag{3.2}$$

and

$$\forall\varepsilon,\sigma[(\mathcal{R}(s_1[t/a[s_2]])(\varepsilon,\sigma) \neq \mathcal{R}(s_2)(\varepsilon,\sigma)) \Rightarrow \tag{3.3}$$
$$(\mathcal{R}(a[s_1[t/a[s_2]]])(\varepsilon,\sigma) = \mathcal{R}(a[s_1])(\varepsilon,\sigma\{\nu/\alpha\}))]$$

First we consider (3.2). By the induction hypothesis, this reduces to: $\forall\varepsilon,\sigma[(\mathcal{R}(s_1)(\varepsilon,\sigma\{\nu/\alpha\}) = \mathcal{R}(s_2)(\varepsilon,\sigma)) \Rightarrow (\nu=\sigma\{\nu/\alpha\}(\varepsilon(a,\mathcal{R}(s_1)(\varepsilon,\sigma\{\nu/\alpha\}))))]$, and the desired result is obtained as follows: By the definition of α, $\alpha = \varepsilon(a,\mathcal{R}(s_2)(\varepsilon,\sigma))$. Now put $\alpha' = \varepsilon(a,\mathcal{R}(s_1)(\varepsilon,\sigma\{\nu/\alpha\}))$. We have $\sigma\{\nu/\alpha\}(\alpha') = \nu$ if $\alpha = \alpha'$, and $\alpha = \alpha'$ iff $\mathcal{R}(s_2)(\varepsilon,\sigma) = \mathcal{R}(s_1)(\varepsilon,\sigma\{\nu/\alpha\})$, using definition 3.2 and the uniqueness condition on ε.

Next, we prove (3.3). Let $\alpha \neq \alpha'$. We have

$$\mathcal{R}(a[s_1[t/a[s_2]]])(\varepsilon,\sigma) =$$
$$\sigma(\varepsilon(a,\mathcal{R}(s_1[t/a[s_2]])(\varepsilon,\sigma))) = \qquad \text{(ind. hyp.)}$$
$$\sigma(\varepsilon(a,\mathcal{R}(s_1)(\varepsilon,\sigma\{\nu/\alpha\})))$$

Also,

$$\mathcal{R}(a[s_1])(\varepsilon,\sigma\{\nu/\alpha\}) =$$
$$\sigma\{\nu/\alpha\}(\varepsilon(a,\mathcal{R}(s_1)(\varepsilon,\sigma\{\nu/\alpha\}))) = (\alpha\neq\alpha')$$
$$\sigma(\varepsilon(a,\mathcal{R}(s_1)(\varepsilon,\sigma\{\nu/\alpha\})))$$

Together, these two derivations establish (3.3), thus completing the proof of theorem 3.6. □

Remark. One might wonder about the role of the environment ε, which remains invariant throughout the calculations. Indeed, properties if statements consisting of sequences of assignments only, may be proven without taking the environment into account. However, we have preferred to include ε into our semantic model since the environment - store duality yields a better picture of actual implementations. Moreover, for the treatment of some of the more difficult programming concepts the environment is certainly necessary, witness the problems raised e.g. by the axiomatic treatment of parameter passing in HOARE & WIRTH [3].

4. AN EXTENDED VERSION OF FLOYD'S AXIOM

In section 1 we introduced Floyd's forward assignment axiom, which we now write as

(4.1) $\models \{p\}\ x := t\ \{\exists y[p[y/x] \wedge x = t[y/x]]\}$
(where y is a variable not occurring in p or t)

Properly speaking, this is syntactically illegal, since we have not yet introduced the ∃x-notation for boolean expressions. In order to remedy this, we extend the definitions of the previous sections with

DEFINITION 4.1 (Extended boolean expressions).

1. For any $p \in \mathcal{P}$ and $x \in \mathcal{SV}$, $\exists x[p]$ belongs to $\mathcal{P}$.
2. $\mathcal{T}(\exists x[p])(\varepsilon,\sigma) = \mathrm{T}$ iff there exists $\nu \in \mathcal{I}$ such that $\mathcal{T}(p)(\varepsilon,\sigma\{\nu/\varepsilon(x)\}) = \mathrm{T}$.

Employing this definition, it is not difficult to prove (4.1). We omit the proof here, since we do give the proof of the extension of (4.1) for assignments a[s] := t.

Before presenting this extension, we devote a brief discussion to the notions of weakest precondition and strongest postcondition. Consider the general correctness formula {p} S {q}. The reader should realize that, trivially, we have that both $\models \{\underline{\text{false}}\}$ S {p} and $\models \{p\}$ S $\{\underline{\text{true}}\}$. Usually, we are interested in pre- and postconditions which are best possible, in the sense of the following definition

DEFINITION 4.2 (Weakest precondition and strongest postcondition).
For any $p \in \mathcal{P}$ and $S \in \mathcal{S}$

1. wp(S,p) is the *weakest precondition* of S with respect to p iff it satisfies both
   (i) $\models \{wp(S,p)\}\ S\ \{p\}$
   (ii) For all r, if $\models \{r\}\ S\ \{p\}$, then $\models r \supset wp(S,p)$.
2. sp(S,p) is the *strongest postcondition* of S with respect to p iff it satisfies both

(i) $\models \{p\}\ S\ \{sp(S,p)\}$

(ii) For all r, if $\models \{p\}\ S\ \{r\}$, then $\models sp(S,p) \supset r$.

It is not difficult to prove that the precondition from HOARE'S assignment axiom is indeed the weakest:

THEOREM 4.3. *For each* $v \in V$, $t \in E$, $p[t/v]$ *is the weakest precondition of* $v := t$ *with respect to* p.

Proof. By theorem 3.6, p[t/v] satisfies clause 1(i) of definition 4.2. To prove 1(ii), assume $\models \{r\}\ v := t\ \{p\}$, for some r. By the validity definition we have

$$\forall \varepsilon,\sigma[\mathcal{T}(r)(\varepsilon,\sigma) \Rightarrow \mathcal{T}(p)(\varepsilon,\sigma\{\mathcal{R}(t)(\varepsilon,\sigma)/\mathcal{L}(v)(\varepsilon,\sigma)\}]$$

and by the proof of theorem 3.6, this is equivalent to

$$\forall \varepsilon,\sigma[\mathcal{T}(r)(\varepsilon,\sigma) \Rightarrow \mathcal{T}(p[t/v])(\varepsilon,\sigma)]$$

i.e. to

$$\models r \supset p\ [t/v]. \qquad \Box$$

We now return to the problem of extending (4.1). Let us, by way of introduction, first consider the case that $p \equiv$ <u>true</u> and $t \equiv n$, for some $n \in C$. We want to find the strongest q such that $\models \{\underline{\text{true}}\}\ a[s] := n\ \{q\}$. We list some possibilities for q which are successively stronger, until the desired strongest postcondition is found.

- $q \equiv$ <u>true</u>. This is clearly too weak.
- $q \equiv \exists z[a[z] = n]$, where z is used to store the old value of s. This is an improvement, but not yet strong enough.
- $q \equiv \exists y,z[a[s[y/a[z]]] = n]$, where y and z are variables not occurring in s, and where z is used to store the old value of s and y the old value of a[s]. This is already much better, but still insufficient.
- $q \equiv \exists y,z[a[s[y/a[z]]] = n \wedge s[y/a[z]] = z]$, where y and z do not occur in s. This is the desired solution. Again, z stores the old value of s and y of a[s]. However, we also have that $s[y/a[z]] = z$, as indicated:

Evaluating s in the new store (i.e., after performing the assignment $a[s] := n$) but making sure that a, indexed by the old value of s, is set to y, should result in the old value of s, i.e., in z.

Generalizing this argument to arbitrary p and t, we arrive at the desired extension of Floyd's axiom:

THEOREM 4.4. *(Extension of Floyd's axiom). For each* $s,t \in \mathcal{E}$, $p \in \mathcal{P}$, $a \in \mathcal{AV}$

$$\begin{aligned} &\models \{p\}\ a[s] := t\ \{\exists y,z \\ (4.2) \qquad &\qquad [p[y/a[z]] \wedge z = s[y/a[z]] \wedge a[z] = t[y/a[z]]]\} \end{aligned}$$

(where y *and* z *do not occur in* p,s *or* t)

Moreover, the postcondition in (4.2) *is strongest in the sense of definition* 4.2.

Before presenting the proof of the theorem, we first discuss the example of the beginning of section 1.

Example. Let $p \equiv (a[1]=2 \wedge a[2]=2)$, $s \equiv a[2]$, and $t \equiv 1$. We then obtain as instance of the postcondition of (4.2):

$$\begin{aligned} \exists y,z[&(a[1]=2 \wedge a[2]=2)[y/a[z]] \wedge \\ &z = a[2][y/a[z]] \wedge \\ &a[z] = 1[y/a[z]]] \end{aligned}$$

By the definition of substitution, this reduces to

$$\begin{aligned} \exists y,z[&\underline{\text{if}}\ 1 = z\ \underline{\text{then}}\ y\ \underline{\text{else}}\ a[1]\ \underline{\text{fi}} = 2 \wedge \\ &\underline{\text{if}}\ 2 = z\ \underline{\text{then}}\ y\ \underline{\text{else}}\ a[2]\ \underline{\text{fi}} = 2 \wedge \\ &z = \underline{\text{if}}\ 2 = z\ \underline{\text{then}}\ y\ \underline{\text{else}}\ a[2]\ \text{fi} \wedge \\ &a[z] = 1] \end{aligned}$$

Comparing the second and third boolean expression in the conjunction, we see that we must take $z = 2$, and the whole expression reduces to $\exists y[a[1] = 2 \wedge y = 2 \wedge y = 2 \wedge a[2] = 1]$, which is in turn equivalent to $a[1] = 2 \wedge a[2] = 1$. From this we obtain, among other facts, that $a[a[2]] = 2$, as desired.

As a slight variant of this example, we take $p \equiv \underline{true}$, $t \equiv 1$, and $s \equiv a[2]$ in (4.2), and obtain
$\models \{\underline{true}\}\ a[a[2]] := 1\ \{\exists y,z[z = \underline{if}\ z = 2\ \underline{then}\ y\ \underline{else}\ a[2]\ \underline{fi} \wedge a[z] = 1\}$.
The postcondition of this formula, which may be simplified to $a[2] = 1 \vee a[a[2]] = 1$, provides the answer to the question posed at the end of section 2.

Proof of theorem 4.4.

A. First we prove (4.2). Let $\alpha = \varepsilon(a,\mathcal{R}(s)(\varepsilon,\sigma))$, $\nu = \mathcal{R}(t)(\varepsilon,\sigma)$. By definitions 3.5 and 4.1, we have to show

$$\begin{aligned}&\forall \varepsilon,\sigma\ \exists \mu_1,\ \mu_2\\ &[\mathcal{T}(p)(\varepsilon,\sigma) \Rightarrow\\ &\ \mathcal{T}(p[y/a[z]] \wedge z = s[y/a[z]] \wedge\\ &\quad a[z] = t[y/a[z]])(\varepsilon,\sigma\{\nu/\alpha\}\{\mu_1/\varepsilon(y)\}\{\mu_2/\varepsilon(z)\})]\end{aligned} \tag{4.3}$$

Now take any (ε,σ), and let $\mu_1 = \sigma(\varepsilon(a,\mu_2))$, $\mu_2 = \mathcal{R}(s)(\varepsilon,\sigma)$. Furthermore, assume $\mathcal{T}(p)(\varepsilon,\sigma)$ and let us write σ''' for $\sigma\{\nu/\alpha\}\{\mu_1/\varepsilon(y)\}\{\mu_2/\varepsilon(z)\}$. Using $\alpha = \varepsilon(a,\mu_2)$ and $\sigma(\varepsilon(a,\mu_2)) = \mu_1$, we obtain $\sigma'''\{\mu_1/\varepsilon(a,\mu_2)\} = \sigma\{\mu_1/\varepsilon(y)\}\{\mu_2/\varepsilon(z)\}$, using the definitions and lemma 3.3. Let us abbreviate $\sigma\{\mu_1/\varepsilon(y)\}\{\mu_2/\varepsilon(z)\}$ to σ''. We show that the three consequences in (4.3) hold:

(i) $\mathcal{T}(p[y/a[z]])(\varepsilon,\sigma''') =$ (cf. the proof of the theorem 3.6)
$\mathcal{T}(p)(\varepsilon,\sigma'''\{\mathcal{R}(y)(\varepsilon,\sigma''')/\mathcal{L}(a[z])(\varepsilon,\sigma''')\}) =$
$\mathcal{T}(p)(\varepsilon,\sigma'''\{\mu_1/\varepsilon(a,\mu_2)\} =$
$\mathcal{T}(p)(\varepsilon,\sigma'') =$ (y and z do not occur in p)
$\mathcal{T}(p)(\varepsilon,\sigma)$, which holds by assumption.

(ii) $\mathcal{R}(s[y/a[z]])(\varepsilon,\sigma''') =$
$\mathcal{R}(s)(\varepsilon,\sigma'''\{\mu_1/\varepsilon(a,\mu_2)\}) =$
$\mathcal{R}(s)(\varepsilon,\sigma'') =$ (y and z do not occur in s)
$\mathcal{R}(s)(\varepsilon,\sigma) \quad = \mu_2 = \mathcal{R}(z)(\varepsilon,\sigma''')$.

(iii) $\mathcal{R}(t[y/a[z]])(\varepsilon,\sigma''') = \mathcal{R}(a[z])(\varepsilon,\sigma''')$. This is shown similar to (ii).

B. We prove that the postcondition of (4.2) is the strongest possible. Let us use the abbreviation $p_0 \equiv (p[y/a[z]] \wedge z = s[y/a[z]] \wedge a[z] = t[y/a[z]])$. Take any r such that $\models \{p\}\ a[s] := t\ \{r\}$ holds. We show that then

$\models \exists y,z[p_0] \supset r$ follows. Let α and ν be as in part A. Using the various definitions, what we have to show amounts to

$$\forall\varepsilon,\sigma[\mathcal{T}(p)(\varepsilon,\sigma) \Rightarrow \mathcal{T}(r)(\varepsilon,\sigma\{\nu/\alpha\})] \Rightarrow$$
$$\forall\varepsilon,\sigma[\mathcal{T}(\exists y,z[p_0])(\varepsilon,\sigma) \Rightarrow \mathcal{T}(r)(\varepsilon,\sigma)]$$

Equivalently, we have to show

$$\forall\varepsilon_0,\sigma_0[\forall\varepsilon,\sigma[\mathcal{T}(p)(\varepsilon,\sigma) \Rightarrow \mathcal{T}(r)(\varepsilon,\sigma\{\nu/\alpha\})] \wedge$$
$$\exists\mu_1,\mu_2[\mathcal{T}(p_0)(\varepsilon,\sigma_0\{\mu_1/\varepsilon(y)\}\{\mu_2/\varepsilon(z)\})] \Rightarrow \mathcal{T}(r)(\varepsilon_0,\sigma_0)]$$

or, writing $\sigma_0'' = \sigma_0\{\mu_1/\varepsilon(y)\}\{\mu_2/\varepsilon(z)\}$,

$$\forall\varepsilon_0,\sigma_0,\mu_1,\mu_2[\forall\varepsilon,\sigma[\mathcal{T}(p)(\varepsilon,\sigma) \Rightarrow \mathcal{T}(r)(\varepsilon,\sigma\{\nu/\alpha\})] \wedge \mathcal{T}(p_0)(\varepsilon_0,\sigma_0'')$$
$$\Rightarrow \mathcal{T}(r)(\varepsilon_0,\sigma_0)]$$

So take any $\varepsilon_0,\sigma_0,\mu_1,\mu_2$ and let us assume (*):
$\forall\varepsilon,\sigma[\mathcal{T}(p)(\varepsilon,\sigma) \Rightarrow \mathcal{T}(r)(\varepsilon,\sigma\{\nu/\alpha\})]$ and $\mathcal{T}(p_0)(\varepsilon_0,\sigma_0'')$. To show $\mathcal{T}(r)(\varepsilon_0,\sigma_0)$.
We use (*), with $\varepsilon = \varepsilon_0$, $\sigma = \sigma_0''\{\mu_1/\varepsilon_0(a,\mu_2\}$. From $\mathcal{T}(p_0)(\varepsilon_0,\sigma_0'')$ we obtain

(i) $\mathcal{T}(p[y/a[z]])(\varepsilon_0,\sigma_0'')$, or, equivalently,
$\mathcal{T}(p)(\varepsilon_0,\sigma_0''\{\mu_1/\varepsilon_0(a,\mu_2)\})$

(ii) $\mathcal{R}(z)(\varepsilon_0,\sigma_0'') = \mathcal{R}(s[y/a[z]])(\varepsilon_0,\sigma_0'')$, or equivalently,
$\mu_2 = \mathcal{R}(s)(\varepsilon_0,\sigma_0''\{\mu_1/\varepsilon_0(a,\mu_2)\}$

(iii) $\mathcal{R}(a[z])(\varepsilon_0,\sigma_0'') = \mathcal{R}(t[y/a[z]])(\varepsilon_0,\sigma_0'')$, or equivalently,
$\sigma_0''(\varepsilon_0(a,\mu_2)) = \mathcal{R}(t)(\varepsilon_0,\sigma_0\{\mu_1/\varepsilon_0(a,\mu_2)\})$.

From (i) and the definitions of ε and σ, we obtain that $\mathcal{T}(p)(\varepsilon,\sigma)$holds. Thus, using (*), $\mathcal{T}(r)(\varepsilon,\sigma\{\nu/\alpha\})$ follows. Also, we have, by the definition of α, $\alpha = \varepsilon(a,\mathcal{R}(s)(\varepsilon,\sigma)) = \varepsilon(a,\mathcal{R}(s)(\varepsilon,\sigma_0''\{\mu_1(\varepsilon_0(a,\mu_2)\})) = \varepsilon(a,\mu_2)$, where the last equality follows from (ii). Hence, $\sigma\{\nu/\alpha\} = \sigma_0''\{\mu_1/\varepsilon(a,\mu_2)\}\{\nu/\alpha\} = \sigma_0''\{\nu/\alpha\}$, by lemma 3.3. From (iii) we derive that $\sigma_0''(\alpha) = \sigma_0''(\varepsilon_0(a,\mu_2)) = \mathcal{R}(t)(\varepsilon_0,\sigma_0''\{\mu_1/\varepsilon_0(a,\mu_2)\}) = \mathcal{R}(t)(\varepsilon,\sigma) = \nu$. Thus, $\sigma_0''\{\nu/\alpha\} = \sigma_0''$. Finally, we have that $\mathcal{T}(r)(\varepsilon_0,\sigma_0) =$ (y and z not in r) $\mathcal{T}(r)(\varepsilon_0,\sigma_0'') = \mathcal{T}(r)(\varepsilon_0,\sigma_0''\{\nu/\alpha\}) = \mathcal{T}(r)(\varepsilon,\sigma\{\nu/\alpha\}$, and the desired result follows, since $\mathcal{T}(r)(\varepsilon,\sigma\{\nu/\alpha\})$was already shown to have the value T.

This completes the proof of part B and, therefore, of theorem 4.4. □

REFERENCES

[1] FLOYD, R.W., *Assigning meanings to programs*, in Proc. of a Symposium in Applied Mathematics Vol. 19 - Math. Aspects of Computer Science (J.T. Schwarz, ed.), p. 19-32. AMS (1967).

[2] HOARE, C.A.R., *An axiomatic basis for computer programming*, C.ACM, 12 (1969), p. 576-580.

[3] HOARE, C.A.R. & N. WIRTH, *An axiomatic definition of the programming language PASCAL*, Acta Inf. 2 (1973), p. 335-355.

[4] IGARASHI, S., R.L. LONDON & D.C. LUCKHAM, *Automatic Program Verification I: A Logical Basis and its Implementation*, Acta Inf. 4 (1975), p. 145-182.

[5] SCOTT, D. & C. STRACHEY, *Towards a mathematical semantics for computer languages*, in Proc. of the Symp. on Computers and Automata (J. Fox, ed.), p. 19-46, Polytechnic Inst. of Brooklyn (1971).

[6] STRACHEY, C., *Towards a formal semantics*, in Proc. IFIP Working Conf. on Formal Language Description Languages (T.B. Steel, Jr., ed.), p. 198-220, North-Holland Publ. Company (1966).