MATHEMATICAL PROBLEMS IN COMPUTATION THEORY
BANACH CENTER PUBLICATIONS, VOLUME 21
PWN – POLISH SCIENTIFIC PUBLISHERS
WARSAW 1988

COUNTABLE NONDETERMINISM AND LOGICS OF PROGRAMS: FAIR-WELLFOUNDEDNESS OF WHILE-PROGRAMS WITH NONDETERMINISTIC ASSIGNMENTS IN THE LOGIC ALNA

MARISA VENTURINI ZILLI

Instituto per le Applicazioni del Calcolo, Roma, Italia

First a semantical analysis of fair-wellfoundedness of while-programs with nondeterministic assignments leading to a simple order-theoretic characterization of that notion is performed. Then fair-wellfoundedness is expressed in the language of ALNA (Algorithmic Logic for while-programs with Nondeterministic Assignments).

Introduction

Expressing total correctness of a fixed family of programs in a given language requires expressing wellfoundedness for the same family of programs. It is a well-known result (see, for instance, [3] and [4]) that the wellfoundedness of a predicate cannot be expressed in a language of type $L_{\omega_1\omega}$ while it can be expressed in one of type $L_{\omega_1\omega_1}$ (for instance as $\sim \exists x_1 x_2 \ldots (\bigwedge_i P(x_i, x_{i+1}))$ with P a binary predicate).

Concerning logics for while-programs with nondeterministic assignments, Apt and Plotkin [2] can express wellfoundedness in the framework of Hoare-logics by using a language based on the μ-calculus of Park with ordinals (in fact, they allow two first order sorts, data and ordinals, and second order variables, of arbitrary arity and sort, not quantified over but such that they can be bound by the least fixed point operator).

Apt and Olderog [1] deal with fair-wellfoundedness by adjoining to a **do-od** program P a set of instructions (including the random assignment $x := ?$) by which a fair-scheduling function is computed, i.e., a function which

chooses fair computations only from any fair-wellfounded computation tree of P. Moreover, they formulate a Hoare-logic for **do-od** programs for reasoning about fairness properties for those programs.

Gabbay, Pnueli, Shelah and Stavi [5] extend the language of temporal logic by adding the modality **until** specifically to deal with fairness properties.

Mascari and Venturini Zilli [7] expressed the wellfoundedness of while-programs with nondeterministic assignments by a language of type $L_{\omega_1\omega}$ plus modalities $\Diamond$ and $\Box$ of Algorithmic Logic, and defined a logic ALNA which is an extension of NAL in Mirkowska [8].

The motivations to consider fair-wellfoundedness inside ALNA are the following:

(i) The first one, common to all authors who deal with fairness problems for nondeterministic programs, is, in particular, that fair-wellfounded computation trees allow to overlook some infinite computations in case they turn out to be unfair with respect to termination.

(ii) The second one is having the possibility of replacing in ALNA, under fairness assumption, the wellfoundedness formula from [7] by a simpler formula, as that of fair-wellfoundedness in this paper. Moreover, since the axiom relative to $\mathrm{Wh}_{\gamma,p}$ of ALNA in [7] is based on the wellfoundedness formula, it can, under fairness assumption, be replaced by a corresponding axiom based on the fair-wellfoundedness formula, so obtaining a logic for fair programs that can be proved to be sound and complete in a way analogous to that used for ALNA.

The notion of fairness that we assume for computation trees is essentially fair reachability of nodes of the tree having a given property, as in Queille and Sifakis [10]. So, in the case of fair-wellfoundedness, fair reachability of terminal nodes is assumed in this paper (see Definition 1 below), notion that can be characterized in a natural way by cofinality in preordered multisets. Hence our aim is to characterize fairness properties in computation trees rather than define fair scheduling functions on computation trees.

Finally, it is worth mentioning that fairness issues in the framework of parallel programs (and hence the relevant literature) are not explicitly considered here. Of course, the parallelism that can be simulated as interleaving inside computation trees obtainable via transition systems is considered, in an indirect way.

This paper is organized as follows: Section 1 gives the syntax and semantics of while-programs with nondeterministic assignments. Section 2 considers wellfoundedness as in [7], fair-wellfoundedness and their characterization. Section 3 gives the wellfoundedness formula as in [7] and considers the fair-wellfoundedness formula in the language of ALNA.

1. While-programs with nondeterministic assignments

1.1. Syntax. Let $P(\Sigma)$ be the set of while-programs with nondeterministic assignments (see [7]) as the least set of programs, on the signature Σ including equality, containing as atomic constructs

(1) $\lceil\ \rfloor$, the *empty program*;

(2) skip, the *identity program*;

(3) $p := \gamma$, *propositional assignment*, where p is a propositional variable in $\mathrm{Var}_p \subset \Phi_0(\Sigma)$, and γ is a formula in $\Phi_0(\Sigma)$, the set of open classical formulas, on Σ, including **true** and **false**;

(4) $x := y \cdot \delta(y)$, *nondeterministic assignment*, where x, y are individual variables in $\mathrm{Var}_{\mathrm{ind}} \subset T_0(\Sigma)$, and $\delta(y)$ is a formula in $\Phi_0(\Sigma)$, to be read as "an arbitrary y such that $\delta(y)$, in x";

and closed under the following program connectives, for P, P_1, P_2 in $P(\Sigma)$ and for every γ in $\Phi_0(\Sigma)$:

(I) $P_1; P_2$, *sequential composition* of P_1 and P_2;

(II) **if** γ **then** P_1 **else** P_2 **fi** abbreviated as $\mathrm{If}_{\gamma, P_1, P_2}$, *conditional composition* of P_1 and P_2 or *branching*;

(III) **while** γ **do** P **od** abbreviated as $\mathrm{Wh}_{\gamma \cdot P}$, *iterative composition* of P or *iteration*.

Notice that $x := t$, with t in $T_0(\Sigma)$, can be used as an abbreviation for $x := y \cdot y = t$, and $x := ?$ as an abbreviation for $x := y \cdot y = y$. Moreover, P^k occurs in the sequel as an abbreviation for $P; P; \ldots; P$, k times.

1.2. Semantics. Let $L_0(\Sigma)$ be a language of nondeterministic while-programs in $P(\Sigma)$, A a structure of signature Σ, B_0 the two-element Boolean algebra with unit element tt and $ff = \neg tt$. A *realization* of $L_0(\Sigma)$ in A and B_0 is a mapping which assigns to every element of Σ an element of A or B_0 and, moreover, assigns, via the usual notion of *valuation* v of $\mathrm{Var}_{\mathrm{ind}}$ in A and of Var_p in B_0, to every t in $T_0(\Sigma)$ and every γ in $\Phi_0(\Sigma)$ a mapping $[\![t]\!]$: $\mathrm{Val}(A) \to A$, $[\![\gamma]\!]$: $\mathrm{Val}(A, B_0) \to B_0$ respectively, defined in the usual way.

Let $\mathrm{States}(A)$ be the set of all states, i.e., of pairs (A, v), or simply v when A is supposed to be fixed. A realization of P in $P(\Sigma)$ is defined by specifying a transition system (Conf, $\to$), where $\mathrm{Conf} = P(\Sigma) \times \mathrm{States}(A)$ is the set of all *configurations*, with c in Conf, and $\to \subseteq \mathrm{Conf} \times \mathrm{Conf}$ is the transition relation defined as the least binary relation in Conf such that, for a fixed A and with $v \models \gamma$ instead of $[\![\gamma]\!](v) = tt$,

$$\langle \mathbf{skip}, v\rangle \to \langle \lceil\ \rfloor, v\rangle,$$

$$\langle p := \gamma, v\rangle \to \langle \lceil\ \rfloor, v[[\![\gamma]\!](v)/p]\rangle,$$

$\langle x := y \cdot \delta(y), v\rangle \to \langle \lceil\ \rfloor, v[a/x]\rangle$ for any a in A such that $v[a/y] \models \delta(y)$,

$$\langle P_1; P_2, v\rangle \to \begin{cases} \langle P_2, v\rangle & \text{if} \quad P_1 \equiv \lceil\ \rfloor, \\ \langle P_1'; P_2, v\rangle & \text{if} \quad \langle P_1, v\rangle \to \langle P_1', v'\rangle, \end{cases}$$

$$\langle \mathrm{If}_{\gamma P_1 . P_2}, v\rangle \to \begin{cases} \langle P_1, v\rangle & \text{if} \quad v \models \gamma, \\ \langle P_2, v\rangle & \text{if} \quad v \models \sim\gamma. \end{cases}$$

$$\langle \mathrm{Wh}_{\gamma . P}, v\rangle \to \begin{cases} \langle \lceil\ \rfloor, v\rangle & \text{if} \quad v \models \sim\gamma, \\ \langle P; \mathrm{Wh}_{\gamma . P}, v\rangle & \text{if} \quad v \models \gamma. \end{cases}$$

Let $[\![P]\!] \subseteq \mathrm{States}^n(A)$, $n \geqslant 2$, be an n-ary relation realized by P via a transition system (Conf, $\to$). For $n = 2$, P is the input-output relation realized by P.

Let $\twoheadrightarrow$ be the transitive closure of $\to$ and $c_0\downarrow$ the multiset of all c such that $c_0 \twoheadrightarrow c$. A *computation tree* $T(c_0)$ or $T(P, v)$ is every transition tree with c_0, i.e., $\langle P, v\rangle$, as label of the root, $\to$ as directed arc, and any other node labelled by c iff c is in $c_0 \twoheadrightarrow c$. A singleton tree consists of one labelled node only and it is easy to see that it is such iff $c_0 \equiv \langle x := y \cdot \delta(y); P, v\rangle$ with P possibly missing and some v and $\delta(y)$ such that $v[a/y] \models \delta(y)$ for no a in A. $T(c)$ is a *proper subtree* of $T(c_0)$, $T(c) < T(c_0)$, iff it is a computation tree with root c such that $c_0 \twoheadrightarrow c$ and with any other node c' in $c\downarrow$.

No distinction is made in the sequel between a node and its label.

Any *terminal configuration* c in $T(c_0)$, i.e., a configuration such that $c \to c'$ for no c', is either *successful* (and then is of the form $\langle \lceil\ \rfloor, v'\rangle$ for some v') or *failing* (and then is of the form $\langle P', v'\rangle$ for some v' and $P \not\equiv \lceil\ \rfloor$). It is easy to check from the definitions that any failing node is of the form $\langle x := y \cdot \delta(y); P, v\rangle$ with P possibly missing.

A *computation* of $T(c_0)$ is every $\to$-maximal path in it. The empty computation consists of a single terminal node only.

Any finite computation of $T(c_0)$ can be either successful or failing according to its terminal configuration. The existence of some failing computation in a computation tree can be expressed in the language of ALNA, as shown in [7]. In every successful computation, $\bar{v}$ in $\langle \lceil\ \rfloor, \bar{v}\rangle$ is a (finite) *result* of P from v in $T(c_0) \equiv T(P, v)$. For a characterization of such results for every while-program see [7], where a characterization from [8] is extended to the unbounded (countable) nondeterminism.

A computation tree is *finite branching* iff every node has at most finitely many arcs, *infinite branching* iff it has at least one node from which countably many arcs start. It is easy to see that any node of $T(c_0)$ from which $k > 2$ arcs do start is of the kind $\langle x := y \cdot \delta(y); P, v\rangle$, with P possibly missing and v, $\delta(y)$ such that for k elements a in A $(k > 2)$, $v[a/y] \models \delta(y)$.

Finite (and infinite) branching can be expressed in the language of ALNA, as is shown in [7].

By using computation graphs instead of trees some distinctions among infinite computations can easily be made, as in [12].

2. Wellfoundedness and fair-wellfoundedness

The notion of wellfoundedness of a computation tree is preliminary to that of total correctness and its negation ensures that there are some infinite computations in the tree.

For all v, γ, $\delta(y)$, P_1, P_2, $T(\text{skip}, v)$, $T(p := \gamma, v)$, $T(x := y \cdot \delta(y), v)$ are wellfounded; $T(P_1; P_2, v)$ is wellfounded iff $T(P_1, v)$ is and, for every $\bar{v}$ in $[\![P]\!](v)$, $T(P_2, \bar{v})$ is wellfounded; $T(\text{If}_{\gamma, P_1, P_2}, v)$ is wellfounded iff either $v \models \gamma$ and $T(P_1, v)$ is wellfounded or $v \models \sim\gamma$ and $T(P_2, v)$ is wellfounded. Following [7], to deal with the wellfoundedness of $T(\text{Wh}_{\gamma,P}, v)$ the notions of If-approximation and k-If-approximation are used.

DEFINITION OF IF-APPROXIMATION. For all γ, P, v in an arbitrary countable semantical structure, $T(\text{Wh}_{\gamma,P}, v)$ is *If-approximable* in case (i) or (ii) holds.

(i) There exists $k < \omega$ such that in every computation of $T(\text{Wh}_{\gamma,P}, v)$ the number of iterations of P is less than or equal to k, so that $[\![\text{Wh}_{\gamma,P}]\!](v) = [\![\text{If}^{k}_{\gamma,P,\text{skip}}]\!](v)$. $T(\text{Wh}_{\gamma,P}, v)$ is then said to be *If-equivalent*.

(ii) There exists a *level* $\mathfrak{F}$ of $T(\text{Wh}_{\gamma,P}, v)$ (i.e., a family of proper subtrees of it whose roots are all reachable by the same number of $\rightarrow$ from c_0) such that every T' in $\mathfrak{F}$ is If-approximable ($\mathfrak{F}$ is then said to be If-approximable).

DEFINITION OF k-IF-APPROXIMATION, AND OF (l, k)-IF-APPROXIMATION. For all γ, P, v in an arbitrary countable semantical structure, $T = T(\text{Wh}_{\gamma,P}, v)$ is *k-If-approximable*, $k > 0$, iff there exists $l > 0$ such that T is (l, k)-If-approximable, where

(a) T is *$(0, 0)$-If-approximable* iff it is If-equivalent;

(b) T is *$(l, 0)$-If-approximable* iff for every $(0, 0)$-If-approximable $\bar{T} < T$ such that no T' with $\bar{T} < T'$ is $(0, 0)$-If-approximable,

$$\bar{T} \equiv T_0 < T_1 < \ldots < T_{l-1} < T_l \equiv T$$

for some $T_1, \ldots, T_{l-1}$ such that every T_j, $0 \leqslant j \leqslant l-1$, is $(j, 0)$-If-approximable and no T'' with $T_j < T''$ is $(j, 0)$-If-approximable;

(c) T is *$(l, k+1)$-If-approximable*, $l > 0$, iff for any k-If-approximable $\bar{T} < T$ such that

$$\bar{T} \equiv T_0 < T_1 < \ldots < T_{l-1} < T_l \equiv T$$

for some $T_1, \ldots, T_{l-1}$ such that every T_j, $0 \leqslant j \leqslant l-1$, is $(j, k+1)$-If-approximable and no T'' with $T_j < T''$ is $(j, k+1)$-If-approximable.

We now state two properties proved in [7] for all γ, P, v in an arbitrary countable semantical structure:

WELLFOUNDEDNESS LEMMA. *$T(\mathrm{Wh}_{\gamma,P}, v)$ is wellfounded iff it is If-approximable.*

WELLFOUNDEDNESS THEOREM. *$T(\mathrm{Wh}_{\gamma,P}, v)$ is wellfounded iff it is k-If-approximable for some $k > 0$.*

For examples of wellfounded computation trees of the considered family of programs see [7] and [6].

Computation trees allow the definition of functions expressing scheduling strategies on their nodes. Some scheduling functions assume fairness criteria in choosing computations with a given property. That is the case for instance in [1], where a scheduling function, written in the form of a set of instructions, is added to the given programs. Every fair scheduling function is defined on a computation tree which is previously assumed to be fair with respect to the property considered. As to termination, a fair strategy on computation trees presupposes a definition of a fair-wellfounded computation tree, i.e., of a computation tree such that if any computation unfair with respect to termination is overlooked, in it, all its remaining computations turn out to be finite.

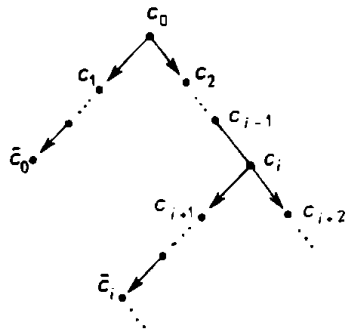
Let us assume the notion of unfair computation and of fair-wellfoundedness as in Definition 1 below.

DEFINITION 1. Let $T(P, v)$ be a computation tree with F as its nonempty multiset of terminal nodes and let C be an infinite computation in it.

(i) C is *termination-unfair* in $T(P, v)$ iff for every configuration c in C there exists some $\bar{c}$ in F such that $c \twoheadrightarrow \bar{c}$.

(ii) $T(P, v)$ is *fair-wellfounded* iff any infinite computation in it is termination-unfair.

So fair-wellfounded is for instance any computation tree of the following kind:



By Definition 1, every finite computation is termination-fair and every wellfounded computation tree is fair-wellfounded.

It is possible to have an order-theoretic characterization of fair-wellfound-

ed computation trees, which distinguishes wellfoundedness, by the notion of cofinality.

Let $(X, \leqslant)$ be a preordered multiset with Y a nonempty multiset properly included in X.

DEFINITION 2. Y is *cofinal in* X iff for every x in X there exists y in Y such that $x \leqslant y$.

THEOREM. *For every* P *in* $P(\Sigma)$ *and every* v *in* $\mathrm{Val}(A, B_0)$

(i) $T(P, v)$ *is fair-wellfounded iff the multiset of all its terminal configurations is cofinal in* $T(P, v)$.

(ii) $T(P, v)$ *is wellfounded iff the multiset of all its terminal configurations is cofinal in the set of all computations of* $T(P, v)$.

Proof. $T(P, v)$ can be considered as a multiset $c_0\downarrow$, with $c_0 \equiv \langle P, v\rangle$, preordered by the transition relation $\rightarrow$. In fact, $\rightarrow$ is transitive, not reflexive and not antisymmetric (it is possible that $c_1 \rightarrow\!\!\!\!\rightarrow c_2$, $c_2 \rightarrow\!\!\!\!\rightarrow c_1$ and $c_1 \neq c_2$, so that $\rightarrow$ is not in general a partial order). The multiset F of all terminal $\bar{c}$ in $c_0\downarrow$ is properly contained in $c_0\downarrow$ when F is not empty. Let C be an arbitrary computation in $T(c_0)$ and let $C \leqslant \bar{c}$ mean that for every c in C there exists $\bar{c}$ in F such that $c \rightarrow\!\!\!\!\rightarrow \bar{c}$ or $c \equiv \bar{c} \cdot T(c_0)$ can also be seen as a multiset of C.

($\Rightarrow$): (i) If $T(c_0)$ is fair-wellfounded, then an arbitrary C in it is either finite, and then F is cofinal in the set of all C, or termination unfair, and then F is disjoint with C and such that for every c in C and for some $\bar{c}$ in F, $c \rightarrow\!\!\!\!\rightarrow \bar{c}$. Since C is arbitrary in $T(c_0)$, F is cofinal in $T(c_0)$.

(ii) If $T(c_0)$ is wellfounded, then F is cofinal in the set of all C of $T(c_0)$.

(Notice that cofinality in the set of all C of $T(c_0)$ implies cofinality in $T(c_0)$ and not vice versa).

($\Leftarrow$): (i) If F is cofinal in $T(c_0)$, then for any infinite C in $T(c_0)$, F is disjoint with C and $c \rightarrow\!\!\!\!\rightarrow \bar{c}$ for every c in C and some $\bar{c}$ in F, i.e., any infinite computation in $T(c_0)$ is termination-unfair, i.e., $T(c_0)$ is fair-wellfounded.

(ii) If F is cofinal in the set of all computations of $T(c_0)$, no infinite computation can be in it, since $C \leqslant \bar{c}$, for every C in $T(c_0)$. ∎

Let π be a given property of configurations in $T(c_0)$. The notion of fairness with respect to π as fair reachability in $T(c_0)$ of all nodes with property π (see [10]) can be analogously characterized by cofinal multisets.

3. Fair-wellfoundedness in ALNA

Let us briefly give the syntax and semantics of a language of ALNA. For a more extensive definition see [7].

3.1. Syntax. Let $L(\Sigma)$, a language over signature Σ, be an extension of $L_0(\Sigma)$ in Part 1 of the present paper, whose set Φ of formulas is such that

(a) $\Phi \supset \Phi_0$, where Φ_0 is the set of classical first order formulas on the alphabet of $L(\Sigma)$, without quantifiers;

(b) If φ is in Φ and P is in $P(\Sigma)$, then $\Diamond P\varphi$ and $\Box P\varphi$ are in Φ;

and is closed under propositional connectives ($\sim$, $\vee$, $\wedge$, $\rightarrow$), quantifiers ($\exists x$, $\forall x$), infinite disjunction and conjunction ($\bigvee_i \varphi_i$, $\bigwedge_i \varphi_i$) with a finite set of free individual variables in (φ_i, $i < \omega$).

3.2. Semantics. For a realization of $L(\Sigma)$, let A, B_0, v be defined as in 1-2. A realization of any term and of any open formula in $L(\Sigma)$ is defined as usual. Realization, in A and B_0, of formulas in $\Phi - \Phi_0$ is such that it associates to every formula an element of B_0 as follows, with A omitted, being supposed to be fixed:

$v \models \Diamond P\varphi$ iff there exists some successful computation in $T(P, v)$ whose result satisfies φ.

$v \models \Box P\varphi$ iff all computations in $T(P, v)$ are successful and their results satisfy φ.

Realizations of $\sim \varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$, $\exists x \cdot \varphi$, $\forall x \cdot \psi$, $\bigvee_i \varphi_i$, $\bigwedge_i \varphi_i$ are defined as usual.

For some program properties expressible in ALNA see [7] and [6]. Let P in $P(\Sigma)$ be said to be fair-wellfounded in case every computation tree of P is.

FAIR-WELLFOUNDEDNESS PROPOSITION. *A program P in $P(\Sigma)$ is fair-wellfounded iff for every realization A, the formula Fair-Wf(P) inductively defined as*

$$\textbf{Fair-Wf}(\text{skip}) \equiv \text{true},$$
$$\textbf{Fair-Wf}(p := j) \equiv \text{true},$$
$$\textbf{Fair-Wf}(x_i = y \cdot \delta(y)) \equiv \text{true},$$
$$\textbf{Fair-Wf}(P_1; P_2) \equiv \textbf{Fair-Wf}(P_1) \wedge \sim \Diamond P_1(\sim \textbf{Fair-Wf}(P_2)),$$
$$\textbf{Fair-Wf}(\text{If}_{\gamma, P_1, P_2}) \equiv (\gamma \wedge \textbf{Fair-Wf}(P_1)) \vee (\sim \gamma \wedge \textbf{Fair-Wf}(P_2)),$$
$$\textbf{Fair-Wf}(\text{Wh}_{\gamma, P}) \equiv \bigwedge_k \Box \text{If}^k_{\gamma, P}(\sim \gamma \vee \bigvee_n \Diamond \text{If}^n_{\gamma, P} \sim \gamma)$$

is valid in A.

Proof. For every countable semantical structure A and every v in A:

$$\text{(I)}\ [\![\textbf{Fair-Wf}(\text{skip})]\!]_A(v) = [\![\textbf{Fair-Wf}(p := \gamma)]\!]_A(v) = [\![\textbf{Fair-Wf}(x := y \cdot \delta(y))]\!]_A(v) = [\![\text{true}]\!]_A(v) = tt$$

because all their computation trees are wellfounded.

$$\text{(II)}\quad [\![\textbf{Fair-Wf}(P_1) \wedge \sim \Diamond P_1(\sim \textbf{Fair-Wf}(P_2))]\!]_A(v) = [\![\textbf{Fair-Wf}(P_1)]\!]_A(v) \cap [\![\sim \Diamond P_1(\sim \textbf{Fair-Wf}(P_2))]\!]_A(v) = tt$$
$$\text{iff}\quad [\![\textbf{Fair-Wf}(P_1)]\!]_A(v) = tt,$$

i.e., $T(P_1, v)$ is fair-wellfounded, and, moreover,

$$[\![\sim \Diamond P_1 (\sim \textbf{Fair-Wf}(P_2))]\!]_A(v) = tt,$$

i.e., every successful C in $T(P_1, v)$ has terminal configuration $\langle \lceil \; \rfloor, \bar{v} \rangle$ such that $T(P_2, \bar{v})$ is fair-wellfounded; otherwise

$$[\![\textbf{Fair-Wf}(P_1) \wedge \sim \Diamond P_1 (\sim \textbf{Fair-Wf}(P_2))]\!]_A(v) = ff.$$

By the semantics of sequential composition,

$$[\![\textbf{Fair-Wf}(P_1) \wedge \sim \Diamond P_1 (\sim \textbf{Fair-Wf}(P_2))]\!]_A(v) = tt$$

iff $T(P_1; P_2, v)$ is fair-wellfounded.

(III) $$[\![\gamma \wedge \textbf{Fair-Wf}(P_1)) \vee (\sim \gamma \wedge \textbf{Fair-Wf}(P_2))]\!]_A(v)$$
$$= ([\![\gamma]\!]_A(v) \cap [\![\textbf{Fair-Wf}(P_1)]\!]_A(v)) \cup ([\![\sim \gamma]\!]_A(v) \cap [\![\textbf{Fair-Wf}(P_2)]\!]_A(v))$$
$$= \begin{cases} tt & \text{iff} \quad v \models \gamma \quad \text{and} \quad T(P_1, v) \quad \text{is fair-wellfounded or} \\ & \quad\; v \models \sim \gamma \quad \text{and} \quad T(P_2, v) \quad \text{is fair-wellfounded,} \\ ff & \text{otherwise.} \end{cases}$$

By the semantics of conditional composition

$$[\![(\gamma \wedge \textbf{Fair-Wf}(P_1)) \vee (\sim \gamma \wedge \textbf{Fair-Wf}(P_2))]\!]_A(v) = tt$$

iff $T(\text{If}_{\gamma, P_1, P_2}, v)$ is fair-wellfounded.

(IV) $$[\![\bigwedge_k \Box \text{If}^k_{\gamma, P} (\sim \gamma \vee \bigvee_n \Diamond \text{If}^n_{\gamma, P} \sim \gamma)]\!]_A(v)$$
$$= \bigcap_{k < \omega} [\![\Box \text{If}^k_{\gamma, P} (\sim \gamma \vee \bigvee_n \Diamond \text{If}^n_{\gamma, P} \sim \gamma]\!]_A(v)$$
$$= \begin{cases} tt & \text{iff for every } k < \omega, \; [\![\Box \text{If}^k_{\gamma, P} (\sim \gamma \vee \bigvee_n \Diamond \text{If}^n_{\gamma, P} \sim \gamma)]\!]_A(v) = tt, \text{ i.e.,} \\ & \quad \text{every } C \text{ of } T(\text{If}^k_{\gamma, P}, v) \text{ has } \langle \lceil \; \rfloor, \bar{v} \rangle \text{ such that} \\ & \quad [\![\sim \gamma \vee \bigvee_n \Diamond \text{If}^n_{\gamma, P} \sim \gamma]\!]_A(\bar{v}) = [\![\sim \gamma]\!]_A(v) \cup [\![\bigvee_n \Diamond \text{If}^n_{\gamma, P} \sim \gamma]\!]_A(v) \\ & \quad = tt, \text{ i.e. either } \bar{v} \models \sim \gamma \text{ or } \bigcup_{n < \omega} [\![\Diamond \text{If}^n_{\gamma, P} \sim \gamma]\!]_A(\bar{v}) = tt, \text{ i.e., some} \\ & \quad T(\text{If}^{k+n}_{\gamma, P}, v) \text{ has some successful computation,} \\ ff & \text{otherwise.} \end{cases}$$

By taking into account that to every finite C' in $T(\text{Wh}_{\gamma, P}, v)$ there corresponds a C in $T(\text{If}^k_{\gamma, P}, v)$, for some $k < \omega$, such that C and C' have the same sequence of states, and vice versa, to every C with $\langle \lceil \; \rfloor, \bar{v} \rangle$, $\bar{v} \models \sim \gamma$, in $T(\text{If}^k_{\gamma, P}, v)$ and to every C with $\langle \lceil \; \rfloor, \bar{v} \rangle$, $\bar{v} \models \sim \gamma$, in some $T(\text{If}^{k+n}_{\gamma, P}, v)$, there correspond some successful C' in $T(\text{Wh}_{\gamma, P}, v)$.

Since for every $k < \omega$ there exists some $n < \omega$ such that $v \models$ **Fair-Wf**$(\text{Wh}_{\gamma, P})$, the multiset of $\langle \lceil \; \rfloor, \bar{v} \rangle$ in $T(\text{Wh}_{\gamma, P}, v)$ with either $\bar{v} \models \sim \gamma$ or $\bar{v} \models \bigvee_n \Diamond \text{If}^n_{\gamma, P} \sim \gamma$ is cofinal in $T(\text{Wh}_{\gamma, P}, v)$. By the given characterization, that happens iff $T(\text{Wh}_{\gamma, P}, v)$ is fair-wellfounded. ∎

Notice that the fair-wellfoundedness formula can be written by using iteration quantifiers.

The wellfoundedness of P, $\mathbf{Wf}(P)$, is expressed in [7] as follows:

$$\mathbf{Wf}(\text{skip}) \equiv \text{true},$$
$$\mathbf{Wf}(p := \gamma) \equiv \text{true},$$
$$\mathbf{Wf}(x := y \cdot \delta(y)) \equiv \text{true},$$
$$\mathbf{Wf}(P_1; P_2) \equiv \mathbf{Wf}(P_1) \wedge \sim \Diamond P_1(\sim \mathbf{Wf}(P_2)),$$
$$\mathbf{Wf}(\text{If}_{\gamma, P_1, P_2}) \equiv (\gamma \wedge \mathbf{Wf}(P_1)) \vee (\sim \gamma \wedge \mathbf{Wf}(P_2)),$$
$$\mathbf{Wf}(\text{Wh}_{\gamma, P}) \equiv \bigvee\!\!\!\bigvee_k ((n_k)\,\text{If}_{\gamma,P}(\ldots((n_1)\,\text{If}_{\gamma,P} \sim \gamma)\ldots)),$$

where $(n)\ \text{If}_{\gamma,P}\varphi$ stands for $\bigvee\!\!\!\bigvee_n \bigvee_{\square}^{n} \text{If}_{\gamma,P}\varphi$, $\bigvee_{\square}^{n} \text{If}_{\gamma,P}\varphi$ stands for $\bigvee_{\square} \text{If}_{\gamma,P}(\ldots(\bigvee_{\square} \text{If}_{\gamma,P}\varphi)\ldots)$ (n times) and $\bigvee_{\square} \text{If}\varphi$ stands for $\bigvee\!\!\!\bigvee_m \square \text{If}^m_{\gamma,P}\varphi \vee \varphi$. (So, in the wellfoundedness formula, iteration quantifiers are not enough).

References

[1] K. R. Apt and R. E. Olderog, *Proof Rules Dealing with Fairness*, Bericht No. 8104, March 1981, University of Kiel.

[2] K. R. Apt and G. D. Plotkin, *Countable Nondeterminism and Random Assignment*, Internal Report CRS-98-82, January 1982, University of Edinburgh.

[3] R. J. R. Back, *Proving Total Correctness of Nondeterministic Programs in Infinitary Logic*, Acta Informatica 15 (1981), 223–249.

[4] H. J. Boom, *A Weaker Precondition for Loops*, Transaction on Programming Languages and Systems 4 (4) (1982), 668–688.

[5] D. Gabbay, A. Pnueli, S. Shelah and J. Stavi, *On the Temporal Analysis of Fairness*, in Proceedings 7th ACM POPL, Las Vegas 1980.

[6] G. F. Mascari and M. Venturini Zilli, *Expressiveness of Algorithmic Logic for While-programs with Nondeterministic Assignments*, Quaderni IAC, Serie III, no. 158, Roma 1983.

[7] –, –, *While-programs with Nondeterministic Assignments and the logic ALNA*, Theoret. Comput. Sci. 39 (1985), 1-25.

[8] G. Mirkowska, *Algorithmic Logic with Nondeterministic Programs*, Fundamenta Informaticae 3 (1980), 45–64.

[9] –, *PAL – Propositional Algorithmic Logic*, ibid. 4 (1981), 675–760.

[10] J. P. Queille and J. Sifakis, *Fairness and Related Properties in Transition Systems. A Temporal Logic to Deal with Fairness*, Acta Informatica 19 (1983), 195–200.

[11] H. Rasiowa, *Lectures on Infinitary Logic and Logics of Programs*, Quaderni IAC, Serie III, No. 142, Roma 1982.

[12] M. Venturini Zilli, *Transition Graphs Semantics and Languages*, Proceedings of the Symposium on Mathematical Foundations of Computer Science, Zaborów (December 1984), Springer LNCS no. 208 (1985), 375–384.

Presented to the semester
Mathematical Problems in Computation Theory
September 16 – December 14, 1985