Abstract fairness and semantics

Doris Nolte, Lutz Priese*

Fachbereich Informatik, Universität Koblenz-Landau, Rheinau 1, 56075 Koblenz, Germany

Abstract

Fairness of a program execution, c, is usually expressed such that all objects which are sufficiently often enabled have to occur also sufficiently often in c. There exists a well-known strong equivalence between fair program executions, Π₁ⁿ-formulae, and convergence of initial program executions. However, these results cannot be applied to a study of "fair semantics" of programs, as such a fair semantics is a Σ₁ⁿ-formula in general. The main reason therefore is that a semantics does not tell which objects are enabled – only the actually occurring objects are usually seen in semantics. Here we study on a very abstract level some quite natural requirements for semantics s.t. fair semantics with invisible "enabledness" can also be characterized with topological techniques.

1. Introduction

A program execution, c, of some program s ∈ L for some programming languages L (with operators for non-determinism and/or concurrency, parallelism, synchronization, etc.) is usually called fair if any program alternative, that could be chosen (is enabled) in c sufficiently often, has to be chosen (has to occur) sufficiently often in c. Such enabled alternatives may be events in Petri-nets, edges in graphs or transition systems, labels of events in transition systems, waiting communication partners or single messages, channels, etc. in OCCAM-like languages, etc. If sufficiently often enabled means always enabled, one speaks of impartiality or unconditional fairness, if it means infinitely often enabled (or almost always enabled) one speaks of strong (or weak, respectively) fairness. Thus, strong fairness is usually defined by a formula

c is strongly fair iff

∀o ∈ alternatives: ∃ⁿ: E(s, c[n], o) ⇒ ∃ᵐ: O(s, c[m], o).

Here E(s, c[n], o) reads that "o is enabled at c[n], where c[n] is an initial program execution of the program s", and O(s, c[n], o) stands for "o occurs at c[n]". ∃ⁿ:
$P(\ldots, n, \ldots)$ abbreviates $\forall n : \exists n' \geq n : P(\ldots, n', \ldots)$. As the predicates $E$ and $O$ are usually decidable, strong fairness is described by a $\Pi^0_2$-formula, i.e. a formula of the form $\forall \exists \forall R$ with a recursive predicate $R$ over finite and infinite program executions, where the quantifiers range only over integers (or finite programs or finite program execution). Analogously, weak fairness is described by

$c$ is weakly fair iff

$$\forall o \in \text{alternatives: } \forall^o n : E(s, c[n], o) \Rightarrow \exists^{(o)} m : O(s, c[m], o)$$

and becomes thus a $\Pi^0_2$-formula of the form $\forall \exists R$. $\forall^o n \ P(\ldots, n, \ldots)$ abbreviates $\exists n : \forall n' \geq n : P(\ldots, n', \ldots)$.

In [4, 8] it is shown that such abstract strong (and weak) fair program executions are exactly those infinite program executions that are the limit points of their finite initial prefixes under some appropriate ultra-metric (that refines the natural Baire metric). Further, any $\Pi^0_3$-($\Pi^0_2$-)set defines canonically some strong (weak) fairness concept for program executions. These results may be regarded as a solution to a research program started by Degano and Montanari. They proved in [6] that a program execution $c$ in some variant of a CCS-language is fair iff its initial segments form a Cauchy sequence in some appropriate metric space. However, the results in [4, 8] cannot be transformed to the semantics of a program. A fair semantics is defined by a $\Sigma^1_2$-formula of the analytical hierarchy, extending $\Pi^0_2$- or $\Pi^0_1$-formula of the arithmetical hierarchy further. We will explain this with two very simple examples.

**Example 1.** Let $A$ denote the finite automaton of Fig. 1. $A$ (as a graph) consists of three states and five edges, $e_1, \ldots, e_5$, that are labelled by $\{a, b\}$. A run, $r$, of $A$ is a finite or infinite chain of edges starting from the initial state.

$Z(A) = \{r \in E^o \cup E^*; \ r \text{ is a run of } A\}$ corresponds to the set of possible program executions of $A$, while the semantics of $A$ would correspond to its language $\mathcal{F}(A) = \text{label}(Z(A))$, the set of words of labellings of allowed runs. We may call a run, $r$, fair iff, whenever $r$ reaches a state infinitely often where an edge with label $x$ leaves,
this label, $x$, has to occur infinitely often as a label of the edges of $r$. Thus, the run $r_1 = e_1(e_2)\omega$ is fair, with label$(r_1) = a\omega$, the run $r_2 = e_3(e_4)\omega$ is fair, with label$(r_2) = a(ab)\omega$. But the run $r_3 = e_5(\epsilon)\omega$, with label$(r_3) = a\omega$, is not fair as $r_3$ might use edge $e_5$ with label $b$ infinitely often. We regard the fair semantics of $A$ as the set of labels of the fair runs. This leads to

$$
\mathcal{C}_{\text{fair}}(A) = \{ r \in \mathcal{C}(A); \text{ $r$ is a fair run in $A$} \},
$$

$$\mathcal{C}^\text{fair}(A) = \text{label}(\mathcal{C}^\text{fair}(A)) = \{ w \in \{a, b\}^\omega; \exists r \in \mathcal{C}^\text{fair}(A) : w = \text{label}(r) \}.$$

$\mathcal{C}^\text{fair}(A)$ is easily shown to be $\Pi^0_3$, but the form of $\mathcal{C}^\text{fair}(A)$ is in $\Sigma_1^1$ as the $\exists$-quantifier may range over infinite runs, (a $\Sigma_1^1$-formula is given by $\exists f : P$, where $f$ is a function (an infinite object) and $P$ some arithmetical predicate).

**Example 2.** The second example regards programs with parallelism, $\parallel$, choice, $\lor$, and recursion, $\mu$. Let $s_1, s_2, s_3$ be the three programs $s_1 := \mu x((a; x) \parallel c)$, $s_2 := \mu x(((a; x) + b) \parallel c)$, $s_3 := s_1 + s_2$.

We regard a true-concurrency semantics with sets of pomsets. Let $p$ be the infinite pomset as described by Fig. 2.

Obviously, $p$ belongs to the true concurrency semantics of $s_1, s_2$, and of $s_3$. Intuitively, $p$ is fair for $s_1$ and unfair for $s_2$, as $b$ is infinitely often enabled but never chosen in $s_2$. One cannot tell whether $p$ is fair or unfair in $s_3$: if at the first step in $s_3$ the alternative $s_1$ is chosen, $p$ is fair, otherwise $p$ is unfair. However, $p$ does not tell whether $s_1$ or $s_2$ was chosen in $s_3$.

Again, enabledness is defineable on the level of program execution, but not necessarily on the level of semantics. As was noticed by Darondeau in [3], one has to introduce indexed pomsets, e.g., to handle fairness.

In our example we may introduce the following program terms:

$$
t_1 := a, \quad t_2 := x, \quad t_3 := c, \quad t_4 := a, \quad t_5 := x, \quad t_6 := b, \quad t_7 := c,
$$

and

$$s_3 := \mu x((t_1; t_2) \parallel t_3) + \mu x((t_4; t_5) + t_6 \parallel t_7).$$

If we index each action by its term we get a program execution that tells exactly how the program was executed. $\hat{p}_1, \hat{p}_2$ of Fig. 3 are such indexed pomsets that are true-concurrency executions of $s_3$. Obviously, $\hat{p}_1$ is a fair, $\hat{p}_2$ an unfair program execution.
Thus, $\mathcal{C}(s)$ becomes a set of indexed pomsets and $\mathcal{S}(s)$ a set of normal pomsets s.t. $\mathcal{S}(s) = \text{label}(\mathcal{C}(s))$, where "label" forgets the indices. Again, the fair semantics $\mathcal{S}^{\text{fair}}(s)$ is a homomorphic image of $\mathcal{C}^{\text{fair}}(s)$, namely $\mathcal{S}^{\text{fair}}(s) := \text{label}(\mathcal{C}^{\text{fair}}(s)) = \{ p \in \mathcal{S}(s); \exists \hat{p} \in \mathcal{C}^{\text{fair}}(s) : p = \text{label}(\hat{p}) \}$. Again, the fair semantics is a $\Sigma^1_1$-set.

Thus, whilst enabledness is easily testable on the level of program executions it becomes usually undetectable on the level of semantics. Nevertheless, we will present a very general and abstract theorem which proves that under some quite natural restrictions even fair semantics become $\Pi^0_3$-sets. Applying the characterization theorem of the following section allows us to express the fair semantics as a limit point in those cases. As our general theorem will be applicable for this simple second example, we can, e.g., conclude:

$$p \in \mathcal{S}^{\text{fair}}(s_3) \gg p = \lim_{n \to \omega} p[n],$$

for some ultra-metric, $d$. Here, $p[n]$ denotes the initial segment of $p$ of length $n$.

We proceed as follows. In Section 2 we present our notations and the characterization theorem for $\Pi^0_3$-sets and present an application to programs. In Section 3 we present and prove the mentioned theorem on fair semantics. In Section 4 we present a few more examples that show the broad applicability of our theorem of Section 3. We will operate with very abstract fairness concepts to keep our results as general as possible.

2. Notations, $\Pi^0_3$-characterization-theorem

Let $\Sigma$ be a finite or infinite alphabet. $\Sigma^*$ and $\Sigma^\omega$ denote the sets of finite and infinite sequences (words) over $\Sigma$. $\Sigma^\omega := \Sigma^* \cup \Sigma^\omega$. For any finite or infinite word $w$ we denote by $w(n)$ the $n$th letter within $w$ and by $w[n]$ its prefix of length $n$. For $u \in \Sigma^*$, $v \in \Sigma^\omega$, $u < v$ ($u$ is a proper prefix of $v$) holds if there exists some $w \neq \varepsilon$ s.t. $uw = v$.

$$d(u,v) := \begin{cases} 0 & \text{if } u = v, \\ \frac{1}{m+1} & \text{where } m = \sup\{n; u[n] = v[n]\} \text{ otherwise} \end{cases}$$

defines the natural (Baire) ultra-metric on $\Sigma^\omega$. 
A $\Pi^0_n$ ($\sum^0_n$) predicate $P$ is defined by a formula $\forall \exists \cdots \forall (\exists \forall \cdots \exists (\forall R)$, respectively) with a recursive kernel, $R$, and $n$ changes of quantifiers, starting with a $\forall$-(for $\Pi^0_n$) or $\exists$-(for $\Sigma^0_n$) quantifier. The quantifiers range over integers but the predicates $P$ and $R$ may range over integers and functions. The $\Pi^1_n$ ($\Sigma^1_n$) case is defined analogously, but now quantifiers range over functions and the kernel may be any $\Pi^0_n \cup \Sigma^0_n$-predicate. As there exists a well-known recursive and bijective Gödel-coding, $g$, between $\mathbb{N}$ and $\mathbb{N}^*$ we also allow quantifiers to range over finite sequences of integers (in the $\Pi^0_n$-$\Sigma^0_n$-hierarchy). Such a Gödel-coding $g$ may be defined by $g(n_1, \ldots, n_k) := \langle k, \langle n_1, \cdots, \langle n_{k-1}, n_k \rangle \cdots \rangle \rangle$, where $\langle \rangle : \mathbb{N}^2 \rightarrow \mathbb{N}$ is a recursive bijection (a Cantor-coding).

Thus, there exists a recursive function length: $\mathbb{N} \rightarrow \mathbb{N}$ s.t. length $(m) := |g^{-1}(m)|$, where $|\langle n_1, \ldots, n_k \rangle| := k$. A metric $d$ is called a $\Pi^0_n$ ($\Sigma^0_n$) metric if the predicate $d(f,g) < 1/n$ over $\mathbb{N}^\infty \times \mathbb{N}^\infty \times \mathbb{N}$ is $\Pi^0_n$ ($\Sigma^0_n$).

For a metric space $(M,d)$, a subset $K \subseteq M$, we define

- the set of cluster points of $K$ by
  $$CP_d(K) := \left\{ x \in M ; \forall n \in \mathbb{N} : \exists x_n \in K : x_n \neq x \; and \; d(x, x_n) < \frac{1}{n} \right\},$$

- the set of limes points of $K$ by
  $$LIM_d(K) := \left\{ x \in M ; \forall i \in \mathbb{N} : \exists x_i \in K : x = \lim_{i \rightarrow \omega} x_i \right\},$$

- the set of Cauchy-sequences by
  $$CS_d := \{ w \in \mathbb{N}^\omega ; (w[n])_{n \in \mathbb{N}} \text{ is a Cauchy-sequence in the metric } d \}.$$ 

Further, whenever we write $\exists n ; \forall n ;$, instead of $\exists n \in M ; \forall n \in M ;$, we refer to $n \in \mathbb{N}$ (or, equivalently, $\mathbb{N}^*$).

A fairness concept, $\mathcal{F}$, is given by two recursive predicates $E, O \subseteq \mathbb{N}^2$. A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is called (strongly) $\mathcal{F}$-fair iff there holds:

$$\forall o : (\exists^\omega n : E(f[n], o) \Rightarrow \exists^\omega m : O(f[m], o)).$$

As $\forall (\exists^\omega \Rightarrow \exists^\omega) = \forall (\forall \exists \Rightarrow \forall \exists) = \forall (\forall \exists \forall \exists) = \forall \exists \forall$ holds, the above formula is $\Pi^0_3$.

In [4] the following theorem is shown.

**Characterization theorem.** For $M \subseteq \mathbb{N}^\omega$ the following statements are equivalent.

- $M$ is a $\Pi^0_3$-set,
- $M = \{ f : \mathbb{N} \rightarrow \mathbb{N} ; f$ is $\mathcal{F}$-fair $\}$ for some fairness concept $\mathcal{F}$,
- $M = CP_d(\mathbb{N})$ for some $\Sigma^0_2$-metric, $d$,
- $M = CP_d(\mathbb{N})$ for some $\Pi^0_1$-ultra metric, $d$, that refines the Baire metric,
- $M = LIM_d(\mathbb{N})$ for some $\Pi^0_1$-ultra metric, $d$, that refines the Baire metric,
- $M = CS_d$ for some $\Pi^0_1$-ultra metric, $d$, that refines the Baire metric.
In addition, $CP_d(\mathbb{N})$ and $LIM_d(\mathbb{N})$ may be replaced by $CP_d(\mathbb{N}^\infty)$ and $LIM_d(\mathbb{N}^\infty)$, respectively. Further, for any $\Pi_3^0$-set $M \subseteq \mathbb{N}^\infty$

$$f \in M \Rightarrow f = \lim_{n \to \infty} f[n]$$

holds for an appropriate $\Pi_3^0$-ultra metric, $d$, that refines the Baire metric.

This characterization theorem has a large variety of applications to fairness in various models of computations. Let $\mathcal{L}$ be some program language. A program $s \in \mathcal{L}$ shall be a finite object that can be coded as an integer, $g: \mathcal{L} \rightarrow \mathbb{N}$, $g(s)$ is the coding of $s$. We assume a program execution, $c$, of $s$ to be some finite or infinite sequence (of actions, e.g.), $c = c(1) \cdots c(n)$, or $c = c(1) \cdots c(n) \cdots$. Further, we assume that

- all possible program alternatives (that define the concept of strong fairness together with the enabledness and occurrence predicates) are finite objects and are thus coded as integers,
- $c$ is coded by $g(c)$ as a finite or infinite sequence of integers s.t. there holds:

$$g(c[n]) = g(c)[n] \quad \forall n \in \mathbb{N}. \tag{1}$$

A program $s$ is thus intuitively strongly $\mathcal{F}$-fair if the following holds for two predicates $E_\mathcal{F}$ and $O_\mathcal{F}$: $\forall o \in \text{alternatives}: (\exists^o n : E_\mathcal{F}(s, c[n], o) \Rightarrow \exists^o m : O_\mathcal{F}(s, c[m], o))$. To get a uniform fairness concept on $\mathcal{L}$ the predicates $E_\mathcal{F}$ and $O_\mathcal{F}$ need $s$ as a parameter. Of course, whether $o$ is enabled in $c[n]$ depends on the program $s$ of which $c$ is a program execution: $E_\mathcal{F}(s, c[n], o)$. If we would drop $s$ as a parameter, $E_\mathcal{F}$ and $O_\mathcal{F}$ would depend only on $c[n]$ and $o$ with the strange consequence that formally we would get different fairness concepts for different programs. In the case of fairness concepts for functions (in our characterization theorem) there are no programs such that predicates over $\mathbb{N}^2$ are sufficient. To be able to talk about recursive predicates $E$ and $O$ we need the coding into $\mathbb{N}$ or $\mathbb{N}^\infty$. Thus, we can express $g(c)$ is $\mathcal{F}$-fair $\iff g(c)$ is a coding of a program execution of $s$ and

$$\forall o \in \mathbb{N}: (\exists^o n : E_\mathcal{F}(g(s), g(c[n]), o)$$

$$\Rightarrow \exists^o m : O_\mathcal{F}(g(s), g(c[m]), o)).$$

Applying Eq. (1) we thus know (by defining $E(x, y) : \mathcal{F} E_\mathcal{F}(s, x, y)$, $O(x, y) : \mathcal{F} O_\mathcal{F}(s, x, y)$, where $s$ is a fixed program):

$$f: \mathbb{N} \rightarrow \mathbb{N} \text{ is } \mathcal{F}\text{-fair } \iff f \in \mathcal{C}(s) \quad \text{and } \forall o: (\exists^o n : E(f[n], o) \Rightarrow \exists^o m : O(f[m], o)),$n

where $E, O \subseteq \mathbb{N}^2$ are recursive predicates and $f \in \mathcal{C}(s)$ is the predicate "that $f$ is the coding of a program execution of $s". We suppose that $f \in \mathcal{C}(s)$ is a $\Pi_3^0$-predicate, too. Thus, the whole previous formula is $\Pi_3^0$ and we may apply our characterization theorem. As a consequence, any fair program execution $c$ of $s$ in $\mathcal{L}$ is approximated by its initial segments, and vice versa, etc., see the characterization theorem.
It should be noted that $f \in \mathcal{C}(s)$ is usually a $\Sigma^0_2$- or $\Pi^0_1$-predicate, as infinite program executions are approximations of finite program executions and a coding on finite program executions has to be computable in both directions. Thus,

$$f \in \mathcal{C}(s) \iff \forall n : f[n] \in \mathcal{C}(s)_{\text{finite}}$$

holds in many concrete examples, where $f[n] \in \mathcal{C}(s)_{\text{finite}}$ is recursive. There are examples where only maximal program executions are allowed (i.e. with some end symbol telling the type of termination) such that $f[n]$ is not a finite program execution. However, in those cases the following usually holds:

$$f \in \mathcal{C}(s) \iff \forall n : f[n] \in \text{Pref}(\mathcal{C}(s)),$$

where $f[n] \in \text{Pref}(\mathcal{C}(s))$ should be recursive for any reasonable coding function $g$. $\text{Pref}(\mathcal{C}(s))$ denotes the set of prefixes (initial segments) of $\mathcal{C}(s)$. In any case, if we claim $f \in \mathcal{C}(s)$ to be a $\Pi^0_3$-predicate we are on the safe side. For simplicity, we may always define $\mathcal{C}(s) := \mathcal{C}(s) \cup \text{Pref}(\mathcal{C}(s))$ and operate thus with prefix-closed sets of program executions.

### 3. Abstract fairness and semantics

In this chapter we prove a very general theorem telling when a fair semantics becomes a $\pi^0_3$-set (instead of a $\Sigma^0_1$-set, the general case). As a consequence, the characterization theorem can be applied to the semantics itself in those cases. We proceed on a very abstract level to get our results in great generality.

**Definition 3.1.** An (abstract step) semantics for a programming language, $\mathcal{L}$, is given by sets $D, D$, and mappings $\mathcal{C}, \mathcal{I}, h, g$ s.t. the following holds:

- $D, D \subseteq \mathbb{N}^\infty$
- $\mathcal{C} : \mathcal{L} \to 2^D$ s.t. $\forall s \in \mathcal{L}$ holds:
  - $\mathcal{C}(s)$ is a $\Pi^0_3$-subset of $\mathbb{N}^\infty$ (see the above discussion),
  - $\mathcal{C}(s)$ is prefixed-closed, i.e. $\forall n \in \mathbb{N} : \mathcal{C}(s)[n] \subseteq \mathcal{C}(s)$,
  - $\mathcal{C}(s)$ is closed under the natural Baire metric, $\delta$
- $h : D_\mathcal{C} \to D_\mathcal{s}$ is a prefix-closed mapping, i.e. $\forall c \in D_\mathcal{C} : \forall n : h(c[n]) = h(c)[n]$
- $\mathcal{I} = h \circ \mathcal{C}$,
- $g : \mathcal{L} \to \mathbb{N}$ is an injective coding of programs into integers.

$\mathcal{C}(s)$ is called the set of (codings of) program executions of $s$, and $\mathcal{I}(s)$ the set of (codings of) meanings of $s$ and is the semantics of $s$.

**Definition 3.2.** An (abstract) fairness concept, $\mathcal{F}$, for $\mathcal{L}$, is a pair $\mathcal{F} = (F, O_\mathcal{F})$ of two recursive predicates $F, O_\mathcal{F}$ on $\mathbb{N}^\infty$. A program execution $c \in \mathcal{C}(s)$ is called $\mathcal{F}$-fair
iff the following holds:

\[ \forall o : (\exists^0 n : E_\mathcal{F}(g(s), c[n], o) \Rightarrow \exists^0 m : O_\mathcal{F}(g(s), c[m], o)). \]

\( P^\mathcal{F}(s) := \{ c \in \mathbb{N}^\infty ; c \in \mathcal{C}(s) \text{ and } c \text{ is } \mathcal{F}\text{-fair} \} \) is the set of all fair program executions of \( s \), \( F^\mathcal{F}(s) := h(P^\mathcal{F}(s)) = \{ w \in \mathbb{N}^\infty ; \exists c \in P^\mathcal{F}(s) : w = h(c) \} \) is the fair semantics of \( s \).

It should be noted that we are only interested in infinite fair program executions in \( P^\mathcal{F}(s) \), as any finite program execution \( c \in \mathcal{C}(s) \) is \( \mathcal{F}\)-fair by definition for any fairness concept. \( P^\mathcal{F}(s) \) is by definition a \( \Pi_3^0 \)-set and \( F^\mathcal{F}(s) \) by definition a \( \Sigma_1^1 \)-set. We now research further restrictions s.t. \( F^\mathcal{F}(s) \) becomes a \( \Pi_3^0 \)-set, too.

In the following definition we define a so-called simple fairness semantics where we require in addition:

- for any \( w \in F^\mathcal{F}(s) \) there are only finitely many program executions, \( l \), of a fixed length, \( n \), with \( h(l) = w[n] \), and we can construct them effectively,
- for any fixed program \( s \in \mathcal{L} \), only finitely many different enabled objects exist in all possible program executions of \( s \),
- the occurring of an object in a program execution can also be detected in \( h(c) \), the meaning of \( c \).

**Definition 3.3.** A simple fairness semantics for a programming language \( \mathcal{L} \) is given by

- an abstract step semantics \( D_c, D_s, \mathcal{C}, \mathcal{I}, h, g \) for \( \mathcal{L} \),
- an abstract fairness-concept \( E_\mathcal{F}, O_\mathcal{F} \) for \( \mathcal{L} \),

s.t. the following holds in addition:

\( (a) \) \( \forall s \in \mathcal{L} \) : \( \forall n \in \mathbb{N} : \forall w \in \mathbb{N}^\infty : h^{-1}_i(w[n]) := \{ l ; l \in \mathcal{C}(s)[n] \text{ and } h(l) = w[n] \} \) is a finite set, and the function

\[ f : \mathbb{N}^\infty \times \mathbb{N}^2 \rightarrow 2^{\mathbb{N}^*} \text{ with } \]

\[ f(w, a, n) := \begin{cases} h^{-1}_i(w[n]) & \text{iff } a = g(s), \\ \emptyset & \text{iff } g^{-1}(a) = \emptyset \end{cases} \]

is recursive,

\( (\beta) \) \( \forall s \in \mathcal{L} : \text{Obj}_s := \{ o \in \mathbb{N} ; \exists c \in \mathcal{C}(s) : \exists n \in \mathbb{N} : E_\mathcal{F}(g(s), c[n], o) \} \) is finite,

\( (\gamma) \exists \hat{O} \subseteq \mathbb{N}^3 \text{ recursive s.t. } \forall s \in \mathcal{L} : \forall c \in \mathcal{C}(s) : \forall o, n : O_\mathcal{F}(g(s), c[n], o) \not\succ \hat{O}(g(s), h(c)[n], o). \)

The requirement \( (\gamma) \) is motivated by the fact that a fair semantics should deal only with objects, \( o \), one is interested in, and those objects should be detectable in the semantics. Let us regard both examples of Section 1 again. The "alternatives", \( o \), for fairness have been the labels \( \{a, b\} \) of the edges of \( A \) (Example 1) and the actions \( \{a, b, c\} \) of \( s_3 \) (Example 2). An occurrence of these alternatives can easily be detected in the semantics (the language of \( A \) and the non-indexed pomsets for \( s_3 \), respectively).
We do not try to convince the reader that these restrictions (a), (β), and (γ) are very 
"natural". On the contrary, they are severe, as they will allow us to reduce a $\Sigma^0_1$ 
to a $\Pi^0_1$-formula. However, there are many examples of concrete semantics of the literature 
that fulfill these requirements, as these requirements say nothing about a visibility of 
enabledness on the semantical level. Let us stress this important aspect again: By (γ) 
we know from the semantics which objects do occur – but we do not know in the 
semantics which objects are enabled.

**Theorem 3.4.** For any simple fairness semantics for $\mathcal{L}$ and for any program $s$ in $\mathcal{L}$ 
the following holds:

- $\mathcal{F}(s)$ is a $(\Pi^0_1 \cap \Sigma^0_3)$-set,
- $\mathcal{F}(s) = \mathcal{F}(\mathcal{F}^\text{fin}(s)) = \mathcal{F}(\mathcal{F}^\text{fin}(s)) = \mathcal{F}(\mathcal{F}^\text{fin}(s)) = \mathcal{F}(\mathcal{F}^\text{fin}(s)) = CS_d$, where 
  $\mathcal{F}^\text{fin}(s) := \{ h(\mathcal{C}(s)[n]) ; n \in \mathbb{N} \}$ – see Definition 3.1 – is the set of finite initial 
  segments of $\mathcal{F}(s)$,
- $w \in \mathcal{F}(s) \rhd w = \lim_{n \to \infty} w[n]$, for some appropriate $\Pi^0_1$-ultra metric $d$.

**Proof.** The second and third statements are an obvious consequence of the first: as 
yany $(\Pi^0_1 \cap \Sigma^0_3)$-set is $\Pi^0_1$ we may apply our characterization theorem with the second 
and third statements as consequences.

For the first statement we prove the following lemma.

**Lemma 3.5.** For any simple fairness semantics (with the above notations) the jbl-
lowing holds:

$$\forall s \in \mathcal{L} : \forall w \in D_s : \forall M \subseteq \text{Obj}_s :$$

$$\exists c \in \mathcal{C}(s) \cap \mathbb{N}^\omega : h(c) = w \quad \text{and} \quad \exists n : \forall n' \geq n : \forall o \in M : \neg E (g(s), c[n'], o)$$

$$\wedge \exists n : \forall n' \geq n : \exists c' \in \mathcal{C}(s)[n'] : h(c') = w[n'] \quad \text{and}$$

$$\forall k : n \leq k \leq n' : \forall o \in M : \neg E (g(s), c'[k], o).$$

**Proof.** 

"\text{\textgreater}" Obvious, set $c' := c[n']$. "\textless" We use the following version of König's 
lemma.

**König's Lemma.** Let $E$ be some set and $R \subseteq E \times E$ some relation with the property :

$$\forall j \in \mathbb{N} : \exists E_j \subseteq E \quad \text{s.t.}$$

- $\emptyset \neq E_j$, $E_j$ is finite
- $\forall j : \forall y \in E_{j+1} : \exists x \in E_j : (x, y) \in R$,

then a sequence $(x_j)_{j \in \mathbb{N}}$ s.t. $\forall j : x_j \in E_j$ and $(x_j, x_{j+1}) \in R$ exists.

We apply König's Lemma as follows:

$$E_j := \{ c \in \mathcal{C}(s)[n + j] ; h(c) = w[n + j] \quad \text{and}$$

$$\forall k : n \leq k \leq n + j : \forall o \in M : \neg E (g(s), c[k], o) \},$$
\[ E := \bigcup_{j \in \mathbb{N}} E_j, \]

\[ (c_1, c_2) \in R : \succ c_1 < c_2 \quad (c_1 \text{ is a prefix of } c_2). \]

The following holds:

- \( E_j \neq \emptyset \), by premise,
- \( E_j \) is finite, by premise \((x)\) of simplicity,
- for \( y \in E_{j+1} \) we know \( y \in \mathcal{G}(s)[n+j+1], \ h(y) = w[n+j+1] \) and \( \forall k : n \leq k \leq n + j + 1 : \forall o \in M : \neg E_{\not F}(g(s), y[k], o) \). Thus:

\[ y[n+j] \in \mathcal{G}(s)[n+j] \quad \text{and} \quad h(y[n+j]) = w[n+j] \]

and

\[ \forall k : n \leq k \leq n + j : \forall o \in M : \neg E_{\not F}(g(s), y[k], o), \]

i.e.:

\[ \forall j : \forall y \in E_{j+1} : \exists x \in E_j : x < y. \]

Thus, by König’s lemma a sequence \((c_j)_{j \in \mathbb{N}}\) in \( E \) s.t. \( \forall j : c_j \in E_j \) and \( c_j < c_{j+1} \) exists.

As \( \mathcal{G}(s) \) is closed under \( \delta \), \( c := \lim_{j \to \omega} c_j \) exists in \( \mathcal{G}(s) \), and, obviously, \( c \in \mathbb{N}^\omega \).

Also, \( \forall n : c[n] = c_n[n] \), thus \( h(c[n]) = (w[n]) \), thus \( h(c) = w \). This proves \("<"\).

Note, only property \((x)\) of simplicity is required for this lemma.

We now continue the proof of the theorem.

Let \( s \in \mathcal{L} \) be fixed.

\[ w \in \mathcal{F}(s) \]

\[ \succ \exists c \in \mathcal{G}(s) : h(c) = w \]

\[ \succ \exists c \in \mathcal{G}(s) \cap \mathbb{N}^\omega : h(c) = w \text{ and } c \text{ is } \mathcal{F}-\text{fair} \]

\[ \succ \exists c \in \mathcal{G}(s) \cap \mathbb{N}^\omega : h(c) = w \text{ and} \]

\[ \forall o : (\exists^\omega n : E_{\not F}(g(s), c[n], o) \Rightarrow \exists^\omega m : O_{\not F}(g(s), c[m], o)) \]

\[ \succ \exists c \in \mathcal{G}(s) \cap \mathbb{N}^\omega : h(c) = w \text{ and} \]

\[ \forall o \in \text{Obj}_s : (\exists^\omega n : E_{\not F}(g(s), c[n], o) \Rightarrow \exists^\omega m : O_{\not F}(g(s), c[m], o)) \]

("\(>\)\) is trivial, and \("<\) is obvious, as for all \( o \notin \text{Obj}_s \)

\[ \neg E_{\not F}(g(s), c[n], o) \text{ holds } \forall n, \text{ i.e. the implication } \Rightarrow \text{ is true} \]

\[ \succ \exists c \in \mathcal{G}(s) \cap \mathbb{N}^\omega : h(c) = w \text{ and } \exists M \subseteq \text{Obj}_s : \]

\[ \forall o \in M : \exists n, \forall n' \geq n : \neg E_{\not F}(g(s), c[n'], o) \]
and $\forall o \in \text{Obj}_s \setminus M : \exists m' \geq m : O_\varphi(g(s), c[m'], o)$

(“>”: set $M := \{ o \in \text{Obj}_s : \exists n' > n : -E_\varphi(g(s), c[n'], o) \}$

“<”: the implication $\Rightarrow$ holds $\forall o \in \text{Obj}_s$, as for $o \in M$ the premise is false and for $o \notin M$ the conclusion is true)

$x \exists \exists \in \mathcal{C}(s) \cap \mathbb{N}^\omega : h(c) = w$ and $\exists M \subseteq \text{Obj}_s$

$\exists n : \forall n' \geq n : \forall o \in M : -E_\varphi(g(s), c[n'], o)$

and $\forall o \in \text{Obj}_s \setminus M : \forall m : \exists m' \geq m : O_\varphi(g(s), c[m'], o)$

(“<” is trivial, and “>” is obvious, as $M$ is a finite set as $\text{Obj}_s$ is finite by (β).)

$x \exists \exists \in \mathcal{C}(s) \setminus \mathbb{N}^\omega : h(c) = w$ and $\exists M \subseteq \text{Obj}_s$

$((\exists c \in \mathcal{C}(s) \cap \mathbb{N}^\omega : h(c) = w \land \\ \exists n : \forall n' \geq n : \forall o \in M : -E_\varphi(g(s), c[n'], o)))$

(Exchange $\exists M, \exists c$, and by (γ) of simplicity)

$x \exists \exists \in \mathcal{C}(s) \setminus \mathbb{N}^\omega : h(c) = w$ and $\exists M \subseteq \text{Obj}_s$

$((\exists n : \forall n' \geq n : \exists c' \in \mathcal{C}(s)[n'] : h(c') = w[n'] \land \\ \forall k : n \leq k \leq n' : \forall o \in M : -E_\varphi(g(s), c'[k], o)))$

(Exchange $\exists M, \exists c$, and by (γ) of simplicity)

We now compute the logical complexity of the last formula.

As $\text{Obj}_s$ is a fixed finite set, “$\exists M \subseteq \text{Obj}_s$” must not be counted (we may replace this by a finite disjunction). The sub-formula

$\exists c' \in \mathcal{C}(s)[n'] : h(c') = w[n'] \land \forall k : n \leq k \leq n' : \forall o \in M : -E_\varphi(g(s), c'[k], o)$

is recursive, as $M$ is a finite set, “$\forall k$” is bounded by “$n'$”, $h_s^{-1}(w[n])$ is a finite set, and $f$ is recursive (see assumption (α)), s.t. “$\exists c' \in \mathcal{C}(s)[n'] : h(c') = w[n']$” is decidable.

Thus the first part of the formula is a $\exists \forall$-formula in $\Sigma_2^0$.

As $\text{Obj}_s \setminus M$ is again a finite set we do not count “$\forall o \in \text{Obj}_s \setminus M$”. Thus, the second part is a $\forall \exists$-formula in $\Pi_2^0$. A conjunction of a $\Sigma_2^0$ and a $\Pi_2^0$-formula is in $\Sigma_3^0 \cap \Pi_3^0$.

This proves the theorem.
4. Applications

In this section we present several applications of the previous theorem to clarify its usability and limitation. We shall present a pomset semantics for Petri nets, two different concepts of fair paths in infinite graphs, and a brief excursion on random real numbers.

**Petri net pomset semantics.** Here, \( \mathcal{L} \) will be the class of Petri nets, and \( s \in \mathcal{L} \) thus is a Petri net \( s = (P, T, F, \lambda, m_0) \) over some finite set \( \Sigma \) of actions. \( P, T, \) and \( F \) are finite sets of places, transitions, and arc-connections, respectively. \( \lambda : T \rightarrow \Sigma \) adds to every transition some action as a label, \( m_0 \) is the initial marking. As the set \( \mathcal{C}(s) \) (of "program executions" of \( s \)) we choose the set of all processes of \( s \), see, e.g., [1] for the definitions. The semantics \( \mathcal{S}(s) \) shall be the set of all possible pomsets of \( s \). A pomset is an abstraction of a process where all places and transitions are dropped. Only the labels (actions) of the transitions and their ordering relation are left. For a definition of pomsets and its natural topology we refer, e.g., to [5]. Fig. 4 presents an example of a simple Petri net with two of its possible processes and their pomsets. Any cut (cf. [1]) in a process \( p \) uniquely defines a marking for the underlying Petri net. We say that a transition \( t \in T \) is enabled in \( p \) if it is enabled in some cut of \( p \). A process \( p \) is called fair if all infinitely often enabled transitions in \( p \) have to occur in \( p \) infinitely often. This defines \( \mathcal{C}^F(s) \).

We are interested in the fair pomset semantics of \( s \), \( \mathcal{S}^F(s) \), consisting of all pomsets which are abstractions of the fair processes of \( s \). Obviously, a cut in a pomset will not define a marking. Thus, enabledness is invisible in pomsets. However, one verifies easily that all requirements for Theorem 3.4 are fulfilled. Let \( \mathcal{P}^\infty(\mathcal{M}(\Sigma)) \) denote the set of all finite pomsets over \( \Sigma \). Thus, by our theorem we conclude the existence of an ultra metric \( d \) (that refines the natural metric on pomsets, see [5]) s.t. \( \mathcal{S}^F(s) = CP^d(\mathcal{P}^\infty(\mathcal{M}(\Sigma))) \), etc. However, we do not claim that our theorem yields a "simple description" for this metric.

**Infinite graphs.** Of course, the previous example also holds true for infinite Petri nets (with some recursive description s.t. assumption (\( \alpha \)) for "simple fairness" holds). We will study a more abstract example now. Let \( G \) denote some infinite arc-labelled graph with an initial node, \( v_0 \), with a "recursive description" (e.g. some Turing machine that generates \( G \) in a breadth-first-search, starting in \( v_0 \)), s.t. assumption (\( \alpha \)) shall hold. We regard \( G \) as \( \mathcal{C}(s) \), the complete branching-time description of all "computation executions" of some program \( s \). A simple computation sequence of \( s \) shall be any path in \( G \) (starting in \( v_0 \)). Let \( \Sigma := \Sigma_1 \cup \Sigma_2 \) be the finite set of arc-labels of \( G \). We are interested in \( \Sigma_1 \)-abstractions of \( G \), i.e. in those words \( w \in \Sigma_1^\infty \) that are labellings of paths of \( G \) where all \( \Sigma_2 \)-labels are mapped to some letters in \( \Sigma_1 \). As a (rather strange) kind of fairness we are interested in those paths in \( G \) only that use \( \Sigma_1 \)-labels almost always, if possible. I.e., whenever \( p \) may choose to follow a \( \Sigma_1 \)-arc or a \( \Sigma_2 \)-arc it will choose a \( \Sigma_2 \)-arc only finitely often. Thus, we are interested in the following "fairness" semantics

\[
\mathcal{G}^F(s) := \{ w \in \Sigma_1^\omega ; \ \exists \ \text{infinite path} \ p \in G \text{ with } w = \lambda(p) \ \text{s.t.} \ \exists k : \forall n \geq 0 : (p(n) \text{ can be prolonged with a } \\
\Sigma_1 \text{-arc implies that } p(n + 1) \text{ is a } \Sigma_1 \text{-arc}) \},
\]
where $\lambda$ is the abstraction that maps paths to $\Sigma_1$-label sequences. This may also be expressed as: $(\forall k : \exists n \geq 0 : p(n) \text{ can be prolonged with a } \Sigma_1\text{-arc and } p(n + 1) \text{ uses a } \Sigma_2\text{-arc}) \Rightarrow \forall k : \exists n \geq 0 : false$. Thus, we get a fairness concept with a trivial occurrence predicate “false”. Again, our Theorem 3.4 is applicable and $F(s) = CP^d(\Sigma^*)$, etc., for an appropriate ultra metric $d$.

As a second example of an (also rather strange) fairness notion we regard only those paths in $G$ that are primitive recursive (as a function from $\mathbb{N}$ into the set of edges).
Thus,

$$\mathcal{F}(s) := \{ w \in \Sigma^\omega; \ \exists \text{ infinite path } p \text{ in } G \text{ s.t.} \ w = \lambda(p) \text{ and } p \text{ is primitive recursive}.\}$$

However, primitive recursiveness is a fairness concept applicable for our theorem: Let $$\{ \varphi_k \}_{k \in \mathbb{N}}$$ be an enumeration of all primitive recursive functions over $$\mathbb{N}$$. Thus, $$p$$ is primitive recursive $$\exists \ \exists n : \varphi_k(n) = p(n)$$. As any $$\Sigma_2$$-formula possesses an equivalent $$\forall$$-formula (see, e.g., [9, Theorem XVII, Section 14.8]), we continue with an appropriate recursive predicate $$R : p$$ is primitive recursive $$\exists \exists k : \forall n \geq k : R(k, p(n))$$. Thus, $$p$$ is primitive recursive $$\exists (\forall k : \exists n \geq k : \neg R(k, p(n))) > \forall k : \exists n \geq k : false.$$ I.e., all requirements of our theorem are fulfilled.

Unfortunately, Theorem 3.4 is not applicable if the abstraction $$\lambda$$ maps some labels of paths into the empty word, or if $$\varepsilon$$-labels are allowed in the case of petrinets, as in these cases the requirement $$\lambda(c[n]) = \lambda(c)[n]$$ of Definition 3.1 is violated.

**Random reals.** We regard a real $$r \in [0, 1]$$ as a function $$r : \mathbb{N} \rightarrow D, D = \{0, 1, \ldots, 9\}$$. For $$p \in D^\omega, q \in D^k$$ the number of occurrences of $$p$$ in $$q$$ is given by $$\#(p, q) := \#\{w \in D^*; \exists w' : q = wpw'\}$$. $$E(p, k) := 10^{-k} \sum_{q \in D^k} \#(p, q)$$ is the expected number of occurrences of $$p$$ in a sequence of length $$k$$. We call a real $$r \in [0, 1]$$ random if $$r$$ is not recursive and Borel-normal, i.e. $$\forall p \in D^\omega(\not\subseteq \mathbb{N}) : \lim_{k \rightarrow \omega} \#(p, r[k])/E(p, k) = 1$$ is there a metric $$d$$ s.t. $$r$$ is random iff $$r = \lim_{n \rightarrow \omega} r[n]$$? Here, our Theorem 3.4 does not apply, but we can use the characterization theorem of Section 2 itself in the following way. Let $$G_D$$ be the graph consisting of one node, $$v_0$$, and ten arcs from $$v_0$$ to $$v_0$$ labelled from 0 to 9. Thus, any real in $$[0, 1]$$ is an infinite path in $$G_D$$, and vice versa. Thus, the characterization theorem applies and proves the existence of such a metric, as randomness is a $$\Pi^0_3$$-predicate:

$$r$$ is not recursive $$\exists \forall k : \exists n : \forall t : \text{ the } k\text{-th Turing machine with input } n \text{ stops after } t \text{ steps with a result different from } r(n) \text{ or does not stop after } t \text{ steps } (\Pi^0_3),$$

$$r$$ is Borel-normal $$\exists \forall p : \forall K : \exists N_0 ; \forall k \geq N_0 : \left|\frac{\#(p, r[k])}{E(p, k)} - 1\right| < \frac{1}{K} (\Pi^0_3).$$

Thus, the answer is yes, such a metric exists.

A real $$r \in [0, 1]$$ is called disjunctive (see, e.g., [7, 2]) iff $$\forall d \in D^\omega : \exists n : r(n + |d|) \ldots r(n + 1) r(n) = d$$; i.e. $$d$$ appears as a consecutive sequence in the series of digits of $$r$$. Thus, disjunctivity is a $$\Pi^0_3$$-predicate. In the case of $$\Pi^0_2$$-predicates the characterization theorem holds in a stronger version with a recursive metric $$d$$ (i.e. the predicate $$d(\cdot, \cdot) < 1/\cdot$$ is recursive), see [8]. Thus a real $$r \in [0, 1]$$ is disjunctive iff $$r = \lim_{n \rightarrow \omega} r[n]$$ holds for an appropriate recursive metric $$d$$, etc. However, in this disjunctivity example the recursive metric can easily be constructed without the characterization theorem.
References