A PROOF RULE FOR FAIR TERMINATION OF GUARDED COMMANDS

by

Orna Grümberg
Nissim Francez
Johann A. Makowsky
Willem P. de Roever

RUU-CS-81-2
January 1981

Rijksuniversiteit Utrecht

Vakgroep informatica

Princetonplein 5
Postbus 80.002
3508 TA Utrecht
Telefoon 030–53 1454
The Netherlands

A PROOF RULE FOR FAIR TERMINATION OF GUARDED COMMANDS

by

Orna Grümberg
Nissim Francez
Johann A. Makowsky
Willem P. de Roever

Technical Report RUU-CS-81-2
January 1981

Department of Computer Science
University of Utrecht
P.O. Box 80.002
3508 TA Utrecht, the Netherlands

Please send any questions concerning soundness and completeness of the rule (i.e. section 3) to Willem P. de Roever, Department of Computer Science, University of Utrecht, P.O. Box 80.002, 3508 TA Utrecht, the Netherlands.

A PROOF RULE FOR FAIR TERMINATION OF GUARDED COMMANDS*

by

Orna Grümberg 1)
Nissim Francez 1)
Johann A. Makowsky 1)
Willem P. de Roever 2)

27 January 1981, Haifa

Summary: We present a proof rule for fairly terminating guarded commands based on a wellfoundedness argument. The rule is applied to several examples, and proved to be sound and complete w.r.t. an operational semantics of computation trees. The rule is related to another rule suggested by Pnueli, Stavi, and Lehmann, by showing that the (semantic) completeness of the [PSL]-rule follows from the completeness of ours'.

Keywords and phrases: Termination, fairness, guarded commands, soundness and semantic completeness, computation trees, infinitary trees.

CR-categories: 5.24.

Affiliation of authors: 1) Department of Computer Science, Technion, Haifa.
2) Vakgroep Informatica, Rijksuniversiteit Utrecht, Utrecht.

---

* Preliminary work regarding this problem was carried out while the 2nd author visited the University of Utrecht, sponsored by a grant from the Netherlands Organization for the Advancement of Pure Research (Z.W.O.); the work was completed while the 4th author visited the Technion sponsored by the Technion; the 2nd author was partly sponsored by an IBM-Israel research grant. The 3rd author was supported by Swiss National Science Foundation grant no. 82.820.0.80.

I. Introduction.

The use of well-ordered sets to prove termination of programs originates from [Floyd] and remained prominent ever since. After the appearance of nondeterministic and concurrent programming language constructs, the notion of termination was generalized to the notion of liveness [Lamport], which also covers properties such as eventual occurrence of events during program execution. One way of specifying and proving such properties is by applying temporal reasoning [Francez & Pnueli]. This may be formalized by using Temporal Logic [Pnueli], a tool suitable for expressing such eventualities.

Within the framework, one of the more interesting concepts that can be studied is the concept of fairness [GPSS]. However, application of temporal reasoning does not appeal to a direct use of well-foundedness arguments, see e.g. [Lamport & Owicki]. Recently, there is a revival of the interest in such direct appeals, see, e.g. [Apt & Plotkin], generalizing arguments hitherto involving finite nondeterminism to a context of infinite nondeterminism, and [PSL], generalizing sequential well-foundedness arguments to the context of concurrency (using a shared variable model).

A common property of well-foundedness arguments for more complicated types of termination is the use of higher countably infinite ordinals, which can be traced back to [Hitchcock & Park], this in contrast to the fact that for deterministic programs (or programs displaying finite nondeterminism) natural numbers suffice.

In this paper, we propose a rule for proving fair termination of guarded loops using well-foundedness arguments, expressed in FGC, a variant of Dijkstra's guarded commands language, restricted to fair execution sequences.

We chose guarded commands [Dijkstra] since it is relatively well-known and simple, has a natural extension to the language Communicating Sequential Processes - CSP [Hoare], and the proof-rule proposed in this paper extends equally naturally to CSP. This extension is the subject of a companion paper.

The ideas in this paper were developed mostly independent of [PSL], in which a similar situation is dealt with. We shall describe the influence of [PSL] on our work in the last section.

In section II we briefly describe an informal operational semantics of FGC programs, introduce the proof rule for fair termination and apply it to several examples. In section III we present soundness and semantic completeness proofs of the suggested rule w.r.t. an operational semantics using computation trees. Section IV ends with a reduction of the semantic completeness of the rule of [PSL], called method F, to the present one.

II. A proof rule for fair termination

Basic notions and definitions

We consider the language FGC, which has the same syntax as "ordinary" guarded commands. Thus:

```
<statement>              ::= <assignment statement> | <composition>
                             |<skip> | <selection> | <repetition>
<assignment statement>   ::= <variable> := <expression>
<composition>            ::= <statement>; <statement>
<skip>                   ::= skip
<selection>              ::= [<boolean-expression> → <statement>
                              {[]
                              <boolean-expression> → <statement>}*
                             ]
<repetition>             ::= *<selection>
```

Boolean expressions are also called guards.

The semantics of FGC programs, however, differs from the usual semantics in that only fair execution sequences are considered.

We follow the usual definition of a computation sequence π: $\xi_0\ \xi_1\ \ldots$, where all ξ_i's denote states (mappings from variables to values).

In the sequel we consider programs of the form of repetitions

$$C :: *[B_1 \rightarrow C_1\ []\ \ldots\ []\ B_n \rightarrow C_n],$$

also abbreviated to $*[\ []_{i\in\{1,\ldots,n\}}\ B_i \rightarrow C_i]$.

C_i is enabled in ξ iff $B_i(\xi)$ holds.

Definition 1. An execution sequence π of C is fair iff it is finite, or it is infinite and for every $1 \leq i \leq n$, if C_i is infinitely often enabled along π, it is also infinitely often chosen along π.

2. C is fairly terminating iff all its infinite execution sequences are not fair, i.e., unfair.

Thus, a fairly terminating program has finite computation sequences (terminating computations), and unfair infinite computation sequences, which are excluded by the FGC semantics, but may not have infinite fair computation sequences.

Actually, for a given initial state ξ, we consider the tree of all possible fairly terminating computation sequences, T_ξ. In case of a repetition, $*[B_1 \to C_1 \ [] \ \ldots \ [] \ B_n \to C_n]$, a state(a node) in T_ξ has subtrees for every i, $1 \leq i \leq n$ s.t. $B_i(n)$ holds. Observe that T_ξ contains finite and unfair infinite computation paths.

Our goal is to characterize deductively the class of all fairly terminating FGC programs. The characterization suggested does carry over directly to concurrent programs with shared variables; a companion paper [Francez & de Roever] extends it to CSP.

We use the notation <r> C <q> to express that C fairly terminates in all initial states satisfying r, and that q holds upon termination.

The intuition behind the suggested proof rule is as follows:
For an always terminating nondeterministic program, there exists a well-founded quantity which decreases along every computation sequence i.e., along every direction in the computation tree.
Now, let us choose the directions along which a certain well-founded quantity decreases, taking care that these directions (certain moves C_i) are always eventually enabled, until they are taken. Let the other directions be non-increasing. Then by the fairness assumption eventually a decreasing move has to accur. Thus all fair computation sequences are guaranteed to be finite.

The proof rule

Choose a well-ordered set $(W,<)$ (without loss of generality we can assume that W is an initial sequence of the countable ordinals, as shown by the completeness proof). Also choose a predicate

$p : W \to [\text{States} \to \{\underline{\text{true}}, \underline{\text{false}}\}]$, assigning a truth value to every pair (w,ξ).

For each $w \in W$, $w > 0$ (or, in general, any non-minimal element in W) choose a partition D_w, S_w of $\{1,\ldots,n\}$, with $D_w \neq \emptyset$ (-D stands for decreasing, S for steady).

Let the following clauses hold:

1) $\langle p(w) \wedge w \gneq 0 \wedge B_j > C_j < \exists v \lneq w.p(v)\rangle$, for all $j \in D_w$,

2) $\langle p(w) \wedge w \gneq 0 \wedge B_i > C_i < \exists v \leq w.p(v)\rangle$, for all $i \in S_w$,

3) $\langle p(w) \wedge w \gneq 0\rangle * [\ [] \ _{i \in S_w} B_i \wedge \neg \vee \ _{j \in D_w} B_j \to C_i] \ \underline{\langle \text{true} \rangle}$

4) $p(0) \supset \bigwedge_{i=1}^{n} \neg B_i$, $w > 0 \wedge p(w) \supset \bigvee_{i=1}^{n} B_i$, $r \supset \exists v.p(v)$

Then we conclude

$$<r> \; C \; <p(0)>,$$

i.e., repetition C fairly terminates.

Explanation

ad 1) This clause guarantees that along every direction in D_w, if it is enabled and taken, then there is a decrease in the well-ordering. (Note again that we use a unique minimal element, denoted by 0, to keep the notation simple.) Note also that at least one decreasing direction is required.

ad 2) This clause guarantees that along every direction in S_w, if enabled and taken, there is no increase in the well-ordering. Thus, an infinite computation proceeding along S_w directions only, and not decreasing, is possible. We have to assume that such a sequence is unfair. Whence clause 3).

ad 3) This clause imposes a recursive application of the rule to an auxiliary program C_w, and hence requires a subproof. C_w terminates because of one of two reasons:

a) $\bigwedge_{i \in S_w} \neg B_i$ is true, hence no S_w move is possible, and only D_w-moves are left.

b) For some $j \in D_w$, B_j is true, i.e., a D_w-move is enabled.

Hence, this clause guarantees that along infinite S_w-computations, D_w-moves are infinitely often enabled, that is, such computations are unfair.

By convention, C_w = skip if $S_w = \emptyset$.

ad 4) This clause guarantees that the program terminates only when reaching a minimal element of $(W,<)$.

Remarks 1. Without clause 3), our rule is complete for programs terminating under the weaker assumption that each of their computation sequences satisfies the following property: either it is finite or every guard is infinitely often tried. This is discussed in section IV.

2. If we take $S_w = \emptyset$ (and hence $D_w = \{1,\ldots,n\}$) for all $w \in W$, the rule reduces to the termination rule for the usual (non-fair) semantics of GC (see, e.g., [Harel]).

3. In proving clauses (1) - (4) of the rule, application of the usual rules (for assignment, etc.) is presupposed.

Examples

Ex. 1. First, consider Dijkstra's example for a random generator of natural numbers [Dijkstra]; this is a possibly non-terminating program, its only infinite computation sequence being unfair. Since this sequence is ruled out by our semantics, the program terminates fairly.

C :: x := 0; b := true;
*[b → x := x + 1
[] b → b := false

We prove <true> C <true>.

Choose as well-ordering {0,1} with 0 < 1, as $S_1 = \{1\}$, $D_1 = \{2\}$, and as ranking predicate $p(w)(x,b) \underset{DEF}{=} (w = 1 \supset b) \wedge (w = 0 \supset \neg b)$.

As to clause 1) : b changes from true to false upon move b := false, and hence w drops from 1 to 0.

As to clause 2) : b remains true under x := x + 1, and p(w) is independent of x, so w stays 1.

As to clause 3) : C_1 :: *[b ∧ ¬b → ...] obviously terminates.

Ex. 2. In ex. 1, a D-move is always enabled (; in the terminology of [PSL], that program is impartial). Next, consider a program, in which D-moves are only eventually enabled, and clause 3) is less trivially satisfied.

C :: b := true; c := true;
*[b → c := ¬c
[] b ∧ c → b := false
]

Again we prove <true> C <true>.

Choose W, p, S_1, D_1 as above. The difference lies in clause 3), with auxiliary program:

C_1' :: *[b ∧ ¬(b ∧ c) → c := ¬c], which terminates after one step at most.

This example is still trivial, but it should give the reader a feeling for the spirit of the rule, which captures eventual enabling of a D-move by means of a proof of termination of the auxiliary program.

Ex. 3. Next, we show that the natural numbers $\mathbb{N}$ are not sufficient for fair termination proofs, since there is no bound on the length of finite computations.

Let x, y, z range over $\mathbb{N}$.

C :: x := 0; y := 0;
*[x = 0 → y := y + 1 [] x = 0 → x := 1
[] x ≠ 0 ∧ y ≠ 0 → y := y - 1 [] x ≠ 0 ∧ y ≠ 0 → z := z + 1
].

To prove <true> C <true>, choose $W = \mathbb{N} \cup \infty$,

$p(w)(x, y, z) = (w = \infty \supset x = 0) \wedge (w \neq \infty \rightarrow x \neq 0 \wedge y = w)$,
$S_\infty = \{1,3,4\}$, $D_\infty = \{2\}$, $S_n = \{1,2,4\}$, $D_n = \{3\}$. For clause 3) we get as auxiliary programs:

C_∞ :: *[x = 0 ∧ x ≠ 0 → ...
[] x ≠ 0 ∧ y ≠ 0 → y := y - 1
[] x ≠ 0 ∧ y ≠ 0 → z := z + 1
].

C_n :: *[x = 0 → ... [] x = 0 → ... [] x ≠ 0 ∧ y ≠0 ∧¬(x ≠ 0 ∧ y ≠ 0) → ...].

To prove $\langle p(n) \wedge n \gtrneq 0 \rangle \xi C_n$ <true> is trivial since $p(n) \supset x \neq 0$, and hence C_n terminates immediately.
To prove <x = 0> C_∞ <true>, choose W' = $\mathbb{N}$, and let $S'_n = \{1,3\}$, $D_n = \{2\}$, $n \in \mathbb{N}$, and $p(n)(x, y, z) = y = n \wedge x \neq 0$.
Note that the alternatives are renumbered.
Clause 1) is satisfied since y := y - 1 decreases y, and clause 2) is satisfied since p(n) is independent of z. As to clause 3), we again construct an auxiliary program, $C_{\infty,n}$,

$C_{\infty,n}$ :: [x = 0 ∧ x ≠ 0 → ... [] x ≠ 0 ∧ y ≠ 0 ∧¬(x ≠ 0 ∧ y ≠ 0) → ...],

which trivially terminates.

Finally, consider the following program:

C :: y := 1; b := true;
*[b → y := y + 1
[] b ∧ prime(y) ∧ prime(y + 2) → b := false.
]

This program fairly terminates iff the conjecture that there exist infinitely many "twin" primes is true.

III. Soundness and semantic completeness

In this section we prove the soundness of the suggested proof rule w.r.t. the semantics of computation trees consisting of fairly terminating sequences, and its semantic completeness. We shall not deal in this paper with the specification language needed to express p(w) and the partitions, an issue dealt with elsewhere (by the fourth author).

a. Soundness

We have to prove that if all premises of the rule hold, so does its conclusion.

Assume that for program C we found a well-ordered set (W,<), a partition S_w, D_w for each w > 0 s.t. $D_w \neq \emptyset$, and a predicate p, satisfying clauses 1) - 4) of the rule.

Assume by way of contradiction that for some state ξ_0, T_{ξ_0} contains an infinite fair path $\langle \xi_i \rangle_{i=0}^{\infty}$. Consider the corresponding sequence of moves $\langle d_i \rangle_{i=0}^{\infty}$. It cannot contain an infinite subsequence $\langle d_{i_j} \rangle_{j=0}^{\infty}$ of D-moves, since by clause 1) this would imply the existence of an infinite decreasing sequence of elements in W, contradicting W's well-foundedness. Thus, from some k onwards $p(w, \xi_k)$ holds, and all moves d_j for $j > k$ are S_w-moves (by clause 2)). By clause 3) there is some $d \in D_w$ which is infinitely often enabled and not taken, contradicting the assumption that $\langle \xi_i \rangle_{i=0}^{\infty}$ is fair.

b. Completeness

This is the harder part. Assume <r> C <q> holds. Then we have to find a well-ordered set (W,<), partitions S_w, D_w for each w > 0 s.t. $D_w \neq \emptyset$, and a predicate p (given by a collection of pairs (w, ξ)) such that clauses 1) - 4) hold.

Since all we "have at hand" is the computation tree, we have to derive everything needed from that tree (compare also [de Roever] for another well-foundedness argument based on the "operational" object ~ the computation stack, for nondeterministic recursive procedures).

We are given that the computation tree T_{ξ_0}, for every state ξ_0 satisfying r, is either well-founded, or contains at least one infinite, hence unfair, computation sequence.

The basic idea is to construct another (possibly infinitely wide) tree $T_{\xi_0}^*$, some of whose nodes are obtained by collapsing certain infinite families of nodes in T_{ξ_0}, all lying on unfair sequences originating in nodes $\xi \in T_{\xi_0}$, such that $T_{\xi_0}^*$ is well-founded, i.e., contains finite path's only. Then we use a standard ranking of $T_{\xi_0}^*$ by means of ordinals. A move which leaves ξ and remains in the same infinite family belongs to S_w for the corresponding rank.

A move which exits such a family belongs to D_w. Special care must be taken that these partitions do not depend on ξ_0, the root of the computation tree.

We now present the details of the construction. Let T_{ξ_0} be given.

case a. T_{ξ_0} is well-founded (; this means that C always terminates in ξ_0). Choose a ranking of the nodes by means of an initial segment of the ordinals, ranking leaves by 0, and proceeding inductively level by level from leaves till root (a standard set-theoretical construction); furthermore, choose uniformly $S_w = \emptyset$, $D_w = \{1,\ldots,n\}$. It is easy to verify that clauses 1) - 4) of the rule hold.

case b. T_{ξ_0} contains at least one unfair, hence infinite, computation path π. This case is dealt with below.

Definition 1. A computation sequence π is d-unfair ($1 \leq d \leq n$) iff along π C_d was infinitely often enabled, but only finitely often chosen.

2. Let $\xi \in T_{\xi_0}$. Define ξ's d-cone $CONE_d(\xi)$ as follows:

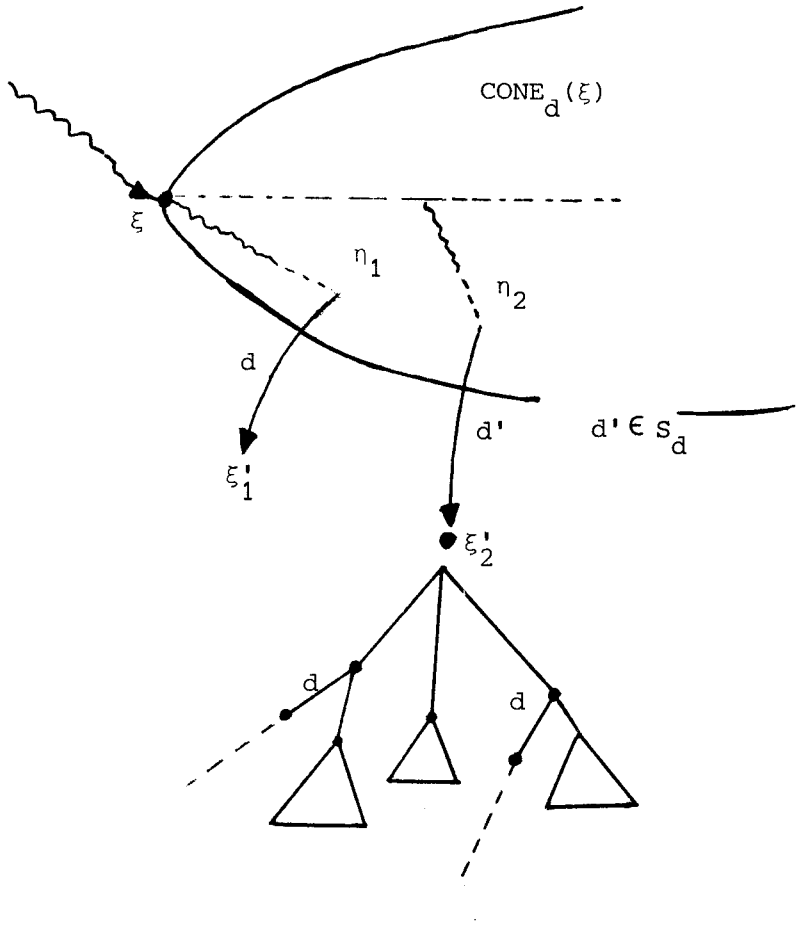
$CONE_d(\xi)$ = the set of all occurrences of states in T_{ξ_0} residing on infinite computation sequences which contain only finitely many d-moves and which start in ξ.

(Obviously, all occurrences of states on d-unfair sequences starting in ξ belong to its d-cone.)

Lemma 1. Let $\xi \in T_{\xi_0}$, and let $\eta \in CONE_d(\xi)$, for some $1 \leq d \leq n$. Then every computation sequence leaving $CONE_d(\xi)$, say at node η, is either finite or contains a d-move.

Proof. Suppose not. Then an infinite path π starts in η, leaves $CONE_d(\xi)$, and does not contain any d-move. Since $\eta \in CONE_d(\xi)$, there is some finite path π' joining ξ to η, along which a d-move was taken at most a finite number of times. Hence concatenation $\pi'!\pi$ of π' and π is contained in $CONE_d(\xi)$, contradicting the assumption that π leaves $CONE_d(\xi)$. Q.E.D.

The situation is described in the following figure, where a triangle denotes a well-founded tree:



Observation: If state ξ resides on a d-unfair sequence, then $CONE_d(\xi) \neq \emptyset$.

Our candidates for families "to be collapsed" into a node in $T^*_{\xi_0}$ are such d-cones.

Next we define inductively a hierarchy of d-cones.

Base step. Since by assumption T_{ξ_0} contains an unfair sequence, fix some $1 \leq d_0 \leq n$ s.t. there exists a d_0-unfair sequence in ξ_0, and let $CONE_{d_0}(\xi_0)$ be defined as above. It is not empty by the observation above. We say that $CONE_{d_0}(\xi)$ is at level 0.

Induction step. Suppose at level i - 1 a d-cone $CONE_d(\xi_{i-1})$ was defined, and let π be some path leaving $CONE_d(\xi_{i-1})$. By lemma 1 either π is finite, or there is a d-move on path π resulting in state ξ_i. If π is finite we finish the construction as far as π is concerned. So assuming state ξ_i as above, construct $CONE_{d'}(\xi_i)$ at level i, where d' is determined as follows:

If there is a move d' not appearing in $\xi_0 \cdots \xi_1 \cdots \xi_{i-1} \cdots \xi_i$, and there is an infinite sequence with a finite number of occurrences of d' starting in ξ_i, choose move d'.

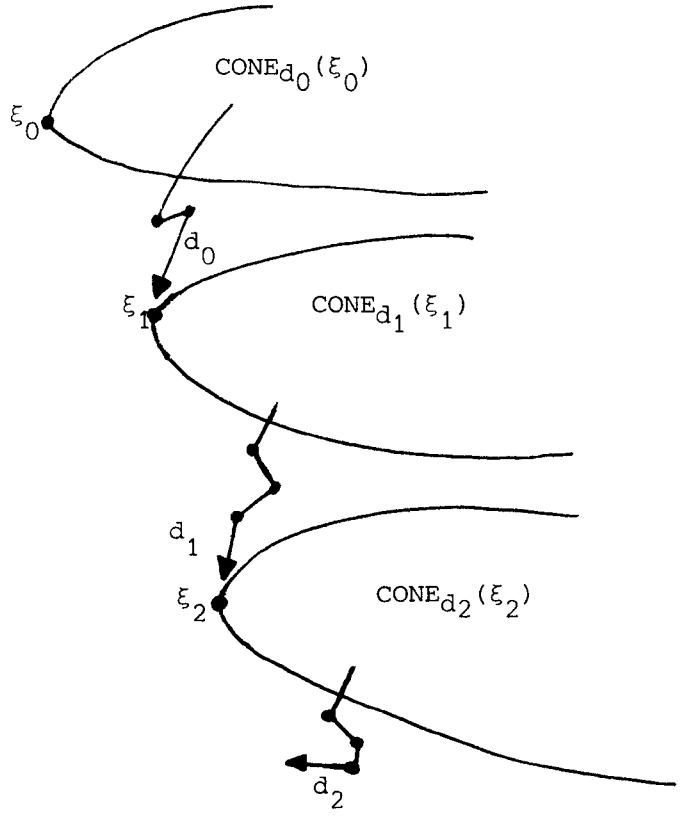
Otherwise, choose the index of the move which didn't appear longest in $\xi_0 \ldots \xi_i$ for which there is an infinite sequence containing finitely many occurrences of that move, starting in ξ_i.

□

Thus, when iterating the cone construction, we vary the move-indices of the cones maximally.

Lemma 2. There does not exist an infinite sequence of cones $\mathrm{CONE}_{d_i}(\xi_i)$ s.t. $\langle \xi_i \rangle_{i=0}^{\infty}$ is an infinite path of T_{ξ_0}.

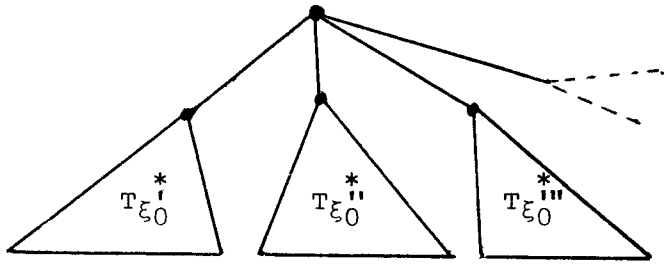
Remark. If we describe the construction of cones as in the following figure, we have by lemma 2 only finite chains of cones.



Proof. Suppose such an infinite sequence $\langle \xi_i \rangle$ exists. Then it is unfair by definition of T_{ξ_0}. Thus, there is some $1 \leq \bar{d} \leq n$ s.t. $\langle \xi_i \rangle$ is $\bar{d}$-unfair. Then there is an i_0 s.t. at ξ_{i_0} either $\bar{d}$ did not occur on $\xi_0 \ldots \xi_{i_0}$ or it occurred less recently than any other move. Hence $\bar{d} = d_{i_0}$ in the inductive construction of $\mathrm{CONE}_{d_{i_0}}(\xi_{i_0})$, and $\langle \xi_i \rangle$ would have been contained in $\mathrm{CONE}_{d_{i_0}}(\xi_{i_0})$, contrary to assumption. Q.E.D.

Now we define $T^*_{\xi_0}$ as suggested above. Its nodes are all the nodes in T_{ξ_0} not belonging to any cone, and the set of all cones. Its edges are either edges entering cones, or edges leaving cones, and, otherwise, edges outside cones. By lemma's 1 and 2, the tree $T^*_{\xi_0}$ is well-founded.

In order to get rid of unwanted ξ_0-dependence of S_w and D_w as suggested above, we do one move construction: combine all $T^*_{\xi_0}$ s.t. $r(\xi_0)$ holds into one infinitary well-founded tree T^*_C:



Next, rank the nodes of T^*_C. However, we must take care that if ξ occurs in two places in T^*_C with the same rank, it determines some (S, D) partition uniquely.

In order to achieve this we perform a rank-shift:
Suppose that at some level of the ranking, say λ, there are equiranked occurrences of a state ξ, say of order type α. Then rerank these consecutively by $\lambda + 1, \ldots, \lambda + \alpha$, and proceed to the next level $\lambda + \alpha + 1$.

Let ρ denote the ranking function of T^*_C. Then we define predicate p and partitions (S_w, D_w). As W we choose the ordinals ranking T^*_C, an initial segment of the countable ordinals.

$$p(w)\xi \underset{\text{DEF}}{=} \exists\eta,d.\xi \in \text{CONE}_d(\eta) \wedge \rho(\text{CONE}_d(\eta)) = w \vee \forall\eta,d.\xi \notin \text{CONE}_d(\eta) \wedge \rho(\xi) = w.$$

$$\text{For } w > 0 : S_w = \begin{cases} S_d, \text{ if } \exists\eta,d.\rho(\text{CONE}_d(\eta)) = w \\ \emptyset, \text{ otherwise.} \end{cases}$$

Note that the rank-shift of T^*_C assures that S_w is well defined.

Next we show that clauses 2) - 4) of the rule hold; and thereafter we refine the cone-construction so as to satisfy clause 1), too.

Lemma 3. W, p, (S_w, D_w) satisfy clause 2) - 4) of the rule. (As we shall see clause 1) need not hold.)

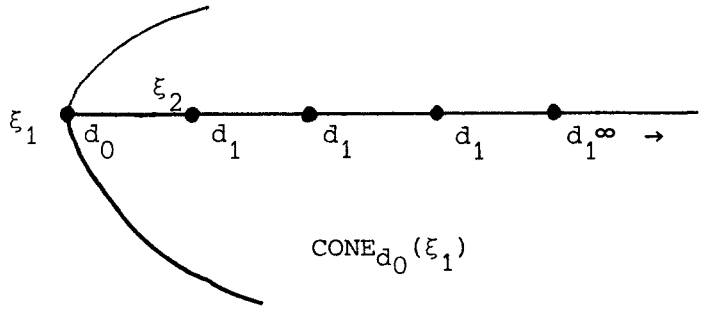
Proof.

Clause 2): Assume $p(w) \wedge w > 0 \wedge B_i$ holds in ξ, for $i \in S_w$. Without loss of generality (by the rank-shift), assume $\xi \in T_{\xi_0}$ and $r(\xi_0)$ holds. Then $\xi \in CONE_d(\eta)$ for some η and d (since, otherwise, $S_w = \emptyset$), and $d \neq i$. If move C_i remains in the cone, by construction the rank remains the same. Otherwise, it leaves the cone, and hence, since T_c^* is ranked from bottom-leaves to top-root, the rank decreases.

Clause 3): Assume again $p(w) \wedge w > 0$ holds in ξ. We have to demonstrate that C_w fairly terminates. Since $S_w \neq \emptyset \supset S_w = S_d$ for some d, the guards of C_w are $B_i \wedge \neg B_d$. Again, assume we are in T_{ξ_0} as above. Let π be a fair computation sequence of C_w starting in ξ. Then π can be extended in front to a fair computation sequence starting in ξ_0, and hence is finite. Thus C_w fairly terminates. (At this point it should be clear to the reader that the whole proof proceeds by induction on the number of alternatives of C, and on the complexity of these.)

Clause 4). By construction, in T_c^* holds $p(\xi) = 0 \leftrightarrow \xi$ is a leaf of T_c^*. Q.E.D.

To see that condition 1) does not hold, consider the case:



I.e., $d_0 d_1^\infty$ is a d_0-unfair computation sequence, contained in $CONE_{d_0}(\xi_1)$, and let $\rho(CONE_{d_0}(\xi_1)) = w$. Then $\rho(w)\xi_1 \wedge w > 0 \wedge B_0$ holds, and hence, $<p(w)\xi_1 \wedge w \not> 0 \wedge B_0 > C_0 < p(w)\xi_2>$, that is, w need not necessarily decrease under the C_0 move as indicated.

Finally, we modify our construction of cones so as to satisfy clause 1) of the rules too. This modification affects the collapsing of a d-cone; instead of collapsing such a cone to a node of $T_{\xi_0}^*$, we collapse it to a well-founded subtree of $T_{\xi_0}^*$.

Let $CONE_d(\xi)$ be given. Now repeat the inductive construction, but modified by defining subcones within $CONE_d(\xi)$ which include only infinite computation sequences containing no occurrences of d at all, and ξ itself (hence never being empty).

Definition. For $\eta \in CONE_d(\xi)$, let $S\text{-}CONE_d(\eta)$ = (the set of all occurrences of states along infinite paths in $CONE_d(\xi)$ starting in η and containing no occurrence of a d-move) $\cup$ $\{\eta\}$.

By an argument similar to the one in the proof of lemma 1 we establish:

Lemma 4. Every computation sequence leaving $S\text{-}CONE_d(\eta)$ is either finite or contains a d-move.

The inductive construction of subcones of $CONE_d(\xi)$ goes as follows: At level 0, define $S\text{-}CONE_d(\eta_0)$ with $\eta_0 = \xi$. Supposing $S\text{-}CONE_d(\eta_{i-1})$ to be defined at level i-1, let, by lemma 4, C_d be the first occurrence of a d-move along a computation sequence leaving $S\text{-}CONE_d(\eta_{i-1})$, or starting in η_{i-1}, in case $S\text{-}CONE_d(\eta_{i-1}) = \emptyset$, s.t. this computation sequence does not leave $CONE_d(\xi)$; this d-move results in $\eta' = \eta_i$. Then $S\text{-}CONE_d(\eta_i)$ at level i is defined.

□

Lemma 5. There does not exist an infinite chain of $S\text{-}CONE_d(\eta_i)$'s with $\eta_0 = \xi$.

Proof. Suppose such a chain exists. Then there exists an infinite computation sequence starting in ξ with an infinite number of occurrences of d-moves, contained in $CONE_d(\xi)$, contradicting the definition of $CONE_d(\xi)$. Q.E.D.

Thus, we now collapse each $CONE_d(\xi)$ into a well-founded sub-tree, with subcones $S\text{-}CONE_d(\eta)$ collapsed to nodes. By lemma 5 this subtree is well-founded, and hence, the whole tree $T^*_{\xi_0}$ is well-founded. Now repeat the previous ranking procedure to $T^*_{\xi_0}$ so obtained.

Now, clause 1) holds, too, because every d-move either leads to a lower ranked node corresponding to a subcone, or leaves the whole cone, therefore also leading to a lower ranked node. Satisfaction of the other clauses is not affected by the modification described above. Hence we established:

Theorem. If C fairly terminates, $(W,<)$, p, $<(S_w, D_w)>_{w\in W, w>0}$, exist satisfying all the clauses appearing as premises in our rule for proving fair termination of guarded loops.

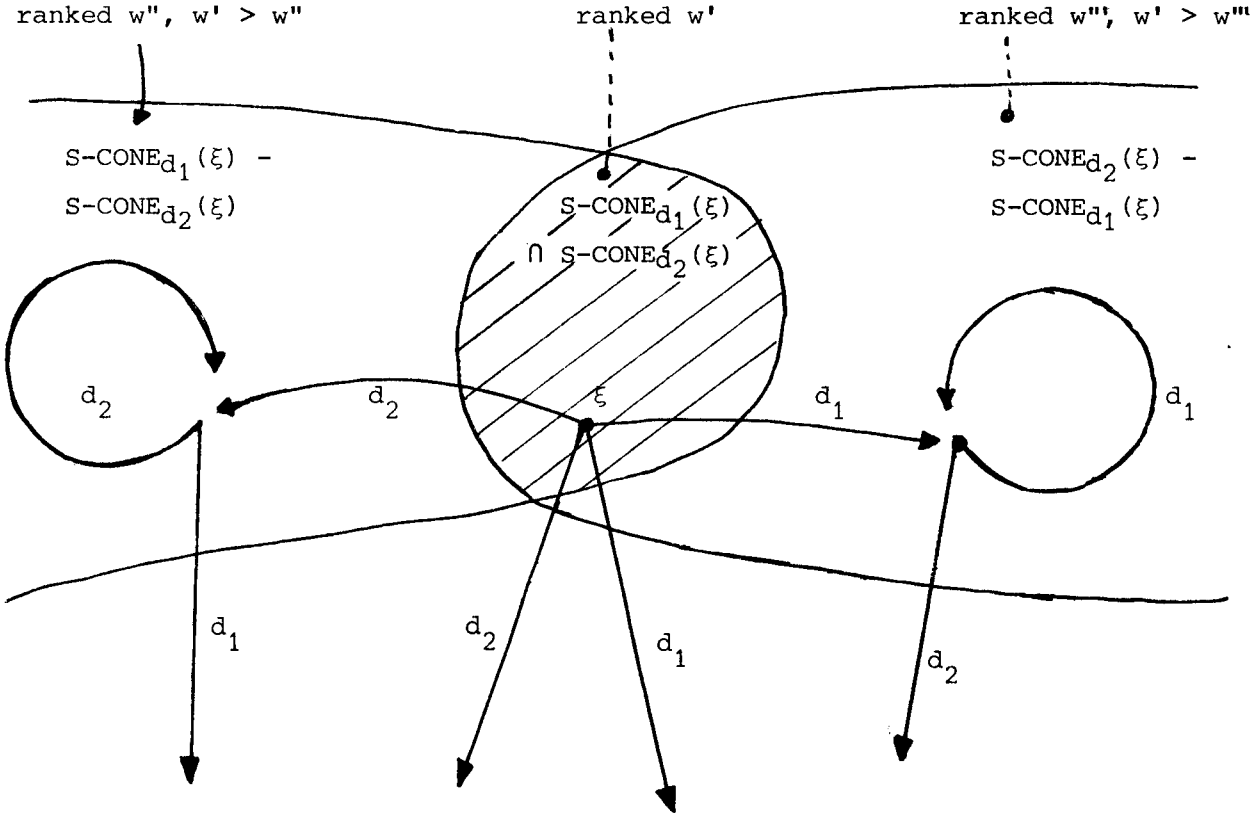
Comparing the construction in the completeness proof with the statement of the rule itself, one cannot help noticing that there is a certain discrepancy between the two. In the construction, we always end up with $|D_w = 1|$ for collapsed nodes, whereas the rule itself allows $|D_w| > 1$. We would like to give some semantic significance to the case $|D_w| > 1$ in the light of the previous construction.

Suppose in ξ there exist infinite computation sequences $\pi_1, \ldots, \pi_k$, not containing, respectively, moves $d_1, \ldots, d_k$ an infinite number of times. Then $\pi_i \in \bigcup_{i=1}^{k} \text{CONE}_{d_i}(\xi)$. Define $\text{CONE}_{\{d_1,\ldots,d_k\}}(\xi) = \bigcup_{i=1,..,k} \text{CONE}_{d_i}(\xi)$, where $\{d_1,\ldots,d_k\}$ is the <u>maximal</u> set of moves s.t. $\text{CONE}_{d_i}(\xi) \neq \emptyset$, $i = 1, \ldots, k$. Next, one verifies:

<u>Lemma 6</u>. Every infinite sequence leaving $\text{CONE}_{\{d_1,\ldots,d_k\}}(\xi)$ contains moves $d_1, \ldots, d_k$.

Then, one modifies the iterative cone construction in that a new (generalized) cone is constructed after all moves $d_1, \ldots, d_k$ occurred. Observe that the analogue of lemma 2. holds again.

Now, generalize the construction of subcones to maximal sets of moves. Assume $k = 2$, for simplicity of notation (the construction generalizes to $k = n$). In order to satisfy clause 1), we refine our ranking, as in the figure below.



Split $S\text{-}CONE_{\{d_1,d_2\}}(\xi)$ into three parts:
$S\text{-}CONE_{d_1}(\xi) - S\text{-}CONE_{d_2}(\xi)$, $S\text{-}CONE_{d_2}(\xi) - S\text{-}CONE_{d_1}(\xi)$,
and $S\text{-}CONE_{d_1}(\xi) \cap S\text{-}CONE_{d_2}(\xi)$, and rank them, respectively, w", w"', w' with w' > w", w' > w"'. (This can be easily accomplished by superposing a lexicographical order on ρ.) Choose $D_{w'} = \{d_1,d_2\}$, $D_{w"} = \{d_1\}$, $D_{w"'} = \{d_2\}$. Now clause 1) is satisfied (as suggested in the figure).

IV. Relation to other work

As already mentioned in the introduction, our work is closely related to [PSL].
In [PSL] three fairness-like notions are introduced:

1. Just execution: along infinite computation sequences all moves appear infinitely often (no reference to being enabled or not).
2. Impartial execution: along infinite computation sequences enabled moves, which once enabled, remain enabled until taken (i.e., are continuously enabled), are taken.
3. Fair execution: along infinite computation sequences, moves infinitely often enabled are eventually taken.

This distinction influenced clause 3) of our rule. Without clause 3), our rule is sound and complete for impartial execution. The difference between impartial termination and fair termination is reflected in examples 1 and 2 in section II.

Actually, omitting clause 3) suffices for termination in case of eventual impartiality, i.e. an enabled move may be disabled finitely often, but eventually, becomes continuously enabled and then it is eventually taken. For such a rule (without recursive application of clause 3)) the underlying assumption about the scheduler (i.e., the underlying semantics) is that each direction is infinitely often tried (in case of infinite computations). Therefore, if it is eventually continuously enabled, some try is bound to succeed.

A notable difference between our rule and the one in [PSL], called method F, is that we partition the moves in an ordinal-dependent way, whereas in [PSL] state predicates play a crucial rule in determining decreasing moves.

Now we show that our rule implies method F, and hence the semantic completeness of our rule implies the semantic completeness of method F. (We do not have a proof of the reverse implication.)

Assume that for program C we found W, p, $<(S_w, D_w)>_{w \in W}$ satisfying clauses 1) - 4) of our rule, relative to precondition r, and that $|D_w| = 1$.

In order to apply method F, we have to:

i. Find a partial ranking function ρ: States $\rightarrow$ W', where W' is ordered by a well-founded ordering, $<$.

ii. Find predicates Q_i, i = 1, ..., n, over states, where $Q = \bigvee_{i=1}^{n} Q_i$, satisfying:

0) $Q(\xi)$ implies $\rho(\xi)$ is defined,

1) $r(\xi) \supset Q(\xi)$,

2) $Q(\xi) \wedge \eta \in C_i(\xi) \supset (Q(\eta) \wedge \rho(\xi) \geq \rho(\eta))$,

3) $Q_i(\xi) \wedge \eta \in C_j(\xi) \wedge \rho(\xi) = \rho(\eta) \supset Q_i(\eta)$, for $i \neq j$,

4) $Q_i(\xi) \wedge \eta \in C_i(\xi) \supset (\rho(\xi) \gneq \rho(\eta))$ (, thus the Q_i determine the decreasing directions),

5) Program C' :: $*[\square_{j=1,\ldots,n} B_j \wedge \neg B_i \rightarrow C_j]$ satisfies $<Q_i>$ C' $<\underline{\text{true}}>$.

To satisfy method F, take W' = W, $<$ = $<$, and define $\rho(\xi) = \min_w p(w)\xi$, $Q_i(\xi) = i \in D_{\rho(\xi)}$. Hence $Q(\xi) \equiv \exists w.p(w)\xi$.

Next we verify conditions 0) - 5) of method F.

condition 0): $\exists w.p(w)\xi \supset \{w|p(w)\xi\} \neq \emptyset$, and the minimum of $\{w|p(w)\xi\}$ exists by a property of the ordinals.

condition 1): $r(\xi) \supset \exists w.p(w)\xi$ holds by clause 4).

condition 2): follows from clauses 1), 2) of our rule, guaranteeing that p(v) holds for $v \leq w$, hence the minimal v s.t. p(v) does not increase, either.

condition 3): $Q_i(\xi) \wedge \eta \in C_j(\xi)$, $i \neq j$, implies that an S-move is taken, and since $\rho(\xi)$ is the minimal w s.t. $p(w)\xi$, this S-move does not decrease the ordinal, hence $Q_i(\xi)$ still holds.

condition 4): follows directly from clause 1), since $\eta \in C_i(\xi)$ and $Q_i(\xi)$ imply a D-move is taken.

condition 5): reduces to clause 3).

Acknowledgements. Amir Pnueli and Samuel Katz are thanked for helpful discussions. Daniel Lehmann suggested the reduction of the [PSL] rule to ours'. The Netherlands Organization for the advancement of Pure Research (Z.W.O.), The Technion, and IBM Israel are thanked for their financial support, and the departments of computer science of the University of Utrecht and the Technion for their hospitality.

V. References.

[Apt & Plotkin]: Apt, K.R. and G. Plotkin, A Cook's tour of countable non-determinism, submitted to ICALP 1981.

[Dijkstra]: Dijkstra, E.W., A discipline of programming, Prentice Hall, 1976.

[Floyd]: Floyd, R.W., Assigning meaning to programs, in: J.T. Schwartz (ed.), Math. Aspects of Computer Science, Proc. Symp. in Apl. Math., AMS, Providence, R.I., 1967.

[Francez & Pnueli]: Francez, N. and A. Pnueli, A proof method for cyclic programs, Acta Informatica 9, 1978.

[Francez & de Roever]: Francez, N. and W.P. de Roever, Fairness in communicating processes, Univ. Of Utrecht, 1980.

[GPSS]: Gabbay, D, A. Pnueli, S. Shelah and Y. Stavi, On the temporal analysis of fairness, Proc. 7th POPL Conf., 1980.

[Harel]: Harel, D., First-order dynamic logic, Lecture Notes in Computer Science 68, Springer-Verlag, 1979.

[Hitchcock & Park]: Hitchcock, P. and D. Park, Induction rules and termination proofs, in M. Nivat (ed.), Automata, Languages and Programming, IRIA, North-Holland, 1973.

[Hoare]: Hoare, C.A.R., Communicating sequential processes, CACM 21, 8, 1978.

[Lamport]: Lamport, L., Proving the correctness of multiprocess programs, IEEE - TSE 3, 2, 1977.

[Lamport & Owicki]: Lamport, L. and S. Owicki, Proving liveness properties of concurrent programs, SRI - TR, 1980.

[Pnueli]: Pnueli, A., The temporal semantics of concurrent programs, TCS, 13, 1, 1981.

[PSL]: Pnueli, A., Y. Stavi and D. Lehmann, Proving termination of just concurrent programs, submitted to ICALP 1981.

[de Roever]: Roever, W.P. de, Dijkstra's predicate transformer, nondeterminism, recursion and termination, MFCS, 1976, Lecture Notes in Computer Science 45, Springer Verlag, 1976.