IMPARTIALITY, JUSTICE AND FAIRNESS: THE ETHICS OF CONCURRENT TERMINATION

by

D. Lehmann - Hebrew University , Jerusalem, Israel

A. Pnueli - Weizmann Institute, Rehovot, Israel

J. Stavi - Bar Ilan University, Ramat Gan, Israel

The research reported in this work was supported in part by a grant from the Israel Academy of Sciences, The Basic Research Foundation.

Abstract

The method of well founded structures for proving termination of programs is extended to concurrent programs. The more complicated case is when a program terminates only for fair executions. Different versions of fairness are introduced: Impartiality, Fairness and Justice, and Methods for proving their termination are presented.

Introduction

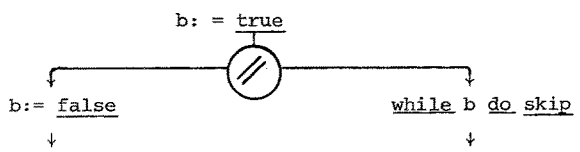
When we examine the development of the proof theory of sequential programs we observe that the foundation laid down in Floyd's pioneering work [F] still serves as the underlying principle upon which more elaborate formalisms have been constructed. The two basic principles suggested there: using inductive assertions for establishing invariances, and using well founded sets for proving termination or total correctness, are implicit in almost every other formalism or programming logic constructed since.

Going over to concurrent programs it seems that the same basic principles may still apply. Numerous works pointed out that the method of inductive assertions certainly works, and [K], [OG], [L] contain explicit directions for its efficient use.

On the other hand, the adaptation of the other principle, that of using well founded sets for proving termination, has not been so readily realized for the general case.

In this paper we study the different aspects of using well founded sets for establishing termination of concurrent programs. We then show that the methodology is easily extendable to cover the general case of an eventuality property and even 'until' properties.

The basic difficulty encountered in the immediate adaptation of the sequential methodology is that one cannot hope for a convergence function which decreases after every execution step. Consider the parallel execution of the following program:



As long as we execute instructions in the right process, nothing changes in the state, so that no convergence function can decrease. On the other hand every fair execution will eventually set b to false causing the whole program to terminate.

This suggests a convergence function which is guaranteed only to decrease eventually. It also shows that the difficulty is involved with the notion of fairness, when only fair executions are guaranteed to terminate while unfair executions may be infinite.

Our model for concurrent programs is rather abstract but we see no difficulties in translating the results to more concrete models. Also, the type of completeness we study is semantic in the sense that we are satisfied with showing that certain predicates and functions exist without investigating the question of their expressibility in a convenient formal language. We are sure that further studies in these directions will be shortly undertaken.

Preliminary Results from Set Theory

A binary relation R over a set S is called well founded if there does not exist an infinite sequence

$$s_0, s_1, \ldots \qquad s_i \in S$$

such that $s_i R s_{i+1}$ for all $i = 0,1,\ldots$

If R , a well founded relation, is also a strict partial order, i.e.

$$s_1 R s_2 \wedge s_2 R s_3 \Rightarrow s_1 R s_3$$

$$s_1 R s_2 \Rightarrow \sim s_2 R s_1$$

Then we define the structure (S,R) to be a well founded structure.

An example of a well founded structure is $(\alpha,>)$ where $\alpha \varepsilon ORD$ is an ordinal, representing the set of all ordinals smaller than α. The ">" is the regular, "greater than" ordering relation which is certainly well founded over the ordinals.

Theorem 1.

Let S be a set and R a well founded relation over it. Then there exist an ordinal α and a ranking function

$$\rho: S \to \alpha$$

such that:

a) $sRs' \Rightarrow \rho(s) > \rho(s')$

b) if for every $s'' \in S$, $s'Rs'' \Rightarrow sRs''$ then

$$\rho(s) \geq \rho(s')$$

Furthermore, if S is finite then the range of ρ is a finite set $\{0,1,\ldots n-1\}$ for some n.

If R is of finite degree, i.e. for every $s \varepsilon S$, the set $\{s' | sRs'\}$ is finite, then the range of ρ may be taken as the natural numbers or a subset of them.

If S is countable the range of ρ is a countable ordinal.

The theorem is proved by showing first that every non-empty set $A \subseteq S$ has an R-minimal element, i.e. an element $s \varepsilon A$ such that there is no $s' \varepsilon A$ such that sRs' This is based on the well foundedness of R. Let min(A) denote the set of all the minimal elements in A.

Then we define the possibly transfinite sequence:

$$S_0 = \min(S)$$
$$\cdot$$
$$\cdot$$
$$S_k = \min(S - \bigcup_{i<k} s_i)$$
$$\cdot$$
$$\cdot$$

and by transfinite induction for every limit ordinal β

$$S_\beta = \min(S - \bigcup_{\alpha<\beta} S_\alpha).$$

We define the ranking function ρ by

$$\rho(s) = \alpha <=> s \in S_\alpha.$$

It can be shown that this function satisfies the theorem's requirements and that there exists an ordinal α such that $\rho(s) < \alpha$ for all $s \in S$. The essential statement of this theorem is that every well founded relation R can be mapped into a well founded structure $(\alpha,>)$ for some ordinal α.

Programs and Total Convergence

In our discussion we will model a concurrent system by a pair

$P = \langle S,F,I \rangle$.

S - is a set of execution states.

$I \subseteq S$ - is the set of initial states.

$F = \{f_1,\ldots f_n\}$ - is a set of transition functions. They model the action of n different processors on a common state. Each $f_i: S \to 2^S$. In general $f_i(s)$ is the set of possible outcomes (i.e. successor states) when the i'th process is applied to the state s. We will refer to f_i as the i'th process.

If $f_i(s) \neq \emptyset$ we say that f_i is enabled in s, otherwise we say that it is disabled at s . A state s such that for all i, $f_i(s) = \emptyset$ is called terminal. We denote the set of terminal states in P by T.

An execution sequence of P is a maximal sequence:

$$s_0 \xrightarrow{f_{i_1}} s_1 \xrightarrow{f_{i_2}} s_2 \ldots$$

such that $s_0 \in I$, each $s_j \in S$, each $f_{i_j} \in F$ and for each j, $s_{j+1} \in f_{i_{j+1}}(s_j)$. Such a sequence can be infinite or finite and then has a last state which must be terminal. A state $s \in S$ is accessible if it occurs in some execution sequence. The set of accessible states is denoted by Acc(I).

We will sometimes refer to execution sequences as computations.

A program is defined to be totally convergent (abbreviated as total) if every computation of it is finite.

Method T. For proving that a program is total.

Find a predicate $Q: S \to \{F,T\}$ and a ranking function $\rho: S \to W$ mapping the set of states into a well founded structure $(W,>)$ such that:

1) For every $s \in I$, $Q(s)$ holds.

2) $Q(s) \wedge s' \in f_i(s) => Q(s') \wedge [\rho(s) > \rho(s')]$

Theorem T

A program P is totally convergent iff method T is applicable.

Assume that ρ and Q which satisfy 1) - 2) of method T have been found. Then the existence of an infinite computation of P

$$s_0 \xrightarrow{f_{i_1}} s_1 \xrightarrow{f_{i_2}} s_2 \ \ldots$$

implies the following: By 1) and 2) Q is true in every s_j. By 2) we have an infinitely decreasing chain

$$\rho(s_o) > \rho(s_1) > \ldots$$

in W, contradicting the well foundedness of W. Thus P must be total.

Let us show now that if P is total then there exist a ρ and a Q which satisfy the requirements of method T.

Define $Q(s) \iff s \in Acc(I)$

On the set of accessible states Acc(I) we define a binary relation

$$sRs' \iff \exists i\ s' \in f_i(s).$$

This relation is well founded. For assuming an infinite chain

$$s^1 R s^2 R s^3 \ldots$$

leads to the execution sequence $s^1 \xrightarrow{f_{i_2}} s^2 \rightarrow \ldots$

Since s^1 is accessible there is a finite computation

$$s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s^1$$

which when combined with the previous chain yields an infinite computation of P:

$$s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s^1 \rightarrow s^2 \rightarrow s^3 \rightarrow \ldots$$

This contradicts the totality of P. Consequently R is well founded. By theorem 1 there exists a ranking function ρ: Acc(I)$\rightarrow \alpha$ such that for s,s'$\in$Acc(I)

$$sRs' \Rightarrow \rho(s) > \rho(s')$$

or by the definition of Q and R

$$Q(s) \wedge \exists i\ s' \in f_i(s) \Rightarrow \rho(s) > \rho(s') .$$

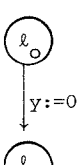
This and the definition of Q establish 2) of method T. Clause 1) is a trivial consequence of the concept of accessibility, i.e. $I \subseteq Acc(I)$. We may arbitrarily extend ρ to S-Acc(I) by setting $\rho(s) = 0$ for every $s \in S\text{-}Acc(I)$.

Impartial Programs

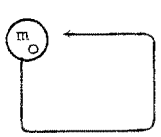
Many useful program are not total but do converge for all fair computations.

Consider the simple two process program:

y: = 1



$-P_1-$



$y > 0 \rightarrow [y:=y+1]$

$-P_2-$

States in this program are of the form $(\ell_0, m_0; u)$, $u \geq 0$ or $(\ell_1, m_0; 0)$. A state contains the current location of each of the processes and the current value of y. The state $(\ell_1, m_0; 0)$ is terminal since neither f_1 nor f_2 are enabled on it. On the other hand for every $(\ell_0, m_0; u)$, $f_1[(\ell_0, m_0; u)] = (\ell_1, m_0; 0)$ while for every $(\ell, m_0; u)$ $u > 0$

$$f_2[(\ell, m_0; u)] = (\ell, m_0; u+1) .$$

According to our definition of computations, this program is not total since it admits the following infinite computation:

$$(\ell_0, m_0; 1) \xrightarrow{f_2} (\ell_0, m_0; 2) \xrightarrow{f_2} (\ell_0, m_0; 3) \xrightarrow{f_2} \ldots$$

However, every realistically concurrent execution of this program will eventually get P_1 to complete its single instruction which will immediately halt the complete program.

Thus we are motivated to study the behavior of programs for more restricted classes of computations. The restrictions will be in the direction of constraining the execution to some degree of fairness.

As a first approximation of fairness we introduce the concept of <u>impartiality</u>.

An execution sequence is defined to be impartial if it is either finite or such that for every k, $1 \leq k \leq n$, there are infinitely many j's such that $f_{i_j} = f_k$, i.e., f_k appears infinitely many times in the sequence.

A program P is said to be <u>impartially convergent</u> or <u>impartial</u> if every <u>impartial</u> computation of P is finite.

Note that this is a first approximation to our intuition that all processes should be equally active and no process should be deprived forever.

<u>Method M.</u> For proving that program P is impartially convergent.

Find a ranking function ρ: $S \to W$ into a well founded set $(W,>)$ and predicates $Q_1, \ldots, Q_n$. Denote $Q = \bigvee_{i=1}^{n} Q_i$. Then the following must be satisfied:

M1) For every $s \in I$, $Q(s)$ holds.

M2) $Q(s) \wedge s' \in f_i(s) \Rightarrow Q(s') \wedge \rho(s) \geq \rho(s')$

M3) $Q_i(s) \wedge s' \in f_j(s) \wedge \rho(s) = \rho(s') \Rightarrow Q_i(s')$

M4) $Q_i(s) \wedge s' \in f_i(s) \Rightarrow \rho(s) > \rho(s')$

Clauses 1) and 2) ensure that Q is an invariant of all accessible states.
Clause 2) ensures that the ranking function never increases in a computation.
Clause 3) states that if Q_i holds at a given state and ρ does not decrease then Q_i remains true for the next state. Clause 4) ensures that taking an f_i transition out of a Q_i state will necessarily decrease ρ.

<u>Theorem M</u>

A program P is <u>impartially convergent</u> iff method M is applicable.

Assume that a ρ and $Q_1, \ldots Q_n$ have been found for a program P. Let

$$s_0 \overset{f_{i_1}}{\to} s_1 \overset{f_{i_2}}{\to} s_2 \ldots$$

be an impartial infinite computation. By 1) and 2) Q holds for all s_j in the computation and the sequence of ρ values must be non increasing $\rho(s_0) \geq \rho(s_1) \geq \ldots$.

Since W is well founded the value of ρ must stop decreasing beyond a certain point in the sequence, s_k say. By the definition of Q, Q_m must hold in s_k for some $1 \leq m \leq n$. Since ρ never decreases beyond s_k we have by 3) that Q_m also holds for all subsequent states s_j, $j \geq k$. By 4) transition f_m could never be taken any more, and hence can appear only a finite number of times in the computation. This contradicts of course the impartiality of the sequence.

Let us show now the completeness of the method.

Assume that P is impartially convergent. Define a relation R on the set of accessible states Acc(I) as follows: $sRs' \Leftrightarrow$ {There exists a computation path

$$s = s_0 \xrightarrow{f_{i_1}} s_1 \xrightarrow{f_{i_2}} s_2 \ldots \longrightarrow s_m = s'$$

such that each of the functions $f_1, \ldots f_n$ appears at least once on the path.}

An infinite chain

$$s_0 R s_1 R s_2, \ldots$$

will necessarily imply an infinite impartial computation since every function appears at least once on the path from s_j to s_{j+1} . Thus R is well founded.

By theorem 1 there exists a ranking function $\rho^0 : Acc(I) \to \alpha \varepsilon ORD$ such that

$$sRs' \Rightarrow \rho^0(s) > \rho^0(s').$$

Let $s' \varepsilon f_i(s)$ for an accessible s. Every path π from s' to some s", $s' \overset{\pi}{*} s"$, on which every f_j $j = 1, \ldots n$ appears can be extended into a path $s \xrightarrow{f_i} s' \xrightarrow{\pi}_{*} s"$ on which also every f_j, $j = 1, \ldots n$ appears. Thus $s'Rs" \Rightarrow sRs"$ for every s". Consequently by b) of theorem 1:

$$s' \varepsilon f_i(s) \Rightarrow \rho^0(s) \geq \rho^0(s') .$$

Consider a computation path:

$$\pi : s_0 \xrightarrow{f_{i_1}} s_1 \xrightarrow{f_{i_2}} \ldots \quad s_{k-1} \xrightarrow{f_{i_k}} s_k$$

Denote by $\sigma(\pi) = \{f_{i_1}, \ldots f_{i_k}\}$ the set of transitions taken in π. Define the weight of a path to be:

$$w(\pi) = \text{least } \{j \geq 1 \mid j \notin \sigma(\pi)\}$$

i.e. the minimal transition not taken in π. Clearly $w(\pi) \varepsilon [1..n+1]$. If π is empty than $w(\pi) = 1$.

Define now for every $s \varepsilon Acc(I)$:

$$\rho'(s) = \max\{w(\pi) \mid s \xrightarrow{\pi}_{*} s' \text{ for some path } \pi \text{ and } s' \text{ such that }, \rho^0(s)=\rho^0(s')\}$$

$\rho'(s)$ is the maximal weight of all paths which connect s to other states with the same value of ρ^0.

Since the empty path is always in the set considered, $\rho'(s) \geq 1$. Clearly $\rho'(s) \leq n+1$. But if $s \xrightarrow{\pi}_{*} s'$ is such that $\rho^0(s) = \rho^0(s')$ then $\sigma(\pi)$ is a strict subset of $[1..n]$ and $w(\pi) \leq n$. Otherwise sRs' which implies $\rho^0(s) > \rho^0(s')$. Con-

sequently $\rho'(s) \varepsilon [1..n]$. Note that if there is no $s' \neq s$ such that $\rho^0(s) = \rho^0(s')$ and $s \xrightarrow[*]{} s'$ then $\rho'(s) = 1$.

The ranking functions ρ^0 and ρ' may be arbitrarily extended to state $s \varepsilon$ S-Acc(I), e.g. $\rho^0(s) = \rho'(s) = 1$.

We are ready to define the ranking functions and predicates required by method M.

The well founded structure is taken to be the cross product of the ranges of ρ^0 and ρ' with the lexicographically defined ordering. Thus we take $W = \alpha \times [1..n]$, with the ordering $\gtrsim_{\ell}$ defined by: $(m,m') \gtrsim_{\ell} (n,n')$ iff $m > n$ or ($m=n$ and $m' > n'$). This is of course the usual ordinal multiplication $\alpha \times n$. It can be shown that this is a well founded structure.

For our function ρ mapping S into W we take:

$$\rho(s) = (\rho^0(s), \rho'(s)) \varepsilon W$$

For our predicates Q_i, i=1,..n we take:

$$Q_i(s) \Longleftrightarrow [s \varepsilon Acc(I) \text{ and } \rho'(s) = i]$$

We will proceed to prove that this choice satisfies clauses M1 to M4.

Since $\rho'(s) \varepsilon [1..n]$ every accessible state s satisfies $Q_i(s)$ for some i, $1 \leq i \leq n$. Consequently

$$Q(s) \Longleftrightarrow \bigvee_{i=1}^{n} Q_i(s) \Longleftrightarrow s \varepsilon Acc(I).$$

Next we show that if s is accessible and $s' \varepsilon f_i(s)$ then $(\rho^0(s), \rho'(s)) \gtrsim (\rho^0(s'), \rho'(s'))$.

We already know that $\rho^0(s) \gtrsim \rho^0(s')$. If $\rho^0(s) > \rho^0(s')$ we are done, so let us assume that $\rho^0(s) = \rho^0(s')$. Let π' be any path leading from s' to s" such that $\rho^0(s') = \rho^0(s")$.

$$\pi' : s' \xrightarrow[*]{} s"$$

We can augment it by the initial f_i transition to obtain:

$$\pi : s \xrightarrow{f_i} s' \xrightarrow[*]{\pi'} s" \qquad \text{and } \rho^0(s) = \rho^0(s")$$

Consequently $\sigma(\pi) \supseteq \sigma(\pi')$ and $w(\pi) \gtrsim w(\pi')$ for every such π'. Thus we have $\rho'(s) \gtrsim \rho'(s')$

To establish M3 let $Q_i(s)$ hold and $s' \varepsilon f_j(s)$, such that $\rho^0(s) = \rho^0(s')$ and $\rho'(s) = \rho'(s')$.

By the fact that $Q_i(s)$ holds we have that s is accessible and $\rho'(s) = i$. This implies that s' is also accessible and $\rho'(s') = i$, leading to $Q_i(s')$.

To establish M4 let $Q_i(s)$ hold and $s' \varepsilon f_i(s)$. We may suppose that $\rho^0(s) = \rho^0(s')$ (otherwise M4 is immediate)and proceed to show that $\rho'(s) > \rho'(s')$. Let π' be the path determining $\rho'(s')$, i.e.

$$s' \xrightarrow[*]{\pi'} s" \quad \text{where } \rho^0(s') = \rho^0(s") \text{ and } \rho'(s') = w(\pi')$$

Consider the path π: $s \xrightarrow{f_i} s' \xrightarrow[*]{\pi'} s"$.

Since $\rho^0(s) = \rho^0(s")$, $\rho'(s) \gtrsim w(\pi) > w(\pi') = \rho'(s')$. Also since $i \varepsilon \sigma(\pi)$, $w(\pi) \neq i$, Hence $i = \rho'(s) > w(\pi) \gtrsim \rho'(s')$ which leads to $\rho(s) > \rho(s')$.

As an example of using method M to prove the impartial convergence of a program, consider the following program for the distributed computation of the gcd function

$$P_1 : *[y_1 > y_2 \longrightarrow [y_1 := y_1 - y_2] \; [] \; y_2 > y_1 \longrightarrow \text{skip}]$$

$$P_2:*[y_1 > y_2 \longrightarrow \text{skip} \;\; [\!] \;\; y_2 > y_1 \longrightarrow [y_2 := y_2 - y_1]]$$

Assuming positive inputs, the set of states for this program consists of pairs, $S = \{(u_1,u_2) \mid u_1,u_2 > 0\}$. The initial set I=S in this case. There are two processes here, f_1 and f_2 given by :

$f_1(y_1,y_2)$ = if $y_1 > y_2$ then (y_1-y_2,y_2) else if $y_1 < y_2$ then (y_1,y_2)

$f_2(y_1,y_2)$ = if $y_1 > y_2$ then (y_1,y_2) else if $y_1 < y_2$ then (y_1,y_2-y_1)

Note that both functions are undefined (disabled) on the set of terminal states.

$$T = \{(u,u) \mid u > 0\} \subseteq S.$$

This program is not totally convergent, because starting for example from a state (u_1,u_2) in which $u_1 > u_2$ we can generate an infinite computation by applying only the f_2 transition which leaves such a state invariant. On the other hand every impartial computation which interleaves applications of f_1 and f_2 in some fair manner will eventually terminate. To prove this formally by method M we choose:

$(W,>)$ the well founded structure to be $(N,>)$, i.e. the natural numbers with the usual "greater than" ordering.

For ranking function ρ we take

$$\rho(u_1,u_2) = u_1 + u_2 .$$

For our predicates we choose:

$$Q_1(u_1,u_2) \;\; : \;\; (u_1 \geq u_2 > 0)$$

$$Q_2(u_1,u_2) \;\; : \;\; (0 < u_1 < u_2)$$

Obviously, either Q_1 or Q_2 are true ($Q_1 \vee Q_2 \equiv u_1,u_2 > 0$) for any state in S.

The function ρ does not increase under any application of f_1 or f_2 which may either preserve the value of ρ or decrease it. This guarantees M2.

The cases in which ρ does not decrease under applications of f_1,f_2 preserve the complete state (u_1,u_2) so that if Q_i i=1,2 was true before it is certainly preserved. This satisfies M3.

To show M4 we observe that if f_1 is applicable (enabled) on a state satisfying Q_1 then $u_1 > u_2$ and the resulting state $(u_1 - u_2,u_2)$ has a strictly lower ρ value. Similarly, application of f_2 on a state satisfying Q_2 yields $(u_1,u_2 - u_1)$ which has a strictly lower ρ value than the original state.

Just Programs

Clearly, the notion of impartiality is only an approximation to the concept of fairness that really interests us. Its inadequacy becomes apparent in cases when one of the process has already terminated. Consider the following simple program:

$$P_1:[\ell_0: \text{skip};\ell_1:] \;\; || \;\; P_2:[m_0:*[y > 0 \longrightarrow [y: =y+1]]; m_1:]$$

Formally, this program is impartially convergent since it does not admit any infinite computation in which both P_1 and P_2 (f_1 and f_2 respectively) are activated infinitely

many times. This is a trivial consequence of the fact that f_1 can be applied at most once in any computation.

However under no reasonable interpretation can such a program be considered to be convergent. Consequently we introduce a more discriminating concept - that of justice.

A computation is said to be just if it is finite or if every transition which is continuously enabled beyond a certain point is taken infinitely many times.

Let us reconsider the example above. Its states are of the form

$S = \{(\ell,m;u) \mid \ell\in\{\ell_0,\ell_1\},\ m\in\{m_0,m_1\},\ u \geq 0\}$

The set of initial state $I = \{(\ell_0,m_0;u) \mid u \geq 0\}$

The transition functions are

$f_1(\ell,m;u)$ = if $\ell=\ell_0$ then $(\ell_1,m;u)$

$f_2(\ell,m;u)$ = if $m=m_0$ then if $u > 0$ then $(\ell,m_0;u+1)$ else $(\ell,m_1;u)$

The infinite computation:

$$(\ell_0,m_0;1) \xrightarrow{f_2} (\ell_0,m_0;2) \xrightarrow{f_2} (\ell_0,m_0;3) \xrightarrow{f_2} \ldots$$

is an unjust computation, since f_1 is enabled on all states in it but is never taken.

On the other hand, the infinite computation:

$$(\ell_0,m_0;1) \xrightarrow{f_2} (\ell_0,m_0;2) \xrightarrow{f_1} (\ell_1,m_0;2) \xrightarrow{f_2} (\ell_1,m_0;3) \xrightarrow{f_2} (\ell_1,m_0;4) \xrightarrow{f_2} \ldots$$

is just. This is so because f_1 is disabled beyond the third state and consequently does not have to be taken anymore. The f_2 transition which is continuously enabled, is in fact taken infinitely many times.

A program P is said to converge justly or be just if every just computation of P is finite.

Thus the program above is not justly convergent, having an infinite just computation.

We offer the following method for proving the just convergence of a program:

Method J for proving that a program P is justly convergent.

Find a ranking function $\rho : S \to W$ into a well founded structure $(W,>)$ and predicates $Q_1,..Q_n$. Denote $Q = \bigvee_{i=1}^{n} Q_i$, and let T denote the set of terminal states in P, $T \subseteq S$.

The $\rho,Q_1,..Q_n$ must satisfy:

J1) For every $s\in I$, $Q(s)$ holds.

J2) $Q(s) \wedge s'\in f_i(s) \Rightarrow Q(s') \wedge \rho(s) \geq \rho(s')$

J3) $Q_i(s) \wedge s'\in f_j(s) \wedge \rho(s) = \rho(s') \Rightarrow Q_i(s')$

J4) $Q_i(s) \wedge s'\in f_i(s) \Rightarrow \rho(s) > \rho(s')$

J5) $Q_i(s) \wedge s\notin T \Rightarrow f_i(s) \neq \emptyset$

Requirements J1-J4 are identical to M1-M4 in method M. The added J5 requires that the transition f_i is always enabled on each non terminal state in Q_i. This adds to the

guarantee in J4 that taking f_i from Q_i will decrease ρ , the assurance that being in Q_i we can always choose to take f_i .

For justifying method J we present:

Theorem J (Soundness and Completeness of method J)

A program is justly convergent iff method J is applicable.

Assume that method J is applicable and the required $(W,>),\rho,Q_1,..Q_n$ have been found.

Consider an infinite computation:

$$s_o \xrightarrow{f_{i_1}} s_1 \xrightarrow{f_{i_2}} s_2 \longrightarrow$$

Arguing as in theorem M we can establish the permanence of some Q_m beyond a certain state s_k in the computation, such that ρ does not decrease beyond that state. Because of condition J4 this implies that f_m is never taken beyond s_k . By J5 and the fact that this is an infinite computation and no states in it are terminal, f_m is continuously enabled beyond s_k but never taken there. The infinite computation we considerd is therefore not a just computation.

To show that just convergence implies applicability of method J we follow a construction similar to that of theorem M.

Given a computation path π of length $k \geq 0$:

$$\pi : s_o \xrightarrow{f_{i_1}} s_1 \xrightarrow{f_{i_2}} \cdots \xrightarrow{f_{i_k}} s_k$$

we define $\sigma(\pi) = \{f_{i_1} .. f_{i_k}\}$ i.e. the set of all transitions taken in π . We also define:

$$e(\pi) = \{f_j \mid f_j(s_i) \neq \emptyset, \forall i = 0,1,..k\}$$

The set $e(\pi)$ is the set of all transitions which are enabled on all the states of π . A path π is called full if $e(\pi) \subseteq \sigma(\pi)$, i.e. all transitions which are everywhere enabled in π are also taken there.

We define now a relation R by:

$sRs' \iff$ {There exists a non empty path $\pi : s \xrightarrow{*} s'$ which is full, i.e. $e(\pi) \subseteq \sigma(\pi)$ }

The proof proceeds similarly to that of theorem M. The only difference is in the definitions of $W(\pi)$ for a computation path π which is given by:

$$W(\pi) = \text{least } \{j \geq 1 \mid j \in e(\pi) - \sigma(\pi)\}$$

To illustrate method J reconsider the program for the computation of the gcd function presented above. Note that for states s such that $Q_1 \vee Q_2$ holds and s is non terminal, $u_1 \neq u_2$ which ensures that both f_1 and f_2 are enabled. This establishes requirement J5.

Semaphores and Fairness

There are however circumstances in which the notion of justice is not strong enough. Typical of these are programs which use semaphores.

Consider the following simplified program in which two processes use a semaphore variable y in order to ensure mutual exclusion in their critical sections ℓ_o and m_o

respectively.

y = 1

ℓ_0 : idle	m_0 : idle
ℓ_1 : P(y)	m_1 : P(y)
ℓ_2 : V(y)	m_2 : V(y)
go to ℓ_0	go to m_0
$-P_1-$	$-P_2-$

The P(y) sempahore operation is equivalent to the guarded command [y>0→[y:=y-1]] and let the respective process through only if y > 0. The V(y) semaphore instruction is equivalent to the statement y:=y+1.

As states in this program we can take:

$$S = \{ (\ell,m;u) \mid \chi(\ell = \ell_2) + \chi(m = m_2) + y = 1 \}$$

The characteristic function χ is defined by $\chi(\underline{\text{true}}) = 1$ and $\chi(\underline{\text{false}}) = 0$. This ensures that $u = 0$ iff either P_1 is at ℓ_2 or P_2 is at m_2. It can be shown that only such states can arise during a proper execution of the program. The transition function f_1 is given by:

$$f_1(\ell,m;u) = \underline{\text{if}}\ \ell = \ell_0\ \underline{\text{then}}\ \{(\ell_0,m;u),(\ell_1,m;u)\}\ \underline{\text{else if}}\ \ell = \ell_1 \wedge u > 0\ \underline{\text{then}}\ (\ell_2,m;u-1)\ \underline{\text{else if}}\ \ell = \ell_2\ \underline{\text{then}}\ (\ell_0,m;u+1)$$

Note that f_2 is not enabled on a state $(\ell_1,m;0)$. The f_2 transition is similarly defined.

Unfortumately, among the just computations of this program we also find the following infinite computation:

$$(\ell_0,m_0;1) \xrightarrow{f_1} [(\ell_1,m_0;1) \xrightarrow{f_2} (\ell_1,m_1;1) \xrightarrow{f_2} (\ell_1,m_2;0) \xrightarrow{f_2}]^*$$

This computation activates f_1 only once and deprives P_1 from ever entering its critical section. It is still a just sequence since f_1 is never continuously enabled. In fact it is infinitely often disabled on the recurring state $(\ell_1,m_2;0)$.

To remedy this situation we introduce the notion of a fair computation.

A computation is said to be fair if it is finite or if every transition which is enabled infinitely many times is also taken infinitely many times.

Thus, the computation above is not fair since the function f_1 while being enabled infinitely many times (on $(\ell_1,m_0;1)$ for example) is taken only once.

Note that for programs that have no semaphore instructions and their transition function are enabled on all but the terminal locations, the notions of fair and just computation coincide.

We define a program P to be fairly convergent if every fair computation of P is finite.

In considering fair termination of arbitrary programs we can offer the following characterization theorems:

C1: A program P which is justly convergent is also fairly convergent.

C2: A program P is fairly convergent iff every program P_K for every $K \subseteq \{1,...n\}$ is impartially convergent.

Here

$$P_K = \langle S, \{f_i \mid i \in K\}, S_K \rangle$$

where

$$S_K = \{s \mid s \in S, \ s \text{ accessible and } \forall j \notin K \ f_j(s) = \emptyset\}$$

This is the set of states which are accessible and whose set of enabled transitions is contained in K.

C3: A program P is fairly convergent iff P is impartially convergent and every program P_i, for i = 1,...n is fairly convergent.

Here

$$P_i = \langle S, \{f_1', ..f_n'\}, Acc(I) \rangle$$

where

$$f_j'(s) = \begin{cases} f_j(s) & \text{if } f_i(s) = \emptyset \\ \emptyset & \text{if } f_i(s) \neq \emptyset \end{cases}$$

The effect of this definition is to make all states in which f_i is enabled, terminal states in P_i.

This result can be worked into a proof method as follows:

Method F

Find $(W,>)$, ρ, $Q_1,..Q_n$ as in method M which satisfy M1-M4 of method M and in addition:

F5) For every i,

$$\Pi_i = \langle \Sigma_i, \{f_1', ..f_n'\}, \Sigma_i \rangle$$

is fairly convergent, where $f_1', ..f_n'$ are defined as above, and

$$\Sigma_i = \{s \mid Q_i(s) \text{ holds}\} .$$

This method calls for the recursive application of itself to prove the fair convergence of Π_i, i = 1,..n. However, since the Π_i contain one transition less this recursion is well founded and will eventually terminate.

Theorem F

A program P converges fairly iff method F is applicable.

The proof is a direct consequence of statement C3 above.

Let us illustrate the application of method F to a proof of accessibility of the critical section ℓ_2 in the semaphore program above. That is show that if P_1 is ever at ℓ_1 it will eventually get to ℓ_2 under all fair computations. While this is not a termination statement it can easily be worked into one by considering the modified program:

$$\hat{P} = (S, \{h_1, h_2\}, I)$$

where S is defined as above,

For i=1,2 $h_i(\ell,m;u)$ = if $\ell \neq \ell_2$ then $f_i(\ell,m;u)$

$I = \{(\ell,m;u) \mid (\ell,m;u) \in S \text{ and } \ell = \ell_1\}$

Thus for initial states we consider all states in which $\ell=\ell_1$ and the h_i's are identical to the f_i's except that under them every state in which $\ell=\ell_2$ is terminal. Certainly, proving the termination of $\hat{P}$ will establish the accessibility in P_1.

According to method F we choose (W,>) = (0..1,>), $\rho(\ell,m;u)$ = if $\ell=\ell_0$ then 1 else 0.

$Q_1(\ell,m;u)$: $\ell \in \{\ell_1,\ell_2\}$

$Q_2(\ell,m,u)$: false

It can be shown that Q_1 is true for every $s \in I$, and is preserved by any h_1 or h_2 application. Also the value of ρ under an h_i, i=1,2 application to a state $s \in Q_1$ does not increase, and actually decreases under the action of h_1.

Consider next the subprogram:

$$\Pi_1 = < \Sigma_1, \{h_1^1, h_2^1\}, \Sigma_1 >$$

$$\Sigma_1 = \{(\ell,m;u) \mid \ell \in \{\ell_1,\ell_2\}\}$$

$h_1^1(s) = \emptyset$ for every state $s \in \Sigma$,

$h_2^1(\ell,m;u)$ = if $(\ell=\ell_2) \wedge (u=0)$ then $h_2(\ell,m;u)$

Thus h_2^1 is disabled on $(\ell_1,m;1)$ and on every $(\ell_2,m;u)$. Consequently, it is only enabled on $(\ell_1,m_2(0))$, and it is easy to see that Π_1 is impartially convergent, the longest computation being of length 2.

Π_2 is even more trivial, having $\Sigma_2 = \emptyset$.

We conclude the discussion of fairness by proving the claims made above, i.e. C1-C3. To prove C1 we observe the following inclusion relations between the concepts discussed:

Every impartial computation is also a fair computation.

Every fair computation is also a just computation.

Consequently every program that is justly convergent is also fairly convergent and every program that is fairly convergent is also impartially convergent.

The statement C2 claims that a program P is fairly convergent iff every P_K for $K \subseteq \{1,..n\}$ is impartially convergent.

Assume that P is fairly convergent. Consider a computation

$$\sigma = s_0, s_1, \ldots$$

which is an infinite impartial computation of P_K for some $K \subseteq \{1,..n\}$. Thus each of $f_i, i \in K$ is taken infinitely many times in σ and no f_j $j \notin K$ is enabled on any s_i (by the definition of S_K). Consequently σ is a fair computation of P and must terminate.

To show the other direction, let each P_K be impartially convergent. Consider a fair computation of P:

$$\sigma = s_0, s_1, \ldots$$

Let K be the set of i's such that f_i is taken infinitely many times in σ. By fairness, there is an s_k such that no f_j for $j \notin K$ is enabled beyond s_k. Consequently the suffix

sequence

$$s_k, s_{k+1}, \cdots$$

is an impartial computation of P_K and must terminate.

Theorem C3 is proved in a similar way.

REFERENCES

[F] Floyd, R.W. - Assigning meaning to programs, in J.T. Schwartz (ed.) - Mathematical aspects of computer science, Proc. Symp. in Appl. Math. 19,A.M.S. 1967, 19-32.

[K] Keller, R.M. - Formal Verification of Parallel Programs, CACM 19 (7) 1976.

[L] Lamport, L. - Proving the Correctness of Multiprocess Programs, IEEE Transactions on Software Engineering, 3 (2) 1977, 125-143.

[MP1] Manna, Z., Pnueli, A. - The Temporal Logic of Programs - A manuscript, 1980.

[MP2] Manna, Z., Pnueli, A. - Temporal Verification of Concurrent Programs, the Nontemporal Alternative, Workshop on Program Logics, Yorktown Heights, May 1981.

[OG1] Owicki, S., Gries, D. - An Axiomatic Proof Technique for Parallel Programs, Acta information 5, 319-339.

[OG2] Owicki, S., Gries, D. - Verifying Properties of Parallel Programs: An Axiomatic Approach, CACM 19(5) 1976, 279-286.