ma
the
ma
tisch

cen
trum

AFDELING INFORMATICA IW 47/75 SEPTEMBER

J.W. DE BAKKER

# FLOW OF CONTROL IN THE PROOF THEORY OF STRUCTURED PROGRAMMING

Prepublication

amsterdam 1975

stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 47/75 SEPTEMBER

J.W. DE BAKKER

# FLOW OF CONTROL IN THE PROOF THEORY OF STRUCTURED PROGRAMMING

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- category: 5.24

Flow of control in the proof theory of structured programming*)

by

J.W. de Bakker

KEY WORDS & PHRASES: *Flow of control, recursion, while and repeat statements, program correctness, axiomatic method, weakest preconditions, termination*

---

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

# FLOW OF CONTROL IN THE PROOF THEORY OF STRUCTURED PROGRAMMING

J.W. de Bakker
Mathematical Centre, Amsterdam


## ABSTRACT

The proof theory of structured programming insofar as concerned with flow of control is investigated. Various proof rules for the while, repeat-until and simple iteration statements - all essentially variants of Hoare's original while rule - are analyzed with respect to their soundness and adequacy. Next, a recently proposed proof rule for recursive procedures due to Dijkstra is - after correction - shown to be a simple instance of Scott's induction rule. Finally, Manna & Pnueli's rule for total correctness of the while statement is formally justified using the Hitchcock & Park theory of program termination based on well-founded relations.


## 0. INTRODUCTION

In this paper we investigate the proof theory of structured programming insofar as it is concerned with the control structure of programs. More specifically, we analyze

- a variety of proof rules for the while, repeat-until and simple iteration ([11]) statements;
- a recently proposed proof rule for recursion due to Dijkstra ([5]);
- Manna & Pnueli's rule ([12]) to prove *total* correctness of the while statement.

*Section 1* of the paper is preliminary and contains the notation - viewing programs as relations between states expressed with the aid of various relational operators -, the basic properties of (parameterless) recursive procedures such as the rule of computational (or Scott's) induction, and the notation for stating partial and total correctness using some auxiliary operators.
*Section 2* (proof rules for iteration): All proof rules - apart from one exception - are *sound*. None of them is fully *adequate*, i.e., none of them allows to prove *all* properties of the statement concerned. It is shown that, for deterministic programs, each proof rule fully characterizes terminating programs only. Also, an extension of the rules yielding full adequacy is proposed.
*Section 3*. Dijkstra's proof rule for recursive procedures which employs his "weakest preconditions" is analyzed. In the form given, the rule is incorrect, at least if our interpretation of his informally stated system is indeed the intended one. A modification of the rule leads to a correct version which is immediately obtained by Scott's induction.
*Section 4* (total correctness of the while statement): It is explained how Dijkstra's rule could be specialized to his "Fundamental Invariance Theorem for Repetition" which, at closer scrutiny, is nothing but a weaker version of Hoare's while rule. Next, we recall some of the theory as introduced in [6], where total correctness is proved using properties of well-founded relations. This enables us to prove the soundness of the rule of Manna & Pnueli which, though intuitively appealing, is not so easy to justify formally.

Our main aim with the paper is to show how the relational theory provides a unified framework for the analysis of a seemingly wide variety of rules. The theory allows one to see these rules in a better perspective, allowing one to compare and, in some cases, to correct or extend them. Instead of remaining isolated ideas, they obtain their due place as propositions in the general theory. An additional feature is the first application - albeit rather modest - of the very interesting theory of [6].

## 1. ITERATION AND RECURSION: NOTATION AND BASIC PROPERTIES

Details about the material in this section can be found e.g. in [1,2,3,4,6].

### 1.1. *Programs and relations*

In our (structured) language we have

- basic actions $A, A_1, \ldots$, and procedure calls $P, P_1, \ldots$ .
- construction rules combining statements $S_1$, $S_2$ and boolean p by
  - . sequential composition : $S_1;S_2$
  - . selection : <u>if</u> p <u>then</u> $S_1$ <u>else</u> $S_2$
  - . while statement : <u>while</u> p <u>do</u> S (or $p*S$, for short)
  - . repeat-until statement : <u>repeat</u> S <u>until</u> p (or $S*\bar{p}$, for short)
  - . simple iteration statement ([11]) : <u>loop</u> S; <u>while</u> $\bar{p}$:T <u>repeat</u> (or $S*\bar{p}*T$, for short (which, by definition, equals $S;\bar{p}*(T;S)$)).

Declarations for (parameterless, possibly recursive) procedures have the format $P \Leftarrow S[P]$, where $S[P]$ is any statement, which may have occurrences of P (e.g., $P \Leftarrow$ <u>if</u> p <u>then</u> $A_1;P$ <u>else</u> $A_2$).

Programs determine relations between states: We write xSy if statement S maps input x to output y. Properly speaking, we need here an *interpretation* of schemes S to relations $\hat{S}$, say. Rigour will be sacrificed to intuition (and brevity), and we *identify* programs and their corresponding input-output relations. Relational operators will then be used as counterpart of the various construction rules of the language. We use, for $V$ the set of states and $x,y,z \in V$:

- $S_1;S_2$ for composition: $xS_1;S_2y$ iff $\exists z[xS_1z \wedge zS_2y]$
- $\cup$, $\cap$, $\subseteq$ with the usual set-theorectical meaning
- $\Omega$ for the empty relation, $I = \{(x,x) \mid x \in V\}$ for the identity
- $S^* = I \cup S \cup S;S \cup \ldots = \bigcup_{i=0}^{\infty} S^i$
- small letters p,q,r, ... for subsets of I
- complementation, denoted by $\bar{\ }$, *only with respect to* I : $\bar{p} = I \backslash p$.

For each boolean p: $V \to$ {true,false}, for convenience's sake always assumed to be *total*, we introduce two relations $p,\bar{p}$, viz. $p = \{(x,x) \mid p(x) = \text{true}\}$, $\bar{p} = \{(x,x) \mid p(x) = \text{false}\}$. This double use of p - as boolean and as a relation - is admittedly ambiguous, but it pays off below. We shall freely use relational identities such as $I;S = S$, $\Omega \cup S = S$, $p \cap \bar{p} = \Omega$, $p \cup \bar{p} = I$, $p;q = p \cap q$, etc. Using the relational operators, with ";" having priority over "$\cup$", we can write:

| | | |
|---|---|---|
| $p;S_1 \cup \bar{p};S_2$ | for | $\underline{\text{if}}$ p $\underline{\text{then}}$ $S_1$ $\underline{\text{else}}$ $S_2$ |
| $(p;S)^*;\bar{p}$ | for | $p*S$ |
| $S;(\bar{p};S)^*;p$ | for | $S*\bar{p}$ |
| $S;(\bar{p};T;S)^*;p$ | for | $S*\bar{p}*T$ |

### 1.2. *Recursive procedures*

We summarize the basic properties of recursive procedures we shall need below. Let S[P] be any statement with possible occurrences of P, and, for any S', let S[S'] denote the result of replacing all occurrences of P in S by S'. Each S has the following properties:

1. *Monotonicity*: If $S_1 \subseteq S_2$ then $S[S_1] \subseteq S[S_2]$.
2. *Continuity*: Let $S_0 \subseteq S_1 \subseteq \dots$ . Then $S[\cup_i S_i] = \cup_i\ S[S_i]$.

We now assume the declaration $P \Leftarrow S[P]$.

3. *Union theorem*: Let, for any T, $S^0[T] = T$, $S^{i+1}[T] = S[S^i[T]]$. Then $P = \cup_{i=0}^{\infty} S^i[\Omega]$.
4. *Least fixed point property*: $P = S[P]$ and, for any T, if $S[T] = T$ (or even $S[T] \subseteq T$) then $P \subseteq T$.
5. *Computational (or Scott's) induction*: Let $T_1[P]$, $T_2[P]$ be any two statements, and let $T_1[X]$, $T_2[X]$ result from these by replacing P be the "program variable" X. Assume that a and b are both satisfied:
   a. $T_1[\Omega] \subseteq T_2[\Omega]$.
   b. For all X, if $T_1[X] \subseteq T_2[X]$ then $T_1[S[X]] \subseteq T_2[S[X]]$.
   Then we may infer that
   c. $T_1[P] \subseteq T_2[P]$.
   *Example*. Let $T_1[P] = p;P$ and $T_2[P] = P;q$. Assume
   a. $p;\Omega \subseteq \Omega;q$ (note that this is trivially satisfied)
   b. For all X, if $p;X \subseteq X;q$ then $p;S[X] \subseteq S[X];q$.
   Then we may infer that
   c. $p;P \subseteq P;q$.
   (Cf. the proof rule for recursive procedures in [8] and its explanation in [13].)

### 1.3. *Partial and total correctness*

S is *partially correct* with respect to p,q iff $\forall x,y[p(x) \wedge xSy \to q(y)]$ or, in our formalism, $p;S \subseteq S;q$ (or {p} S {q}, as in [7]). S is *totally correct* with respect to p,q iff $\forall x[p(x) \to \exists y[xSy \wedge q(y)]]$.
We introduce two new operators between a statement S and a boolean p, viz.

$$(S \circ p)(x) \overset{df.}{\Longleftrightarrow} \exists y[xSy \wedge p(y)]$$

$$(S \to p)(x) \overset{df.}{\Longleftrightarrow} \forall y[xSy \to p(y)]$$

We have, e.g.,

(i) S is totally correct with respect to p,q iff $p \subseteq S \circ q$

(ii) $S \to p = \overline{S \circ \bar{p}}$

(iii) $S \circ (p \cup q) = (S \circ p) \cup (S \circ q)$, $S_1 \to (S_2 \to p) = (S_1;S_2) \to p$, and, for S a *function* (i.e., $\forall x,y,z[xSy \wedge xSz \to y=z]$): $S \circ (p \cap q) = (S \circ p) \cap (S \circ q)$, $S \to (p \cup q) = (S \to p) \cup (S \to q)$, etc.

## 2. PROOF RULES FOR ITERATION

We discuss the soundness and adequacy of a number of proof rules for the while, repeat-until and simple iteration statements. Apart from one exception, the rules are all easily seen to be sound. None of them is fully adequate, however, and it is shown how to extend them to achieve adequacy.

### 2.1. *Rules for the while statement*

In the literature we encountered the following four versions (taking a few liberties with the notation):

$$\mathcal{W}_1([7]): \frac{\{u \wedge p\}\ S\ \{u\}}{\{u\}\ p*S\ \{u \wedge \bar{p}\}}$$

(In words, if the assertion u is an invariant of S - under the additional assumption that p holds - then u is an invariant of p*S. Moreover, upon termination of p*S, $\bar{p}$ holds.)

$$\mathcal{W}_2([12]): \frac{\{u \wedge p\}\ S\{u\},\ u \wedge \bar{p} \supset v}{\{u\}\ p*S\ \{v\}}$$

$$\mathcal{W}_3([9]): \frac{\{w\}\ S\{u\},\ u \supset \text{if } p \text{ then } w \text{ else } v}{\{u\}\ p*S\ \{v\}}$$

$$\mathcal{W}_4([8]): \frac{\{u\}\ S\{w\},\ w \supset \text{if } p \text{ then } u \text{ else } v}{\{u\}\ p*S\ \{v\}}$$

In the relational notation these rules are written as

$\mathcal{W}_1$: $\forall u[u;p;S \subseteq S;u \Rightarrow u;p*S \subseteq p*S;\bar{p};u]$

$\mathcal{W}_2$: $\forall u,v[u;p;S \subseteq S;u$ and $\bar{p};u \subseteq v \Rightarrow u;p*S \subseteq p*S;v]$

$\mathcal{W}_3$: $\forall u,v,w[w;S \subseteq S;u$ and $u \subseteq p;w \cup \bar{p};v \Rightarrow u;p*S \subseteq p*S;v]$

$\mathcal{W}_4$: $\forall u,v,w[u;S \subseteq S;w$ and $w \subseteq p;u \cup \bar{p};v \Rightarrow u;p*S \subseteq p*S;v]$

to which we add (cf. [3]):

$\mathcal{W}_5$: $\forall u,v[\exists w[u \subseteq w,\ w;p;S \subseteq S;w,\ w;\bar{p} \subseteq v] \Rightarrow u;p*S \subseteq p*S;v]$

Of course, $\mathcal{W}_5$ is nothing but the inductive assertion method: In order to show that p*S is partially correct with respect to u,v, try to find intermediate w satisfying the three "verification conditions" $u \subseteq w$, $w;p;S \subseteq S;w$, and $w;\bar{p} \subseteq v$.

Our analysis of the five rules is summarized in the following three lemmas:

LEMMA 2.1 (Soundness).
a. $\mathcal{W}_1$, $\mathcal{W}_2$, $\mathcal{W}_3$ and $\mathcal{W}_5$ are sound.
b. $\mathcal{W}_4$ is not sound.

PROOF.
a. Straightforward from the fact that $p*S = (p;S)^*;\bar{p} = (\cup_i (p;S)^i);\bar{p}$, by applying induction on i.
b. Taking $v = w = p = S = \Omega$, and $u = I$, and using the fact that $\Omega*\Omega = I$, $\mathcal{W}_4$ yields the contradiction that $I \subseteq \Omega$. □

It is maybe not immediately clear how to understand the notion of *adequacy* of a proof rule. We take the following approach: Let $\mathcal{W}_1(p,S,X)$, or $\mathcal{W}_1(X)$ when p and S are understood, be defined as:

$\mathcal{W}_1(p,S,X)$: $\forall u[u;p;S \subseteq S;u \Rightarrow u;X \subseteq X;\bar{p};u]$

Then we call $\mathcal{W}_1(p,S,X)$ *adequate* with respect to the while statement p*S iff for all X, if $\mathcal{W}_1(p,S,X)$ holds, then X = p*S.

From now on, we omit reference to p and S in our notation. Also $\mathcal{W}_i(X)$, i = 2,3,5 are defined similarly to $\mathcal{W}_1(X)$. We then have

LEMMA 2.2 (Adequacy). For all X:
a. $\mathcal{W}_1(X) \Leftrightarrow \mathcal{W}_2(X) \Leftrightarrow \mathcal{W}_3(X) \Leftrightarrow \mathcal{W}_5(X)$
b. $\mathcal{W}_1(X) \Rightarrow [X \subseteq p*S]$. Hence, $\mathcal{W}_i(X) \Rightarrow [X \subseteq p*S]$, i = 2,3,5.
c. None of the $\mathcal{W}_i(X)$, i = 1,2,3,5, is adequate.

PROOF.
a. We show that $\mathcal{W}_1(X) \overset{(i)}{\Rightarrow} \mathcal{W}_5(X) \overset{(ii)}{\Rightarrow} \mathcal{W}_2(X) \overset{(iii)}{\Rightarrow} \mathcal{W}_3(X) \overset{(iv)}{\Rightarrow} \mathcal{W}_1(X)$.
(i) Assume $\mathcal{W}_1(X)$ and the assumptions of $\mathcal{W}_5(X)$ for some $u_0$, $v_0$, $w_0$. Then the assumption of $\mathcal{W}_1(X)$ is satisfied for $w_0$, hence $w_0;X \subseteq X;\bar{p};w_0$ follows. Since $u_0 \subseteq w_0$ and $\bar{p};w_0 \subseteq v_0$, the result $u_0;X \subseteq X;v_0$ follows, thus establishing $\mathcal{W}_5(X)$.
(ii) Assume $\mathcal{W}_5(X)$ and the assumptions of $\mathcal{W}_2(X)$ for

$u_0, v_0$. Taking $w_0 = u_0$, we see that the assumptions of $W_5(X)$ are satisfied for $u_0$, $v_0$, $w_0$, hence $u_0;X \subseteq X;v_0$ follows, thus establishing $W_2(X)$.

(iii),(iv) Similar.

b. (This proof is a slight variant of an unpublished argument due to Scott.) First we show: For any R, S, if $\forall u,v[u;R \subseteq R;u \Rightarrow u;S \subseteq S;u]$ then $S \subseteq R^*$: Take an arbitrary state $x_0$, and define $u_0(x) \overset{df.}{=} x_0R^*x$. It is easy to check that $u_0;R \subseteq R;u_0$ holds. Thus $u_0;S \subseteq S;u_0$ follows, i.e. $\forall x,y[x_0R^*x \wedge xSy \rightarrow x_0R^*y]$. In particular, $\forall x,y[x=x_0 \wedge xSy \rightarrow x_0R^*y]$, i.e., $\forall y[x_0Sy \rightarrow x_0R^*y]$. Since $x_0$ was arbitrary, $S \subseteq R^*$ follows. Applying this result, from $W_1(X)$ we obtain that $X \subseteq (p;S)^*$. Also, taking $u = I$ in $W_1(X)$ yields $X \subseteq X;\bar{p}$. Combining the two inclusions for X yields $X \subseteq (p;S)^*;\bar{p} = p*S$.

c. Observe that, e.g. $W_i(\Omega)$ holds for each i = 1,2,3,5. Only from additional requirements, e.g. that X be total ($\forall x \exists y[xXy]$) and all our relations be functions (i.e., if we restrict ourselves to deterministic programs) can we infer from $X \subseteq p*S$ that $X = p*S$. □

We now show how to extend the rule $W_5(X)$ to a new rule which *is* adequate.

LEMMA 2.3 (Extended while rule).
Let $W_5^{\Leftrightarrow}(X)$ be defined as:

$$W_5^{\Leftrightarrow}(X) : \forall u,v[\exists w[u \subseteq w,\ w;p;S \subseteq S;w,\ w;\bar{p} \subseteq v] \Leftrightarrow u;X \subseteq X;v]$$

Then $W_5^{\Leftrightarrow}(X)$ iff $X = p*S$.

PROOF.
(If). We show that $W_5^{\Leftrightarrow}(p*S)$ holds. $\Rightarrow$ is lemma 2.1, part a. $\Leftarrow$ follows by taking, for some given $u_0, v_0$, the (inductive assertion) $w_0$ as $w_0 \overset{df.}{=} (p;S)^* \circ u_0$. It is easily checked that $w_0$ satisfies the three verification conditions.
(Only if). We use the auxiliary result: For all R,S : $\forall u,v[u;R \subseteq R;v \Rightarrow u;S \subseteq S;v]$ iff $S \subseteq R$. The proof of this is left to the reader. Now assume $W_5^{\Leftrightarrow}(X)$, and take any $u_0, v_0$. We have, using $W_5^{\Leftrightarrow}(X)$ and $W_5^{\Leftrightarrow}(p*S)$ respectively: $u_0;X \subseteq X;v_0 \Leftrightarrow \exists w[u_0 \subseteq w,\ w;p;S \subseteq S;w,\ w;\bar{p} \subseteq v_0] \Leftrightarrow u_0;p*S \subseteq p*S;v_0$. Thus we obtain: $\forall u,v[u;X \subseteq X;v \Leftrightarrow u;p*S \subseteq p*S;v]$, and $X = p*S$ follows by the auxiliary result. □

## 2.2. *Rules for the repeat-until statement*

Similar results as in 2.1 are obtained for the repeat-until statement. We consider

$R_1$([14]): $\forall u,v[(u \cup \bar{p};v);S \subseteq S;v \Rightarrow u;S*\bar{p} \subseteq S*\bar{p};v]$

$R_2$([10]): $\forall u,v[u;S \subseteq S;v$ and $\bar{p};v \subseteq u \Rightarrow u;S*\bar{p} \subseteq S*\bar{p};v]$

$R_3$([9]): $\forall u,v,w[w;S \subseteq S;u$ and $u \subseteq p;v \cup \bar{p};w \Rightarrow u;S*\bar{p} \subseteq S*\bar{p};v]$

$R_4$([3]): $\forall u,v[\exists w[u;S \subseteq S;w,\ w;\bar{p};S \subseteq S;w,\ w;p \subseteq v] \Rightarrow u;S*\bar{p} \subseteq S*\bar{p};v]$.

We have (notation as in section 2.1):

LEMMA 2.4.
a. $R_1$, $R_2$, $R_3$ and $R_4$ are sound.
b. For all X, $R_1(X) \Leftrightarrow R_2(X) \Leftrightarrow R_3(X) \Leftrightarrow R_4(X)$.
c. For all X, $R_i(X) \Rightarrow [X \subseteq S*\bar{p}]$, i = 1,2,3,4.
d. For all X, $R_4^{\Leftrightarrow}(X)$ iff $X = S*\bar{p}$.

PROOF. Similar to the proofs in section 2.1. □

## 2.3. *Rules for the simple iteration statement*

We consider

$S_1$([11]): $\forall u,v[u;S \subseteq S;v$ and $v;\bar{p};T \subseteq T;u \Rightarrow u;S*\bar{p}*T \subseteq S*\bar{p}*T;v]$

$S_2$([3]): $\forall u,v[\exists w[u;S \subseteq S;w,\ w;\bar{p};T;S \subseteq T;S;w,\ w;p \subseteq v] \Rightarrow u;S*\bar{p}*T \subseteq S*\bar{p}*T;v]$.

We have

LEMMA 2.5.
a. $S_1$ and $S_2$ are sound
b. For all X, $S_1(X) \Leftrightarrow S_2(X)$
c. For all X, $S_i(X) \Rightarrow [X \subseteq S*\bar{p}*T]$, i = 1,2.
d. For all X, $S_2^{\Leftrightarrow}(X)$ iff $X = S*\bar{p}*T$.

PROOF. Similar to the proofs in section 2.1. In the proof of 2.5c we use the auxiliary result: For all R, S, T, if $\forall u,v[u;R \subseteq R;v$ and $v;S \subseteq S;u \Rightarrow u;T \subseteq T;v]$ then $T \subseteq (R;S)^*;R$. To show this, take $u_0(x) \overset{df.}{=} x_0(R;S)^*x$, and $v_0(x) \overset{df.}{=} x_0(R;S)^*;Rx$, etc. □

# 3. A PROOF RULE FOR RECURSION

*In sections 3 and 4 all programs are assumed deterministic, i.e., all relations are functions.*

## 3.1. *Weakest preconditions*

We quote from [5]: "We consider the semantics of a program S fully determined when we can derive for any postcondition q satisfied by the final state, the weakest precondition that for this purpose should be satisfied by the initial state. We regard this weakest precondition as a function of the postcondition q, and denote it by fS(q)." Though not stated in this quotation, the rest of [5] makes it clear that Dijkstra is only interested in *total* correctness. Consider once more its formulation in our notation: $\forall x[p(x) \rightarrow \exists y[xSy \wedge q(y)]]$, or, using the $\circ$-operator: $\forall x[p(x) \rightarrow (S\circ q)(x)]$. Thus, we see that whatever condition p guarantees total correctness with final q, such p always implies $S\circ q$. Hence, the *weakest* such p is nothing but $S\circ q$ itself, which we therefore propose to identify with fS(q). Our interpretation is supported by (i) All basic properties and additional rules from [5] are provable for $S\circ q$. (ii) The main theorem of [5] is, after correction, also provable.

We now give a selection of the basic properties of fS(q) as mentioned in [5]:

$\mathcal{D}_1$: $p = q$ implies $fS(p) = fS(q)$, i.e., $p = q \Rightarrow S\circ p = S\circ q$.

$\mathcal{D}_2$: $fS(f) = f$ (f the identically false predicate), i.e., $S\circ\Omega = \Omega$.

$\mathcal{D}_3$: $fS(p \cap q) = fS(p) \cap fS(q)$, i.e., $S\circ(p\cap q) = (S\circ p) \cap (S\circ q)$. Similarly for $\cup$.

$\mathcal{D}_4$: $f(S_1;S_2)(p) = fS_1(fS_2(p))$, i.e., $(S_1;S_2)\circ p = S_1\circ(S_2\circ p)$.

As an example, we exhibit the proof of $\mathcal{D}_4$:
For all x, we have
$((S_1;S_2)\circ p)(x)$ iff (def. "$\circ$")
$\exists y[xS_1;S_2y \wedge p(y)]$ iff (def. ";")
$\exists y[\exists z[xS_1z \wedge zS_2y] \wedge p(y)]$ iff
$\exists y,z[xS_1z \wedge zS_2y \wedge p(y)]$ iff
$\exists z[xS_1z \wedge \exists y[zS_2y \wedge p(y)]]$ iff (def. "$\circ$")
$\exists z[xS_1z \wedge (S_2\circ p)(z)]$ iff (def. "$\circ$")
$(S_1\circ(S_2\circ p))(x)$.
The other proofs are equally simple.

## 3.2. *The Fundamental Invariance Theorem for Recursive Procedures (F.I.T.R.P.).*

We quote from [5]: "Consider a text, called H", of the form H": ...H'...H'...H'..., to which corresponds a predicate transformer fH", such that for a specific pair of predicates q and r, the assumption $q \subseteq fH'(r)$ is a sufficient assumption about fH' for proving $q \subseteq fH''(r)$. In that case, the recursive procedure H defined by <u>proc</u> H;...H...H...H...<u>corp</u> enjoys the property that $q \cap fH(I) \subseteq fH(r)$". Thus, the theorem reads: From
a. If $q \subseteq fH'(r)$ then $q \subseteq fH''(r)$
one may infer that

b. $q \cap fH(I) \subseteq fH(r)$.
In this form the theorem is incorrect. Take $q = I$ and $r = \Omega$. Then we obtain, using $\mathcal{D}_2$ above: From
a'. If $I \subseteq \Omega$ then $I \subseteq \Omega$
one may infer that
b'. $I \cap fH(I) \subseteq \Omega$.
Since a' is always satisfied we obtain that, for arbitrary procedure H, $fH(I)$ (or $H \circ I$) $= \Omega$, i.e., H is nowhere defined, which is absurd. Next, we propose a modified version: From
a". If $q \cap fH'(I) \subseteq fH''(r)$ then $q \cap fH''(I) \subseteq fH''(r)$
one may infer that
b". $q \cap fH(I) \subseteq fH(r)$.
Proof of this version: First we show that we can rewrite the inclusion $q \cap fH(I) \subseteq fH(r)$ as the inclusion $q;H \subseteq H;r$. We have $\forall x[q(x) \wedge \exists y[xHy \wedge I(y)] \rightarrow \exists z[xHz \wedge r(z)]]$ iff $\forall x,y[q(x) \wedge xHy \rightarrow \exists z[xHz \wedge r(z)]]$ iff (H a function) $\forall x,y[q(x) \wedge xHy \rightarrow r(y)]$ iff $q;H \subseteq H;r$.
Applying the same rewriting for H' and H" we obtain: From
a"'. If $q;H' \subseteq H';r$ then $q;H'' \subseteq H'';r$
one may infer that
b"'. $q;H \subseteq H;r$.
We now use that $H'' = \ldots H' \ldots H' \ldots H' \ldots = S[H']$, say, and that $H \Leftarrow S[H]$. (Remember that H is the recursive procedure declared by <u>proc</u> H; ...H...H...H... <u>corp</u>, i.e., <u>proc</u> H; S[H] <u>corp</u>, or $H \Leftarrow S[H]$ in our notation.)
We thus obtain as next step: From
$a^{iv}$. If $q;H' \subseteq H';r$ then $q;S[H'] \subseteq S[H'];r$
one may infer that
$b^{iv}$. $q;H \subseteq H;r$.
Finally, we apply a renaming of the program and procedure variables and obtain: For P satisfying $P \Leftarrow S[P]$, from
$a^{v}$. If $q;X \subseteq X;r$ then $q;S[X] \subseteq S[X];r$
one may infer that
$b^{v}$. $q;P \subseteq P;r$
and this is nothing but the special case of Scott's induction rule mentioned at the end of section 1.2. □

## 4. TOTAL CORRECTNESS OF THE WHILE STATEMENT

In this section we first explain how Dijkstra's theorem could be specialized to a correct rule for the while statement, which turns out to be nothing but a weaker version of $\mathcal{W}_1$ from section 2.1. Next, we briefly review some of the results of Hitchcock & Park [6] on proving program termination using the notion of well-founded relation, and then finally justify the proof rule for total correctness due to Manna & Pnueli [12].

### 4.1. *The Fundamental Invariance Theorem for Repetition (F.I.T.R)*

In [5], it is asserted that: From
a. If $q \cap p \subseteq fS(q)$
one may infer that
b. $q \cap f(p{*}S)(I) \subseteq f(p{*}S)(q \cap \bar{p})$.
Using our interpretation of $fS(q)$, and the same rewriting argument as applied in section 3.2, we obtain instead: From
a'. $q;p \subseteq S \circ q$
one may infer that
b'. $q;p{*}S \subseteq p{*}S;q;\bar{p}$.
Since, clearly, if $q;p \subseteq S \circ q$ then $q;p;S \subseteq S;q$ (i.e., if S is totally correct with respect to $p \cap q$ and q, then S is partially correct with respect to $p \cap q$ and q) we see that the F.I.T.R. is just a weaker version of $\mathcal{W}_1$.

The derivation of the (correct) F.I.T.R. from the (incorrect) F.I.T.R.P. is based on the alleged equivalence of a) $q \cap p \subseteq fS(q)$, and a") if $q \subseteq fH'(q \cap \bar{p})$ then $q \subseteq (q \cap \bar{p}) \cup (p \cap fS(fH'(q \cap \bar{p})))$. One easily sees that though indeed a) $\Rightarrow$ a"), it is not true that a") $\Rightarrow$ a): Take $p = I$ to obtain a counterexample.

### 4.2. *Termination proofs according to Hitchcock & Park*

We quote some of the results of [6] to be used here.

DEFINITION 4.1. A relation R is well-founded in $x = x_0$ iff there does not exist an infinite sequence $x_0 R x_1 R x_2 \ldots$ .

DEFINITION 4.2. For $T\lceil X \rceil$ any statement which is monotonic in X, let $\mu X \lceil T \lceil X \rceil \rceil$ denote the least fixed point of $T \lceil X \rceil$, and $\nu X \lceil T \lceil X \rceil \rceil$ its greatest fixed point (note that both exist according to the Knaster-Tarski theorem).

LEMMA 4.3 ([6]). R is well-founded in x iff $\mu X \lceil R \rightarrow X \rceil (x)$ holds.

PROOF. First we show
(i) For all x, $\nu X \lceil R \circ X \rceil (x)$ iff there exist $x_0 = x$, $x_1, x_2, \ldots$ such that $x_0 R x_1 R x_2 \ldots$, i.e., iff R is *not* well-founded in x.
Let $r \overset{df.}{=} \nu X \lceil R \circ X \rceil$.
(Only if). Let $x = x_0$, and assume $r(x_0)$. Since $r = R \circ r$, by the definition of "$\circ$" there exists $x_1$ such that $x_0 R x_1 \wedge r(x)$. Similarly, there exists $x_2$ such that $x_1 R x_2 \wedge r(x_2)$, etc. Hence $x_0 R x_1 R x_2 \ldots$ .
(If). Let the predicate s be defined by: s(x) iff there exists an infinite R-sequence starting in x. Then $s \subseteq R \circ s$. By the definition of r, then $s \subseteq r$.
Furthermore, we need
(ii) $R \rightarrow X = \overline{R \circ \bar{X}}$ (section 1.3)
(iii) $\mu X \lceil T \lceil X \rceil \rceil = \overline{\nu X \lceil \overline{T \lceil \bar{X} \rceil} \rceil}$ (direct from the definitions).
Combining (i), (ii) and (iii) yields that R is well-founded in x iff $\mu X \lceil R \rightarrow X \rceil (x)$ holds. □

Next, we consider the question: For what reason could the while statement $p{*}S$ fail to deliver a value for some argument $x = x_0$. Either a) The sequence $x_0\ p;S\ x_1\ p;S\ x_2 \ldots$ can be continued ad inf., or b) There exists $x_n$ such that $x_0\ p;S\ x_1\ p;S\ x_2 \ldots x_{n-1}\ p;S\ x_n$, with $x_n$ satisfying $p \cap \overline{S \circ I}$ (i.e., $x_n$ satisfies the test of the loop but S is undefined in $x_n$). From a) and b) together we see that $p{*}S$ does terminate properly iff the relation $p;S \cup p;\overline{S \circ I}$ is well-founded. Thus we obtain

THEOREM 4.4 ([6]). $p{*}S$ terminates properly for all x iff $\mu X \lceil p;S \cup p;\overline{S \circ I} \rightarrow X \rceil = I$.

### 4.3. *A justification of the Manna & Pnueli rule*

First we apply theorem 4.4 to obtain an extension of $\mathcal{W}_1$ to total correctness.

LEMMA 4.5. From
a. $u;p \subseteq S \circ u$, and
b. S is well-founded
one may infer that
c. $u \subseteq \mu X \lceil p;S \cup p;\overline{S \circ I} \rightarrow X \rceil$

(In words, if u is an invariant of the well-founded S guaranteeing termination of S - under the additional assumption that p holds - then u implies proper termination of $p{*}S$.)

PROOF. We use the auxiliary result that for R a *function*, $\mu X \lceil R \rightarrow X \rceil = \bigcup_i (R^i \rightarrow \Omega)$. (By [6], this does not hold for arbitrary R, but we recall the restriction stated at the beginning of section 3, that all programs considered be deterministic.) Therefore, we can apply Scott's induction in the following way: Let $r \overset{df.}{=} \mu X \lceil p;S \cup p;\overline{S \circ I} \rightarrow X \rceil$. We shall show c'): If $u;X \subseteq r$, then $u;(S \rightarrow X) \subseteq r$. Once c') has been established, we conclude, by Scott's rule, that $u;\mu X \lceil S \rightarrow X \rceil \subseteq r$, and, since by assumption b) we have $\mu X \lceil S \rightarrow X \rceil = I$, the desired result c) follows. In order to prove c'), assume $u;X \subseteq r$, and u(y) and $(S \rightarrow X)(y)$ for some y. We must show that the r(y), or, by the fixed point property, that $(p;S \cup p;\overline{S\circ I} \rightarrow r)(y)$, or, equivalently, that both $(p;S\rightarrow r)(y)$ and $(p;\overline{S\circ I}\rightarrow r)(y)$. To show $(p;S\rightarrow r)(y)$, we assume $y\ p;S\ z$, and show that then $r(z)$. Since $u(y)$ and $p(y)$, by assumption a) we have that $ySt$ and $u(t)$ for some $t$. Since $S$ is a function, $t = z$. Since $(S\rightarrow X)(y)$, also $X(z)$. From $u(z)$ and $X(z)$, and since $u;X \subseteq r$, $r(z)$ follows as desired. The proof of $(p;\overline{S\circ I}\rightarrow r)(y)$ is straightforward and therefore omitted. □

We now give the justification of the Manna & Pnueli rule. They write $\{p(x)\}\ S\ \{xQy\}$ for: For all states $x$ satisfying $p$, $S$ terminates properly with output $y$ satisfying $xQy$. Now let $(W,<)$ be a well-founded set (no infinite decreasing <-chains) and $f$ a partial function mapping the set of states $V$ to $W$. Then Manna & Pnueli's rule reads as follows: From

a. $\{u(x) \wedge p(x)\}\ S\ \{xQy \wedge (f(x) > f(y))\}$
b. $\forall x,y[xQy \wedge p(y) \rightarrow u(y)]$
c. $\forall x,y,z[xQy \wedge yQz \rightarrow xQz]$
d. $\forall x[u(x) \wedge \bar{p}(x) \rightarrow xQx]$

one may infer that

e. $\{u(x)\}\ p{*}S\ \{xQy \wedge \bar{p}(y)\}$

In our relational formulation this takes the form as given in

THEOREM 4.6. From

a. $u;p \subseteq (S\cap Q)\circ I$
b. $\mu X[S\rightarrow X] = I$
c. $Q;p \subseteq Q;u$
d. $Q;Q \subseteq Q$
e. $u;\bar{p} \subseteq Q$

one may infer that

f. $u \subseteq \mu X[p;S \cup p;\overline{S\circ I} \rightarrow X]$
g. $u;p{*}S \subseteq Q;\bar{p}$

PROOF. We use the auxiliary result that, for any $R$, $p$, we have $\mu X[p;R \rightarrow X] = \mu X[p;R;p \rightarrow X]$, the simple proof of which is omitted. In order to show f), we use Scott's induction (cf. the proof of lemma 4.5) and prove f'): if $u;X \subseteq r$, then $u;(S\rightarrow X) \subseteq r$, where $r$ is defined as: $r = \mu X[p;S \cup p;\overline{S\circ I} \rightarrow X]$. So assume $u;X \subseteq r$, and $u(y)$ and $(S\rightarrow X)(y)$ for some $y$. To show $r(y)$, or, by the fixed point property, both $(p;S\rightarrow r)(y)$ and $(p;\overline{S\circ I}\rightarrow r)(y)$. The second of these is again obvious, and we prove only the first. By the auxiliary result, it is sufficient to show $(p;S;p\rightarrow r)(y)$. So assume $y\ p;S;p\ z$ and show $r(z)$. Since $p(y)$ and $u(y)$, by a) we have $ySt$ and $yQt$ for some $t$. Since $ySz$ and $S$ is a function, $t = z$. Since $yQz$ and $p(z)$, from c) we infer that $u(z)$. Since $ySz$ and $(S\rightarrow X)(y)$, also $X(z)$. Then, using $u;X \subseteq r$, the result $r(z)$ follows, and the proof of f') is completed.

Next we prove g) by first applying simultaneous Scott's induction to show $u;p{*}S \subseteq Q$ and $Q;p{*}S \subseteq Q$. Thus, assume $u;X \subseteq Q$ and $Q;X \subseteq Q$. We verify

- $u;(p;S;X \cup \bar{p}) \subseteq Q$:
  $u;p;S;X \subseteq$ (ass. a) $u;p;Q;X \subseteq Q;X \subseteq Q$
  $u;\bar{p} \subseteq Q$ (ass. e)
- $Q;(p;S;X \cup \bar{p}) \subseteq Q$:
  $Q;p;S;X \subseteq$ (ass. c) $Q;p;u;S;X \subseteq$ (ass. a) $Q;p;u;Q;X \subseteq$
  $\subseteq Q;Q;X \subseteq Q;Q \subseteq$ (ass. d) $Q$, and
  $Q;\bar{p} \subseteq Q$

Thus, we have shown that $u;p{*}S \subseteq Q$. Clearly, then also $u;p{*}S \subseteq Q;\bar{p}$, and the proof of g) and hence of theorem 4.6 is completed. □

REFERENCES


[1] J.W. de Bakker, Recursive Procedures, Mathematical Centre Tracts 24, Amsterdam (1971).

[2] J.W. de Bakker, Least fixed points revisited, to appear in Theoretical Computer Science.

[3] J.W. de Bakker & L.G.L.T. Meertens, On the completeness of the inductive assertion method, to appear in J. of Comp. Syst. Sci.

[4] J.W. de Bakker & W.P. de Roever, A calculus for recursive program schemes, in Automata, Languages and Programming (M. Nivat, ed.), p.167-196, North-Holland, Amsterdam (1973).

[5] E.W. Dijkstra, A simple axiomatic basis for programming language constructs, Proc. Kon. Ned. Akad., Ser. A, 77 (or Indagationes Math., 36), 1-15 (1974).

[6] P. Hitchcock & D. Park, Induction rules and proofs of termination, in Automata, Languages and Programming (M. Nivat, ed.), p.225-251, North-Holland, Amsterdam (1973).

[7] C.A.R. Hoare, An axiomatic basis for computer programming, CACM 12 (1969), 576-580.

[8] C.A.R. Hoare, Procedures and parameters, an axiomatic approach, in Symp. on Sem. Alg. Lang. (E. Engeler, ed.), Lecture Notes in Math., Vol. 188, p.102-116, Springer (1971).

[9] C.A.R. Hoare, An axiomatic definition of the programming language PASCAL, in Int. Symp. on Theor. Progr. (A. Ershov, ed.), Lecture Notes in Comp. Sci., vol. 5, p.1-16, Springer (1974).

[10] C.A.R. Hoare & N. Wirth, An axiomatic definition of the programming language PASCAL, Acta Inf. 2 (1973), 335-355.

[11] D.E. Knuth, Structured programming with goto statements, Comp. Surveys 6 (1974), 261-302.

[12] Z. Manna, Mathematical Theory of Computation, McGraw-Hill (1974).

[13] Z. Manna & J. Vuillemin, Fixpoint approach to the theory of computation, CACM 15 (1972), 528-536.

[14] N. Wirth, On the composition of well-structured programs, Comp. Surveys 6 (1974), 247-260.