Weakest Preconditions for Progress

Johan J. Lukkien, Jan L.A. van de Snepscheut
Computer Science
California Institute of Technology
Pasadena, CA 91125

26 March 1991

Abstract Predicate transformers that map the postcondition and all intermediate conditions of a command to a precondition are introduced. They can be used to specify certain progress properties of sequential programs.

1. Introduction

The semantics of a program notation defines the meaning of each program written in that notation. The semantics describes relevant aspects of the execution of the programs by a computer. Execution of a program may be viewed as a, possible nonterminating, sequence of state transitions. Program semantics described by wlp and wp predicate transformers (cf. [3,4]) relates initial states and final states, if any. One of the attractive aspects of this style of semantic definitions is that it supports the development of programs: program verification and construction boil down to calculation with predicates, sometimes referred to as equational reasoning (cf. [3,5]).

The two predicate transformers wlp and wp relate initial and final states only and are independent of the intermediate states reached by program execution. The advantages are many, both to the programmer and to the implementer of the program notation. In some cases, however, properties of the intermediate states are of interest. Owicki and Lamport (cf. [9]) proposed the study of liveness properties in the context of concurrent programs. They were the first to to introduce the property p *leads–to* q. In [2], Chandy and Misra show how properties like p *leads–to* q can be used to good advantage in the specification and design of programs. If p *leads–to* q is a property of the program, then this expresses the fact that if a state satisfying p is encountered during execution of the program, it will eventually be followed by a state satisfying q. Such a property is called a progress property. (Here and in the remainder of this note, 'followed by' means that the second state coincides with the first, or comes later during execution of the program.)

A major difference in the semantic style between UNITY logic and the wp calculus is that properties in UNITY logic are not calculated through equational reasoning but are derived from axioms and inference rules. Although p and q are predicates, p *leads–to* q is not a predicate: it is a property that may or may not be derivable. Constructs like p *leads–to* $(q$ *leads–to* $r)$ are out of the question. In this note we propose predicate transformers for expressing progress. They relate the initial state with intermediate states and the final state. Because we take the final state into account, sequential composition fits in well. However, it turns out that we win and loose: although it is easy to deal with sequential composition, parallel composition is hard. In this note we do not discuss parallel composition at all.

We begin our study with an overview of results on predicates and predicate transformers as used in this context. It provides the notation and terminology for the remainder of this paper. Next, we link a proposed predicate transformer for expressing progress properties to an operational interpretation of program execution. This leads to a set of requirements that seem reasonable to expect from any predicate transformer that satisfies the interpretation. In the following section we define the predicate transformer by induction over the program structure, and we establish some theorems about it. Finally, we repeat these steps for another progress property and its predicate transformer.

2. A summary of results in predicate calculus

We use predicates to express conditions that may or may not hold in a certain state during program execution. Because of the state's omnipresence we would like it to be as anonymous as possible. Therefore, we have the following two conventions. If e and f are expressions and $\oplus$ is an operator of the appropriate type, $e \oplus f$ is also an expression. The value of $e \oplus f$ is in every point in the state space given by the value of e in that point combined through $\oplus$ with the value of f in the same point, i.e. operator $\oplus$ is applied pointwise. This is the usual interpretation for, say arithmetic operations, and we stick to the same rule for boolean operations. Hence, operators $\vee$ and $\wedge$ are applied pointwise, as usual, and so are $\equiv$ and $\Rightarrow$, which is unusual. The price that we have to pay for this convention is that $P \equiv Q$ does not express the fact that P and Q are the same predicates, i.e. that P and Q are boolean expressions whose values are pointwise equivalent, but it is again a predicate: a boolean expression that is $true$ in those points where P and Q are equal, and $false$ elsewhere. This is where the second convention comes in. We write $[P]$ to denote universal quantification of P over all points in the state space. (Exactly what the state space is depends on the collection of program variables at hand and the advantage of the square brackets is that this collection can be left anonymous under the assumption that the collection of program variables is constant.) If P is a predicate that is $true$ in every point in the state space, $[P]$ is $true$, and $[P]$ is $false$ for each predicate that is $false$ in at least one point in the state space. Hence $[P \equiv Q]$ is $true$ if P and Q are the same predicates, i.e. have the same value in every point in the state space, and $false$ otherwise. This would traditionally be written as $P = Q$ and we avoid confusion by never applying $=$ to predicates.

As a result of our conventions, for arithmetic expressions e and f, $(e+f)\cdot(e-f)$ is an arithmetic expression, $(e+f)\cdot(e-f) \;=\; e^2+f^2$ is a boolean expression that is $true$ in those points in space where f is 0, and $false$ elsewhere. Notice that $(e+f)\cdot(e-f) \;=\; e^2-f^2$ is a boolean expression that happens to be $true$ in every point in space. Consequently $[(e+f)\cdot(e-f) \;=\; e^2-f^2]$ is the boolean $true$. Hence, we use the square brackets to express statements of fact.

Observe that without quantifications all our expressions would be pointwise. We know from experience that many interesting functions are not pointwise, such as the determinant of a matrix.

It turns out that we are often proving properties like $[A \Rightarrow C]$, for predicates A and C in a number of steps by showing for example $[A \equiv B]$ and $[B \Rightarrow C]$ for judiciously chosen intermediate predicate B. If the predicate B is long, as is often the case in our proofs, we do not want to write it twice as $[A \equiv B]$ and $[B \Rightarrow C]$. In such a case we will write

$$\begin{array}{ll} & A \\ \equiv & \{ \text{ hint why } [A \equiv B] \ \} \\ & B \\ \Rightarrow & \{ \text{ hint why } [B \Rightarrow C] \ \} \\ & C \quad . \end{array}$$

Besides implication and equivalence for connecting lines in a proof, we use the reverse of implication, called follows from and written $\Leftarrow$.

We also follow the convention introduced in [4] to write function application with an explicit period rather than implicitly, without a symbol. Hence, $f.x$ is function f applied to argument x. Function application has a higher binding power than the other operators. The highest binding power of these is assigned to $\neg$, next come $\wedge$ and $\vee$, and finally $\equiv$, $\Rightarrow$, and $\Leftarrow$.

Quantified expressions mention the bound variables. For example $\langle \forall r : r \in R : q \vee r \rangle$ is a universal quantification over terms of the form $q \vee r$ where q is a free variable and r is a bound variable that ranges over all values in set R. If the range of r is understood we abbreviate to $\langle \forall r :: q \vee r \rangle$.

For f a predicate transformer, i.e. a function from predicates to predicates, we say that f is monotonic if

$$[p \Rightarrow q] \Rightarrow [f.p \Rightarrow f.q]$$

for all predicates p and q. We say that f is universally conjunctive if

$$[f.\langle \forall r : r \in R : r \rangle \equiv \langle \forall r : r \in R : f.r \rangle]$$

holds for all sets of predicates R, independent of whether R is finite, infinite, or empty. If the property holds for all nonempty sets R, we say that f is positively conjunctive. Clearly, universal conjunctivity implies positive conjunctivity. We give a little theorem that will be used a lot. It shows that conjunctivity implies monotonicity.

Theorem

If f is positively conjunctive then f is monotonic.

Proof For all predicates x and y,

$$\begin{array}{ll} & [x \Rightarrow y] \\ \equiv & \\ & [x \wedge y \equiv x] \\ \Rightarrow & \{ \text{ Leibniz's rule } \} \\ & [f.(x \wedge y) \equiv f.x] \\ \equiv & \{ \text{ conjunctivity } \} \\ & [f.x \wedge f.y \equiv f.x] \\ \equiv & \\ & [f.x \Rightarrow f.y] \quad . \end{array}$$

(End of proof)

In the proof we have referred to the rule of Leibniz. It is often paraphrased as "substitution of equals for equals". There is a dual theorem for disjunctive function, i.e. for functions that distribute over disjunction.

Theorem

If f is positively disjunctive then f is monotonic.

A "half-sided" counter-part to these theorems exists. For monotonic f, and any set P of predicates,

$$[f.\langle\forall p : p \in P : p\rangle \ \Rightarrow\ \langle\forall p : p \in P : f.p\rangle] \qquad \text{and} \tag{0}$$

$$[f.\langle\exists p : p \in P : p\rangle \ \Leftarrow\ \langle\exists p : p \in P : f.p\rangle] \qquad . \tag{1}$$

Sometimes, we have to solve equations in predicates. They are written in the form

y : [some predicate in y]

and y is a predicate variable. We say that y is a solution if [some predicate in y] is *true*. Some equations have no solution, some have one solution, and others have more than one solution. Our equations generally fall in the latter category. They are often of the form

$$y : [y \ \equiv\ f.y] \tag{2}$$

for monotonic predicate transformer f. Sometimes we encounter

$$y : [y \ \Leftarrow\ f.y] \tag{3}$$

and its dual

$$y : [y \ \Rightarrow\ f.y] \qquad . \tag{4}$$

Observe that (3) has at least one solution, viz. *true*, but in general it has more. Similarly, *false* is a solution of (4). For monotonic f, the conjunction of an arbitrary set of solutions of (3) is not necessarily a solution of (3). The following theorem shows, however, that the conjunction of all solutions is a solution. This conjunction q is special in the sense that it is the strongest solution, i.e. $[q \ \Rightarrow\ z]$ for every solution z. If f is not only monotonic but also universally conjunctive then the conjunction of any set of solutions is a solution.

Theorem

For monotonic f, equation (3) has a strongest solution, viz., the conjunction of all solutions of (3). Similarly, (4) has a weakest solution, viz., the disjunction of all solutions of (4).

Proof Let S be the set of all solutions of equation (3), and let q be their conjunction, i.e. $[q \ \equiv\ \langle\forall z : z \in S : z\rangle]$. We formulate $z \in S$ as $[z \ \Leftarrow\ f.z]$ and we prove $q \in S$ by showing $[q \ \Leftarrow\ f.q]$.

$$\begin{array}{ll}
 & f.q \\
\equiv & \qquad \{ \text{ definition of } q \ \} \\
 & f.\langle\forall z : [z \ \Leftarrow\ f.z] : z\rangle \\
\Rightarrow & \qquad \{ \ f \text{ is monotonic, (0) } \} \\
 & \langle\forall z : [z \ \Leftarrow\ f.z] : f.z\rangle \\
\Rightarrow & \qquad \{ \text{ quantification ranges only over } z \text{ for which } f.z \text{ implies } z \ \} \\
 & \langle\forall z : [z \ \Leftarrow\ f.z] : z\rangle \\
\equiv & \qquad \{ \text{ definition of } q \ \} \\
 & q
\end{array}$$

(End of proof)

Solutions to equations (2), (3), and (4) are related by the following important theorem, attributed to Knaster and Tarski (cf. [10]).

Theorem (Knaster-Tarski)

For monotonic f, equations (2) and (3) have the same strongest solution, and (2) and (4) have the same weakest solution.

Proof We restrict ourselves to (2) and (3). From the previous theorem we know that (3) has a strongest solution q, i.e. we have

$$[f.z \;\Rightarrow\; z] \;\Rightarrow\; [q \;\Rightarrow\; z] \quad \text{for all } z \tag{5}$$

and

$$[f.q \;\Rightarrow\; q] \quad . \tag{6}$$

It remains to show that q is a solution of (2), i.e. $[f.q \;\equiv\; q]$, and that q is stronger than all other solutions, i.e. $[f.z \;\equiv\; z] \;\Rightarrow\; [q \;\Rightarrow\; z]$ for all z. We have

$$\begin{array}{ll} & [q \;\Rightarrow\; z] \\ \Leftarrow & \qquad \{\ (5)\ \} \\ & [f.z \;\Rightarrow\; z] \\ \Leftarrow & \\ & [f.z \;\equiv\; z] \end{array}$$

and

$$\begin{array}{ll} & [f.q \;\equiv\; q] \\ \equiv & \\ & [f.q \;\Rightarrow\; q] \;\;\wedge\;\; [f.q \;\Leftarrow\; q] \\ \equiv & \qquad \{\ (6)\ \} \\ & [f.q \;\Leftarrow\; q] \\ \Leftarrow & \qquad \{\ (5) \text{ with } f.q \text{ for } z\ \} \\ & [f.(f.q) \;\Rightarrow\; f.q] \\ \Leftarrow & \qquad \{\ f \text{ is monotonic}\ \} \\ & [f.q \;\Rightarrow\; q] \\ \equiv & \qquad \{\ (6)\ \} \\ & \mathit{true} \quad . \end{array}$$

(End of proof)

The function f that figures in our equations usually depends on another predicate also, z say. Therefore, we consider equation

$$y : [y \;\equiv\; f.z.y] \quad . \tag{7}$$

According to Knaster-Tarski, if f is monotonic in y, (7) has a strongest solution, $g.z$ say, and a weakest solution, $h.z$ say. We mention some additional results (cf. [4]). If f is universally or positively conjunctive in (z,y), h is universally or positively conjunctive in z respectively. If f is universally or positively disjunctive in (z,y), g is universally or positively disjunctive in z respectively. Function f is said to be universally conjunctive in (z,y) if

$$[\langle\forall z,y : z \in Z \wedge y \in Y : f.z.y\rangle \;\equiv\; f.\langle\forall z : z \in Z : z\rangle.\langle\forall y : y \in Y : y\rangle]$$

for all Z and Y. The other definitions are similar. If f is conjunctive, we have a property that relates g and h. This property is crucial in the remainder.

$$[g.(x \wedge y) \;\equiv\; g.x \quad \wedge \quad h.y] \tag{8}$$

In the cases that we encounter, $f.z.y$ is of the form $z \wedge k.y$, where z is independent of y, and k is conjunctive. We may then rewrite equation (7) as

$$y : [y \;\equiv\; z \wedge k.y] \tag{9}$$

We have the following property.

Theorem For any solution x of (9),

$$x \wedge h.z \text{ is the weakest solution of } \; y : [y \;\equiv\; z \wedge k.(x \wedge y)] \quad . \tag{10}$$

Proof First, we show that $x \wedge h.z$ solves the equation in (10).

$$\begin{array}{ll}
 & x \wedge h.z \\
\equiv & \qquad \{ \; x \text{ and } h.z \text{ solve } (9) \; \} \\
 & z \wedge k.x \wedge z \wedge k.(h.z) \\
\equiv & \qquad \{ \; \text{calculus} \; \} \\
 & z \wedge k.x \wedge k.(h.z) \\
\equiv & \qquad \{ \; k \text{ is conjunctive} \; \} \\
 & z \wedge k.(x \wedge h.z)
\end{array}$$

We rewrite the equation in (10) by observing

$$\begin{array}{ll}
 & z \wedge k.(x \wedge y) \\
\equiv & \qquad \{ \; k \text{ is conjunctive} \; \} \\
 & z \wedge k.x \wedge k.y \\
\equiv & \\
 & z \wedge k.x \wedge z \wedge k.y \\
\equiv & \qquad \{ \; x \text{ solves } (9) \; \} \\
 & x \wedge z \wedge k.y
\end{array}$$

which yields equation

$$y : [y \;\equiv\; x \wedge z \wedge k.y]$$

with weakest solution $h.(x \wedge z)$. It remains to show that the latter implies $x \wedge h.z$.

$$\begin{array}{ll}
 & h.(x \wedge z) \\
\equiv & \qquad \{ \; h \text{ is conjunctive} \; \} \\
 & h.x \wedge h.z \\
\Rightarrow & \qquad \{ \; [h.x \;\equiv\; x \wedge k.(h.x)] \text{ hence } [h.x \;\Rightarrow\; x] \; \} \\
 & x \wedge h.z
\end{array}$$

(End of proof)

3. Semantics of programs through wlp and wp

In this section we summarize the definitions of predicate transformers wlp and wp as given in [3,4,5]. The interpretation of wlp and wp is discussed in the next section; here we just list the definition and the requirements that are imposed on the definition of wlp and wp for all programs s in order to admit a sensible interpretation. The two requirements, or healthiness conditions are (R0) and (R1).

$$wlp.s \text{ is universally conjunctive} \tag{R0}$$

$$[wp.s.r \;\equiv\; wp.s.true \wedge wlp.s.r] \text{ for all } r \tag{R1}$$

There is a third requirement, usually referred to as the Law of the Excluded Miracle,

$$[wp.s.false \;\equiv\; false] \tag{R2}$$

but we have no use for it and ignore it in this paper. The definition of wlp and wp is by induction over the structure of the program.

$$[wlp.skip.r \;\equiv\; r]$$

$$[wp.skip.r \;\equiv\; r]$$

$$[wlp.abort.r \;\equiv\; true]$$

$$[wp.abort.r \;\equiv\; false]$$

$$[wlp.(s;t).r \;\equiv\; wlp.s.(wlp.t.r)]$$

$$[wp.(s;t).r \;\equiv\; wp.s.(wp.t.r)]$$

$$[wlp.(x := e).r \;\equiv\; r^x_e]$$

$$[wp.(x := e).r \;\equiv\; r^x_e]$$

For $IF = \mathbf{if}\langle[\!]\, i :: b_i \rightarrow s_i\rangle\mathbf{fi}$

$$[wlp.IF.r \;\equiv\; \langle\forall i : b_i : wlp.s_i.r\rangle]$$

$$[wp.IF.r \;\equiv\; \langle\forall i : b_i : wp.s_i.r\rangle \quad\wedge\quad \langle\exists i :: b_i\rangle]$$

For $DO = \mathbf{do}\ b \rightarrow s\ \mathbf{od}$ we obtain a defining equation by equating DO and its first unfolding $\mathbf{if}\ b \rightarrow s; DO\ [\!]\ \neg b \rightarrow skip\ \mathbf{fi}$. If the two are to be equal, the predicate transformers of DO and its unfolding are to be equal also, i.e.

$$[wlp.DO.r \;\equiv\; (b \vee r) \wedge (\neg b \vee wlp.s.(wlp.DO.r))]$$

$$[wp.DO.r \;\equiv\; (b \vee r) \wedge (\neg b \vee wp.s.(wp.DO.r))] \qquad .$$

We want to use the equations for defining the left-hand sides, but the two equations may have many solutions. Using (R1), the latter equation is equivalent to

$$[wp.DO.r \;\equiv\; (b \vee r) \wedge (\neg b \vee wp.s.true) \wedge (\neg b \vee wlp.s.(wp.DO.r))] \qquad .$$

Abbreviating $\mathbf{if}\ b \rightarrow s\ \mathbf{fi}$ to IF, we see that $wlp.DO.r$ and $wp.DO.r$ are solutions of equations (11) and (12).

$$y : [y \;\equiv\; (b \vee r) \wedge wlp.IF.y] \tag{11}$$

$$y : [y \;\equiv\; (b \vee r) \wedge (\neg b \vee wp.s.true) \wedge wlp.IF.y] \tag{12}$$

Both (11) and (12) are of the form

$$y : [y \;\equiv\; z \wedge wlp.IF.y] \tag{13}$$

and differ in the choice for z only. Since $z \wedge wlp.IF.y$ is monotonic, (13) has a strongest solution $g.z$ and a weakest solution $h.z$. The predicate transformer $wlp.DO$ is chosen to be the weakest solution of its defining equation, and $wp.DO$ is chosen to be the strongest solution of its defining equation.

$$[wlp.DO.r \ \equiv \ h.(b \vee r)] \tag{14}$$

$$[wp.DO.r \ \equiv \ g.((b \vee r) \wedge (\neg b \vee wp.s.true))] \tag{15}$$

The last line is (cf. (8)) equivalent to

$$[wp.DO.r \ \equiv \ h.(b \vee r) \wedge g.(\neg b \vee wp.s.true)] \qquad . \tag{16}$$

The definitions of $wlp.DO$ and $wp.DO$ as given above are as in [4]. The definitions given in [3,5] are different: they give a definition in terms of the limit of a sequence. The two definitions are equivalent if the predicate transformer of the repeated statement, i.e. $wlp.s$ or $wp.s$, is continuous. (A predicate transformer f is continuous if

$$[\langle \exists p : p \in P : f.p \rangle \ \equiv \ f.\langle \exists p : p \in P : p \rangle]$$

for every nonempty set of predicates P that can be totally ordered into a sequence by implication.) If it is merely conjunctive the definitions are different and the limit need not solve the equations on which the present definitions are based. Restricting statements to continuous ones makes it hard, if not impossible, to construct programs by stepwise refinement, as argued in [0].

4. An operational appreciation of predicate transformers

In this section we introduce a classification of program executions. An execution of a program can be viewed as a sequence of states; the actual sequence depends on the program text and on implementation choices. One classification is based on termination: execution does or does not terminate. In the latter case, we say that we have an eternal execution. Executions that terminate do so in some final state, and they are further distinguished by whether or not they satisfy some given condition on the final state. The condition is expressed as a predicate r on the program's state space, and each state satisfies either r or $\neg r$. The wlp and wp calculus is based on partitioning the executions of a (fixed) program s into three classes:

eternal : the executions that do not terminate,

finally r : the executions that terminate in a final state satisfying r,

finally $\neg r$: the executions that terminate in a final state satisfying $\neg r$.

Every execution falls into exactly one of these three classes. Predicate transformers wlp and wp can be defined in terms of these classes. For program s, predicates $wlp.s.r$ and $wp.s.r$ are as follows.

$wlp.s.r$: holds in those initial states for which no execution of s belongs to the class 'finally $\neg r$'

$wp.s.r$: holds in those initial states for which no execution of s belongs to the class 'eternal' or to the class 'finally $\neg r$'

Predicate transformer wlp corresponds to what is called partial correctness whereas wp corresponds to what is called total correctness. Usually, wp is used in the specification of programs, whereas wlp has more attractive mathematical properties. The two are related through

$$[wp.s.r \ \equiv \ wp.s.true \wedge wlp.s.r] \qquad . \tag{R1}$$

In this paper we propose a different characterization by considering a predicate q and partition executions in two classes, depending on whether q holds at some state. We consider all states: the initial state, the final state (if any), and all states in between.

ever q : the executions such that q holds in some state

never q : the executions such that q holds in no state

Observe that this partitioning is independent of the previous one: an execution in 'ever q' can be in 'eternal', 'finally r', or in 'finally $\neg r$', and so can an execution in 'ever $\neg q$'. Of course, an execution in 'finally q' is in 'ever q'.

As indicated before, we aim at a predicate transformer that maps predicate q to a predicate on the initial state (the precondition) such that an execution is in 'ever q' if the precondition is satisfied. Since we anticipate sequential composition of programs, and since the state in which q holds during execution of $(s;\ t)$ may, loosely speaking, be either a state of s or of t, we are also interested in the state in which s terminates. For a fixed program we, therefore, partition program executions into four classes.

ever q : the executions such that q holds in some state

never q and eternal : the executions such that q holds in no state and that do not terminate

never q and finally r : the executions such that q holds in no state and that terminate in a final state satisfying r

never q and finally $\neg r$: the executions such that q holds in no state and that terminate in a final state satisfying $\neg r$.

For given s, q, and r, this classification leads us to define two predicates.

$wlev.s.q.r$: holds in those initial states for which no execution of s belongs to the class 'never q and finally $\neg r$'

$wev.s.q.r$: holds in those initial states for which no execution of s belongs to the class 'never q and eternal' or to the class 'never q and finally $\neg r$'

The name wev is derived from 'weakest ever' and is chosen to reflect the property that $wev.s.q.r$ is the weakest condition on the initial state such that execution of s is guaranteed to ever visit a state that satisfies q or to terminate in a state that satisfies r. The l in $wlev$ stands for $wlev$ being the liberal version of wev. In a later section we discuss an even more involved predicate transformer that expresses properties like 'a state in which p holds is eventually followed by a state in which q holds'. We postpone the introduction thereof until after we have studied $wlev$ and wev.

5. Requirements on predicate transformers

Not all predicate transformers can meaningfully be interpreted as a $wlev.s$ or as a $wev.s$ for some program s. We discuss a number of requirements that are motivated by operational concerns. When we define $wlev$ and wev for a variety of statements in the next section, we check that these requirements are met.

If we choose for q the predicate *false*, the class 'ever q' is empty and the partitioning into the remaining three classes coincides with the partitioning underlying the definition of $wlp.s$ and

$wp.s$. Hence, we require that substitution of *false* for q reduces $wlev$ and wev to wlp and wp respectively.

$$[wlev.s.false.r \;\equiv\; wlp.s.r] \qquad \text{(E0)}$$

$$[wev.s.false.r \;\equiv\; wp.s.r] \qquad \text{(E0')}$$

Since wlp is universally conjunctive, and because of (E0), we expect a similar conjunctive property for $wlev$. What exactly should the requirement be? Let R be any (possible empty) set of predicates. We have

no execution belongs to the class 'never q and finally $\neg\langle\forall r : r \in R : r\rangle$ '

$\equiv$

$\langle\forall r : r \in R$: no execution belongs to the class 'never q and finally $\neg r$ '$\rangle$

and conclude that we need to require universal conjunctivity of $wlev$: for each q and each program s

$wlev.s.q$ is universally conjunctive . (E1)

A consequence of (E1) is (take the conjunction of zero predicates)

$$[wlev.s.q.true] \quad . \qquad (17)$$

We have already mentioned formula (R1), which expresses the relation between wlp and wp. Since wlp and wp are in turn related to $wlev$ and wev we expect a similar relation between the latter two. We have

$wev.s.q.true$: holds in those initial states for which no execution of s belongs to the class 'never q and eternal'

and conclude that, for each program s, and all predicates q and r, we need to require

$$[wev.s.q.r \;\equiv\; wev.s.q.true \wedge wlev.s.q.r] \quad . \qquad \text{(E2)}$$

On account of (17) both left- and right-hand side of (E2) reduce to $wev.s.q.true$. From (E1) and (E2) it follows that for each program s, and all predicates q and r, we have

$wev.s.q$ is positively conjunctive . (E1')

So much for the properties that were inspired by wlp and wp. The remaining three properties are peculiar to $wlev$ and wev. If a state satisfies some predicate q, it also satisfies every weaker predicate q', and therefore every execution in class 'ever q ' is also in the larger class 'ever q' '. As a result we expect $wev.s.q.r$ and $wlev.s.q.r$ to be monotonic in q. This leads to our next requirements.

$$[q \Rightarrow q'] \Rightarrow [wlev.s.q.r \Rightarrow wlev.s.q'.r] \qquad \text{(E3)}$$

$$[q \Rightarrow q'] \Rightarrow [wev.s.q.r \Rightarrow wev.s.q'.r] \qquad \text{(E3')}$$

Observe that neither of (E3) and (E3'), combined with (E2), implies the other one. Finally we consider the fact that the initial and final states of an execution are also intermediate states. This suggests the requirements

$$[wlev.s.q.r \;\equiv\; wlev.s.q.(q \vee r)] \qquad \text{(E4)}$$

$$[q \Rightarrow wev.s.q.r] \qquad \text{(E5')}$$

which in combination with (E2) imply

$$[wev.s.q.r \;\equiv\; wev.s.q.(q \vee r)] \tag{E4'}$$

$$[q \Rightarrow wlev.s.q.r] \quad . \tag{E5}$$

This completes our list of requirements. We conclude this section with a little theorem. It expresses that we can strengthen the postcondition of a $wlev$ with that of a wlp. If either the $wlev$ or wlp is replaced by its nonliberal counterpart, the conclusion is nonliberal.

Theorem

For all programs s, and predicates q, r, and w, we have

$$[wlev.s.q.r \wedge wlp.s.w \Rightarrow wlev.s.q.(r \wedge w)] \tag{18a}$$

$$[wlev.s.q.r \wedge wp.s.w \Rightarrow wev.s.q.(r \wedge w)] \tag{18b}$$

$$[wev.s.q.r \wedge wlp.s.w \Rightarrow wev.s.q.(r \wedge w)] \tag{18c}$$

$$[wev.s.q.r \wedge wp.s.w \Rightarrow wev.s.q.(r \wedge w)] \tag{18d}$$

Proof We give the proof for (18a); the other three are similar.

$$\begin{array}{ll} & wlev.s.q.r \wedge wlp.s.w \\ \equiv & \quad \{ \text{ (E0) } \} \\ & wlev.s.q.r \wedge wlev.s.false.w \\ \Rightarrow & \quad \{ \text{ (E3) } \} \\ & wlev.s.q.r \wedge wlev.s.q.w \\ \equiv & \quad \{ \text{ (E1) } \} \\ & wlev.s.q.(r \wedge w) \end{array}$$

(End of proof)

One might be tempted to replace the implications in this theorem by equivalences. The example

$$\begin{array}{ll} s \;=\; \textbf{if} & true \rightarrow \cdots \{q\} \cdots \{r \wedge \neg w\} \\ & [\!]\; true \rightarrow \cdots \{r \wedge w\} \\ \textbf{fi} & \end{array}$$

(in which $\cdots$ stands for a terminating statement that establishes the condition following it) shows that such temptations had better be resisted since

$$[wlev.s.q.(r \wedge w) \;\equiv\; true]$$

$$[wlev.s.q.r \;\equiv\; true]$$

$$[wlp.s.q.w \;\equiv\; false] \quad .$$

We have a long list of requirements on $wlev$ and wev and one wonders whether they are independent. Here we give a number of functions that meet each but one of the requirements for $wlev$. A similar list can be compiled for wev. For $wlev.s.q.r$ that meets all requirements except one, choose

all except (E0) : $true$

all except (E1) : $wlp.s.(q \vee r) \vee q \vee (\neg[\neg q] \wedge \neg r)$

all except (E2) : not applicable, because we consider $wlev$ only

all except (E3) : $q \;\vee\; (([\neg q] \vee r) \wedge wlp.s.(q \vee r))$

all except (E4) : $q \vee wlp.s.r$

all except (E5) : $wlp.s.(q \vee r)$

We omit the proof that, for example, the choice $q \vee wlp.s.r$ satisfies (E0) through (E5). The most complicated case is $wlp.s.(q \vee r) \vee q \vee (\neg[\neg q] \wedge \neg r)$ which meets all requirements except (E1), the conjunctivity. An example is: substitute $x := 4$ for s, *false* for q, and $\{false, x = 3\}$ for the set of predicates R.

One might also wonder whether our requirements capture all properties of *wlev* and *wev*. This is not the case. For example, assume that one has an implementation that executes a program without any intermediate states, i.e., it proceeds in one step from initial to final state. The requirements that we have discussed do not address this issue at all. Therefore, one may expect that the pair consisting of $q \vee wlp.s.(q \vee r)$ and $q \vee wp.s.(q \vee r)$ satisfies all requirements. Fortunately, it does. In the next section we discuss a definition of *wlev* and *wev* by induction on the structure of programs. The definitions admit the traditional operational interpretation of programs (cf. [4]) in which, for example, an intermediate state exists between execution of s and t in $(s;\ t)$.

6. Definition of *wlev* and *wev*

In this section we give definitions of *wlev* and *wev* by induction on the structure of the programs. For each construct we have to verify that requirements (E0), (E0'), (E1), (E2), (E3), (E3'), (E4), and (E5') are met. Most proofs are omitted; only the proofs for the case of the iterative command are listed in the appendix.

We begin our list of definitions with perhaps the simplest statement of all: *skip*, the statement that does not change the state whatsoever. The definition of *skip* is

$$[wlev.skip.q.r \ \equiv\ q \vee r]$$

$$[wev.skip.q.r \ \equiv\ q \vee r] \qquad .$$

The second statement that we define is *abort*. It is the statement that has an initial state but no other intermediate state and has no final state. Or, equivalently, it repeats the initial state indefinitely. This suggests the following definition.

$$[wlev.abort.q.r \ \equiv\ true]$$

$$[wev.abort.q.r \ \equiv\ q]$$

The next construct that we define is sequential composition. The execution of $s;t$ is the execution of s, followed by execution of t if s terminates. Hence, we are led to define

$$[wlev.(s;t).q.r \ \equiv\ wlev.s.q.(wlev.t.q.r)]$$

$$[wev.(s;t).q.r \ \equiv\ wev.s.q.(wev.t.q.r)] \qquad .$$

The simplicity of these two definitions was one of our motivations for introducing predicate transformers that operate on two predicates in the first place.

(Intermezzo) This is the point where the second predicate, r, plays its role. We tried to avoid r and defined $wev.s.q$ to be the weakest precondition such that execution of s would ever visit a state in which q holds. The problem is that a definition like

$$[wev.(s;t).q \ \equiv\ wev.s.q \vee wp.s.(wev.t.q)]$$

is inappropriate because of possible nondeterminism in s.

We show that a few properties that one might expect on operational grounds are indeed theorems. First we show that $skip$ is the left and right unit element of sequential composition. Next we show that $abort$ is the left zero element, and finally we show that sequential composition is associative.

Theorem

For all programs s we have

$s; skip \;=\; s \;=\; skip; s \qquad .$

Proof Equality of programs depends on the context: it is with respect to the properties that one looks at. In our present context, equality of programs means that their predicate transformers $wlev$ and wev are equal. It follows from (E0) and (E0') that wlp and wp are special cases of $wlev$ and wev. Therefore, equality of programs with respect to $wlev$ and wev implies equality with respect to wlp and wp. We give the proof for equality with respect to wev only; the proof for $wlev$ is similar. For all s, q, and r we have

$$
\begin{array}{ll}
 & wev.(s;skip).q.r \\
\equiv & \qquad \{ \text{ definition } ; \; \} \\
 & wev.s.q.(wev.skip.q.r) \\
\equiv & \qquad \{ \text{ definition } skip \; \} \\
 & wev.s.q.(q \vee r) \\
\equiv & \qquad \{ \text{ (E4') } \} \\
 & wev.s.q.r \\
\equiv & \qquad \{ \text{ (E5') } \} \\
 & q \vee wev.s.q.r \\
\equiv & \qquad \{ \text{ definition } skip \; \} \\
 & wev.skip.q.(wev.s.q.r) \\
\equiv & \qquad \{ \text{ definition } ; \; \} \\
 & wev.(skip;s).q.r \qquad .
\end{array}
$$

(End of proof)

Theorem

For all programs s we have

$abort; s \;=\; abort$

Proof We give the proof for equality with respect to wev only; the proof for $wlev$ is similar. For all s, q, and r we have

$$
\begin{array}{ll}
 & wev.(abort;s).q.r \\
\equiv & \qquad \{ \text{ definition } ; \; \} \\
 & wev.abort.q.(wev.s.q.r) \\
\equiv & \qquad \{ \text{ definition } abort \; \} \\
 & q \\
\equiv & \qquad \{ \text{ definition } abort \; \} \\
 & wev.abort.q.r \qquad .
\end{array}
$$

(End of proof)

In general, *abort* is not the right zero element of sequential composition. If a state is reached by execution of s from some initial state then that same state is reached by executing $s; abort$. Hence, $wev.(s; abort).q.r$ is weaker, holds in more initial states, than $wev.abort.q.r$ does.

Theorem

Sequential composition is associative.

Proof We give the proof for associativity with respect to wev only; the proof for $wlev$ is similar. For all s, t, u, q, and r we have

$$\begin{array}{ll} & wev.(s;(t;u)).q.r \\ \equiv & \quad \{ \text{ definition } ; \ \} \\ & wev.s.q.(wev.(t;u).q.r) \\ \equiv & \quad \{ \text{ definition } ; \ \} \\ & wev.s.q.(wev.t.q.(wev.u.q.r)) \\ \equiv & \quad \{ \text{ definition } ; \ \} \\ & wev.(s;t).q.(wev.u.q.r) \\ \equiv & \quad \{ \text{ definition } ; \ \} \\ & wev.((s;t);u).q.r \qquad . \end{array}$$

(End of proof)

We continue our list of definitions and tackle the only statement that changes the state: the assignment statement. We consider the assignment $x := e$ and ignore the concurrent assignment. The definition uses a function dom. We have $dom.e = true$ if the evaluation of e is within its domain, and $dom.e = false$ otherwise. For example, $dom.(1/x) \;=\; (x \neq 0)$. The operational interpretation of the assignment $x := e$ reveals that there are two states: one before and one after the assignment. They differ in the value for the x coordinate only.

$$wlev.(x := e).q.r \;\equiv\; q \vee (\neg dom.e \;\textbf{cor}\; (q^x_e \vee r^x_e))$$

$$wev.(x := e).q.r \;\equiv\; q \vee (dom.e \;\textbf{cand}\; (q^x_e \vee r^x_e))$$

Usually we ignore the possibility of an expression being evaluated outside its domain, in which case we can simplify the two formulae by replacing $dom.e$ with $true$.

$$[wlev.(x := e).q.r \;\equiv\; q \vee q^x_e \vee r^x_e]$$

$$[wev.(x := e).q.r \;\equiv\; q \vee q^x_e \vee r^x_e]$$

In the sequel we stick to the latter two.

Example

We give an example to show that programs that are equal with respect to wlp and wp semantics can be different with respect to $wlev$ and wev semantics. The two programs that we compare are $i := 3; j := 4$ and $j := 4; i := 3$. Since for both programs $wp.s.r$ and $wlp.s.r$ is $r^{i,j}_{3,4}$ they are indistinguishable with respect to wlp and wp. However, we have

$$wev.(i := 3; j := 4).(i > j).false$$

$$\equiv$$

$$wev.(i := 3).(i > j).(wev.(j := 4).(i > j).false)$$

$$\equiv$$

$$wev.(i := 3).(i > j).(i > j \vee i > 4)$$

$$\equiv$$

$$i > j \vee 3 > j \vee 3 > j \vee 3 > 4$$

$$\equiv$$

$$i > j \vee 3 > j$$

and

$$wev.(j := 4; i := 3).(i > j).false$$

$$\equiv$$

$$wev.(j := 4).(i > j).(wev.(i := 3).(i > j).false)$$

$$\equiv$$

$$wev.(j := 4).(i > j).(i > j \vee 3 > j)$$

$$\equiv$$

$$i > j \vee i > 4 \vee i > 4 \vee 3 > 4$$

$$\equiv$$

$$i > j \vee i > 4$$

which are distinct.

(End of example)

The last statement but one that we consider is the alternative statement. Abbreviated to IF, the alternative statement is written as a list of guarded commands, joined by vertical bars, $[\!]$, and surrounded by the bracket pair **if fi**. We write $b_i \rightarrow s_i$ for a guarded command, and assume the range for i to be understood.

$[wlev.IF.q.r \;\equiv\; \langle \forall i : b_i : wlev.s_i.q.r \rangle]$

$[wev.IF.q.r \;\equiv\; \langle \forall i : b_i : wev.s_i.q.r \rangle \wedge (q \vee \langle \exists i :: b_i \rangle)]$

The last, and more challenging statement to define is the repetition. We look at the case of a repetition consisting of one guarded command only. At the expense of more notational inconvenience it can readily be generalized to more (or fewer) guarded commands. We use the abbreviation DO for the repetition with guard b and body s.

$$DO \quad = \quad \mathbf{do}\ b \rightarrow s\ \mathbf{od}$$

We shall find it convenient to have IF stand for the same statement in which the outer bracket pair is replace by that of an alternative statement

$$IF \quad = \quad \mathbf{if}\ b \rightarrow s\ \mathbf{fi}$$

with the same b and s. The definition of $wlev.DO$ and $wev.DO$ is inspired by the operational interpretation of DO. It begins by stipulating that DO equals its first 'unfolding'

$$DO \quad = \quad \mathbf{if}\ b \rightarrow s;\ DO\ [\!]\ \neg b \rightarrow skip\ \mathbf{fi}$$

which suggests

$$[wlev.DO.q.r \;\equiv\; (\neg b \vee wlev.(s;DO).q.r) \wedge (b \vee wlev.skip.q.r)]$$

$$[wev.DO.q.r \;\equiv\; (\neg b \vee wev.(s;DO).q.r) \wedge (b \vee wev.skip.q.r)] \qquad .$$

By substituting the definitions of ; and *skip* we can simplify these definitions to the following two.

$$[wlev.DO.q.r \;\equiv\; (\neg b \vee wlev.s.q.(wlev.DO.q.r)) \wedge (b \vee q \vee r)]$$

$$[wev.DO.q.r \;\equiv\; (\neg b \vee wev.s.(wev.DO.q.r)) \wedge (b \vee q \vee r)]$$

This is the easy part. The harder part is formed by the observation that, when viewed as an equation in the unknown $wlev.DO.q.r$ and $wev.DO.q.r$, these equations may, for given b and s, have more than one solution. We have to specify which solutions we want to be the definitions of $wlev.DO$ and $wev.DO$. We have a closer look at the two equations. We replace the unknown quantities by a fresh variable y to make the character of the equations more explicit. We find that $wlev.DO.q.r$ is a solution of equation

$$y:[y \;\equiv\; (\neg b \vee wlev.s.q.y) \wedge (b \vee q \vee r)] \tag{19}$$

in y, and that $wev.DO.q.r$ is a solution of equation

$$y:[y \;\equiv\; (\neg b \vee wev.s.q.y) \wedge (b \vee q \vee r)] \qquad . \tag{20}$$

Applying (E2) to the last equation we obtain

$$y:[y \;\equiv\; (\neg b \vee wev.s.q.true) \wedge (\neg b \vee wlev.s.q.y) \wedge (b \vee q \vee r)] \qquad .$$

We rewrite the equations in terms of $wlev.IF$.

$$y:[y \;\equiv\; wlev.IF.q.y \wedge (b \vee q \vee r)] \tag{21}$$

$$y:[y \;\equiv\; wlev.IF.q.y \wedge (\neg b \vee wev.s.q.true) \wedge (b \vee q \vee r)] \tag{22}$$

Observe that both equations are of the form

$$y:[y \;\equiv\; z \quad\wedge\quad wlev.IF.q.y] \qquad , \tag{23}$$

differing in the choice for z only. (z depends on b, s, q, and r, but not on y.) Therefore, we concentrate on equation (23). As a function of y, $z \;\wedge\; wlev.IF.q.y$ is positively conjunctive and, hence, monotonic. According to Knaster–Tarski's theorem this implies that (23) has a strongest solution, $g.z$ say, and a weakest solution, $h.z$ say. Since $z \;\wedge\; wlev.IF.q.y$ is universally conjunctive in (z,y), we have

$$[g.z \;\equiv\; g.true \wedge h.z] \qquad .$$

Furthermore, h is universally conjunctive.

Next, we give an operational interpretation of $g.true$ and $h.z$ in order to motivate our choices for $wlev.DO$ and $wev.DO$. Observe that $g.true$ is the strongest solution of

$$y:[y \;\equiv\; wlev.IF.q.y] \qquad . \tag{24}$$

Let c be any solution of this equation. An execution started in a state satisfying c is started in a state in which $wlev.IF.q.c$, i.e. $\neg b \vee wlev.s.q.c$, holds. If $\neg b$ execution of the loop terminates; if $wlev.s.q.c$ execution of s is in the class 'eternal', or in 'ever q', or in 'finally c'. By induction, we find that an execution of DO started in a state satisfying c is in one of four classes: the iteration stops after a finite number of steps in a state satisfying $c \wedge \neg b$, or one of the executions of s visits a state in which q holds, or after a finite number of iterations the loop goes into an execution of

s that does not terminate, or the loop repeats s forever. These are the four possibilities for an arbitrary solution of (24). Next we show that the last of these four possibilities is ruled out if c happens to be the strongest solution of (24). Let us define d to be the predicate that holds precisely in those initial states such that, after a finite number of executions of s, DO terminates or another execution of s is started which is in the class 'ever q' or in the class 'eternal'.

From the definition of d we have $[\neg b \Rightarrow d]$ because initial state $\neg b$ implies immediate termination of DO, which is a possibility included in d. We also have $[wlev.s.q.d \Rightarrow d]$ because initial state $wlev.s.q.d$ implies that execution of s visits q, or terminates in q, or does not terminate at all; each of these three is included in d. Together they imply that d is a solution of $y : [y \Leftarrow \neg b \vee wlev.s.q.y]$, i.e. of

$$y : [y \Leftarrow wlev.IF.q.y] \quad . \tag{25}$$

From Knaster–Tarski's theorem, we know that equations (24) and (25) have the same strongest solution. From the definition of d we conclude $[d \Rightarrow c]$, for every solution c of (24), including the strongest solution $g.true$ of (24). Since $g.true$ is also the strongest solution of (25), we find that d is a solution of (25), and it is at least as strong as the strongest solution of (25). This all goes to say that d is the strongest solution of (25), i.e. $d \equiv g.true$. (How did the infinite repetition of s disappear? It is handled by Knaster–Tarski's theorem, simple as it may appear to be.)

Next we consider, for fixed z, $h.z$, the weakest solution of

$$y : [y \equiv z \wedge wlev.IF.q.y] \quad .$$

This turns out to be a hard job, the hard part being the interaction between q and z, and we make it a bit easier by restricting ourselves to predicates z that are implied by q, i.e. we assume $[q \Rightarrow z]$. Define e to be the predicate that holds exactly in those initial states of DO for which z holds prior to every iteration of the loop as long as q has not been $true$.

From the definition of e we have that, if execution of DO is started in a state that satisfies e, either $\neg b$ holds and the loop terminates, or b holds and leads to an execution of s that either does not terminate, or visits q, or terminates in a state in which e holds again. This amounts to the implication $[e \Rightarrow \neg b \vee wlev.s.q.e]$. Furthermore, if e holds initially then also either q or z holds. Since $[q \Rightarrow z]$ we conclude that $[e \Rightarrow z]$. (By the way, this was the reason for restricting our attention to $[q \Rightarrow z]$.) Combined with the other implication, we find that e satisfies $[e \Rightarrow z \wedge (\neg b \vee wlev.s.q.e)]$ which simplifies to $[e \Rightarrow z \wedge wlev.IF.q.e]$. This shows that e is a solution of equation (26).

$$y : [y \Rightarrow z \wedge wlev.IF.q.y] \tag{26}$$

According to Knaster–Tarski's theorem, equations (26) and (23) have the same weakest solution $h.z$. Since e is a solution of (26), it implies the weakest solution.

$$[e \Rightarrow h.z] \tag{27}$$

Let f be an arbitrary solution of (26). An execution of DO starting a state that satisfies f either terminates right away or initiates an execution of IF that —because f is a solution of (26)— does not terminate, or visits q, or terminates in f. Hence, unless q has been $true$, prior to each iteration f holds. From (26) we conclude that f implies z and, unless q has been $true$, prior to each iteration z holds as well. From the definition of e it follows that f implies e. Since this conclusion holds for every f that solves (26) it also holds for the weakest solution $h.z$.

$$[h.z \Rightarrow e] \tag{28}$$

The combination of (27) and (28) shows that $h.z$ equals e, which provides the characterization of $h.z$ that we were looking for (but only for those z that satisfy $[q \Rightarrow z]$!). Since $[q \Rightarrow b \vee q \vee r]$ we may substitute (cf. (21)) $b \vee q \vee r$ for z in the interpretation of $h.z$ and find that $h.(b \vee q \vee r)$ holds exactly in those initial states of DO for which $b \vee q \vee r$ holds prior to every iteration of the loop, as long as q has not been $true$. In other words: $h.(b \vee q \vee r)$ holds exactly in those initial states of DO such that no execution is in the class 'never q and finally $\neg r$'.

Since $[q \Rightarrow \neg b \vee wev.s.q.true]$ we may substitute (cf. (22)) $\neg b \vee wev.s.q.true$ for z in the interpretation of $h.z$ and find that $h.(\neg b \vee wev.s.q.true)$ holds exactly in those initial states of DO for which $\neg b \vee wev.s.q.true$ holds prior to every iteration of the loop, as long as q has not been $true$. Together with $g.true$ we have: $g.true \wedge h.(\neg b \vee wev.s.q.true)$ holds exactly in those initial states of DO such that no execution is in the class 'never q and eternal'.

From this lengthy detour through equations in predicates we find our definitions for $wlev.DO.q.r$ and $wev.DO.q.r$.

$$[wlev.DO.q.r \;\equiv\; h.(b \vee q \vee r)] \tag{29}$$

$$[wev.DO.q.r \;\equiv\; g.true \wedge h.(\neg b \vee wev.s.q.true) \wedge h.(b \vee q \vee r)] \tag{30}$$

The latter can, thanks to $g.(x \wedge y) \;\equiv\; g.x \wedge h.y$, be simplified to

$$[wev.DO.q.r \;\equiv\; g.((\neg b \vee wev.s.q.true) \wedge (b \vee q \vee r))] \tag{31}$$

We have shown that they satisfy equations (21) and (22) that were derived from the first unfolding of DO and we have shown that they agree with the operational interpretation. In the appendix we show that they satisfy the seven requirements that we have imposed earlier. We now show that $skip$ and $abort$ are two extreme cases of the DO-statement.

Theorem

For each statement s,

$$skip \;=\; \mathbf{do}\ false \rightarrow s\ \mathbf{od}$$

Proof We give the proof of equality with respect to $wlev$ only; the proof for wev is similar. We have

$$\begin{array}{ll}
 & wlev.(\mathbf{do}\ false \rightarrow s\ \mathbf{od}).q.r \\
\equiv & \quad \{ \text{ definition } DO \ \} \\
 & \textit{the weakest solution of } y : [y \;\equiv\; wlev.(\mathbf{if}\ false \rightarrow s\ \mathbf{fi}).q.y \wedge (false \vee q \vee r)] \\
\equiv & \quad \{ \text{ definition } IF \ \} \\
 & \textit{the weakest solution of } y : [y \;\equiv\; true \wedge (false \vee q \vee r)] \\
\equiv & \quad \{ \text{ calculus } \} \\
 & \textit{the weakest solution of } y : [y \;\equiv\; q \vee r] \\
\equiv & \quad \{ \text{ definition of weakest solution } \} \\
 & q \vee r \\
\equiv & \quad \{ \text{ definition } wlev.skip \ \} \\
 & wlev.skip.q.r
\end{array}$$

(End of proof)

Theorem

$$abort \;=\; \mathbf{do}\ true \rightarrow skip\ \mathbf{od}$$

Proof Again, we give the proof of equality with respect to $wlev$ only; the proof for wev is similar. We have

$$\begin{array}{ll} & wlev.(\textbf{do}\ true \rightarrow skip\ \textbf{od}).q.r \\ \equiv & \quad \{ \text{ definition } DO \ \} \\ & the\ weakest\ solution\ of\ y : [y \ \equiv \ wlev.(\textbf{if}\ true \rightarrow skip\ \textbf{fi}).q.y \wedge (true \vee q \vee r)] \\ \equiv & \quad \{ \text{ definition } IF \text{ and } skip \ \} \\ & the\ weakest\ solution\ of\ y : [y \ \equiv \ q \vee y] \\ \equiv & \quad \{ \text{ definition of weakest solution } \} \\ & true \\ \equiv & \quad \{ \text{ definition } wlev.abort \ \} \\ & wlev.abort.q.r \end{array}$$

(End of proof)

Next we present an invariance theorem for wev. It is a generalization of the invariance theorem for wp.

Theorem

Consider the program

$$DO \quad = \quad \textbf{do}\ b \rightarrow s\ \textbf{od} \qquad .$$

Let $(D, \leq)$ be a partially ordered set and let C be a subset of D such that $(C, \leq)$ is well founded. Let I, q, and r be predicates on the state and t an expression of type D such that

$$[I \wedge b \ \Rightarrow \ t \in C] \tag{32}$$

$$\langle \forall x : x \in C : [I \wedge b \wedge t = x \ \Rightarrow \ wev.s.q.(I \wedge (b \vee q \vee r) \wedge t < x)] \rangle \qquad ; \tag{33}$$

then

$$[I \wedge (b \vee q \vee r) \ \Rightarrow \ wev.DO.q.r] \qquad . \tag{34}$$

Before we give the proof, we show that this theorem is a true generalization of the theorem for wp. With no great surprise, we substitute $false$ for q and $I \wedge \neg b$ for r and find: if

$$[I \wedge b \ \Rightarrow \ t \in C]$$

and

$$\langle \forall x : x \in C : [I \wedge b \wedge t = x \ \Rightarrow \ wp.s.(I \wedge t < x)] \rangle$$

then

$$[I \ \Rightarrow \ wp.DO.(I \wedge \neg b)] \qquad .$$

This is exactly the familiar invariance theorem for wp.

Observe that the replacement of r by $I \wedge \neg b$ is something that we could have done in our theorem and proof also. We have chosen not to do so because that would allow us to weaken the postcondition easily, but does not permit strengthening it. And when proving progress properties that is exactly what we often do: we strengthen the postcondition r to $false$ and the 'true' progress condition q remains.

Proof The proof that we give is similar to the proof for wp's invariance theorem as found in [4]. If we let w stand for $wev.DO.q.r$ then w is the strongest solution of equation

$$[y \;\equiv\; f.y] \tag{35}$$

where $[f.y \;\equiv\; (b \vee q \vee r) \wedge (\neg b \vee wev.s.q.y)]$. In fact, we show that the invariance theorem holds for any solution w of (35).

Our proof obligation (34) is $[I \wedge (b \vee q \vee r) \;\Rightarrow\; w]$. From requirement (E5') and the definition of f we have, for any e,

$$[q \;\Rightarrow\; f.e] \tag{36}$$

which in conjunction with $[w \;\equiv\; f.w]$ implies $[q \;\Rightarrow\; w]$ thereby reducing our proof obligation to $[I \wedge (b \vee r) \;\Rightarrow\; w]$. Since

$$[I \wedge (b \vee r) \;\Rightarrow\; w] \;\equiv\; [I \wedge (b \vee r) \wedge t \in C \;\Rightarrow\; w] \wedge [I \wedge (b \vee r) \wedge t \notin C \;\Rightarrow\; w]$$

and

$$\begin{array}{ll}
 & [I \wedge (b \vee r) \wedge t \notin C \;\Rightarrow\; w] \\
\equiv & \quad \{\ [b \vee r \;\equiv\; b \vee (\neg b \wedge r)]\ \} \\
 & [I \wedge b \wedge t \notin C \;\Rightarrow\; w] \wedge [I \wedge \neg b \wedge r \wedge t \notin C \;\Rightarrow\; w] \\
\equiv & \quad \{\ (32)\ \} \\
 & [I \wedge \neg b \wedge r \wedge t \notin C \;\Rightarrow\; w] \\
\equiv & \quad \{\ w \text{ is a solution of (35)}\ \} \\
 & [I \wedge \neg b \wedge r \wedge t \notin C \;\Rightarrow\; f.w] \\
\equiv & \quad \{\ \text{definition of } f\ \} \\
 & [I \wedge \neg b \wedge r \wedge t \notin C \;\Rightarrow\; (b \vee q \vee r) \wedge (\neg b \vee wev.s.q.w)] \\
\equiv & \quad \{\ \text{calculus}\ \} \\
 & true
\end{array}$$

we have

$$[I \wedge (b \vee r) \;\Rightarrow\; w] \;\equiv\; [I \wedge (b \vee r) \wedge t \in C \;\Rightarrow\; w] \tag{37}$$

and our proof obligation reduces to

$$[I \wedge (b \vee r) \wedge t \in C \;\Rightarrow\; w] \qquad .$$

Rewriting it even further we obtain

$$\begin{array}{ll}
 & [I \wedge (b \vee r) \wedge t \in C \;\Rightarrow\; w] \\
\equiv & \\
 & [\langle \forall x : t = x : I \wedge (b \vee r) \wedge x \in C \;\Rightarrow\; w \rangle] \\
\equiv & \\
 & [\langle \forall x : x \in C : I \wedge (b \vee r) \wedge t = x \;\Rightarrow\; w \rangle] \\
\equiv & \\
 & \langle \forall x : x \in C : [I \wedge (b \vee r) \wedge t = x \;\Rightarrow\; w] \rangle
\end{array}$$

and we prove the latter by mathematical induction, i.e. we prove it under the assumption

$$\langle \forall y : y \in C \wedge y < x : [I \wedge (b \vee r) \wedge t = y \;\Rightarrow\; w] \rangle \qquad .$$

In order to apply (33), we rewrite it in terms of f first. For $x \in C$ we have

$$
\begin{array}{ll}
 & I \wedge b \wedge t = x \;\Rightarrow\; wev.s.q.(I \wedge (b \vee r) \wedge t < x) \\
\equiv & \\
 & I \wedge b \wedge t = x \;\Rightarrow\; b \wedge wev.s.q.(I \wedge (b \vee r) \wedge t < x) \\
\equiv & \\
 & I \wedge (b \vee q \vee r) \wedge (b \vee \neg(q \vee r)) \wedge t = x \;\Rightarrow\; b \wedge wev.s.q.(I \wedge (b \vee r) \wedge t < x) \\
\equiv & \\
 & I \wedge (b \vee q \vee r) \wedge t = x \;\Rightarrow\; (\neg b \wedge (q \vee r)) \vee (b \wedge wev.s.q.(I \wedge (b \vee r) \wedge t < x)) \\
\equiv & \\
 & I \wedge (b \vee q \vee r) \wedge t = x \;\Rightarrow\; f.(I \wedge (b \vee r) \wedge t < x) \\
\equiv & \quad \{\ (36)\ \} \\
 & I \wedge (b \vee r) \wedge t = x \;\Rightarrow\; f.(I \wedge (b \vee r) \wedge t < x)
\end{array}
$$

so that we can rewrite (33) as

$$\langle \forall x : x \in C : [I \wedge (b \vee r) \wedge t = x \;\Rightarrow\; f.(I \wedge (b \vee r) \wedge t < x)] \rangle \tag{38}$$

and proceed with the proof.

$$
\begin{array}{ll}
 & \langle \forall y : y \in C \wedge y < x : [I \wedge (b \vee r) \wedge t = y \;\Rightarrow\; w] \rangle \\
\equiv & \\
 & [\langle \forall y : y \in C \wedge y < x : I \wedge (b \vee r) \wedge t = y \;\Rightarrow\; w \rangle] \\
\equiv & \\
 & [\langle \forall y : t = y \wedge y < x : I \wedge (b \vee r) \wedge y \in C \;\Rightarrow\; w \rangle] \\
\equiv & \\
 & [I \wedge (b \vee r) \wedge t < x \wedge t \in C \;\Rightarrow\; w] \\
\equiv & \quad \{\ (37)\ \} \\
 & [I \wedge (b \vee r) \wedge t < x \;\Rightarrow\; w] \\
\Rightarrow & \quad \{\ f \text{ is monotonic }\ \} \\
 & [f.(I \wedge (b \vee r) \wedge t < x) \;\Rightarrow\; f.w] \\
\Rightarrow & \quad \{\ (38)\ \} \\
 & [(I \wedge (b \vee r) \wedge t = x) \;\Rightarrow\; f.w] \\
\equiv & \quad \{\ w \text{ is a solution of (35) }\ \} \\
 & [(I \wedge (b \vee r) \wedge t = x) \;\Rightarrow\; w]
\end{array}
$$

(End of proof)

We conclude this section with an example. Consider the following program.

$S \;=\; i := 10;\ \mathbf{do}\ i \neq 0 \rightarrow i := i - 1\ \mathbf{od}$

We want to prove $[wev.S.(i = 3).false]$, i.e. during execution of S a state is encountered during which i equals 3. We guess that an appeal to the invariance theorem is required with

$I: \quad 3 \leq i$

$t: \quad i$

$C: \quad natural\ numbers$

and we boldly go ahead, pretending that we can work out the invariance later.

$$
\begin{array}{ll}
 & wev.S.(i = 3).\mathit{false} \\
\equiv & \quad \{ \text{ definition of } S \ \} \\
 & wev.(i := 10).(i = 3).(wev.DO.(i = 3).\mathit{false}) \\
\Leftarrow & \quad \{ \ [I \wedge (i \neq 0 \vee i = 3) \ \Rightarrow \ wev.DO.(i = 3).\mathit{false}], \text{ see below; (E1')} \ \} \\
 & wev.(i := 10).(i = 3).(I \wedge (i \neq 0 \vee i = 3)) \\
\equiv & \quad \{ \ [I \wedge (i \neq 0 \vee i = 3) \ \equiv \ 3 \leq i] \ \} \\
 & wev.(i := 10).(i = 3).(3 \leq i) \\
\equiv & \quad \{ \text{ definition of assignment } \} \\
 & i = 3 \vee 10 = 3 \vee 3 \leq 10 \\
\equiv & \\
 & \mathit{true}
\end{array}
$$

It remains to check $[I \wedge (i \neq 0 \vee i = 3) \ \Rightarrow \ wev.DO.(i = 3).\mathit{false}]$, which we do next. By the invariance theorem, this boils down to checking

$$[I \wedge i \neq 0 \ \Rightarrow \ i \geq 0]$$

which is immediate from the definition of I, and checking

$$\langle \forall x : x \geq 0 : [I \wedge i \neq 0 \wedge i = x \ \Rightarrow \ wev.(i := i - 1).(i = 3).(I \wedge (i \neq 0 \vee i = 3) \wedge i < x)] \rangle$$

which follows from

$$
\begin{array}{ll}
 & wev.(i := i - 1).(i = 3).(I \wedge (i \neq 0 \vee i = 3) \wedge i < x) \\
\equiv & \\
 & wev.(i := i - 1).(i = 3).(3 \leq i < x) \\
\equiv & \\
 & i = 3 \vee i - 1 = 3 \vee 3 \leq i - 1 < x \\
\equiv & \\
 & i = 3 \vee i = 4 \vee 4 \leq i \leq x \\
\Leftarrow & \\
 & 3 \leq i \wedge i = x \\
\equiv & \\
 & I \wedge i \neq 0 \wedge i = x
\end{array}
$$

for all x.

7. An exploration of leads–to

As mentioned before, one of our main sources of inspiration is the progress property p *leads–to* q as used in UNITY. The property $[I \ \Rightarrow \ wev.s.q.\mathit{false}]$ expresses that execution of s from an initial state that satisfies I will lead to a state that satisfies q. It is more restricted than $leads-to$ in the sense that one predicate is restricted to the initial state. In this section we explore a method for expressing something like $leads-to$ as a predicate transformer. Similar to what we have done before, we partition executions of a program into classes. Instead of partitioning them into 'ever q'

and 'never q' we use two predicates, p and q, and the partitioning is in two classes, depending on whether in an execution every state in which p holds is eventually followed by a state in which q holds.

p leads to q : the executions such that every state in which p holds is eventually followed by a state in which q holds

p without q : the executions such that there is a state in which p holds which is not followed by a state in which q holds

Observe that the executions in which no state satisfies p are in 'p leads to q'. Observe also that the class '*true* leads to q' is a proper subset of the class 'ever q'. As before we anticipate sequential composition, i.e. we anticipate interest in the final state, and distinguish four classes.

p leads to q : the executions such that every state in which p holds is eventually followed by a state in which q holds

p without q and eternal : the executions such that there is a state in which p holds which is not followed by a state in which q holds and that do not terminate

p without q and finally r : the executions such that there is a state in which p holds which is not followed by a state in which q holds and that terminate in a state satisfying r

p without q and finally $\neg r$: the executions such that there is a state in which p holds which is not followed by a state in which q holds and that terminate in a state satisfying $\neg r$

For given s, p, q, and r, this classification leads us to define two predicates.

$wlto.s.p.q.r$: holds in those initial states for which no execution belongs to the class 'p without q and finally $\neg r$'

$wto.s.p.q.r$: holds in those initial states for which no execution belongs to the class 'p without q and finally $\neg r$' or to the class 'p without q and eternal'

As before, not all predicate transformers can meaningfully be interpreted as a $wlto.s$ or a $wto.s$ for some program s, and we discuss a number of requirements inspired by operational interpretations. Again, we choose constants for some of the parameters to obtain wlp and wp as special cases of $wlto$ and wto respectively. Substituting *true* for p and *false* for q we obtain

$$[wlto.s.true.false.r \;\equiv\; wlp.s.r]$$

$$[wto.s.true.false.r \;\equiv\; wp.s.r] \quad . \tag{T0'}$$

However, a stronger result is possible for $wlto$. According to the operational interpretation, $wlto.s.p.q.(\neg p \vee q \vee r)$ holds exactly in those initial states for which no execution is in the class 'p without q and finally $p \wedge \neg q \wedge \neg r$'. If an execution terminates in a state satisfying $p \wedge \neg q \wedge \neg r$', then p holds but there is no way to establish either q or r. Hence $wlto.s.p.q.(\neg p \vee q \vee r)$ holds exactly in those initial states for which no execution is in the class 'finally $p \wedge \neg q \wedge \neg r$'. Consequently

$$[wlto.s.p.q.(\neg p \vee q \vee r) \;\equiv\; wlp.s.(\neg p \vee q \vee r)] \quad . \tag{T0}$$

For wto and wp we see no way to strengthen (T0').

Since wlp is conjunctive, and because of (T0), we expect a conjunctivity property for $wlto$. We require, for all s, p, and q,

$$wlto.s.p.q \text{ is universally conjunctive} \tag{T1}$$

which implies the special case (the conjunction of zero predicates)

$$[wlto.s.p.q.true] \quad . \tag{39}$$

We have come to expect a relation between the liberal and the conservative predicate transformer. They are coupled by substituting $true$ for the the last parameter of the conservative function.

$$[wto.s.p.q.r \;\equiv\; wto.s.p.q.true \wedge wlto.s.p.q.r] \tag{T2}$$

Thanks to (39) both left- and right-hand side of (T2) reduce to $wto.s.p.q.true$ when substituting $true$ for r. From (T1) and (T2) it follows that, for all s, p, and q,

$$wto.s.p.q \text{ is positively conjunctive} \quad . \tag{T1'}$$

So much for the properties related to the r argument, Next we turn to the second argument, viz. q. Just like in the case of $wlev$ and wev we require monotonicity of the predicate transformers as a function of q, and, again, the requirements are independent for the two functions.

$$[q \;\Rightarrow\; q'] \;\Rightarrow\; [wlto.s.p.q.r \;\Rightarrow\; wlto.s.p.q'.r] \tag{T3}$$

$$[q \;\Rightarrow\; q'] \;\Rightarrow\; [wto.s.p.q.r \;\Rightarrow\; wto.s.p.q'.r] \tag{T3'}$$

Properties (E4) and (E4') express that the final state is also an intermediate state. The pendant for $leads-to$ is similar. We have

$$[wlto.s.p.q.r \;\equiv\; wlto.s.p.q.(q \vee r)] \tag{T4}$$

which implies

$$[wto.s.p.q.r \;\equiv\; wto.s.p.q.(q \vee r)] \tag{T4'}$$

Properties (E5) and (E5') express that the initial state is also an intermediate state. The pendant for $leads-to$ is that a state satisfying p is also a state that coincides with or follows a state satisfying p.

$$[wto.s.p.p.r] \tag{T5'}$$

In combination with the monotonicity in q this can be phrased as

$$[p \;\Rightarrow\; q] \;\Rightarrow\; [wto.s.p.q.r] \quad .$$

In combination with (T2), (T5') implies

$$[wlto.s.p.p.r] \tag{T5}$$

and

$$[p \;\Rightarrow\; q] \;\Rightarrow\; [wlto.s.p.q.r] \quad .$$

This brings us to consider the first predicative argument, p. Suppose a computation is in the class '$p0$ leads to q or terminates in r' and also in the class '$p1$ leads to q or terminates in r'. Then it is also in the class '$p0 \vee p1$ leads to q or terminates in r', and vice versa. We extend this observation to the 'junctivity' requirements (T6) and (T6'). For each, possibly empty, set of predicates P we require

$$[\langle \forall p : p \in P : wlto.s.p.q.r \rangle \;\equiv\; wlto.s.\langle \exists p : p \in P : p \rangle.q.r] \quad ; \tag{T6}$$

$$[\langle \forall p : p \in P : wto.s.p.q.r \rangle \;\equiv\; wto.s.\langle \exists p : p \in P : p \rangle.q.r] \quad . \tag{T6'}$$

The remaining two pairs of requirements capture important 'progress' characteristics of our predicate transformers. The first pair expresses that an execution that is in the class 'ever p' and in the class 'p leads to q' is also in the class 'ever q'.

$$[wlto.s.p.q.r \wedge wlev.s.p.r \;\Rightarrow\; wlev.s.q.r] \tag{T7}$$

$$[wto.s.p.q.r \wedge wev.s.p.r \;\Rightarrow\; wev.s.q.r] \tag{T7'}$$

The second and last pair expresses transitivity: an execution that is in the class 'p leads to q' and in the class 'q leads to w' is also in the class 'p leads to w'.

$$[wlto.s.p.q.r \wedge wlto.s.q.w.r \;\Rightarrow\; wlto.s.p.w.r] \tag{T8}$$

$$[wto.s.p.q.r \wedge wto.s.q.w.r \;\Rightarrow\; wto.s.p.w.r] \tag{T8'}$$

This completes our list of requirements for $wlto$ and wto. Since they have three predicates as arguments instead of two, we are not surprised that the list is longer than for $wlev$ and wev. The next section defines $wlto$ and wto for a number of statements. this section concludes with a theorem that is similar to a theorem about $wlev$ and wev, and we have a theorem on monotonicity in the p argument.

Theorem

For all programs s, and predicates p, q, r, and w, we have

$$[wlto.s.p.q.r \wedge wlp.s.w \Rightarrow wlto.s.p.q.(r \wedge w)] \tag{40a}$$

$$[wlto.s.p.q.r \wedge wp.s.w \Rightarrow wto.s.p.q.(r \wedge w)] \tag{40b}$$

$$[wto.s.p.q.r \wedge wlp.s.w \Rightarrow wto.s.p.q.(r \wedge w)] \tag{40c}$$

$$[wto.s.p.q.r \wedge wp.s.w \Rightarrow wto.s.p.q.(r \wedge w)] \tag{40d}$$

Proof We give the proof for (40a); the other three are similar.

$$\begin{array}{ll}
 & wlto.s.p.q.r \wedge wlp.s.w \\
\equiv & \qquad \{\ (\mathrm{T0})\ \} \\
 & wlto.s.p.q.r \wedge wlto.s.true.false.w \\
\Rightarrow & \qquad \{\ (\mathrm{T3}) \text{ and } (\mathrm{T6})\ \} \\
 & wlto.s.p.q.r \wedge wlto.s.p.q.w \\
\equiv & \qquad \{\ (\mathrm{T1})\ \} \\
 & wlto.s.p.q.(r \wedge w)
\end{array}$$

(End of proof)

Theorem

For all programs s, and predicates p, p', q, r, and w, we have

$$[p' \Rightarrow p] \;\Rightarrow\; [wlto.s.p.q.r \Rightarrow wlto.s.p'.q.r] \tag{41a}$$

$$[p' \Rightarrow p] \;\Rightarrow\; [wto.s.p.q.r \Rightarrow wto.s.p'.q.r] \tag{41b}$$

Proof We give the proof for (41a); the other one is similar. Assume $[p' \Rightarrow p]$.

$$\begin{array}{ll}
 & wlto.s.p.q.r \wedge wlto.s.p'.q.r \\
\equiv & \qquad \{\ (\mathrm{T6})\ \} \\
 & wlto.s.(p \vee p').q.r \\
\equiv & \qquad \{\ [p' \Rightarrow p]\ \} \\
 & wlto.s.p.q.r
\end{array}$$

(End of proof)

8. Definition of *wlto* and *wto*

In this section we give the definitions of $wlto$ and wto by induction on the structure of the programs. For each construct we have to verify that the requirements are met. Again, most proofs are omitted; only the proofs for the case of the iterative command are listed in the appendix.

We begin our list of definitions with $skip$, the statement that does not change the state whatsoever. The definition of $skip$ is

$$[wlto.skip.p.q.r \;\equiv\; \neg p \vee q \vee r]$$

$$[wto.skip.p.q.r \;\equiv\; \neg p \vee q \vee r] \qquad .$$

The second statement that we define is $abort$. It is the statement that has an initial state but no other intermediate state and has no final state. This suggests the following definition.

$$[wlto.abort.p.q.r \;\equiv\; true]$$

$$[wto.abort.p.q.r \;\equiv\; \neg p \vee q]$$

The next construct that we define is sequential composition. The execution of $s;t$ is the execution of s, followed by execution of t if s terminates. If p leads to q in $s;t$ then a state satisfying p that is encountered during execution of s leads to a state satisfying q that is encountered during s or else s terminates in a state from which t is started and that leads to a state satisfying q. Furthermore, if s terminates, then it does so in a state in which p leads to q during execution of t. Hence, we are suggested to define

$$[wlto.(s;t).p.q.r \;\equiv\; wlto.s.p.q.(wlev.t.q.r) \wedge wlp.s.(wlto.t.p.q.r)]$$

$$[wto.(s;t).p.q.r \;\equiv\; wto.s.p.q.(wev.t.q.r) \wedge wlp.s.(wto.t.p.q.r)] \qquad .$$

We show that a few properties that one might expect on operational grounds are indeed theorems. First we show that $skip$ is the left and right unit element of sequential composition. Next we show that $abort$ is the left zero element, and finally we show that sequential composition is associative.

Theorem

For all programs s we have

$$s;skip \;=\; s \;=\; skip;s$$

Proof The equality of programs means that their predicate transformers $wlto$ and wto are equal. Observe that, since wlp and wp are special cases of $wlto$ and wto, equality of programs with respect to $wlto$ and wto implies equality with respect to wlp and wp. It does not imply equality with respect to $wlev$ and wev, however. We give the proof for equality with respect to $wlto$ only;

the proof for wto is similar. For all s, p, q, and r we have

$$
\begin{array}{ll}
 & wlto.(s;skip).p.q.r \\
\equiv & \quad \{ \text{ definition } ; \ \} \\
 & wlto.s.p.q.(wlev.skip.q.r) \wedge wlp.s.(wlto.skip.p.q.r) \\
\equiv & \quad \{ \text{ definition } skip \ \} \\
 & wlto.s.p.q.(q \vee r) \wedge wlp.s.(\neg p \vee q \vee r) \\
\equiv & \quad \{ \text{ (T0) } \} \\
 & wlto.s.p.q.(q \vee r) \wedge wlto.s.p.q.(\neg p \vee q \vee r) \\
\equiv & \quad \{ \text{ (T1) } \} \\
 & wlto.s.p.q.(q \vee r) \\
\equiv & \quad \{ \text{ (T4) } \} \\
 & wlto.s.p.q.r \\
\equiv & \quad \{ \text{ see below } \} \\
 & (\neg p \vee wlev.s.q.r) \wedge wlto.s.p.q.r \\
\equiv & \quad \{ \text{ (E5) } \} \\
 & (\neg p \vee q \vee wlev.s.q.r) \wedge wlto.s.p.q.r \\
\equiv & \quad \{ \text{ definition } skip \ \} \\
 & wlto.skip.p.q.(wlev.s.q.r) \wedge wlp.skip.(wlto.s.p.q.r) \\
\equiv & \quad \{ \text{ definition } ; \ \} \\
 & wlto.(skip;s).p.q.r
\end{array}
$$

in which we have used the fact that

$$
\begin{array}{ll}
 & [wlto.s.p.q.r \ \Rightarrow \ \neg p \vee wlev.s.q.r] \\
\equiv & \quad \{ \text{ calculus } \} \\
 & [wlto.s.p.q.r \wedge p \ \Rightarrow \ wlev.s.q.r] \\
\Leftarrow & \quad \{ \text{ (E5) } \} \\
 & [wlto.s.p.q.r \wedge wlev.s.p.r \ \Rightarrow \ wlev.s.q.r] \\
\equiv & \quad \{ \text{ (T7) } \} \\
 & true \qquad .
\end{array}
$$

(End of proof)

Theorem

For all programs s we have

$abort; s \ = \ abort$

Proof We give the proof for equality with respect to wto only; the proof for $wlto$ is similar. For all s, q, and r we have

$$
\begin{array}{ll}
 & wto.(abort;s).p.q.r \\
\equiv & \quad \{ \text{ definition } ; \ \} \\
 & wto.abort.p.q.(wto.s.q.r) \wedge wlp.abort.(wto.s.p.q.r) \\
\equiv & \quad \{ \text{ definition } abort \ \} \\
 & \neg p \vee q \\
\equiv & \quad \{ \text{ definition } abort \ \} \\
 & wto.abort.p.q.r \qquad .
\end{array}
$$

(End of proof)

Theorem

Sequential composition is associative.

Proof We give the proof for associativity with respect to wto only; the proof for $wlto$ is similar. For all s, t, u, q, and r we have

$$
\begin{array}{cl}
 & wto.(s;(t;u)).p.q.r \\
\equiv & \quad \{ \text{ definition } ; \ \} \\
 & wto.s.p.q.(wev.t.q.(wev.u.q.r)) \\
 & \wedge\ wlp.s.(wto.t.p.q.(wev.u.q.r) \wedge wlp.t.(wto.u.p.q.r)) \\
\equiv & \quad \{ \text{ conjunctivity of } wlp \ \} \\
 & wto.s.p.q.(wev.t.q.(wev.u.q.r)) \\
 & \wedge\ wlp.s.(wto.t.p.q.(wev.u.q.r)) \wedge wlp.s.(wlp.t.(wto.u.p.q.r)) \\
\equiv & \quad \{ \text{ definition } ; \ \} \\
 & wto.(s;t).p.q.(wev.u.q.r) \wedge wlp.(s;t).(wto.u.p.q.r) \\
\equiv & \quad \{ \text{ definition } ; \ \} \\
 & wto.((s;t);u).p.q.r \qquad .
\end{array}
$$

(End of proof)

Next, we consider the assignment $x := e$. We ignore the possibility of an expression being evaluated outside its domain. The definitions are obtained by considering the fact that there are two states in the execution of the assignment statement: one before and one after the assignment. If p leads to q, and p holds initially then either q holds also or $q \vee r$ holds in the final state; if p holds in the final state then $q \vee r$ holds also.

$$[wlto.(x := e).p.q.r \ \equiv \ (\neg p \vee q \vee q_e^x \vee r_e^x) \wedge (\neg p_e^x \vee q_e^x \vee r_e^x)]$$

$$[wto.(x := e).p.q.r \ \equiv \ (\neg p \vee q \vee q_e^x \vee r_e^x) \wedge (\neg p_e^x \vee q_e^x \vee r_e^x)]$$

The last statement but one that we consider is the alternative statement. Abbreviated to IF, the alternative statement is written as a list of guarded commands, joined by vertical bars, $[\!]$, and surrounded by the bracket pair **if fi**. We write $b_i \rightarrow s_i$ for a guarded command, and assume the range for i to be understood.

$$[wlto.IF.p.q.r \ \equiv \ \langle \forall i : b_i : wlto.s_i.p.q.r \rangle]$$

$$[wto.IF.p.q.r \ \equiv \ \langle \forall i : b_i : wto.s_i.p.q.r \rangle \wedge (\neg p \vee q \vee \langle \exists i :: b_i \rangle)]$$

The last, and more challenging statement to define is the repetition. We use the abbreviation DO for the repetition with guard b and body s.

$$DO \quad = \quad \mathbf{do}\ b \rightarrow s\ \mathbf{od}$$

Just as in the previous section, we find it convenient to have IF stand for the same statement in which the outer bracket pair is replace by that of an alternative statement

$$IF \quad = \quad \mathbf{if}\ b \rightarrow s\ \mathbf{fi}$$

with the same b and s. The definition of $wlto.DO$ and $wto.DO$ is inspired by its operational interpretation. Equating DO and its first 'unfolding'

$$DO \quad = \quad \textbf{if}\ b \rightarrow s;\ DO \ [\!]\ \neg b \rightarrow skip\ \textbf{fi}$$

suggests

$$[wlto.DO.p.q.r \ \equiv\ (\neg b \vee wlto.(s;DO).p.q.r) \wedge (b \vee wlto.skip.p.q.r)]$$

$$[wto.DO.p.q.r \ \equiv\ (\neg b \vee wto.(s;DO).p.q.r) \wedge (b \vee wto.skip.p.q.r)] \qquad .$$

By substituting the definitions of ; and $skip$ we can simplify these definitions. Replacing the unknown quantities by a fresh variable y to make the character of the equation more explicit we find that $wlto.DO.p.q.r$ is a solution of

$$y : [y \ \equiv\ (\neg b \vee (wlp.s.y \wedge wlto.s.p.q.(wlev.DO.q.r))) \wedge (b \vee \neg p \vee q \vee r)] \tag{42}$$

and that $wto.DO.p.q.r$ is a solution of the similar equation (43).

$$y : [y \ \equiv\ (\neg b \vee (wlp.s.y \wedge wto.s.p.q.(wev.DO.q.r))) \wedge (b \vee \neg p \vee q \vee r)] \tag{43}$$

Both equations are of the form

$$y : [y \ \equiv\ z \wedge wlp.IF.y] \qquad , \tag{44}$$

differing in the choice for z only. Remember that we have studied the similar equation (23) earlier. In (23) we have $wlev.IF.q.y$ instead of $wlp.IF.y$. Therefore, equations (23) and (44) are equal if we substitute $false$ for q (cf. (E0)). Since equation (44) equals equation (13), the g and h that we have now are the same as the g and h that we have in the context of $wlp.DO$ and $wp.DO$. In order to see which solution we have to choose as the appropriate one, we argue as follows. We claim that $wto.DO.p.q.r$ is the weakest solution of (43). Let y be any solution of (43). It suffices to show that y implies $wto.DO.p.q.r$. Assume that y is a precondition of the loop. From the conjunct $\neg b \vee wlp.s.y$ in (43), it follows that y is an invariant of the loop. Whenever p holds in some intermediate state of command s, predicate $b \wedge y$ holds at the beginning of that command and, hence, also $wto.s.p.q.(wev.DO.q.r)$. This implies that s establishes, after the state in which p holds, an intermediate state in which q holds or that s terminates in a state in which $wev.DO.q.r$ holds. In the latter case it follows that the remainder of the execution of the loop reaches an intermediate state in which q holds or terminates in a state in which r holds. Consequently, the initial state satisfies $wto.DO.p.q.r$. A similar reasoning applies to $wlto.DO.p.q.r$. We are thus led to the following definitions.

$$[wlto.DO.p.q.r \ \equiv\ h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r))] \tag{45}$$

$$[wto.DO.p.q.r \ \equiv\ h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wto.s.p.q.(wev.DO.q.r))] \tag{46}$$

Observe that both $wlto.DO$ and $wto.DO$ have been defined as the weakest solution of an equation, not as a weakest and a strongest solution respectively. This is our main reason for extensively including the operational interpretation of the predicates. In an earlier attempt we defined wto as the strongest solution, and were unable to prove one of the requirements (T5'). One might say that the strongest solution restricts the loop to a finite number of iterations, whereas the operational interpretation of $wto.DO$ does not. The weakest solution does not limit the number of iterations either. We mention two extreme cases of the DO-statement. The proofs are omitted.

Theorem

For each statement s,

$skip \quad = \quad \mathbf{do}\ false \rightarrow s\ \mathbf{od}$

Theorem

$abort \quad = \quad \mathbf{do}\ true \rightarrow skip\ \mathbf{od}$

We conclude this section with an invariance theorem for wto. It is somewhat different from the invariance theorems for wev and wp in the sense that there is no variant function, and all 'real' work is delegated to the wev of the same loop.

Theorem

Consider program $DO \quad = \quad \mathbf{do}\ b \rightarrow s\ \mathbf{od}$. We have

$$[J \wedge b \ \Rightarrow\ wto.s.p.q.(wev.DO.q.r) \wedge wlp.s.(J \wedge (b \vee \neg p \vee q \vee r))] \tag{47}$$

implies

$$[J \wedge (b \vee \neg p \vee q \vee r) \ \Rightarrow\ wto.DO.p.q.r] \quad .$$

Proof The weakest solution $h.z$ of $y : [y \ \equiv\ f.z.y]$ is, for monotonic f, characterized by

$$[x \Rightarrow f.z.x] \ \Rightarrow\ [x \Rightarrow h.z] \quad .$$

Since $wto.DO.p.q.r$ is the weakest solution of (43), it suffices to show $[x \Rightarrow f.z.x]$, where

$$[x \ \equiv\ J \wedge (b \vee \neg p \vee q \vee r)]$$

$$[f.z.x \ \equiv\ wlp.IF.x \wedge (\neg b \vee wto.s.p.q.(wev.DO.q.r)) \wedge (b \vee \neg p \vee q \vee r)]$$

$$IF = \mathbf{if}\ b \rightarrow s\ \mathbf{fi} \quad .$$

Here we go.

$$
\begin{array}{ll}
 & [x \ \Rightarrow\ f.z.x] \\
\equiv & \qquad \{ \text{ substitution } \} \\
 & [J \wedge (b \vee \neg p \vee q \vee r) \ \Rightarrow \\
 & \qquad (\neg b \vee (wlp.s.(J \wedge (b \vee \neg p \vee q \vee r)) \wedge wto.s.p.q.(wev.DO.q.r))) \wedge (b \vee \neg p \vee q \vee r)] \\
\Leftarrow & \qquad \{ \ [\neg b \wedge (\neg p \vee q \vee r) \ \Rightarrow\ (\neg b \vee \text{anything}) \wedge (b \vee \neg p \vee q \vee r)] \ \} \\
 & [J \wedge b \ \Rightarrow \\
 & \qquad (\neg b \vee (wlp.s.(J \wedge (b \vee \neg p \vee q \vee r)) \wedge wto.s.p.q.(wev.DO.q.r))) \wedge (b \vee \neg p \vee q \vee r)] \\
\equiv & \qquad \{ \text{ (47), calculus } \} \\
 & true
\end{array}
$$

(End of proof)

Again, we give an example to demonstrate the use of the invariance theorem. The example is the loop

$$DO \quad = \quad \mathbf{do}\ i \neq 0 \rightarrow i := i - 1\ \mathbf{od}$$

and we show

$$[m \geq n \geq 0 \vee i < m \ \Rightarrow\ wto.DO.(i = m).(i = n).false] \quad .$$

The choice for J is $m \geq n \geq 0 \vee i < m$. Observe $[\,J \ \equiv\ J \wedge (b \vee \neg p \vee q \vee r)]$. The result follows from the invariance theorem for wto provided

$[J \wedge b \;\Rightarrow\; wto.s.p.q.(wev.DO.q.r) \wedge wlp.s.J]$.

We deal with the last term first.

$$\begin{array}{ll} & wlp.s.J \\ \equiv & \\ & m \geq n \geq 0 \vee i-1 < m \\ \Leftarrow & \\ & (m \geq n \geq 0 \vee i < m) \wedge i \neq 0 \end{array}$$

In order to tackle the first term, we proceed with showing $[i \geq n \geq 0 \;\Rightarrow\; wev.DO.(i=n).false]$. We do so by applying the invariance theorem for wev with the following choices:

$I: \;\; i \geq n \geq 0$

$t: \;\; i$

$C: \;\; natural\ numbers$

Observe $[I \wedge (i \neq 0 \vee i = n) \;\equiv\; I]$. The invariance theorem applies since

$[I \wedge b \;\Rightarrow\; t \in C]$

$[I \wedge b \wedge t = x \;\Rightarrow\; wev.(i := i-1).(i=n).(I \wedge t < x)]$

for all x. Both proofs are just a matter of substitution, and are omitted. This leaves us to show

$[J \wedge b \;\Rightarrow\; wto.s.p.q.(i \geq n \geq 0)]$.

We have

$$\begin{array}{ll} & wto.(i := i-1).(i=m).(i=n).(i \geq n \geq 0) \\ \equiv & \\ & (i \neq m \vee i = n \vee i-1 = n \vee i-1 \geq n \geq 0) \wedge \\ & (i-1 \neq m \vee i-1 = n \vee i-1 \geq n \geq 0) \\ \Leftarrow & \\ & i \geq m \geq n \geq 0 \vee i < m \\ \Leftarrow & \\ & (m \geq n \geq 0 \vee i < m) \wedge i \neq 0 \end{array}$$

which completes the proof.

9. Concluding Remarks

In this paper we have introduced new predicate transformers to express progress properties of programs. These predicate transformers were shown to be extensions of the familiar predicate transformers wp and wlp. The inherent complexity of these new predicate transformers led to a large number of requirements or healthiness conditions to be imposed in order to pin down the characteristics of the predicate transformers. Verification of these healthiness conditions is a laborious task. It is difficult only for the case of the iterative command. As an alternative one might give an operational semantics of the program notation. In such an operational semantics, verification of the healthiness conditions would be easier, since induction over the syntax can be avoided. It also allows for a more rigorous (but non-trivial) proof of which solution of a defining equation must be chosen as the predicate transformer of the iterative command. Yet another advantage is that the definitions of the predicate transformers for the various commands that we have given in this paper can be derived from the operational semantics. In this paper we have chosen to follow the path that avoids the operational semantics since it makes the analysis cleaner: only the aspects that are relevant to our present purpose are involved. It also allows a more relaxed implementation of the programming language.

The application of our predicate transformers in the specification of of programs is mainly in the area of parallel programs. Although parallel composition is not part of our program notation, we can "simulate" parallelism by constructing a repetition of an alternative statement. Such a program is called an action system and was first introduced by Back and Kurki-Suonio in [1]. If the alternatives are further restricted to assignment statements and if a fair choice between the alternatives is postulated we end up with UNITY (cf. [2]).

Some recent papers have been published that deal with various predicate transformers. In [6] the predicate transformer "weakest leads to" is considered. The weakest leads to of a program and a predicate q is the weakest predicate p such that p leads to q holds for that program. No obvious relationship to the program's precondition exists. The relation to commands in a program notation is investigated for UNITY programs only. Because UNITY's individual statements are deterministic and because of the absence of sequential composition, the equations that result are slightly simpler than ours. In [8], Morris discusses a predicate transformer that is very similar to our $wev/wlev$, the main difference being that the intermediate states considered by Morris are only those states that directly precede the call of a recursive procedure.

The development of our predicate transformers seems to follow a pattern, and one might wonder whether it is possible to make the pattern explicit and specialize the pattern to the cases that we have discussed. We have not succeeded in doing so, the main problem being the fact that wev is the strongest solution of its defining equation (22) whereas wto is the weakest solution of its defining equation (43). This seems to be a formidable obstacle in coming up with a single pattern. One is referred to the proof of (T0') for the iterative command (see last appendix) to see the complexity of linking a weakest solution to a strongest solution.

Acknowledgement

We are very grateful to one of the referees for insightful comments that helped us improve the presentation and that clarified some technical problems.

10. Appendix – proofs for (E0) through (E5)

In this appendix we give the proofs that the predicate transformers $wlev$ and wev of the statements satisfy the requirements that we have imposed. Since most proofs are quite simple, we only consider the case of the repetition. Notice that (E1'), (E4') and (E5) follow from the other requirements. The proofs of (E0') and (E3') have been omitted because they are similar to the proofs for (E0) and (E3) respectively.

(E0):

$$\begin{array}{ll}
& wlev.DO.false.r \\
\equiv & \quad \{ \text{ definition of } wlev.DO \ \} \\
& \text{the weakest solution of } y : [y \ \equiv \ (b \vee r) \wedge wlev.IF.false.y] \\
\equiv & \quad \{ \text{ (E0) for } IF \ \} \\
& \text{the weakest solution of } y : [y \ \equiv \ (b \vee r) \wedge wlp.IF.y] \\
\equiv & \quad \{ \text{ definition of } wlp.DO \ \} \\
& wlp.DO.r
\end{array}$$

(E1): $z \wedge wlev.IF.q.y$ is a universally conjunctive function of (z, y) i.e.

$$\begin{array}{ll}
& \langle \forall z, y : z \in Z \wedge y \in Y : z \wedge wlev.IF.q.y \rangle \\
\equiv & \\
& \langle \forall z : z \in Z : z \rangle \wedge wlev.IF.q.\langle \forall y : y \in Y : y \rangle
\end{array}$$

It follows that $h.z$, the weakest solution of (23), is a universally conjunctive function of z.

$$\begin{array}{ll}
& wlev.DO.q.\langle \forall r : r \in R : r \rangle \\
\equiv & \quad \{ \ (29) \ \} \\
& h.(b \vee q \vee \langle \forall r : r \in R : r \rangle) \\
\equiv & \quad \{ \text{ calculus } \} \\
& h.\langle \forall r : r \in R : b \vee q \vee r \rangle \\
\equiv & \quad \{ \ h \text{ is universally conjunctive; so is } b \vee q \vee r \ \} \\
& \langle \forall r : r \in R : h.(b \vee q \vee r) \rangle \\
\equiv & \quad \{ \ (29) \ \} \\
& \langle \forall r : r \in R : wlev.DO.q.r \rangle
\end{array}$$

(E2):

$$\begin{array}{ll}
& wev.DO.q.r \\
\equiv & \quad \{ \ (30); (8) \ \} \\
& g.(\neg b \vee wev.s.q.true) \wedge h.(b \vee q \vee r) \\
\equiv & \quad \{ \ [true \equiv b \vee q \vee true] \ \} \\
& g.((\neg b \vee wev.s.q.true) \wedge (b \vee q \vee true)) \wedge h.(b \vee q \vee r) \\
\equiv & \quad \{ \ (31) \text{ and } (29) \ \} \\
& wev.DO.q.true \quad \wedge \quad wlev.DO.q.r
\end{array}$$

(E3): In order to show that, given $[q \ \Rightarrow \ q']$, $wlev.DO.q.r$ implies $wlev.DO.q'.r$ we use the fact that the latter is the weakest solution of its defining equation. According to Knaster–Tarski it is also the weakest solution of the equation in which equivalence is replaced by implication. We are left to show that $wlev.DO.q.r$ is also a solution of the latter equation.

$$
\begin{array}{ll}
 & [wlev.DO.q.r \Rightarrow wlev.DO.q'.r] \\
\Leftarrow & \quad \{\ wlev.DO.q'.r \text{ is the weakest solution of (19); see above }\} \\
 & wlev.DO.q.r \text{ solves } y : [y \ \Rightarrow\ (\neg b \vee wlev.s.q'.y) \wedge (b \vee q' \vee r)] \\
\equiv & \quad \{\ \text{substitution}\ \} \\
 & [wlev.DO.q.r \ \Rightarrow\ (\neg b \vee wlev.s.q'.(wlev.DO.q.r)) \wedge (b \vee q' \vee r)] \\
\equiv & \quad \{\ wlev.DO.q.r \text{ solves (19) }\} \\
 & [(\neg b \vee wlev.s.q.(wlev.DO.q.r)) \wedge (b \vee q \vee r) \ \Rightarrow \\
 & \quad (\neg b \vee wlev.s.q'.(wlev.DO.q.r)) \wedge (b \vee q' \vee r)] \\
\Leftarrow & \quad \{\ \text{(E3) for } s \text{; calculus }\} \\
 & [q \Rightarrow q']
\end{array}
$$

(E4):

$$
\begin{array}{ll}
 & wlev.DO.q.r \\
\equiv & \quad \{\ (29)\ \} \\
 & h.(b \vee q \vee r) \\
\equiv & \quad \{\ \text{calculus}\ \} \\
 & h.(b \vee q \vee (q \vee r)) \\
\equiv & \quad \{\ (29)\ \} \\
 & wlev.DO.q.(q \vee r)
\end{array}
$$

(E5'):

$$
\begin{array}{ll}
 & wev.DO.q.r \\
\equiv & \quad \{\ (30)\ \} \\
 & (\neg b \vee wev.s.q.(wev.DO.q.r)) \wedge (b \vee q \vee r) \\
\Leftarrow & \quad \{\ \text{(E5') for } s\ \} \\
 & q
\end{array}
$$

We draw attention to the fact that we can use the definitions of $wlev.DO$ and $wev.DO$ in terms of the function h as long as the second argument, viz. q, is fixed. Since q is hidden in the definition of h, we have to go back to the original equation when we analyze the dependence on q. For example, the proof of (E4) is valid since h does not depend on r. The similar-looking argument

$$
\begin{array}{ll}
 & wlev.DO.q.r \\
\equiv & \quad \{\ (29)\ \} \\
 & h.(b \vee q \vee r) \\
\equiv & \quad \{\ \text{calculus}\ \} \\
 & h.(b \vee (q \vee r) \vee r) \\
\equiv & \quad \{\ (29)\ \} \\
 & wlev.DO.(q \vee r).r
\end{array}
$$

is invalid (and fortunately so) because h does depend on q and we, therefore, have a different function h when the second argument of $wlev$ is $q \vee r$.

11. Appendix – proofs for (T0) through (T8)

In this appendix we give the proofs that the predicate transformers $wlto$ and wto of the statements satisfy the requirements that we have imposed. Again, we restrict attention to the loop. Notice that (T1'), (T4') and (T5) follow from the other requirements. The proofs of (T0'), (T3'), (T6'), (T7'), and (T8') have been omitted because they are similar to the proofs for the corresponding 'liberal' requirements. Remember that the g and h of (44) are the same as the g and h of (13).

(T0):

$$
\begin{array}{ll}
 & wlto.DO.p.q.(\neg p \vee q \vee r) \\
\Leftarrow & \quad \{ \text{ (T3) and (T6) for } wlto.DO \ \} \\
 & wlto.DO.true.false.(\neg p \vee q \vee r) \\
\equiv & \quad \{ \text{ (45) } \} \\
 & h.((b \vee (\neg p \vee q \vee r)) \wedge (\neg b \vee wlto.s.true.false.(wlev.DO.false.(\neg p \vee q \vee r)))) \\
\equiv & \quad \{ \text{ (T0) for } s, \text{ (E0) for } DO \ \} \\
 & h.((b \vee (\neg p \vee q \vee r)) \wedge (\neg b \vee wlp.s.(wlp.DO.(\neg p \vee q \vee r)))) \\
\equiv & \quad \{ \text{ (10) } \} \\
 & wlp.DO.(\neg p \vee q \vee r) \wedge h.(b \vee (\neg p \vee q \vee r)) \\
\equiv & \quad \{ \text{ (14) } \} \\
 & h.(b \vee \neg p \vee q \vee r) \\
\Leftarrow & \quad \{ \ h \text{ is monotonic } \} \\
 & h.((\neg b \vee wlto.s.p.q.(wlev.DO.q.r)) \wedge (b \vee \neg p \vee q \vee r)) \\
\equiv & \quad \{ \text{ (45) } \} \\
 & wlto.DO.p.q.(\neg p \vee q \vee r)
\end{array}
$$

Observe that, in the first step of this proof, we have used (T3) and (T6). In the proofs thereof, we avoid references to (T0).

(T0'):

$$
\begin{array}{ll}
 & wto.DO.true.false.r \\
\equiv & \quad \{ \text{ (46) } \} \\
 & h.((b \vee r) \wedge (\neg b \vee wto.s.true.false.(wev.DO.false.r))) \\
\equiv & \quad \{ \text{ (T0') for } s, \text{ (E0') for } DO \ \} \\
 & h.((b \vee r) \wedge (\neg b \vee wp.s.(wp.DO.r))) \\
\equiv & \quad \{ \text{ (R1) } \} \\
 & h.((b \vee r) \wedge (\neg b \vee wp.s.true) \wedge (\neg b \vee wlp.s(wp.DO.r))) \\
\equiv & \quad \{ \text{ (10) with } (b \vee r) \wedge (\neg b \vee wp.s.true) \text{ for } z \ \} \\
 & wp.DO.r \wedge h.((b \vee r) \wedge (\neg b \vee wp.s.true)) \\
\equiv & \quad \{ \text{ (15) } \} \\
 & g.((b \vee r) \wedge (\neg b \vee wp.s.true)) \wedge h.((b \vee r) \wedge (\neg b \vee wp.s.true)) \\
\equiv & \quad \{ \text{ (8) } \} \\
 & g.((b \vee r) \wedge (\neg b \vee wp.s.true)) \\
\equiv & \quad \{ \text{ (15) } \} \\
 & wp.DO.r
\end{array}
$$

(T1):

$$
\begin{array}{ll}
 & wlto.DO.p.q.\langle\forall r : r \in R : r\rangle \\
\equiv & \quad \{\ (45)\ \} \\
 & h.((b \vee \neg p \vee q \vee \langle\forall r : r \in R : r\rangle) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.\langle\forall r : r \in R : r\rangle))) \\
\equiv & \quad \{\ \text{(T1) for } s,\ \text{(E1) for } DO\ \} \\
 & h.((b \vee \neg p \vee q \vee \langle\forall r : r \in R : r\rangle) \wedge \langle\forall r : r \in R : \neg b \vee wlto.s.p.q.(wlev.DO.q.r)\rangle) \\
\equiv & \quad \{\ \text{calculus}\ \} \\
 & h.\langle\forall r : r \in R : (b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r))\rangle \\
\equiv & \quad \{\ h \text{ is universally conjunctive}\ \} \\
 & \langle\forall r : r \in R : h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r)))\rangle \\
\equiv & \quad \{\ (45)\ \} \\
 & \langle\forall r : r \in R : wlto.DO.p.q.r\rangle
\end{array}
$$

(T2):

$$
\begin{array}{ll}
 & wto.DO.p.q.r \\
\equiv & \quad \{\ (46)\ \} \\
 & h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wto.s.p.q.(wev.DO.q.r)) \\
\equiv & \quad \{\ \text{(E2)}\ \} \\
 & h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wto.s.p.q.(wev.DO.q.true \wedge wlev.DO.q.r)) \\
\equiv & \quad \{\ \text{(T1)}\ \} \\
 & h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wto.s.p.q.(wev.DO.q.true)) \wedge \\
 & \quad (\neg b \vee wto.s.p.q.(wlev.DO.q.r))) \\
\equiv & \quad \{\ \text{(T2) for } s\ \} \\
 & h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wto.s.p.q.(wev.DO.q.true)) \wedge \\
 & \quad (\neg b \vee wto.s.p.q.true) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r))) \\
\equiv & \quad \{\ \text{(T1)}\ \} \\
 & h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wto.s.p.q.(wev.DO.q.true)) \wedge \\
 & \quad (\neg b \vee wlto.s.p.q.(wlev.DO.q.r))) \\
\equiv & \quad \{\ h \text{ is conjunctive}\ \} \\
 & h.(\neg b \vee wto.s.p.q.(wev.DO.q.true)) \quad \wedge \\
 & h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r))) \\
\equiv & \\
 & h.((b \vee \neg p \vee q \vee true) \wedge (\neg b \vee wto.s.p.q.(wev.DO.q.true))) \quad \wedge \\
 & h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r))) \\
\equiv & \quad \{\ \text{(45) and (46)}\ \} \\
 & wto.DO.p.q.true \wedge wlto.DO.p.q.r
\end{array}
$$

(T3): $[wlto.DO.p.q.r \;\Rightarrow\; wlto.DO.p.q'.r]$

$$\equiv \quad \{\ (45)\ \}$$

$$[h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r)) \;\Rightarrow$$
$$h.((b \vee \neg p \vee q' \vee r) \wedge (\neg b \vee wlto.s.p.q'.(wlev.DO.q'.r))]$$

$$\Leftarrow \quad \{\ h \text{ is monotonic }\}$$

$$[wlto.s.p.q.(wlev.DO.q.r) \;\Rightarrow\; wlto.s.p.q'.(wlev.DO.q'.r)] \wedge [q \;\Rightarrow\; q']$$

$$\Leftarrow \quad \{\ \text{(E3) and (T1) for } DO\ \}$$

$$[wlto.s.p.q.(wlev.DO.q.r) \;\Rightarrow\; wlto.s.p.q'.(wlev.DO.q.r)] \wedge [q \;\Rightarrow\; q']$$

$$\Leftarrow \quad \{\ \text{(T3) for } s\ \}$$

$$[q \;\Rightarrow\; q']$$

(T4): $wlto.DO.p.q.r$

$$\equiv \quad \{\ (45)\ \}$$

$$h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r))$$

$$\equiv \quad \{\ (E4)\ \}$$

$$h.((b \vee \neg p \vee q \vee (q \vee r)) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.(q \vee r)))$$

$$\equiv \quad \{\ (45)\ \}$$

$$wlto.DO.p.q.(q \vee r)$$

(T5'): $wto.DO.p.p.r$

$$\equiv \quad \{\ (46)\ \}$$

$$h.((b \vee \neg p \vee p \vee r) \wedge (\neg b \vee wto.s.p.p.(wev.DO.q.r))$$

$$\equiv \quad \{\ \text{(T5') for } s\ \}$$

$$h.true$$

$$\equiv \quad \{\ h \text{ is universally conjunctive }\}$$

$$true$$

(T6): $\langle \forall p : p \in P : wlto.DO.p.q.r \rangle$

$$\equiv \quad \{\ (45)\ \}$$

$$\langle \forall p : p \in P : h.((b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r))) \rangle$$

$$\equiv \quad \{\ h \text{ is universally conjunctive }\}$$

$$h.\langle \forall p : p \in P : (b \vee \neg p \vee q \vee r) \wedge (\neg b \vee wlto.s.p.q.(wlev.DO.q.r)) \rangle$$

$$\equiv \quad \{\ \text{(T6) for } s\ \}$$

$$h.((b \vee \neg \langle \exists p : p \in P : p \rangle \vee q \vee r) \wedge (\neg b \vee wlto.s.\langle \exists p : p \in P : p \rangle.q.(wlev.DO.q.r))))$$

$$\equiv \quad \{\ (45)\ \}$$

$$wlto.DO.\langle \exists p : p \in P : p \rangle.q.r$$

(T7): $wlev.DO.q.r$ is the weakest solution of

$$y : [y \;\Rightarrow\; (\neg b \vee wlev.s.q.y) \wedge (b \vee q \vee r)]$$

and hence it suffices to show that a solution w exists that is implied by the premise of (T7), i.e.

$$[wlto.DO.p.q.r \wedge wlev.DO.p.r \;\Rightarrow\; w]$$

$$[w \;\Rightarrow\; (\neg b \vee wlev.s.q.w) \wedge (b \vee q \vee r)]$$

By choosing

$$[w \;\equiv\; wlto.DO.p.q.r \wedge wlev.DO.(p \vee q).r]$$

the first of these proof obligations follows from (E3). The second proof obligation is settled by

$$
\begin{array}{ll}
 & w \\
\equiv & \quad \{ \text{ definition of } w \ \} \\
 & wlto.DO.p.q.r \wedge wlev.DO.(p \vee q).r \\
\equiv & \quad \{ \ wlto.DO \text{ solves (42); } wlev.DO \text{ solves (19) } \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlto.s.p.q.(wlev.DO.q.r))) \wedge (b \vee \neg p \vee q \vee r) \ \ \wedge \\
 & (\neg b \vee wlev.s.(p \vee q).(wlev.DO.(p \vee q).r)) \wedge (b \vee p \vee q \vee r) \\
\equiv & \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlto.s.p.q.(wlev.DO.q.r) \wedge wlev.s.(p \vee q).(wlev.DO.(p \vee q).r))) \\
 & \ \ \wedge \ \ (b \vee q \vee r) \\
\Rightarrow & \quad \{ \text{ (E3) for } DO, \text{ monotonicity of } wlto.s.p.q, \ \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlto.s.p.q.(wlev.DO.(p \vee q).r) \ \ \wedge \\
 & \qquad wlev.s.(p \vee q).(wlev.DO.(p \vee q).r)) \ \ \wedge \\
 & (b \vee q \vee r) \\
\equiv & \quad \{ \text{ (T5) } \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlto.s.p.q.(wlev.DO.(p \vee q).r) \ \ \wedge \\
 & \qquad wlev.s.(p \vee q).(wlev.DO.(p \vee q).r) \wedge wlto.s.q.q.(wlev.DO.(p \vee q).r) \\
 & (b \vee q \vee r) \\
\equiv & \quad \{ \text{ (T6) } \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlto.s.(p \vee q).q.(wlev.DO.(p \vee q).r) \ \ \wedge \\
 & \qquad wlev.s.(p \vee q).(wlev.DO.(p \vee q).r)) \ \ \wedge \\
 & (b \vee q \vee r) \\
\Rightarrow & \quad \{ \text{ (T7) for } s \ \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlev.s.q.(wlev.DO.(p \vee q).r)) \ \ \wedge \ \ (b \vee q \vee r) \\
\Rightarrow & \quad \{ \text{ (18a) } \} \\
 & (\neg b \vee (wlev.s.q.(wlto.DO.p.q.r \wedge wlev.DO.(p \vee q).r)) \ \ \wedge \ \ (b \vee q \vee r) \\
\equiv & \quad \{ \text{ definition of } w \ \} \\
 & (\neg b \vee wlev.s.q.w) \ \ \wedge \ \ (b \vee q \vee r)
\end{array}
$$

(T7'): The proof of (T7') is quite different from the proof of (T7) and is, therefore, included. we have to show

$$[wto.DO.p.q.r \ \ \wedge \ \ wev.DO.p.r \ \Rightarrow \ wev.DO.q.r]$$

which is equivalent to

$$[wev.DO.p.r \ \Rightarrow \ \neg wto.DO.p.q.r \vee wev.DO.q.r] \qquad .$$

Since $wev.DO.p.r$ is the strongest solution of its defining equation, it suffices to show that

$$\neg wto.DO.p.q.r \vee wev.DO.q.r$$

is also a solution of that same equation (with equivalence replaced by implication). Here is the proof.

$$
\begin{array}{ll}
 & [(\neg b \vee wev.s.p.(\neg wto.DO.p.q.r \vee wev.DO.q.r)) \wedge (b \vee p \vee r) \\
 & \quad \Rightarrow\ \neg wto.DO.p.q.r \vee wev.DO.q.r] \\
\equiv & \quad \{ \text{ calculus } \} \\
 & [(\neg b \vee wev.s.p.(\neg wto.DO.p.q.r \vee wev.DO.q.r)) \wedge (b \vee p \vee r) \wedge wto.DO.p.q.r \\
 & \quad \Rightarrow\ wev.DO.q.r] \\
\equiv & \quad \{\ wto.DO.p.q.r \text{ solves (43) } \} \\
 & [(\neg b \vee wev.s.p.(\neg wto.DO.p.q.r \vee wev.DO.q.r)) \ \wedge \\
 & (\neg b \vee (wlp.s.(wto.DO.p.q.r) \wedge wto.s.p.q.(wev.DO.q.r))) \ \wedge \\
 & (b \vee p \vee r) \wedge (b \vee \neg p \vee q \vee r) \\
 & \quad \Rightarrow\ wev.DO.q.r] \\
\equiv & \quad \{ \text{ (18c) } \} \\
 & [(\neg b \vee (wev.s.p.(wev.DO.q.r) \wedge wlp.s.(wto.DO.p.q.r) \wedge wto.s.p.q.(wev.DO.q.r))) \ \wedge \\
 & (b \vee (p \wedge q) \vee r) \\
 & \quad \Rightarrow\ wev.DO.q.r] \\
\Leftarrow & \quad \{ \text{ (T7') for } s \text{ ; calculus } \} \\
 & [(\neg b \vee wev.s.q.(wev.DO.q.r)) \wedge (b \vee q \vee r) \ \Rightarrow\ wev.DO.q.r] \\
\equiv & \quad \{\ wev.DO.q.r \text{ solves (20) } \} \\
 & true
\end{array}
$$

(T8): We have

$$
\begin{array}{ll}
 & wlto.DO.p.q.r \wedge wlto.DO.q.w.r \\
\equiv & \quad \{\ wlto.DO \text{ solves (42) } \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlto.s.p.q.(wlev.DO.q.r))) \ \wedge \\
 & (\neg b \vee (wlp.s.(wlto.DO.q.w.r) \wedge wlto.s.q.w.(wlev.DO.w.r))) \ \wedge \\
 & (b \vee \neg p \vee q \vee r) \wedge (b \vee \neg q \vee w \vee r) \\
\Rightarrow & \quad \{ \text{ calculus } \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlp.s.(wlto.DO.q.w.r) \ \wedge \\
 & \qquad wlto.s.p.q.(wlev.DO.q.r) \wedge wlto.s.q.w.(wlev.DO.w.r))) \ \wedge \\
 & (b \vee \neg p \vee w \vee r) \\
\Rightarrow & \quad \{ \text{ (40a); (T7) for } DO \ \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r) \wedge wlp.s.(wlto.DO.q.w.r) \ \wedge \\
 & \qquad wlto.s.p.q.(wlev.DO.w.r) \wedge wlto.s.q.w.(wlev.DO.w.r))) \ \wedge \\
 & (b \vee \neg p \vee w \vee r) \\
\Rightarrow & \quad \{ \text{ conjunctivity of } wlp.s \text{ ; (T8) for } s \ \} \\
 & (\neg b \vee (wlp.s.(wlto.DO.p.q.r \wedge wlto.DO.q.w.r) \wedge wlto.s.p.w.(wlev.DO.w.r))) \ \wedge \\
 & (b \vee \neg p \vee w \vee r)
\end{array}
$$

which shows that $wlto.DO.p.q.r \wedge wlto.DO.q.w.r$ is a solution of (42), with equivalence replaced by implication. Since $wlto.DO.p.w.r$ is the weakest solution of (42), the result follows.

12. References

[0] R.J.R. Back; On the correctness of refinement in program development; PhD-thesis, University of Helsinki, 1978.

[1] R.J.R. Back, R. Kurki-Suonio; Decentralization of process nets with centralized control, *2nd ACM SIGACT-SIGOPS Symposium on principles of Distributed Computing*, Montreal, Canada, August 1983, 131-142.

[2] K. Mani Chandy, J. Misra; Parallel Program Design: A Foundation, Addison Wesley, 1988.

[3] Edsger W. Dijkstra; A Discipline of Programming, Prentice-Hall, 1976.

[4] Edsger W. Dijkstra, Carel S. Scholten; Predicate Calculus and Program Semantics, Springer-Verlag, 1990.

[5] David Gries; The Science of Programming, Springer-Verlag, 1981.

[6] C.S. Jutla, E. Knapp, J.R. Rao; A Predicate Transformer Approach to Semantics of Parallel Programs; *Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing 1989*, 249-263.

[7] Johan J. Lukkien; Parallel Program Design and Generalized Weakest Preconditions. Ph-D Thesis Groningen University, 1991.

[8] Joseph M. Morris; Temporal Predicate Transformers and Fair Termination; *Acta Informatica*, vol 27 (1990) 287-313.

[9] S. Owicki, L. Lamport; Proving Liveness Properties of Concurrent Programs; *ACM TOPLAS*, vol. 4, no. 3, July 1982, 455-495.

[10] A. Tarski; A lattice-theoretical fixpoint theorem and its applications; *Pacific Journal of Mathematics*, vol.5(1955), 285–309.