Acta Informatica 26, 309–332 (1989)

Predicate-Transformer Semantics of General Recursion

Wim H. Hesselink
University of Groningen, Department of Computing Science, P.O. Box 800, NL-9700 AV Groningen, The Netherlands

Summary. We develop the semantics of a language with arbitrary atomic statements, unbounded nondeterminacy, and mutual recursion. The semantics is expressed in weakest preconditions and weakest liberal preconditions. Individual states are not mentioned. The predicates on the state space are treated as elements of a distributive lattice. The semantics of recursion is constructed by means of the theorem of Knaster-Tarski. It is proved that the law of the excluded miracle can be preserved, if that is wanted. The universal conjunctivity of the weakest liberal precondition, and the connection between the weakest precondition and the weakest liberal precondition are proved to remain valid. Finally we treat Hoare-triple methods for proving correctness and conditional correctness of programs.

0. Introduction

0.0. Recursion, Repetition, and Nondeterminacy

Most treatments of the semantics of repetitive or recursive constructs depend on an assumption of bounded nondeterminacy. As recursion is less operational and more general than repetition, we concentrate on recursion, cf. [10]. An outline for a treatment of recursion was given as early as 1976 in [18], but more detailed treatments such as [10] and [2], Chap. 7, depended on continuity, i.e. bounded nondeterminacy. In [1], the repetition is treated, under the weaker assumption of countable nondeterminacy. Unbounded nondeterminacy for the repetition is achieved in [7] and [9]. Inspired by [9], we constructed in [13] an operational model for recursion with unbounded nondeterminacy. This model leads to nice denotational characterizations of the semantics, but, as it heavily depends on consideration of individual states, it tends to encourage operational reasoning.

The present paper contains a purely formal treatment of recursion with unbounded nondeterminacy. We express the semantics of our programming language by means of weakest preconditions, cf. [4] and [10]. This method is more suitable for programming methodology than the method of [2] and [13], in which the semantics of a construct is determined by the set of possibly resulting states for a given initial state.

0.1. *The Semantic Functions wp and wlp*

Following Dijkstra, cf. [4], we express the semantics of our programming language by means of functions *wp* and *wlp*. If x is a predicate and t is a command, the predicate $wp(t, x)$ is the weakest precondition such that execution of t terminates in a state where x holds. The predicate $wlp(t, x)$ is the weakest precondition such that x holds whenever execution of t terminates. Thus the weakest liberal precondition *wlp* does not imply termination. In other words, *wp* represents correctness and *wlp* represents conditional correctness. The predicate $wp(t, true)$ represents guaranteed termination. This leads to the requirement, cf. [5]:

$$wp(t, x) = wp(t, true) \wedge wlp(t, x). \tag{0}$$

It is easy to argue that the *wlp* of a conjunction of postconditions should be the conjunction of the *wlp*'s of the postconditions, cf. [5]. So, if Y is a set of predicates, then

$$wlp(t, (\forall y \in Y::y)) = (\forall y \in Y::wlp(t, y)). \tag{1}$$

As a formalization of the fact that no precondition can guarantee termination in a state where *false* holds, Dijkstra [4] postulates the so-called law of the excluded miracle

$$wp(t, false) = false. \tag{2}$$

In the semantics of recursion, however, syntactic enforcement of this law leads to complicated formulae. Therefore, we follow De Bakker, cf. [2], p. 270, and Nelson, cf. [17], in allowing violations of (2). A command t that violates (2) is said to be *partial*. In states where $wp(t, false)$ holds, command t is said to *fail*, and implementations are supposed to perform backtracking, cf. [14] p. 470. A command t that satisfies (2) is said to be *total*.

0.2. *The Alternative Construct*

The starting point for our treatment of recursion is Dijkstra's nondeterministic alternative construct, cf. [4], in the form

$$h = \mathbf{if}\ (\square j \in J : b.j \rightarrow r.j)\ \mathbf{fi}, \tag{3}$$

where J is a set of indices, each guard $b.j$ is a predicate, and each $r.j$ is a command. The operational interpretation is that one command $r.j$ is executed for which the guard $b.j$ holds. If no guard holds, the construct does not terminate. Formally, the semantics is defined by the preconditions

$$\begin{aligned} wp(h,x) &= (\exists j::b.j) \wedge (\forall j::\neg b.j \vee wp(r.j,x)) \\ \wedge\ wlp(h,x) &= (\forall j::\neg b.j \vee wlp(r.j,x)) \end{aligned} \tag{4}$$

for any postcondition x.

0.3. Discussion of our Modifications

(a) In order to treat general recursion, formula (3) is regarded as the declaration of a procedure h. Every procedure is supposed to be declared in this way. The commands $r.j$ are strings of simple commands. Every simple command is either a statement of which the semantics is assumed to be known, or a procedure which has some declaration. In this way recursion is possible, as is mutual recursion.

(b) The index set J in (3) is allowed to be infinite. In this way, a value parameter can be passed to the procedure, but unbounded nondeterminacy can arise. Notice that the statements can also be unboundedly nondeterministic.

(c) Because of the conjunct $(\exists j::b.j)$ in $wp(h,x)$, the procedure h leads to abortion if all guards are false. This is an exception, which complicates the semantics of recursion. Therefore, in this paper we delete this conjunct, cf. [2], p. 272, [17]. This has the effect that if none of the guards holds, abortion is replaced by failure, so that h is a partial command, cf. 0.1. In implementations, one may want to impose the semantic restriction

$$(\exists j::b.j) = true. \tag{5}$$

Remark. The equality sign between predicates on the state space is used to denote that the predicates are everywhere equivalent.

0.4. Procedure Declarations

Summarizing our modifications, we propose procedure declarations of the form

$$h::=(\square j \in J : b.j \rightarrow r.j) \tag{6}$$

where J can be any set of indices, $b.j$ is a predicate, $r.j$ is a command. If all predicates $\neg b.j$ hold, execution fails, cf. 0.1. Declaration (6) is defined to be *total*, if and only if

$$(\forall j::\neg b.j) = false. \tag{7}$$

Notice that $(7) \equiv (5)$. We do not require that the commands $r.j$ be total. For, in case of recursion, such a condition would lead to circularity. We come back to this in Sect. 2.8. In view of (4) and modification (c), the semantic functions *wp* and *wlp* will be defined as solutions of the following equation in an unknown function *w*

$$w(h, x) = (\forall j :: \neg b.j \vee w(r.j, x)). \tag{8}$$

As the commands $r.j$ may contain recursive calls of h, this equation need not have a unique solution. We come back to this in Sect. 0.8 below. Notice that step 0.3 (c) was crucial in obtaining a single equation for both *wp* and *wlp*.

0.5. Using a Partially Ordered Set

In programming practice, we write $[x \Rightarrow y]$ to denote that predicate x implies predicate y everywhere on the state space. For the present purpose, however, it is convenient to treat the implication abstractly as a partial order. Therefore, the predicates are treated as elements of a partially ordered set $\langle X, \leqq \rangle$ where relation "$\leqq$" is given by

$$x \leqq y \equiv [x \Rightarrow y]. \tag{9}$$

The strongest predicate *false* corresponds to the smallest element 0 of X. The weakest element *true* corresponds to the biggest element 1. The conjunction operators $\wedge$ and $\forall$ are replaced by the greatest lower bound operators $\wedge$ and $\bigwedge$, pronounced "meet". The disjunction operators $\vee$ and $\exists$ are replaced by the least upper bound operators $\vee$ and $\bigvee$, pronounced "join". The negation operator "$\neg$" used in (7) and (8) could be an arbitrary function from X to X. We abstract from all its special properties.

We need arbitrary least upper bounds and greatest lower bounds, and a general form of distributivity. Therefore, X will be a complete distributive lattice, cf. 2.0 and 3.1 below. This abstraction is justified by the fact that the set of predicates on a state space satisfies all these axioms.

0.6. Notations for Functions

For the sake of uniformity, the application of a function is denoted by the infix operator dot ".", cf. [8]. Composition of functions is denoted by the usual circle "$\circ$". Dot and circle have the same high binding power. They bind from left to right, so that $f \circ g.x = f.(g.x)$. We use the currying convention that a function of more than one argument is treated as a function of the first argument which yields a function of the remaining arguments.

By convention, functions f and g on the same domain U are equal, if and only if $f.u = g.u$ for every $u \in U$. Thus the well-known rule

$$(\forall x :: wp((p; q), x) = wp(p, wp(q, x)))$$

of [4] is expressed by

$$(\forall x :: wp.(p; q).x = (wp.p) \circ (wp.q).x),$$

or equivalently

$$wp.(p; q) = (wp.p) \circ (wp.q).$$

0.7. The Language

Let A be a given set of symbols, to be called *simple commands*. Let A^* denote the set of finite strings of elements of A. The empty string is denoted by ε. The concatenation of strings p and q is denoted by "$p; q$", as used already. The elements of A^* are called *commands*.

We assume that A is the union of two disjoint subsets S and H, where S is the set of the statements and H is the set of the procedure names. For each statement $s \in S$ the semantics is supposed to be given, so that the predicate transformers $wp.s$ and $wlp.s$ are known. Functions w, such as wp and wlp, will be defined on the set A, and then extended to A^* by the rule

$$w.(p;q) = (w.p) \circ (w.q). \tag{10}$$

Therefore, it will remain to specify $wp.h$ and $wlp.h$ for each procedure $h \in H$. Every procedure h has a declaration of the form

$$h ::= (\square j \in J.h : b.j \rightarrow r.j) \tag{11}$$

with $b.j \in X$ and $r.j \in A^*$, where $J.h$ is an index set, which may depend on h. Actually, some readers may want to impose the condition that the set $J.h$ for different values of h are disjoint. By formula (8), the functions wp and wlp should be solutions of the system of equations in w

$$w.h.x = (\textstyle\bigwedge j \in J.h :: \neg b.j \vee w.(r.j).x). \tag{12}$$

0.8. Main Results

It will turn out that wp can be defined as the smallest solution of (12) that extends wp as given on the statements. Similarly, wlp will be defined as the greatest solution of (12) that extends wlp as given on the statements. This choice for extending wp and wlp goes back to [18] and [10]. In [13], we showed that a suitable operational model leads to functions wp and wlp which are indeed the extreme solutions of Eq. (12).

Another justification for this choice is provided by the fact that the functions wp and wlp thus defined have the expected properties. In fact we prove that if all statements satisfy the requirements (0) and (1), then the procedures and all commands satisfy these requirements. We also prove that if all statements are total, cf. 0.1, and all procedure declarations are total, cf. 0.4, then all procedures and all commands are total, cf. 0.1. Finally, if all statements and all declarations are deterministic, then all commands are deterministic, see Sect. 3.5.

In Chap. 4, we develop Hoare-triple methods for proving termination, correctness and conditional correctness of programs. The methods are well-known. New is that the validity is proved in a very general semantic formalism. As an illustration we treat a very simple example in Sect. 4.4.

0.9. Aspects of Presentation

The braces "{" and "}" are used exclusively for comment within formal proofs. We use the proof format exposed in [19] p. 2. We use a typed version of Dijkstra's quantification format, cf. [6]. The formula

$$(\forall v \in V{:}\, b.v{:}\, f.v)$$

denotes that $f.v$ holds for every value $v \in V$ for which $b.v$ holds. If the range condition $b.v$ is identically true, it can be omitted. The existential quantification is determined by

$$(\exists v \in V{:}\, b.v{:}\, f.v) \equiv \neg(\forall v \in V{:}\, b.v{:}\, \neg f.v).$$

In each chapter, the referenced formulae are numbered consecutively. Formula (j) of chapter i is referred to from other chapters as formula $i(j)$.

1. A Variation of the Theorem of Knaster-Tarski

1.0. This chapter contains some background material on partially ordered sets. In particular, we present a version of the theorem of Knaster-Tarski which is slightly stronger than usual.

1.1. Partially Ordered Sets

Let W be a partially ordered set with order relation "$\leqq$". An element $w0$ of W is said to be a greatest lower bound of a subset U of W, if and only if

$$(\forall w \in W{::}\, w \leqq w0 \equiv (\forall u \in U{::}\, w \leqq u)). \qquad (0)$$

If it exists, the greatest lower bound of a set U is unique. It is denoted by $(\bigwedge u \in U{::}\, u)$.

Reversing the order, we say that $w1 \in W$ is a least upper bound of U, if and only if

$$(\forall w \in W{::}\, w1 \leqq w \equiv (\forall u \in U{::}\, u \leqq w)). \qquad (1)$$

If it exists, the least upper bound of U is also unique. It is denoted by $(\bigvee u \in U{::}\, u)$.

1.2. Completeness and Closure

The partially ordered set W is said to be *complete* if and only if every subset of W has a greatest lower bound and a least upper bound in W.

A subset V of W is said to be *closed under least upper bounds in* W if and only if for every subset U of V the set V contains an element that is the least upper bound of U in W. Similarly, V is said to be *closed under greatest lower bounds in* W if and only if for every subset U of V the set V contains an element that is the greatest lower bound of U in W.

Remark. If V is closed under least upper bounds in W, then V is nonempty. For, V contains the least upper bound of the empty subset $U=\emptyset$. Similarly, V is nonempty if it is closed under greatest lower bounds in W.

1.3. Fixed Points of Monotone Functions

A function $f\colon W\rightarrow W$ is said to be *monotone*, if and only if

$$(\forall v, w\in W{:}v\leqq w{:}f.v\leqq f.w).$$

An element $w\in W$ is called a *fixed point* of function f, if and only if $f.w=w$. A subset V of W is said to be *invariant under* f, if and only if

$$(\forall v\in V{:}\,{:}f.v\in V).$$

The following result is a variation of the theorem of Knaster-Tarski, cf. [20].

Theorem. *Let $f\colon W\rightarrow W$ be a monotone function. Let V be a subset of W, which is closed under least upper bounds in W and which is invariant under f. Then V contains a fixed point $v0$ of f, which satisfies $v0\leqq w$ for every $w\in W$ with $f.w\leqq w$. In particular, $v0$ is the smallest fixed point of f in W.*

Proof. Let U be the subset of W given by

$$u\in U\equiv u\in V\wedge(\forall w\in W{:}f.w\leqq w{:}u\leqq w). \tag{2}$$

Since U is a subset of V and V is closed under least upper bounds in W, the set V contains an element that is the least upper bound of U in W. Calling this element $v0$, we have from (1) with $w1:=v0$

$$v0\in V\wedge(\forall w\in W{:}\,{:}v0\leqq w\equiv(\forall u\in U{:}\,{:}u\leqq w)). \tag{3}$$

For any $w\in W$ we observe

$$\begin{aligned}&f.w\leqq w\\ \Rightarrow&\{(2)\}\ (\forall u\in U{:}\,{:}u\leqq w)\\ \equiv&\{(3)\}\ v0\leqq w.\end{aligned}$$

By (2) and (3) this implies that $v0\in U$. Instantiation of (3) with $w:=v0$ gives

$$(\forall u\in U{:}\,{:}u\leqq v0). \tag{4}$$

Now we observe

$$
\begin{array}{ll}
\quad v0 \leqq f.v0 & (5) \\
\Leftarrow \{(2) \text{ with } u := v0 \text{ and } w := f.v0\} & \\
\quad f.(f.v0) \leqq f.v0 & \\
\Leftarrow \{f \text{ is monotone}\} & \\
\quad f.v0 \leqq v0 & (6) \\
\Leftarrow \{(4)\} & \\
\quad \mathrm{f}.v0 \in U & \\
\Leftarrow \{(2), \text{ and } V \text{ is invariant under } f\} & \\
\quad (\forall w \in W{:}f.w \leqq w{:}f.v0 \leqq w) & \\
\Leftarrow \{\text{transitivity of } \leqq \text{ and range condition}\} & \\
\quad (\forall w \in W{:}f.w \leqq w{:}f.v0 \leqq f.w) & \\
\Leftarrow \{f \text{ is monotone}\} & \\
\quad (\forall w \in W{:}f.w \leqq w{:}v0 \leqq w) & \\
\equiv \{v0 \in U \text{ and } (2)\} & \\
\quad \mathit{true}. &
\end{array}
$$

This reduction shows that the inequalities (5) and (6) hold. These inequalities combine to yield $v0 = f.v0$. This proves that $v0$ is a fixed point of f. Since $v0 \in U$, we have $v0 \leqq w$ for every $w \in W$ with $f.w \leqq w$. The last assertion of the theorem follows from the fact that every fixed point w satisfies $f.w \leqq w$. (End of proof.)

Remark. I first used the above argument in [12] 8.2. I assume that the result is known. Using transfinite induction one can prove the result under the weaker assumptions that V is invariant under f and closed under least upper bounds of totally ordered subsets.

2. The Semantics of Recursion

2.0. The Semantic Domain

In this chapter we formally construct the extended functions *wp* and *wlp* on the set A, cf. 0.7.

Let X be a given partially ordered set with order relation $\leqq$. We assume that X is complete, cf. 1.2. The smallest element of X is denoted by 0, the biggest one by 1. The elements of X may be regarded as predicates on a state space. A function from X to X may be regarded as a predicate transformer. We only consider monotone functions from X to X.

We use M to denote the set of the monotone functions $X \to X$. The composition $f \circ g$ of functions $f, g \in M$ is an element of M. The identify function *ident* of X is the unit element for the composition in M.

2.1. Semantic Functions

We refer to Sect. 0.7 for the introduction of the alphabet $A = S \cup H$ and of the language A^*. We define W to be the set of all functions $w \colon A \to M$. Since composition in M is associative and has *ident* as unit element, any function $w \in W$ has precisely one extension to a function $w \colon A^* \to M$ such that

$$w.\varepsilon = ident \tag{0}$$

$$(\forall p, q \in A^* :: w.(p;q) = (w.p) \circ (w.q)). \tag{1}$$

For any $w \in W$, we shall use the extended function w whenever appropriate. Compare formula 0(10). From 0.7, every procedure $h \in H$ is equipped with a declaration of the form

$$h :: = (\square j \in J.h : b.j \to r.j).$$

A function $w \in W$ is called a *semantic function*, if and only if it satisfies formula 0(12), that is

$$(\forall h \in H, x \in X :: w.h.x = (\bigwedge j \in J.h :: \neg b.j \vee w.(r.j).x)). \tag{2}$$

Remark. The semantic functions used here correspond to the preparator functions of [13].

2.2. Unfolding

In order to treat formula (2) as a fixed point equation, we introduce the unfolding function F. For any given function $wg \colon S \to M$, we define the function $F.wg \colon W \to W$ by

$$(\forall w \in W, s \in S, h \in H, x \in X ::$$
$$F.wg.w.s.x = wg.s.x \wedge F.wg.w.h.x = (\bigwedge j \in J.h :: \neg b.j \vee w.(r.j).x)). \tag{3}$$

So $F.wg.w$ agrees with wg on the statements s. For a procedure name h, function $F.wg.w.h$ is given as the righthand side of formula (2). One verifies that $F.wg.w.a \in M$ for every $w \in W$ and every $a \in A$. Clearly, $w \in W$ is a semantic function if and only if $w = F.wg.w$ for some function wg, – in which case wg is the restriction of w to the set S.

2.3. A Partial Order on W

We shall use the theorem of Knaster-Tarski to ensure that every given function $wg\colon S \to M$ can be extended to a semantic function w. Therefore, we introduce a partial order $\leqq$ on W by

$$w \leqq w' \equiv (\forall a \in A, x \in X \colon\colon w.a.x \leqq w'.a.x). \tag{4}$$

The partially ordered set $\langle W, \leqq \rangle$ is complete. In fact, a subset U of W has a greatest lower bound $w0$ and a least upper bound $w1$ in W given by

$$w0.a.x = (\bigwedge u \in U \colon\colon u.a.x) \wedge w1.a.x = (\bigvee u \in U \colon\colon u.a.x). \tag{5}$$

The verifications are straightforward. For example, monotony of $w0$ and $w1$ is easy to show. The calculation

$$\begin{aligned} & w \leqq w0 \\ \equiv & \{(4)\}\ (\forall a, x \colon\colon w.a.x \leqq w0.a.x) \\ \equiv & \{(5)\}\ (\forall a, x \colon\colon w.a.x \leqq (\bigwedge u \colon\colon u.a.x)) \\ \equiv & \{1(0)\}\ (\forall a, x, u \colon\colon w.a.x \leqq u.a.x) \\ \equiv & \{(4)\}\ (\forall u \colon\colon w \leqq u) \end{aligned}$$

proves that $w0$ is the greatest lower bound.

Since the definition of F involves application of w to strings $r.j$, we extend formula (4) to strings. For functions $w, w' \in W$, we have

$$w \leqq w' \equiv (\forall t \in A^*, x \in X \colon\colon w.t.x \leqq w'.t.x). \tag{6}$$

The implication "$\Leftarrow$" follows from (4) and $A \subset A^*$. The implication "$\Rightarrow$" is proved by induction on the length of string t. The case $t = \varepsilon$ follows from (0). The induction step is

$$\begin{aligned} & w.t.x \leqq w'.t.x \\ \Rightarrow & \{\text{if } a \in A \text{ then } w.a \text{ is monotone}\}\ w.a.(w.t.x) \leqq w.a.(w'.t.x) \\ \Rightarrow & \{(4) \text{ with } x := w'.t.x \text{ and transitivity of } \leqq\}\ w.a.(w.t.x) \leqq w'.a.(w'.t.x) \\ \equiv & \{(1)\}\ w.(a;t).x \leqq w'.(a;t).x. \end{aligned}$$

This concludes the proof of (6).

2.4. Theorem. *For any function $wg\colon S \to M$, the function $F.wg\colon W \to W$ is monotone.*

Proof. For any w, $w' \in W$, we observe

$$\begin{aligned}
& F.wg.w \leqq F.wg.w' \\
\equiv & \{(4)\} \\
& (\forall a \in A, x \in X :: F.wg.w.a.x \leqq F.wg.w'.a.x) \\
\equiv & \{(3), \text{use } J.h \text{ as range for } j\} \\
& (\forall h \in H, x \in X :: (\bigwedge j :: \neg b.j \vee w.(r.j).x) \leqq (\bigwedge j :: \neg b.j \vee w'.(r.j).x)) \\
\Leftarrow & \{\text{monotony of } \bigwedge \text{ and } \vee\} \\
& (\forall t \in A^*, x \in X :: w.t.x \leqq w'.t.x) \\
\equiv & \{(6)\} \\
& w \leqq w'.
\end{aligned}$$

(End of proof.)

Remark. This proof relies on (6). The proof of (6) relies on the monotony of the functions in M. In 2.0, monotony was imposed for this reason. It does not help to define the order of W by means of (6), since the above proof also uses (4).

2.5. Theorem. *For any function $wg: S \to M$, the function $F.wg: W \to W$ has a smallest fixed point wg^0, and a greatest fixed point wg^1 in W. The functions wg^0 and wg^1 are the smallest and greatest semantic functions which agree with wg on S.*

Proof. Since W is complete and $F.wg$ is monotone, the existence of wg^0 follows from Theorem 1.3 with $V := W$. The dual version of that theorem yields the existence of wg^1. The last assertion of the theorem follows from 2.2. (End of proof.)

Remark. Here we just use the ordinary version of the theorem of Knaster-Tarski.

2.6. The Postulate of Predicate-Transformer Semantics

We assume that the semantics of the statements $s \in S$ is given by weakest preconditions and weakest liberal preconditions, cf. [4]. So we assume given a weakest precondition function $wp: S \to M$ and a weakest liberal precondition function $wlp: S \to M$.

The weakest precondition function on A is defined as the smallest semantic function which extends wp on S. So, it is the function $wp^0: A \to M$, cf. 2.5. The weakest liberal precondition function on A is defined as the greatest semantic function which extends $wlp: S \to M$. So it is the function $wlp^1: A \to M$, cf. 2.5. If no ambiguity can arise we write wp and wlp instead to wp^0 and wlp^1. This is rather harmless, as wp^0 and wlp^1 are extensions of the functions wp and wlp, respectively.

2.7. Remarks. The above definition can be justified by its simplicity, or by many references, cf. [10, 16, 18], or by an operational model. Following [9], we developed in [13] an operational model in which the functions wp and wlp are proved to be the extreme semantic functions. Actually, they remain extreme if the monotony condition of 2.0 is dropped. I cannot achieve this in the abstract setting. For the translation, it may be noted that in [13] the symbol X stands for the set of states, so that the power set $\mathscr{P}(X)$ corresponds to the set X of the present paper.

The recursive equation $w=F.wg.w$ is an equation of second order functions. In the case of simple recursion, however, the set H consists of one procedure, say h. So, in that case, the equation boils down to one equation in the "predicate transformer" $w.h\in M$. In the last part of [17], it is shown that in the case of tail recursion the equation can be specialized to equations in the predicates $w.h.x$.

2.8. The Law of the Excluded Miracle

In this section we show that the law of the excluded miracle is equivalent to that same law restricted to the statements in conjunction with the totality of the declarations. In other words we prove

Theorem. *All commands are total if and only if all statements and all declarations are total.*

Proof. By 0.1 and 0.5, a command t is total if and only if $wp.t.0=0$, and the declaration of a procedure h is total if and only if $(\bigwedge j\in J.h::\neg b.j)=0$.

Let all commands be total. Clearly, all statements are total. For a procedure $h\in H$, totality of h implies totality of its declaration because of

$$\begin{aligned} & wp.h.0=0 \\ \equiv & \{2.6\}\ F.wp.wp.0=0 \\ \equiv & \{(3)\}\ (\textstyle\bigwedge j\in J.h::\neg b.j\vee wp.(r.j).0)=0 \\ \Rightarrow & \{\text{calculus}\}\ (\textstyle\bigwedge j\in J.h::\neg b.j)=0. \end{aligned}$$

For the proof of the other implication, we construct an auxiliary function. Let $u\in W$ be given by

$$(\forall a\in A::u.a.0=0\wedge(\forall x\in X:x\neq 0:u.a.x=1)). \qquad (7)$$

Function u is the biggest function that treats all simple commands as total. In fact, by (4) and (7), we have

$$(\forall w\in W::w\leq u\equiv(\forall a\in A::w.a.0=0)). \qquad (8)$$

Now we observe that

$$(\forall t \in A^* :: wp.t.0 = 0)$$
$$\equiv \{\text{induction on the length of string } t\text{, using (0) and (1)}\}$$
$$(\forall a \in A :: wp.a.0 = 0)$$
$$\equiv \{(8)\}$$
$$wp \leqq u$$
$$\Leftarrow \{\text{Theorem 1.3 with } V := W, f := F.wp, w := u, \text{ and hence } v0 = wp \text{, the smallest fixed point of } F.wp\}$$
$$F.wp.u \leqq u$$
$$\equiv \{(8)\}$$
$$(\forall a \in A :: F.wp.u.a.0 = 0)$$
$$\equiv \{(3)\}$$
$$(\forall s \in S :: wp.s.0 = 0) \wedge (\forall h \in H :: (\textstyle\bigwedge j \in J.h :: \neg b.j \vee u.(r.j).0) = 0)$$
$$\equiv \{\text{calculus}\}$$
$$(\forall s \in S :: wp.s.0 = 0) \wedge (\forall h \in H :: (\textstyle\bigwedge j \in J.h :: \neg b.j) = 0).$$

This proves that totality of all commands follows from totality of all statements and all declarations. (End of proof.)

3. Properties of *wp* and *wlp*

3.0. In this chapter we show that the semantic functions *wp* and *wlp* have the properties 0(0) and 0(1), provided that these properties hold for the statements and that the partially ordered set X satisfies certain distributivity laws. We use $Pow.X$ to denote the power set of X. In the last subsection we consider the deterministic theory. If all statements are deterministic and all declarations are deterministic, we prove that all comands are deterministic.

3.1. *Postulates*

We assume that all statements satisfy the formulae 0(0) and 0(1). By 0.5 and 0.6, this gives the postulates

$$(\forall s \in S, x \in X :: wp.s.x = wp.s.1 \wedge wlp.s.x), \tag{0}$$

$$(\forall s \in S,\ Y \in Pow.X :: wlp.s.(\textstyle\bigwedge y \in Y :: y) = (\textstyle\bigwedge y \in Y :: wlp.s.y)). \tag{1}$$

Moreover, we postulate the distributivity laws

$$(\forall x \in X,\ Y \in Pow.X::x \vee (\bigwedge y \in Y::y) = (\bigwedge y \in Y::x \vee y)), \tag{2}$$

$$(\forall x \in X,\ Y \in Pow.X::x \wedge (\bigvee y \in Y::y) = (\bigvee y \in Y::x \wedge y)), \tag{3}$$

Remark. It is justified to postulate these laws, as they are satisfied by the set of boolean functions on a state space. In [3], p. 229, these laws are denoted by D_1 and $D_1^{\sim}$.

3.2. *Universal Conjunctivity*

It turns out that in order to extend formula (0) we need a weak version of the extension of formula (1). Therefore, we start by extending formula (1) to command strings t.

Theorem. *For every $t \in A^*$ and $Y \in Pow.X$ we have*

$$wlp.t.(\bigwedge y \in Y::y) = (\bigwedge y \in Y::wlp.t.y).$$

Proof. Let $M0$ denote the subset of M given by

$$f \in M0 \equiv (\forall Y \in Pow.X::f.(\bigwedge y \in Y::f.y) = (\bigwedge y \in Y::f.y)). \tag{4}$$

Let $W0$ denote the subset of W given by

$$w \in W0 \equiv (\forall a \in A::w.a \in M0). \tag{5}$$

For elements $f, g \in M0$, the composition $f \circ g$ is also element of $M0$. Therefore, it follows from (5) by the extension rules 2(0) and 2(1) that

$$(\forall w \in W0,\ t \in A^*::w.t \in M0). \tag{6}$$

Now the theorem is equivalent to $wlp \in W0$. By 2.6, wlp is the greatest fixed point of the monotone function $F.wlp$ in W. Therefore, by the dual version of Theorem 1.3 with $V := W0$, it suffices to prove

(a) $W0$ is closed under greatest lower bounds in W.
(b) $W0$ is invariant under $F.wlp$.

These two facts are proved in the Lemmas (a) and (b) below. (End of proof.)

Lemma (a). *The set $W0$ is closed under greatest lower bounds in W.*

Proof. Let U be a subset of $W0$. By 2.3, the set U has a greatest lower bound $u0$ in W given by

$$u0.a.x = (\bigwedge u \in U::u.a.x). \tag{7}$$

For any subset Y of X and any element $a \in A$, we observe

$$\begin{aligned} & u0.a.(\bigwedge y \in Y::y) \\ = & \{(7)\}\ (\bigwedge u \in U::u.a.(\bigwedge y \in Y::y)) \\ = & \{U \subset W0, \text{and (5) and (4)}\}\ (\bigwedge u \in U::(\bigwedge y \in Y::u.a.y)) \\ = & \{\text{interchange}\}\ (\bigwedge y \in Y::(\bigwedge u \in U::u.a.y)) \\ = & \{(7)\}\ (\bigwedge y \in Y::u0.a.y). \end{aligned}$$

By (4) and (5), this proves that $u0 \in W0$. (End of proof).

Lemma (b). *The set $W0$ is invariant under $F.wlp$.*

Proof. Let $w \in W0$ be given. For any $s \in S$ we have

$$\begin{aligned} & F.wlp.w.s \\ = & \{2(3)\}\ wlp.s \\ \in & \{(1), (4)\}\ M0. \end{aligned}$$

For any $h \in H$ and any subset Y of X, we observe

$$\begin{aligned} & F.wlp.w.h.(\bigwedge y \in Y::y) \\ = & \{2(3)\}\ (\bigwedge j \in J.h::\neg b.j \vee w.(r.j).(\bigwedge y \in Y::y)) \\ = & \{w \in W0, (5), (6)\}\ (\bigwedge j \in J.h::\neg b.j \vee (\bigwedge y \in Y::w.(r.j).y)) \\ = & \{\text{distributivity law (2)}\}\ (\bigwedge j \in J.h::(\bigwedge y \in Y::\neg b.j \vee w.(r.j).y)) \\ = & \{\text{interchange and } 2(3)\}\ (\bigwedge y \in Y::F.wlp.w.h.y), \end{aligned}$$

so that $F.wlp.w.h \in M0$ by (4). By formula (5), this proves that $F.wlp.w \in W0$, as required. (End of proof.)

3.3. The Separation of Termination and Conditional Correctness

Theorem. $(\forall t \in A^*, x \in X::wp.t.x = wp.t.1 \wedge wlp.t.x)$.

Proof. Let $W1$ be defined as the subset of W given by

$$w \in W1 \equiv (\forall a \in A, x \in X::w.a.x = w.a.1 \wedge wlp.a.x). \tag{8}$$

Below in Lemma (c) we prove

$$(\forall w \in W1, t \in A^*, x \in X::w.t.x = w.t.1 \wedge wlp.t.x).$$

Now the theorem is equivalent to $wp \in W1$. By 2.6, wp is the smallest fixed point of the monotone function $F.wp$ in W. Therefore, by 1.3 with $V := W1$, it suffices to prove

(d) $W1$ is closed under least upper bounds in W.

(e) $W1$ is invariant under $F.wp$.

These facts are proved below in the Lemmas (d) and (e). (End of proof.)

Lemma (c). *Let* $w \in W1$ *and* $t \in A^*$. *Then* $(\forall x \in X :: w.t.1 \wedge wlp.t.x)$.

Proof. This is proved by induction in the length of string t. For $t := \varepsilon$, we have

$$\begin{aligned} & w.\varepsilon.x = w.\varepsilon.1 \wedge wlp.\varepsilon.x \\ \equiv & \{2(0)\}\ x = 1 \wedge x \\ \equiv & \{x \leq 1\}\ \mathit{true}. \end{aligned}$$

For $t := a; t$ with $a \in A$, we observe

$$\begin{aligned} & w.(a;t).x = w.(a;t).1 \wedge wlp.(a;t).x \\ \equiv & \{2(1)\} \\ & w.a.(w.t.x) = w.a.(w.t.1) \wedge wlp.a.(wlp.t.x) \\ \equiv & \{(8) \text{ with } x := w.t.x, \text{ and also with } x := w.t.1\} \\ & w.a.1 \wedge wlp.a.(w.t.x) = w.a.1 \wedge wlp.a.(w.t.1) \wedge wlp.a.(wlp.t.x) \\ \Leftarrow & \{\text{calculus}\} \\ & wlp.a.(w.t.x) = wlp.a.(w.t.1) \wedge wlp.a.(wlp.t.x) \\ \Leftarrow & \{\text{Theorem 3.2 with } t := a, \text{ and where } Y \text{ consists of } w.t.1 \text{ and } wlp.t.x\} \\ & w.t.x = w.t.1 \wedge wlp.t.x. \end{aligned}$$

This proves the induction. (End of proof.)

Lemma (d). *The subset* $W1$ *is closed under least upper bounds in* W.

Proof. Let $w1$ be the least bound in W of a subset U of $W1$. For any $a \in A$ and $x \in X$ we observe

$$\begin{aligned} & w1.a.x = w1.a.1 \wedge wlp.a.x \\ \equiv & \{2(5)\}\ (\bigvee u \in U :: u.a.x) = (\bigvee u \in U :: u.a.1) \wedge wlp.a.x \\ \equiv & \{\text{distributivity law (3)}\}\ (\bigvee u \in U :: u.a.x) = (\bigvee u \in U :: u.a.1 \wedge wlp.a.x) \\ \equiv & \{U \subset W1 \text{ and } (8)\}\ \mathit{true}. \end{aligned}$$

This proves that $w1 \in W1$. (End of proof.)

Lemma (e). *The set* $W1$ *is invariant under* $F.wp$.

Proof. Let $w \in W1$ and $x \in X$. For $s \in S$, we observe

$$\begin{aligned} & F.wp.w.s.x = F.wp.w.s.1 \wedge wlp.s.x \\ \equiv & \{2(3)\}\ wp.s.x = wp.s.1 \wedge wlp.s.x \\ \equiv & \{\text{postulate (0)}\}\ \mathit{true}. \end{aligned}$$

For any $h \in H$ we have

$$\begin{aligned}
& F.wp.w.h.x \\
= & \{2(3), \text{use } J.h \text{ as range for } j\} \\
& (\bigwedge j :: \neg b.j \vee w.(r.j).x) \\
= & \{w \in W1 \text{ and Lemma (c)}\} \\
& (\bigwedge j :: \neg b.j \vee (w.(r.j).1 \wedge wlp.(r.j).x)) \\
= & \{\text{distributivity law (2) and calculus}\} \\
& (\bigwedge j :: \neg b.j \vee w.(r.j).1) \wedge (\bigwedge j :: \neg b.j \vee wlp.(r.j).x) \\
= & \{2(3) \text{ with } wg, w := wp, w, \text{ and also with } wg, w := wlp, wlp\} \\
& F.wp.w.h.1 \wedge F.wlp.wlp.h.x \\
= & \{2.6\} \\
& F.wp.w.h.1 \wedge wlp.h.x.
\end{aligned}$$

This proves that $F.wp.w \in W1$. (End of proof).

Remarks. In definition 2(3) we kept the first argument of F explicit because of the proof of this lemma. If it was endurable, that is due to our currying convention. The above theorem can be compared with Hehner's result in [11] Sect. 5.3.2. It should be noticed, however, that the semantics of [11] is defined by means of the limit of the iterated unfoldings, and not by means of extreme fixed points. So, in the case of unbounded nondeterminacy, our semantics differ from those of [11].

3.4. Remark. As is well known, the Theorems 3.2 and 3.3 imply that the predicate transformers $wp.t$ commute with nonempty conjunctions. In fact, for any command t and any nonempty set Y of predicates we have

$$\begin{aligned}
& wp.t.(\bigwedge y \in Y :: y) \\
= & \{3.3\}\ wp.t.1 \wedge wlp.t.(\bigwedge y \in Y :: y) \\
= & \{3.2\}\ wp.t.1 \wedge (\bigwedge y \in Y :: wlp.t.y) \\
= & \{Y \neq \emptyset\}\ (\bigwedge y \in Y :: wp.t.1 \wedge wlp.t.y) \\
= & \{3.3\}\ (\bigwedge y \in Y :: wp.t.y).
\end{aligned}$$

3.5. Determinacy of Commands

In this section we need the negation of predicates with all its usual properties. Therefore, X is supposed to be a complete Boolean lattice, and $\neg$ is the negation operator.

A command t is defined to be *deterministic* if for every predicate x the following holds. If t starts in a state not satisfying $wp.t.x$, then t does not terminate or terminates in a state that satisfies $\neg x$. So, formally, t is deterministic if and only if

$$(\forall x \in X :: \neg wp.t.x \leqq wlp.t.(\neg x)). \qquad (9)$$

A declaration $h ::= (\square j \in J.h : b.j \rightarrow r.j)$ is called *deterministic* if and only if

$$(\forall i, j \in J.h : i \neq j : b.i \wedge b.j = 0). \qquad (10)$$

Theorem. *Let all statements and all declarations be deterministic. Then all commands are deterministic.*

Proof. We define the subset $W2$ of W by

$$w \in W2 \equiv (\forall a \in A, x \in X :: \neg wp.a.x \leqq w.a.(\neg x)). \qquad (11)$$

Below in Lemma (f) we prove that

$$(\forall w \in W2, t \in A^*, x \in X :: \neg wp.t.x \leqq w.t.(\neg x)).$$

Therefore, the theorem is equivalent to $wlp \in W2$. By 2.6, wlp is the biggest fixed point of $F.wlp$ in W. Thus, by the dual version of 1.3 with $V := W2$, it suffices to prove

(g) $W2$ is closed under greatest lower bounds in W.
(h) $W2$ is invariant under $F.wlp$.

These facts are proved in the Lemmas (g) and (h) below. (End of proof.)

Lemma (f). *Let $w \in W2$ and $t \in A^*$. Then $(\forall x \in X :: \neg wp.t.x \leqq w.t.(\neg x))$.*

Proof. We use induction on the length of string t. For $t = \varepsilon$, it suffices by 2(0) to note that $\neg x \leqq \neg x$. For $t := a; t$ with $a \in A$ we have

$$\begin{aligned} & \neg wp.(a;t).x \\ = & \{2(1)\}\ \neg wp.a.(wp.t.x) \\ \leqq & \{(11)\}\ w.a.(\neg wp.t.x) \\ \leqq & \{\text{induction and monotony of } w.a\}\ w.a.(w.t.(\neg x)) \\ = & \{2(1)\}\ w.(a;t).(\neg x). \end{aligned}$$

This proves the induction. (End of proof.)

Lemma (g). *The subset $W2$ is closed under greatest lower bounds in W.*

Proof. Let w be the greatest lower bound in W of a subset U of $W2$. For any $a \in A$ and $x \in X$ we have

$$\begin{aligned} & \neg wp.a.x \leqq w.a.(\neg x) \\ \equiv & \{2(5) \text{ and } 1(0)\}\ (\forall u \in U :: \neg wp.a.x \leqq u.a.(\neg x)) \\ \equiv & \{(11)\}\ (\forall u \in U :: u \in W2). \end{aligned}$$

By (11), this proves that $w \in W2$. (End of proof.)

Lemma (h). *The set $W2$ is invariant under $F.wlp$.*

Proof. For any $w \in W2$ we have

$$\begin{aligned}
& F.wlp.w \in W2 \\
\equiv & \{(11) \text{ and } 2(3)\} \\
& (\forall x \in X, s \in S : : \neg wp.s.x \leqq wlp.s.(\neg x)) \\
& \wedge (\forall x \in X, h \in H : : \neg wp.h.x \leqq F.wlp.w.h.(\neg x)) \\
\equiv & \{\text{all } s \in S \text{ are deterministic, (9), } wp = F.wp.wp\} \\
& (\forall x \in X, h \in H : : \neg F.wp.wp.h.x \leqq F.wlp.w.h.(\neg x)).
\end{aligned}$$

We fix x and h. It remains to observe that

$$\begin{aligned}
& \neg F.wp.wp.h.x \leqq F.wlp.w.h.(\neg x) \\
\equiv & \{2(3), \text{ use } J.h \text{ as range for } i \text{ and } j\} \\
& \neg (\textstyle\bigwedge i : : \neg b.i \vee wp.(r.i).x) \leqq (\textstyle\bigwedge j : : \neg b.j \vee w.(r.j).(\neg x)) \\
\equiv & \{\text{De Morgan}\} \\
& (\textstyle\bigvee i : : b.i \wedge \neg wp.(r.i).x) \leqq (\textstyle\bigwedge j : : \neg b.j \vee w.(r.j).(\neg x)) \\
\equiv & \{1(0) \text{ and } 1(1)\} \\
& (\forall i, j : : b.i \wedge \neg wp.(r.i).x \leqq \neg b.j \vee w.(r.j).(\neg x)) \\
\Leftarrow & \{\text{transitivity of } \leqq\} \\
& (\forall i, j : i \neq j : b.i \leqq \neg b.j) \wedge (\forall j : : \neg wp.(r.j).x \leqq w.(r.j).(\neg x)) \\
\equiv & \{\text{calculus, and } w \in W2 \text{ and Lemma (f)}\} \\
& (\forall i, j \in J.h : i \neq j : b.i \wedge b.j = 0) \\
\equiv & \{\text{all declarations are deterministic, (10)}\} \\
& \textit{true}.
\end{aligned}$$

(End of proof.)

Remark. If command t is total, the inequality in formula (9) is equivalent to equality. In fact, for any command t and any predicate x we have

$$\begin{aligned}
& wlp.t.(\neg x) \leqq \neg wp.t.x \\
\equiv & \{\text{calculus}\}\ wp.t.x \wedge wlp.t.(\neg x) = 0 \\
\equiv & \{3.3\}\ wp.t.1 \wedge wlp.t.x \wedge wlp.t.(\neg x) = 0 \\
\equiv & \{3.2\}\ wp.t.1 \wedge wlp.t.(x \wedge \neg x) = 0 \\
\equiv & \{3.3 \text{ and calculus}\}\ wp.t.0 = 0 \\
\equiv & \{0.1 \text{ and } 0.5\} \text{ command } t \text{ is total.}
\end{aligned}$$

4. Hoare-Triple Methods

4.0. Hoare Triples

Usually, one is interested in establishing specific postconditions, not arbitrary ones. Moreover, one may be happy with suitable preconditions which are not weakest preconditions. Therefore, Hoare triples are introduced as specifications. Hoare triples of recursive procedures generalize both the invariant of a loop, and the variant function.

We define a *Hoare triple* to be a triple $\langle x, h, y\rangle$ with $x, y \in X$ and $h \in H$. Hoare's relation $\{x\}\, h\{y\}$ is represented by $[x \Rightarrow w.h.y]$. So it depends on a semantic function w. In our formalism this relation is represented by $x \leqq w.h.y$, cf. 0.5.

We first treat termination and correctness. For the case of simple recursion, the result is contained in [15] Sect.6. We refer to [7] for the use of well-founded sets in termination proofs.

4.1. Theorem. *Let C be a well-founded set. Let $(\langle x.c, h.c, y.c\rangle : c \in C)$ be a family of Hoare triples such that for every $c \in C$ we have*

$$(\forall d \in C : d < c : x.d \leqq wp.(h.d).(y.d)) \Rightarrow (\forall j \in J.(h.c) :: x.c \leqq \neg b.j \vee wp.(r.j).(y.c)). \quad (0)$$

Then for every $c \in C$, it holds that $x.c \leqq wp.(h.c).(y.c)$.

Proof. By induction over the well-founded set C, this follows from the fact that for every $c \in C$ we have

$$\begin{aligned} & x.c \leqq wp.(h.c).(y.c) \\ \equiv \{2.6\}\ & x.c \leqq F.wp.wp.(h.c).(y.c) \\ \equiv \{2(3)\}\ & x.c \leqq (\textstyle\bigwedge j \in J.(h.c) :: \neg b.j \vee wp.(r.j).(y.c)) \\ \equiv \{1(0)\}\ & (\forall j \in J.(h.c) :: x.c \leqq \neg b.j \vee wp.(r.j).(y.c)) \\ \Leftarrow \{(0)\}\ & (\forall d \in C : d < c : x.d \leqq wp.(h.d).(y.d)). \end{aligned}$$

(End of proof.)

4.2. Remark. The method of proving termination by means of 4.1 is complete for the following reason. Let the set H be ordered by $(h \leqq h') \equiv (h = h')$. This ordering is well-founded. Now the family of Hoare triples $\langle wp.h.1, h, 1\rangle$ with $h \in H$ satisfies formula (0). In fact, that formula reduces to

$$(\forall h \in H, j \in J.h :: wp.h.1 \leqq \neg b.j \vee wp.(r.j).1),$$

and the calculation in the above proof shows that this is *true*. The triviality of this completeness proof indicates that in general it may be difficult to establish formula (0).

4.3. The Induction Theorem of Conditional Correctness

The next result generalizes Hoare's recursion rule, cf. [0] p. 444, to the case of unbounded nondeterminacy and mutual recursion. If T is a set of Hoare triples, a function $w \in W$ is said to satisfy specification T, notation $w \models T$, if and only if

$$(\forall \langle x, h, y\rangle \in T : : x \leqq w.h.y). \tag{1}$$

Let *WLP* denote the subset of W given by

$$w \in WLP \equiv (\forall s \in S : : w.s = wlp.s). \tag{2}$$

Theorem. *Let T be a set of Hoare triples such that*

$$(\forall w \in WLP : w \models T : F.wlp.w \models T). \tag{3}$$

Then $wlp \models T$.

Proof. We define $W3$ to be the subset of W given by

$$w \in W3 \equiv w \in WLP \wedge w \models T. \tag{4}$$

By 2(3), we have $F.wlp.w \in WLP$ for any $w \in W$. Therefore, the set *WLP* is invariant under $F.wlp$. By (3), it follows that $W3$ is invariant under $F.wlp$ as well.

(*Remark.* We are heading for an application of the dual of Theorem 1.3, but *WLP* and $W3$ need not be closed under greatest lower bounds in W. The greatest lower bound of the empty set in W is the function $w1$ given by $w1.a.x = 1$ for all $a \in A$ and $x \in X$. Usually, $w1 \notin WLP$.)

The set *WLP* with the order relation "$\leqq$" inherited from W is a complete partially ordered set. For example, the greatest lower bound in *WLP* of a subset U is the function $w0$ given by $w0 \in WLP$ and

$$(\forall h \in H, x \in X : : w0.h.x = (\bigwedge u \in U : : u.h.x)). \tag{5}$$

The subset $W3$ of *WLP* is closed in *WLP* under greatest lower bounds. For, with U and $w0$ as above, we have

$$\begin{aligned} & w0 \in W3 \\ \equiv & \{(1) \text{ and } (4)\}\ (\forall \langle x, h, y\rangle \in T : : x \leqq w0.h.y) \\ \equiv & \{(5)\}\ (\forall \langle x, h, y\rangle \in T : : x \leqq (\bigwedge u \in U : : u.h.y)) \\ \equiv & \{1(0)\}\ (\forall \langle x, h, y\rangle \in T, u \in U : : x \leqq u.h.y) \\ \equiv & \{(1) \text{ and } (4)\}\ (\forall u \in U : : u \in W3). \end{aligned}$$

Since *wlp* is the greatest fixed point in *WLP* of the monotone function $F.wlp: WLP \rightarrow WLP$, it follows from the dual of Theorem 1.3 that $wlp \in W3$. This proves that $wlp \models T$. (End of proof.)

Remark. This theorem shows that in order to prove $[x \Rightarrow wlp.h.y]$ for all triples $\langle x, h, y\rangle \in T$, it suffices to prove that $[x \Rightarrow F.wlp.w.h.y]$ for all triples $\langle x, h, y\rangle \in T$ and all functions $w \in W$ with

$$(\forall s \in S :: w.s = wlp.s) \wedge (\forall \langle x, h, y\rangle \in T :: [x \Rightarrow w.h.y]).$$

In other words, to prove conditional correctness of a specification T, it suffices to prove correctness of the first unfoldings with respect to all functions w that are equal to wlp on the statements and that satisfy specification T on the procedures.

4.4. Example: A Euclidean Algorithm

We illustrate Theorem 4.3 by means of a nondeterministic and not necessarily terminating implementation of the euclidean algorithm. Procedure h works on a state space of five integer variables p, q, s, t, k. If p and q have initial values m and n, procedure h (if terminating) determines values for s and t such that $s \times m + t \times n$ is the greatest common divisor $gcd.m.n$ of m and n. By convention, $gcd.0.0 = 0$. Formally, procedure h is specified by means of the set T of the Hoare triples $\langle x.m.n, h, y.m.n\rangle$ where m and n range over the integers and where the preconditions $x.m.n$ are given by

$$x.m.n = (p = m \wedge q = n)$$

and the postconditions $y.m.n$ are given by

$$y.m.n = (s \times m + t \times n = gcd.m.n).$$

We now turn to the implementation. Procedure h is declared by

```
h::=(b0: $p \geq 0 \wedge q = 0$ → r0: $s, t := 1, 0$
     ▯b1: true              → r1: $k :=$ arbitrary
                                  ; $p, q := q, p - k \times q$
                                  ; $h$
                                  ; $s, t := t, s - k \times t$
    ).
```

In this declaration we use labels $b0$, $b1$ to indicate the guards, and labels $r0$, $r1$ to indicate the commands. Clearly, procedure h need not terminate. Therefore, we only prove conditional correctness, that is $wlp \models T$. By Theorem 4.3, it suffices to verify formula (3). Let $w \in WLP$ be such that $w \models T$. In order to prove $F.wlp.w \models T$ we observe that

$$\begin{aligned}
&[x.m.n \Rightarrow F.wlp.w.h.(y.m.n)]\\
\equiv&\ \{2(3) \text{ and declaration of } h\}\\
&[x.m.n \Rightarrow (\neg b0 \vee w.r0.(y.m.n)) \wedge (\neg b1 \vee w.r1.(y.m.n))]\\
\equiv&\ \{b1 = \mathit{true} \text{ and calculus}\}\\
&[x.m.n \wedge b0 \Rightarrow w.r0.(y.m.n)] \wedge [x.m.n \Rightarrow w.r1.(y.m.n)]. \qquad (6)
\end{aligned}$$

The two conjuncts of (6) are verified separately. The first conjunct is proved in

$$\begin{aligned}
&[x.m.n \wedge b0 \Rightarrow w.r0.(y.m.n)]\\
\equiv&\ \{w \in WLP \text{ so that } w.r0 = wlp.r0\}\\
&[x.m.n \wedge b0 \Rightarrow wlp.r0.(s \times m + t \times n = gcd.m.n)]\\
\equiv&\ \{r0 = (s, t := 1, 0)\}\\
&[p = m \wedge q = n \wedge p \geqq 0 \wedge q = 0 \Rightarrow 1 \times m + 0 \times n = gcd.m.n]\\
\equiv&\ \{\text{calculus}\}\\
&\mathit{true}.
\end{aligned}$$

In order to prove the second conjunct of formula (6), we write $r1 = (s0;\ s1;\ h;\ s2)$ with $s0$, $s1$, $s2 \in S$. We use a different proof format. The second conjunct is proved in

$$\begin{aligned}
&w.r1.(y.m.n)\\
=&\ \{w \in WLP\}\ w.(s0; s1; h).(wlp.s2.(y.m.n))\\
=&\ \{s2, y.m.n\}\ w.(s0; s1; h).(t \times m + (s - k \times t) \times n = gcd.m.n)\\
=&\ \{\text{calculus}\}\ w.(s0; s1; h).(s \times n + t \times (m - k \times n) = gcd.n.(m - k \times n))\\
\Leftarrow&\ \{w \in WLP, w \models T\}\ wlp.(s0; s1).(p = n \wedge q = m - k \times n)\\
=&\ \{s1\}\ wlp.s0.(q = n \wedge p - k \times q = m - k \times n)\\
=&\ \{\text{calculus}\}\ wlp.s0.(x.m.n)\\
=&\ \{k \text{ does not occur in } x.m.n\}\ x.m.n.
\end{aligned}$$

This concludes the proof of the conditional correctness of h. Conditional correctness in itself is useless. Termination can be effected, however, by guiding the nondeterminacy. In other words,

$$wlp.h.\mathit{false} = \mathit{false}.$$

As it does not illustrate Theorem 4.3, the verification of this fact is left to the reader.

5. Concluding Remarks

We developed the semantics of recursion with unbounded nondeterminacy, expressed in terms of predicate transformers. The predicates were treated as elements of a complete distributive lattice. The main technical features were a strong version of the theorem of Knaster-Tarski, cf. 1.3, a treatment of unfolding by means of the higher order function F of 2.2, and the use of the complete partially ordered set W of 2.3, The proofs of most theorems were straightforward applications of Theorem 1.3.

Acknowledgements. During the academic year 1986–1987, when I was on sabbatical leave in Austin, Texas, I relearned mathematical methodology and predicate calculus from E.W. Dijkstra. My view of mathematics and computing science was influenced immensely. This paper was written at the end of that fruitful year. It was mainly prompted by Dijkstra's appreciation of [17]. I am indebted to the Austin Tuesday Afternoon Club for constructive comments on an early version of the paper, and to the Department of Computing Sciences at Austin for its hospitality and support. Many improvements in the presentation are due to suggestions of the members of the Wednesday Morning Club at Groningen.

References

0. Apt, K.R.: Ten years of Hoare's logic: a survey – Part 1. ACM Trans. Program. Lang. Syst. **3**, 431–483 (1981)
1. Apt, K.R., Plotkin, G.D.: Countable nondeterminism and random assignment. J. ACM **33**, 724–767 (1986)
2. De Bakker, J.W.: Mathematical theory of program correctness. London: Prentice-Hall 1980
3. Balbes, R., Dwinger, P.: Distributive lattices. University of Missouri Press 1974
4. Dijkstra, E.W.: A discipline of programming. London: Prentice-Hall 1976
5. Dijkstra, E.W.: Semantics of straight-line programs (draft of Chap. 4) EWD 910 (1985)
6. Dijkstra, E.W.: The calculus of boolean structures (Part 1). EWD 1002, March 1987
7. Dijkstra, E.W., van Gasteren, A.J.M.: A simple fixpoint argument without the restriction to continuity. Acta Inf. **23**, 1–7 (1986)
8. Dijkstra, E.W., van Gasteren, A.J.M.: On notation, AvG 65a/EWD 950a, January 1986
9. Dijkstra, E.W., Scholten, C.S.: The operational interpretation of extreme solutions. EWD 883 (1984)
10. Hehner, E.C.R.: do considered od: a contribution to programming calculus. Acta Inf. **11**, 287–304 (1979)
11. Hehner, E.C.R.: The logic of programming. New York: Prentice-Hall 1984
12. Hesselink, W.H.: Nondeterminism in data types, a mathematical approach. ACM Trans. Program. Lang. Syst. **10**, 87–117 (1988)
13. Hesselink, W.H.: Interpretations of recursion under unbounded nondeterminacy. Theor. Comput. Sci. **59**, 211–234 (1988)
14. Hoare, C.A.R.: Some properties of predicate transformers. J. ACM **25**, 461–480 (1978)
15. Martin, A.: A general proof rule for procedures in predicate transformer semantics. Acta Inf. **20**, 301–313 (1983)
16. Meyer, J.-J.Ch.: Programming calculi based on fixed point transformations: semantics and applications. Thesis, Vrije Universiteit Amsterdam 1985
17. Nelson, G.: A generalization of Dijkstra's calculus. SRC Report (DEC), April 1987
18. De Roever, W.P.: Dijkstra's predicate transformer, non-determinism, recursion, and termination. In: Mathematical foundations of computer science. LNCS 45, Springer 1976, pp. 472–481
19. Van de Snepscheut, J.L.A.: Trace theory and VLSI design. LNCS 200. Berlin Heidelberg New York: Springer 1985
20. Tarski, A.: A lattice theoretical fixpoint theorem and its applications. Pacific J. Math. **5**, 285–309 (1955)

Received December 3, 1987/August 30, 1988