PROOF RULES DEALING WITH FAIRNESS

- Extended Abstract - *)

Krzysztof R. Apt
University of Rotterdam

Ernst-Rüdiger Olderog
University of Kiel

Abstract. We provide proof rules allowing to deal with two fairness assumptions in the context of Dijkstra's do-od programs. These proof rules are obtained by considering a translated version of the original program which uses random assignment x:=? and admits only fair runs. The proof rules use infinite ordinals and deal with the original programs and not their translated versions.

1. Introduction

One of the troublesome issues concerning non-deterministic and parallel programs is that of fairness. This assumption states roughly speaking that each possible continuation is scheduled for execution sufficiently often. The meaning of a continuation depends on the language considered. For example, in the case of Dijkstra's guarded commands a possible continuation is a branch guarded by a guard evaluating to true. "Sufficiently often" can be interpreted here in a variety of ways the simplest of them being "eventually".

The aim of this paper is to develop a simple proof theoretic approach to the issue of fairness. This approach was originally suggested in APT & PLOTKIN [1].

We restrict our attention to Dijkstra's do-od-programs whose components are simple while-programs. Each fairness assumption (we study here two of them) can be incorporated here by providing an appropriate equivalent version of the original program which uses the random assignment x:=? (set x to an arbitrary non-negative integer) for scheduling purposes

---

Author's addresses: K.R. Apt, Faculty of Economics, University of Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands; E.-R. Olderog, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, Olshausenstr. 40-60, D-2300 Kiel 1, West Germany.

*) The full version of this paper is available as Bericht Nr. 8104, Institut für Informatik und Praktische Mathematik, University of Kiel, March 1981, and has been submitted for publication.

and admits only fair computations. By applying to this version of progra Hoare-style proof-rules considered in APT & PLOTKIN [1] we arrive at proof rules dealing with fairness. It should be stressed that these proc rules deal with the original program - the applied transformations are "absorbed" into the assertions leaving the program in question intact. Using these proof rules total correctness of do-od-programs under the assumption of (weak and strong) fairness can be proved. The proof rules use infinite ordinals.

The use of such infinitistic methods seems to be needed in view of the results of EMERSON & CLARKE [3] who show that termination under fairness assumption is not first order definable. The results of APT & PLOTKIN [1] imply soundness and relative completeness of our system for a special type of assertion languages - those which allow the use of the least fixed point operator and ordinals.

2. Definitions

We consider programs of the form

$$S = \underline{\text{do}}\ B_1 \rightarrow S_1 \,[]\, \ldots \,[]\, B_n \rightarrow S_n\ \underline{\text{od}}$$

where the B_i are quantifier-free formulas and the S_i are deterministic while-programs. We have a simple model of state in mind, viz. $\sigma : \mathcal{V}ar \rightarrow \mathcal{D}$ $\mathcal{V}ar$ denotes here the set of program variables and $\mathcal{D}$ is a domain of an interpretation. Thus the meaning of a subprogram S_i is a partially defin mapping $\mathcal{M}(S_i)$ from states to states. To state the notions of fairness and total correctness properly we employ so-called computation sequences of S defined as follows: For $i \in \{1,\ldots,n\}$ and states σ, σ' we write

$\sigma \xrightarrow{i} \sigma'$ iff $\models B_i(\sigma)$ and $\mathcal{M}(S_i)(\sigma) = \sigma'$ and

$\sigma \xrightarrow{i}$ iff $\models B_i(\sigma)$ and $\mathcal{M}(S_i)(\sigma)$ is undefined.

Computation sequences of S are now exactly those sequences ξ of states which fall into one of the following cases:

1° $\xi = \sigma_1 \xrightarrow{i_1} \sigma_2 \xrightarrow{i_2} \ldots \xrightarrow{i_{m-1}} \sigma_m$

where $i_j \in \{1,\ldots,n\}$ and $\models (\neg B_1 \wedge \ldots \wedge \neg B_n)(\sigma_m)$.

Then ξ is said to properly terminate.

2° $\xi = \sigma_1 \xrightarrow{i_1} \sigma_2 \xrightarrow{i_2} \ldots \xrightarrow{i_{m-1}}$

where $i_j \in \{1,\ldots,n\}$.

Then ξ is said to fail.

3° $\xi = \sigma_1 \xrightarrow{i_1} \sigma_2 \xrightarrow{i_2} \ldots \xrightarrow{i_{j-1}} \sigma_j \xrightarrow{i_j} \ldots$

where $i_j \in \{1,\ldots,n\}$ and ξ is infinite.

Then ξ is said to <u>diverge</u>.

(Note that all finite sequences ξ of maximal length must fall into the cases 1° or 2° because $B_i(\sigma)$ is always defined, i.e. $\models B_i(\sigma)$ or $\models \neg B_i(\sigma)$ holds.)

A computation sequence ξ of S is said to be <u>weakly fair</u> iff ξ is either finite (i.e. properly terminates or fails) or infinite, i.e. of the form

$$\xi = \sigma_1 \xrightarrow{i_1} \sigma_2 \xrightarrow{i_2} \ldots \sigma_j \xrightarrow{i_j} \ldots$$

with $i_j \in \{1,\ldots,n\}$, but then fulfils the following condition

$$\forall i \in \{1,\ldots,n\}\ ((\overset{\infty}{\forall} j \in \mathbb{N} \models B_i(\sigma_j)) \rightarrow (\overset{\infty}{\exists} j \in \mathbb{N}\ i_j = i))\ ^{*})$$

i.e. if B_i is almost always true then the i-th component is infinitely often chosen. In other words we explicitly <u>disallow</u> infinite sequences ξ with

$$\exists i \in \{1,\ldots,n\}\ (\overset{\infty}{\forall} j \in \mathbb{N} \models B_i(\sigma_j) \wedge \overset{\infty}{\forall} j \in \mathbb{N}\ i_j \neq i)\ .$$

A computation sequence ξ of S is said to be <u>strongly fair</u> iff ξ is either finite or infinite, i.e. of the form

$$\xi = \sigma_1 \xrightarrow{i_1} \sigma_2 \xrightarrow{i_2} \ldots \sigma_j \xrightarrow{i_j} \ldots$$

with $i_j \in \{1,\ldots,n\}$, but then fulfils the following condition

$$\forall i \in \{1,\ldots,n\}\ ((\overset{\infty}{\exists} j \in \mathbb{N} \models B_i(\sigma_j)) \rightarrow (\overset{\infty}{\exists} j \in \mathbb{N}\ i_j = i))$$

i.e. if B_i is infinitely often true then the i-th component is infinitely often chosen. In other words we explicitly <u>disallow</u> infinite sequences ξ with

$$\exists i \in \{1,\ldots,n\}\ (\overset{\infty}{\exists} j \in \mathbb{N} \models B_i(\sigma_j) \wedge \overset{\infty}{\forall} j \in \mathbb{N}\ i_j \neq i)\ .$$

Now we can state precisely what we understand by total correctness of programs with or without fairness assumptions. For arbitrary first order formulas P and Q we define:

*) The quantifier $\overset{\infty}{\forall}$ means "for all, but finitely may" and $\overset{\infty}{\exists}$ means "there exist infinitely many".

$\models \{P\}\ S\ \{Q\}$ [under weak (strong) fairness assumption]

iff every [weakly (strongly) fair] computation sequence of S starting in a state σ with $\models P(\sigma)$ is properly terminating, i.e. is of the form

$\sigma \xrightarrow{i_1} \ldots \xrightarrow{i_m} \sigma'$, and σ' fulfils $\models Q(\sigma')$.

Thus under fairness assumption we need not bother about unfair computation sequences.

3. The Transformations

Let

$S = \underline{do}\ B_1 \to S_1 \ [] \ \ldots \ [] \ B_n \to S_n \ \underline{od}.$

We consider the weak fairness assumption first. We use the following transformation

```
T*weak(S) = if B1 → turn:=1 [] ... [] Bn → turn:=n [] ¬(B1 ∨ ... ∨ Bn)→skip fi;
            for i:= 1 to n do z[i] :=? od;
            do
             ...
            [] Bi ∧ turn=i →
                 Si;
                 if ¬Bi ∨ mini ≤ 0 then turn:=indexi;
                                        z[turn] :=? fi;
                 for j:=1 to n do
                   if j≠turn then if Bj then z[j]:=z[j] -1
                                        else z[j]:=?      fi fi
                 od
             ...
            od
```

where i ranges from 1 to n.

The random assignment z[i]:=? means "Set z[i] to an arbitrary non-negative integer". min_i and $index_i$ are shorthands defined as follows

$$min_i = \begin{cases} \min\{z[j] \mid j \neq i \wedge B_j\} & \text{if } \bigvee_{j \neq i} B_j \\ +\infty & \text{otherwise} \end{cases}$$

$$index_i = \begin{cases} \min\{j \mid j \neq i \wedge B_j \wedge z[j] = min_i\} & \text{if } \bigvee_{j \neq i} B_j \\ i & \text{otherwise} \end{cases}$$

We require that turn and z [1],...,z[n] are variables of sort $\{1,\ldots,n\}$ resp. integer which do not occur in S.

Some informal explanations may help to understand why T^*_{weak} (S) admits only weakly fair runs: If z [j]> 0 holds, z [j] is equal to the number of times B_j is still to be continuously true before control is switched to S_j (via turn:=j). Since more than one z [j] can become 0 at the same time, we allow z [j] to be negative. Each time when B_j is true but S_j is not executed z [j] gets decreased by 1. If B_j is not true, z [j]is reset to an arbitrary non-negative value. turn=i means that the i-th component has the control. Control is transferred to another component if $\neg B_i$ holds or there exists a j≠i with B_j and z [j]≤0. In both cases control gets transferred to the least component j with the minimal z [j] for which at this moment B_j holds. If no such j exists, then all guards $B_1,\ldots,B_n$ are false and the program terminates.

We now pass to the issue of <u>strong fairness</u>. As basis for our proof rule we take the following transformation

T^*_{strong}(S) =

```
if B1→turn:=1[]...[]Bn→turn:=n[]¬(B1∨...∨Bn)→skip fi;
for i:= 1 to n do z [ i]:=? od;
do
 ...
[] Bi∧ turn=i →
    Si; do Bi ∧ ¬ ⋁(j≠i) Bj→Si od;
    if ¬Bi∨ mini ≤ 0 then turn:=indexi;
                          z[turn]:=? fi;
    for j:=1 to n do
      if j≠turn∧Bj then z[j]:= z [ j]-1 fi
    od
 ...
od
```

where i ranges from 1 to n.

This transformation is very similar to the one used for the case of weak fairness. The main difference is that the value of z [j] is not reset in the case when B_j is false.

The following lemma relates T^*_{weak}(S) and T^*_{strong}(S) to S and is of independent interest.

Lemma 1 (a) If ξ is a weakly (strongly) fair computation sequence of S then an element-wise extension ξ' of ξ is a computation sequence of $T^*_{weak}(S)$ $(T^*_{strong}(S))$.

(b) If ξ' is a computation sequence of $T^*_{weak}(S)$ $(T^*_{strong}(S))$ then its element-wise restriction ξ to the variables of S is a weakly (strongly) fair computation sequence of S.

Intuitively this lemma states that $T^*_{weak}(S)(T^*_{strong}(S))$ admits exactly all weakly (strongly) fair computation sequences of S.

4. The Proof Rules

The transformed programs use random assignments. In APT & PLOTKIN [1] proof rules have been developed which allow to prove total correctness of such programs. The relevant proof rule is the following

$$\frac{\begin{array}{l} P_0 \longrightarrow \exists \alpha\, p(\alpha) \\ p(0) \longrightarrow \neg(B_1 \vee \ldots \vee B_n) \wedge Q_0 \\ p(\alpha) \wedge \alpha > 0 \longrightarrow B_1 \vee \ldots \vee B_n \\ \{p(\alpha) \wedge \alpha > 0 \wedge B_i\}\ S_i\ \{\exists \beta < \alpha\ p(\beta)\} \quad i=1,\ldots,n \end{array}}{\{P_0\}\ \underline{\text{do}}\ B_1 \rightarrow S_1 \,[]\, \ldots \,[]\, B_n \rightarrow S_n\ \underline{\text{od}}\ \{Q_0\}}$$

where S_i are while-programs allowing random assignments and where α and β are ordinals.

Below we use the following notation

$\bar{z} = (z[1],\ldots,z[n])$, $\underset{\sim}{\bar{z}} = (\underset{\sim}{z}[1], \ldots, \underset{\sim}{z}[n])$

$\bar{z}_{-i}{}^1 = (y_1, \ldots, y_n)$

$$\text{where } y_j = \begin{cases} z[j]^{-1} & \text{if } j \neq i \text{ and } B_j \\ z[j] & \text{otherwise} \end{cases}$$

$[\bar{z} \cdot \underset{\sim}{\bar{z}}] = (y_1,\ldots,y_n)$

$$\text{where } y_j = \begin{cases} z[j] & \text{if } B_j \\ \underset{\sim}{z}[j] & \text{otherwise} \end{cases}$$

Applying the above proof rule to $T^*_{weak}(S)$ we derive the following

Proof rule for weak fairness ($n \geq 2$)

1^0 $P_0 \wedge \bigvee_i (B_i \rightarrow turn=i) \rightarrow \forall \bar{z} \geq 0 \; \exists \alpha \; p(\alpha, turn, \bar{z})$

2^0 $p(0, turn, \bar{z}) \rightarrow \bigwedge_i \neg (B_i \wedge turn=i) \wedge Q_0$

3^0 $p(\alpha, turn, \bar{z}) \wedge \alpha > 0 \rightarrow \bigvee_i (B_i \wedge turn=i)$

$4i^0$ $(i=1,\ldots,n)$

$\{ p(\alpha, i, \bar{z}) \wedge \alpha > 0 \wedge B_i \}$

S_i

$\{ ((\neg B_i \vee min_i \leq 0) \wedge index_i = k \longrightarrow$

$\forall \underset{\sim}{z} \geq 0 \; \forall \bar{z}[k] \geq 0 \; \exists \beta < \alpha \; p(\beta, k, [\bar{z} -_k 1 \cdot \underset{\sim}{z}]))$

$(B_i \wedge min_i > 0 \longrightarrow \forall \underset{\sim}{z} \geq 0 \; p(\alpha, i, [\bar{z} -_i 1 \cdot \underset{\sim}{z}]))$

$\{P_0\}$ do $B_1 \rightarrow S_1 \; [] \; \ldots \; [] \; B_n \rightarrow S_n$ od $\{Q_0\}$

under weak fairness assumption

where α, β are ordinals and $turn, z[1], \ldots, z[n]$ variables not occurring freely in P_0, Q_0 or $B_1, \ldots, B_n, S_1, \ldots, S_n$.

An analogous rule for strong fairness can be derived from $T^*_{strong}(S)$. These proof rules can be considerably simplified in the case of two guards.

Corollary (to Lemma 1). For all formulas P and Q without free variables turn, $z[1], \ldots, z[n]$ the following holds

$\models \{P\} \; S \; \{Q\}$ under weak fairness assumption

iff $\models \{P\} \; T^*_{weak}(S) \; \{Q\}$ (under no assumption)

and similarly for the strong fairness assumption.

5. An Example

We provide here a natural example for fairness suggested by P. Cousot. Let L be a complete lattice (with ordering $\sqsubseteq$ and least element $\perp$) which fulfils the finite chain property, i.e. every strictly increasing $\sqsubseteq$-chain in L is finite. Further on, let $f_i : L^n \rightarrow L$ be monotone for $i=1,\ldots,n$ and $A, \bar{x}, \bar{\perp}, lfp(f_1, \ldots, f_n)$ be the following abbreviations:

$A = x_1 \neq f_1(\bar{x}) \vee \ldots \vee x_n \neq f_n(\bar{x})$.

$\bar{x} = (x_1,\ldots,x_n)$ and $\bar{\perp} = (\perp,\ldots,\perp)$ (n times).

$lfp(f_1,\ldots,f_n)$ denotes the least fixed point of the monotone operator $(f_1,\ldots,f_n): L^n \to L^n$.

Then using our proof rules we can prove

$\models \{\bar{x}=\bar{\perp}\}$

do $A \to x_1 := f_1(\bar{x}) \,[]\, \ldots \,[]\, A \to x_n := f_n(\bar{x})$ od

$\{\bar{x}=lfp(f_1,\ldots,f_n)\}$

under weak (resp. strong) fairness assumption

where $\models$ refers to validity in L. (This correctness result is a special case of a more general theorem proved in COUSOT[2] .) The full proof can be found in the full version of the paper.

References

[1] APT, K.R. & G.D. PLOTKIN, A Cook's Tour of countable nondeterminism, Technical Report, Department of Computer Science, University of Edinburgh, 1980 (to appear in Proc. ICALP 81).

[2] COUSOT, P., Asynchronous iterative methods for solving a fixed point system of monotone equations in a complete lattice, Rapport de Recherche No 88, L.A.7, Université Scientifique et Medicale de Grenoble, 1977.

[3] EMERSON, E.A. & E.M. CLARKE, Characterizing correctness properties of parallel programs using fixpoints, in: Proc. ICALP 80, Lecture Notes in Computer Science 85, Springer Verlag, pp. 169-181, 1980.