Concurrency, Fairness and Logical Complexity

Marta Kwiatkowska
mzk@mcs.le.ac.uk
Department of Mathematics and Computer Science
University of Leicester, Leicester LE1 7RH, United Kingdom
also visiting
Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ, United Kingdom

Abstract
We consider a connection between fairness and Π^0_3 sets of functions made recently by Darondeau, Nolte, Priese and Yoccoz, and extend their work to cater for effective asynchronous transition systems with concurrency structure represented by a Mazurkiewicz independency relation.

1 Introduction

Fairness is usually expressed in terms of logical statements of substantial complexity. A typical notion of (strong process) fairness is defined as follows:

$$\forall f\,\forall P.\ P\ \textit{infinitely often enabled in}\ f \Rightarrow P\ \textit{proceeds infinitely often in}\ f$$

where P denotes a process and f is a computation. Weak (process) fairness is obtained by replacing the premiss of the implication with P *almost always enabled in* f (that is, enabled continuously from some point on), giving rise to a weaker statement that each process P proceeds infinitely often unless it terminates or is disabled infinitely often. A variety of fairness notions of different "strength" arises if one replaces the premiss with more complex statements such as *there exists infinitely many points in* f *at which all processes are equally often enabled* (probabilistic fairness). One may also wish to take a different notion of a component, such as an action or direction in a non-deterministic guarded command language, instead of the process, thus giving rise to fairness notions of different "granularity". For an overview of fairness issues and various proof systems see *e.g.* [14, 9].

Fairness is known to lead to an increase in the complexity of methods for dealing with it. It is an easy observation that fairness cannot be expressed unless infinite computations are allowed. Moreover, fairness properties (viewed as sets of computations) are not closed with respect to the usual topologies employed in denotational semantics (*i.e.* Scott and metric topologies). Park [21] exhibited the failure of continuity of fairmerge (a relational view of fairness), and proposed monotonicity and transfinite fixpoint induction instead (a similar point was also elaborated on by Manna and Pnueli [15]). Plotkin constructs a powerdomain for countable non-determinism [22]. Apt and Plotkin [3] showed that proving termination under fairness is Π^1_1, and, what is more, it is Π^1_1 complete. Informally, Apt and Plotkin's results are not surprising if they are understood as the statement:

$$\forall f.\ \textit{if } f \textit{ is fair then it is finite}$$

or, equivalently, *there is no infinite fair computation*. This involves universal quantification over possibly infinite computations, and thus does not belong to the arithmetic hierarchy (Π^0), in which only quantification over numbers is allowed, but to the analytical hierarchy (Π^1), where it is possible to quantify over functions, or, equivalently, over infinite sequences. Thus, on the one hand, fairness is characterized in terms of highly undecidable statements, while, on the other hand, it can be guaranteed by relatively simple algorithms (for instance, a busy waiting mutual exclusion algorithm guarantees weak fairness, while a scheduler with a queueing mechanism will guarantee strong fairness). The intuition (already observed by Abramsky [1]) is that when fairness is considered as a statement of some abstract domain of the computations of *all* programs then it is Π^1_1. However, if it is considered as a statement about the computations of a *particular* program, there may be reasonable assumptions one can make to place it in the arithmetic hierarchy.

We find it useful to follow the intuition originally introduced by Harel in [10], where recursive transformations on trees were considered. The main result showed there was the proof of the existence of a one-one effective transformation between well-founded countably-branching trees (which can be viewed as inductive proofs) and "φ-abiding" finitely-branching infinite trees (which can be viewed as program unfoldings). φ is a formula in an infinitary propositional modal logic, and a tree is said to be φ-abiding iff it has no infinite paths satisfying φ. The claim is that this transformation gives rise to a generic proof system for (quoting Harel [10]) "almost any conceivable notion of fairness".

While Harel's notion of fairness may reach arbitrary levels of the arithmetic hierarchy, in their recent work Darondeau, Nolte, Priese and Yoccoz [7] showed a connection between fairness and Π^0_3 sets of functions. The authors proposed a notion of a recursive (recursively enumerable) transition system, in which (under certain strong assumptions, which we shall explain in the remainder of the paper) every fairness property (viewed extensionally as a set of infinite computations that are fair in some sense) can be identified with a Π^0_3 subset of ω^ω through a reduction to a normal form. Thanks to a well-known recursion theory correspondence, this characterization gives rise to a topological characterization of fairness in terms of $\mathbf{\Pi}^0_3$ ($F_{\sigma\delta}$ in the Borel hierarchy) sets.

We propose to take the work of [7] one step further by applying it to a non-interleaving semantic model. As exhibited in some earlier papers of the author [12], non-interleaving gives rise to the phenomenon of confusion, which is not present in standard interleaving descriptions of behaviour. We define a notion of a recursive (recursively enumerable) *asynchronous transition system* [25, 12] by enhancing it with concurrency information given in the form of a Mazurkiewicz independency relation [17] (*i.e.* an irreflexive and symmetric relation on transitions), and show that our definition is indeed reasonable since such systems are determined by existing languages (*e.g.* Condition-Event nets, CCS). We then propose a suitable notion of a concurrent computation of such a system, which requires a Π^0_2 preorder on infinite computations. Finally, we characterize the fair sets of infinite computations in terms of an adjustment of the normal form.

2 Fairness, Logical Complexity and Recursion Theory

2.1 Notation

Let A be a set. The set of finite (resp. infinite) sequences over A is denoted by $A^\star$ (resp. A^ω); A^∞ is taken to mean $A^\star \cup A^\omega$. The prefix order on A^∞ is denoted by $\leq$.

We use ω to denote natural numbers, ω^ω the set of functions on natural numbers, and Ω denotes $\omega \cup \omega^\omega$. It is known that there exists a recursive one-one coding of sequences $\omega^\star$ of natural numbers to natural numbers ω [23], which gives rise to a bijection between sets ω^∞ and Ω. The function $[k] : \omega^\infty \longrightarrow \omega^\star$ which truncates the argument string to the string of length k (or to the length of the argument string if its length is less than k), induces a function $[k]' : \Omega \longrightarrow \omega$, which yields the coding of the truncation. We overload the notation and refer to this function by $[k]$. The application of this function to the argument f is in the postfix notation $f[k]$.

We shall use letters such as x, y to range over ω, f, g over ω^ω, α, β over Ω, and σ, τ over ω^∞.

We say a set (relation) $X \subseteq \omega^m$ is *recursive* iff there is an algorithm for testing its membership, *i.e.* an algorithm which terminates with answer "yes" or "no" depending on whether the element is in the set or not (formally, the characteristic function χ_X is recursive). A set (relation) $X \subseteq \omega^m$ is *recursively enumerable* if there exists an algorithm for enumerating its elements (formally, X is the range of some recursive function).

The *arithmetic* hierarchy (see *e.g.* [23], page 301) of relations on ω^m is defined as follows. Let Δ^0_1 be the recursive relations. Define inductively the family of relations:

$$\begin{array}{rcl}
\Sigma^0_0 = \Pi^0_0 & = & \Delta^0_1 \\
\Pi^0_{n+1} & = & \forall \Sigma^0_n \\
\Sigma^0_{n+1} & = & \exists \Pi^0_n \\
\Delta^0_{n+1} & = & \Pi^0_{n+1} \cap \Sigma^0_{n+1}
\end{array}$$

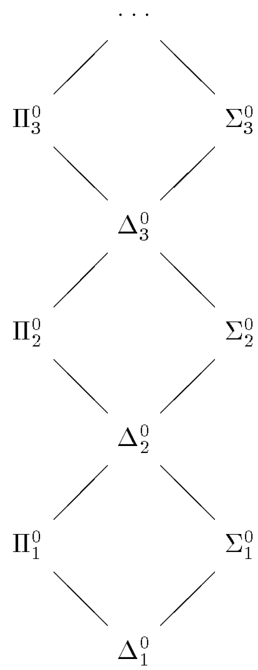
where $\forall \Sigma^0_n$ (resp. $\exists \Pi^0_n$) denotes the class of relations S such that for some recursive relation R belonging to Σ^0_{n-1} (resp. Π^0_{n-1}):

$$< x_1, ..., x_m > \in S \quad \Longleftrightarrow \quad \forall i R(i, x_1, ..., x_m).$$

The family is shown in Figure 1. Note that the recursively enumerable relations are in Σ^0_1. The arithmetic hierarchy is defined similarly for relations in arbitrary products $\omega^m \times (\omega^\omega)^n$.

The arithmetic relations may also be defined (from recursive relations) by logical formulas which include logical connectives, as well as the quantifiers. These formulas may always be transformed into equivalent ones with a single sequence of leading quantifiers (as required by the definition of the hierarchy) by means of the Tarski–Kuratowski algorithm ([23], page 307).

Figure 1: The arithmetic hierarchy



The usual metric on Ω is the ultra-metric δ defined by

$$\delta(\alpha, \beta) \quad = \quad \begin{cases} 0 & \text{if } \alpha = \beta \\ \frac{1}{1+\alpha\wedge\beta} & \text{otherwise} \end{cases}$$

where $\alpha\wedge\beta = \sup\{k \mid \alpha[k] = \beta[k]\}$, and $\sup\{\} = 0$. As in [7], we call this metric *recursive* (since the relation $\delta(\alpha, \beta) < \frac{1}{n}$ is recursive in (α, β, n)). Π^0_1 metrics are defined similarly. The topology that arises from δ is often called the Baire space; it is zero-dimensional and non-compact. The *derived* set $Der_d(A)$ of the set $A \subseteq X$ in a metric space (X, d) is the set of all $x \in X$ such that for any $n \in \omega$ the open ball with the centre x and radius $\frac{1}{n}$ contains at least one element $a \in A$ distinct from x. In topological terms, $x \in Der_d(A)$ iff $x \in Cl(A \setminus \{x\})$ [11]. It is a straightforward observation that ω^ω is Π^0_1 and is the derived set of ω with respect to the metric δ, since $Der_\delta(\omega) = \left\{f \mid \forall n\, \exists k.\delta(f, f[k]) < \frac{1}{n}\right\}$. We shall see later on that each fair set of functions (in some sense) will be characterized as the derived set of ω for a suitable refinement of the usual metric δ.

2.2 Recursive Transition Systems

We find it convenient to work with the notion of an (unlabelled, rooted) transition system $S = (Q, T, \mathrm{s}, \mathrm{e}, q_0)$, where Q is a set of states, T is a set of transitions, and $\mathrm{s}, \mathrm{e} : T \longrightarrow Q$ are maps yielding the source and end state of each transition, and $q_0 \in Q$ is the initial state. Given $q \in Q, t \in T$, we shall write $q \xrightarrow{t}$ if $\mathrm{s}(t) = q$ and $\xrightarrow{t} q$ if $\mathrm{e}(t) = q$. A transition system $S = (Q, T, \mathrm{s}, \mathrm{e}, q_0)$ is *recursive* (resp. *recursively enumerable*) iff all its components are recursive (resp. Σ^0_1). As a simplification, we shall only consider transition systems which have no sink states, that is, states p such that $\forall t \in T.\mathrm{s}(t) \neq p$. A transition system with sink states can easily be transformed to one without by adding "loops" at sink states. This transformation does not have impact on fairness.

A *computation* of a (recursive or recursively enumerable) transition system S is a (finite or infinite) sequence of transitions $(t_i)_{i \in I} \in T^\infty$, $I \subseteq \omega$, such that $\mathrm{s}(t_0) = q$ and $\mathrm{s}(t_i) = \mathrm{e}(t_{i-1})$ for $i > 0$. Clearly, under the assumption of no sink states, a computation of a system S is partial iff it is finite, so we can easily separate the approximations from the "completed" computations. Note that every finite computation can be encoded as an element of ω (this includes T), while every infinite computation can be viewed as a function $f \in \omega^\omega$ (mapping position index to the coding of a transition). We shall abuse the notation by identifying the two representations of computations, *i.e.* an infinite sequence over ω and and a function ω^ω. We use $\mathit{Fin}(S), \mathit{Inf}(S)$ to denote all finite and infinite computations of S respectively. Note that if S is recursive then $\mathit{Inf}(S) = \{f \mid \forall k.f[k] \in \mathit{Fin}(S)\}$ is Π^0_1 (Π^0_2 for r.e. systems [7]). $\mathit{Inf}(S)$ is the derived set of $\mathit{Fin}(S)$ with respect to the Baire metric δ.

2.3 Normal Form for Fairness

A typical notion of (strong) fairness (of a computation f) would be written in the form:

$$\forall P. [\exists^\omega k\, \mathrm{E}(f, k, P) \Rightarrow \exists^\omega k\, \mathrm{T}(f, k, P)]$$

where $\exists^\omega k$, $\forall^\omega k$ are defined to be $\forall i\, \exists k \geq i$ (exists infinitely many) and $\exists i\, \forall k \geq i$ (for all but finitely many) respectively, P denotes a process, and E, T are (recursive) predicates which evaluate to true iff the process P is enabled (resp. taken) at the kth stage of the computation f. This can be shown to be Π^0_3 (with a function argument) by means of a calculation using the Tarski-Kuratowski algorithm. In the following we show how the above form may be reduced to a simpler form by putting additional conditions on the transition system.

If we assume that each process is identified with a single transition (an "agent"), then the predicate T reduces to testing for $f(k) = P$. We can simplify further, by requiring that each P "dies" as soon as it is taken in the computation, that is

$$\exists k.\mathrm{T}(f, k, P) \quad \Rightarrow \quad \forall j > k.\neg\mathrm{E}(f, j, P).$$

This in fact imposes an infinite number of agents, and allows us to rewrite the above in the (equivalent under these assumptions) form:

$$\forall P \forall^\omega k.\neg\mathrm{E}(f, k, P)$$

Since we are dealing with computations produced by the reduction process for a particular machine, we can (uniformly) simplify further to:

$$\forall P \forall^\omega k.\neg\mathrm{E}(f[k], P) \tag{1}$$

where we shall treat the derived statement as a *filter* on computations:

$$\{f \mid \forall P \forall^{\omega} k. \neg \mathrm{E}(f[k], P)\}.$$

Now observe that since $\mathrm{E}(f[k], P)$ is a recursive relation (and it is in $\omega \times \omega$ if $f[k]$ and P are identified with their codings), we have obtained a Π_3^0 characterization (*i.e.* of the form $\forall i \, \exists j \, \forall k. S(f, i, j, k)$ where S is a recursive relation) of the strong (process) fairness we started with, under the simplifying assumptions as outlined above. The predicate E is understood as defining if the agent coded by i is enabled at the truncation of f of length k. The following examples illustrate its meaning.

Example 2.1 Consider the tree unfolding of the process $\mathrm{fix}X.aX + bX$, then the predicate E for the computation a^{ω} is summarised in the table:

	a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_4	$\cdots$
0	$\times$	$\times$							
1			$\times$	$\times$					
2					$\times$	$\times$			
3							$\times$	$\times$	
4				$\cdots$					

where a_i, b_i symbolically denote the codes of the corresponding transitions and $0, 1, 2, \cdots$ denote stages of the computation.

Example 2.2 Consider the tree unfolding of the process $\mathrm{fix}X.aX \mid \mathrm{fix}X.bX$, then the predicate E for the computation a^{ω} is summarised in the table:

	a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_4	$\cdots$
0	$\times$	$\times$							
1		$\times$	$\times$						
2		$\times$			$\times$				
3		$\times$					$\times$		
4		$\times$		$\cdots$					

where a_i, b_i symbolically denote the codes of the corresponding transitions and $0, 1, 2, \cdots$ denote stages of the computation. On the other hand, the predicate E for the computation $(ab)^{\omega}$ is given by:

	a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_4	$\cdots$
0	$\times$	$\times$							
1			$\times$	$\times$					
2					$\times$	$\times$			
3							$\times$	$\times$	
4				$\cdots$					

The interest in the form (1) for fairness is motivated by the fact that it was shown in [7] that *any* Π_3^0 set of functions can be reduced to such form, thus giving (1) the status of a normal form. The claim is that, since extreme fairness [9] is Π_3^0, and extreme fairness implies many known notions of fairness, this normal form is in effect a generic notion of effective fairness. (Strong) extreme fairness [9] can be informally defined as follows. Let $\Gamma = \left\{\varphi^0, \varphi^1, ...\right\}$ be any countable collection of state properties (*e.g.* first order definable predicates); then a computation f is *extremely fair* iff for all $\varphi \in \Gamma, i \in \omega$, i infinitely often enabled in states satisfying φ implies i infinitely often taken in states satisfying φ.

The following result links the different concepts introduced so far – recursive transition systems, derived sets in metric spaces, and the normal form for fairness. It shows that there are three equivalent ways of viewing fairness. Recall that the derived set of the finite computations of some recursive or r.e. transition system with respect to the usual (recursive) Baire metric is the set of *all* infinite computations. This includes the fair, as well as the unfair ones. The significance of the result below is that it provides a Π^0_3 method for excluding the unfair infinite sequences from the derived set of the finite computations. More precisely, the set of fair computations coincides with the derived set of the finite computations with respect to a metric, which *refines* the Baire metric.

Proposition 2.3 [7] For any rooted recursively enumerable transition system S the following are equivalent:

1. F is a Π^0_3 subset of $Inf(S)$.
2. F is the derived set of $Fin(S)$ for some Π^0_1 ultra-metric refining the usual Baire metric δ.
3. $F = \{f \mid \forall i\, \forall^\omega k. \neg \mathrm{E}(f[k], i)\}$ for some recursive predicate $E \subseteq \omega \times \omega$.

The refined metric is constructed uniformly from the enabledness predicate E as follows. Firstly, define the predicate completed(i, j), meaning the agent i has finished by stage j of the computation α, by:

$$\text{completed}(i,j) \iff \forall k \geq j. \neg \mathrm{E}(\alpha[k], i).$$

Then the distance between computations (finite or infinite) is given by:

$$\mathrm{d}(\alpha,\beta) = \begin{cases} 0 & \text{if } \alpha = \beta \\ \dfrac{1}{\max\left\{\frac{1}{1+\eta(\alpha,\alpha\wedge\beta)}, \frac{1}{1+\eta(\beta,\alpha\wedge\beta)}\right\}} & \text{otherwise} \end{cases}$$

where $\eta(\alpha, j) = \max\{i \leq j \mid \forall i' \leq i \,.\, \text{completed}(i', j)\}$. Intuitively speaking, the greater the code of the last completed agent, the smaller the distance between the computations, where the "last completed agent" $\eta(\alpha, j)$ of the computation α at stage j is the agent i, all of whose predecessors (*i.e.* with codes below i) are never enabled beyond j, meaning that the computation either terminated before the agents became enabled, or they have been taken. Thus, f is E-fair iff $\lim_{j\to\omega} \eta(f, j) = \omega$.

Note that we have a *different* metric (*i.e.* possibly a different topology!) for each fairness notion.

Example 2.4 In Example 2.2 the computation $(ab)^\omega$ is fair, while a^ω is not fair because $\lim_{j\to\omega} \eta((ab)^\omega, j) = \omega$, while $\lim_{j\to\omega} \eta(a^\omega, j) = a_1$.

3 Recursive Asynchronous Transition Systems

The interesting question arises whether the theory described in the previous sections is capable of handling fairness in a non-interleaving model for concurrency. It has already been observed [20] that since pomsets can be encoded as functions, the results of the previous sections can be applied in any setting which uses a pomset representation of runs. However, we believe that this approach is indirect, because it depends on the coding of a representation of the behaviour, instead of being a direct outcome of a reduction strategy. Therefore, we propose a generalization of effective transition systems with additional concurrency structure. This way, the outcome of a reduction strategy is a sequence, but certain sequences will be rendered "equivalent", in the sense that they only contain inessential differences. Inessential differences are those which arise from permutations of consecutive, concurrent transitions.

We propose to view recursive transition systems as the *outcome of some reduction*, or derivation, following a set of SOS rules. The codes (of states and transitions) are assigned *as the reduction proceeds*, so that the coding is fixed by the reduction strategy. This approach has the advantage of leading to results, which will be preserved through restrictions of transition systems to subsystems accessible from some initial state. For an arbitrary recursive coding, such a restriction does not lead to a recursive coding of a subsystem [6].

To illustrate our idea, consider, for example, CCS [19] with the one-step transition relation $\xrightarrow{a}$, and consider the process $\mathrm{fix}X.aX + bX$. Note that each a-move generates a "new" copy of both a and b by the rule:

$$\frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'}$$

Since P and Q disappear, so do the codes generated at the previous step of reduction. In contrast, b-moves in the process $\mathrm{fix}X.aX \mid \mathrm{fix}X.bX$ do not give rise to a "new" copy of a because of the rule:

$$\frac{P \xrightarrow{a} P'}{P \mid Q \xrightarrow{a} P' \mid Q}$$

where Q does not disappear, so neither does the code of the a-move.

We now proceed with the definition of an effective asynchronous transition system. Define an *independency* relation [18] to be any irreflexive and symmetric relation $\iota \subseteq T \times T$. The purpose of this relation is to establish the potential for concurrency within the system. Any two transitions that are independent can happen concurrently; for example, they can be executed by separate processors. Note that a transition cannot be concurrent with itself, and concurrency is symmetric. Also note that this defines a *global* relation; when considering languages such as CCS additional labelling on actions has to be used to handle correctly cases such as $a + a$ and $ab(a \mid b)$. This is not a problem for us, since we can assume that these labels would be produced by the reduction process, and hence will be unique.

We shall also require the notion of a *residual* operation (notation $/$) [5] on transitions. As each transition contains information about start and end states, and start and end states of consecutive transitions in a computation must agree, we cannot just permute two independent transitions because we would not necessarily obtain a valid computation. Instead, we require that a transition commutes with the residual of another transition.

As introduced in [25, 12], we define an *asynchronous transition system* (abbrev. ATS) to be the structure $S = (Q, T, \mathrm{s}, \mathrm{e}, \iota, /, q_0)$ such that $S = (Q, T, \mathrm{s}, \mathrm{e}, q_0)$ is a rooted transition system, $\iota \subseteq T \times T$ is an independency relation, $/ : T \times T \longrightarrow T$ is an operation such that t_1 / t_2 is defined whenever $t_1 \ \iota \ t_2$, subject to the following axioms being satisfied:

$$\begin{array}{ll} (\text{Forw-}\Diamond) & \forall q \in Q.(q \xrightarrow{t} p, q \xrightarrow{t'} p', t \ \iota \ t' \Rightarrow \exists! q' \in Q. p \xrightarrow{t'/t} q', p' \xrightarrow{t/t'} q') \\ (\text{Perm-}\Diamond) & \forall q \in Q.(q \xrightarrow{t} p \xrightarrow{t'/t} q', t \ \iota \ t' \Rightarrow \exists! p' \in Q. q \xrightarrow{t'} p' \xrightarrow{t/t'} q'). \end{array}$$

The first of the two axioms is a conditional Church-Rosser property. The second axiom ensures that the set of computations of an ATS is closed under permutation of consecutive, independent transitions (both modulo residual). We assume that, just as the codes of transitions are computed as the reduction proceeds, the independency relation will be determined in the same manner. Note that, as soon as the reduction has been performed, the actual labels of moves have been lost, so the independency relation is defined over action occurrences, rather than action labels as is usually the case.

$S = (Q, T, \mathrm{s}, \mathrm{e}, \iota, /, q_0)$ is said to be a *recursive* (resp. *recursively enumerable*) asynchronous transition system iff all its components are recursive (resp. Σ^0_1).

Given an ATS $S = (Q, T, \mathrm{s}, \mathrm{e}, \iota, /, q_0)$ define the relation $\equiv^f$ on finite sequences of transitions as follows. Let $\sigma, \tau \in \mathit{Fin}(S)$. Then

$$\begin{array}{lll} \sigma \equiv^f \tau & \Longleftrightarrow & \exists \gamma_1, \cdots, \gamma_n. \quad \sigma = \gamma_1, \gamma_n = \tau, \\ & & \qquad \gamma_i = \gamma'_i t_1 (t_2 / t_1) \gamma''_i, t_1 \ \iota \ t_2, \gamma_{i+1} = \gamma'_i t_2 (t_1 / t_2) \gamma''_i \end{array}$$

Thus, two (finite) sequences are equivalent iff they differ in the order of consecutive independent transitions. The case of infinite sequences is a bit more complex, since we need to handle an infinite number of permutations as in $(ab)^\omega$ and $(ba)^\omega$, which clearly should be equivalent if $a \ \iota \ b$. We first define a preorder $\preceq^f$ on $T^\star$:

$$\sigma \preceq^f \tau \quad \Longleftrightarrow \sigma \leq \gamma \equiv^f \tau.$$

We now extend this preorder to T^∞ by:

$$\sigma \preceq \tau \quad \Longleftrightarrow \quad \forall \sigma' \leq \sigma \, \exists \tau' \leq \tau. \, \sigma' \preceq^f \tau'.$$

The equivalence relation induced by this preorder (*i.e.* $\equiv \stackrel{\mathrm{df}}{=} \preceq \cap \succeq$) is called the *trace equivalence* on T^∞. A calculation using the Tarski-Kuratowski algorithm ([23], page 307) shows this equivalence is Π^0_2.

The quotient $\mathit{Inf}(S)/_\equiv$ is the *asynchronous behaviour* of the asynchronous transition system S. An equivalence class $[\sigma]$ of a sequence is called a (Mazurkiewicz) *trace*. That this is a reasonable definition of a behaviour of S (meaning that all the sequences added by the closure under the equivalence relation are actually possible computations of the system) follows from the axiom (Perm-$\Diamond$). Moreover, any two finite equivalent sequences lead to the same state.

Proposition 3.1 For any $\sigma, \tau \in \mathit{Fin}(S)$, $\sigma \equiv \tau$, $q \xrightarrow{\sigma} q'$, $q \xrightarrow{\tau} q''$ implies $q' = q''$.

Proof: By induction from axiom (Forw-$\diamond$). $\square$

There exists a strong relationship between asynchronous transition systems and certain classes of trace languages [25, 13].

The discussion in [7] gave examples of effective transition systems, *e.g.* the λ-calculus with β-reduction. What is of interest to us here is whether the notion of an effective *asynchronous* transition system is reasonable, in the sense of whether effective ATS's "occur in nature". It is an easy observation that (finite) Condition-Event nets determine finite asynchronous transition systems, see *e.g.* [12]. CCS with the one-step transition $\xrightarrow{a}$ determines a recursive transition system, while CCS with the $\overset{a}{\Rightarrow} \overset{\mathrm{df}}{=} (\xrightarrow{\tau})^\star \xrightarrow{a} (\xrightarrow{\tau})^\star$ determines a recursively enumerable transition system. The question of the independency relation being determined is a bit more difficult to settle, because it requires labelling of transitions, but nevertheless Boudol and Castellani [5] showed that CCS determines an ATS. The idea is to label transitions with *proofs* of how to obtain a given transition from a process expression, so, for example, the b-move of $(a \,|\, b) + c$ is labelled with $+_1(|_2(b))$, meaning that it comes from the lefthand-side of the $+$, followed by the righthand-side of the $|$ operator. In addition, passing through a guard may be encoded as γ_a, so that the proof of ab is represented by $\gamma_a(b)$. Under these conditions, given a CCS expression, each label uniquely determines the start and end states. For the subcalculus without communication and restriction (see [5] for full definition), the independency relation on such proof terms is the least symmetric relation satisfying:

$$
\begin{array}{ll}
\text{(i1)} & |_1\theta \;\iota\; |_2\theta' \\
\text{(i2)} & \theta \;\iota\; \theta' \Rightarrow |_i\theta \;\iota\; |_i\theta' \\
\text{(i3)} & \theta \;\iota\; \theta' \Rightarrow +_i\theta \;\iota\; +_i\theta'
\end{array}
$$

The essential rule is (i1), which states that two proof terms are independent iff they appear on either side of the parallel. The remaining rules preserve the relation through the syntax of proof terms.

The intuition for the residual operation is as follows. In the term such as $(a \mid b) + c$, the transitions $+_1(|_1(a))$ and $+_1(|_2(b))$ are independent, but after $+_1(|_1(a))$ has been performed, what remains from $+_1(|_2(b))$ is $|_2(b)$. Formally, the residual operation, again without communication and restriction, is defined by:

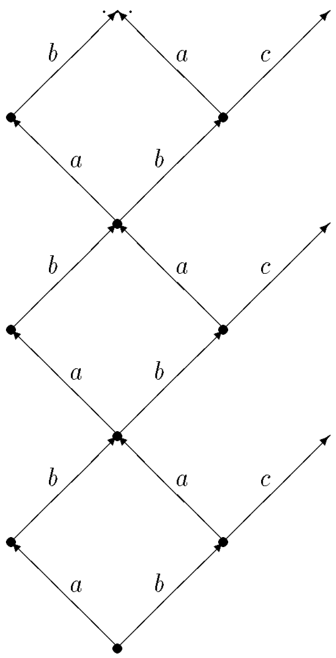
$$
\begin{array}{ll}
\text{(r1)} & (|_1\theta) \,/\, (|_2\theta') = |_1\theta \\
\text{(r2)} & (|_2\theta') \,/\, (|_1\theta) = |_2\theta' \\
\text{(r3)} & \theta \;\iota\; \theta' \Rightarrow (|_i\theta) \,/\, (|_i\theta') = |_i(\theta \,/\, \theta') \\
\text{(r4)} & \theta \;\iota\; \theta' \Rightarrow (+_i\theta) \,/\, (+_i\theta') = \theta \,/\, \theta'
\end{array}
$$

Observe that both the independency relation and the residual are recursive if the system is. Both diamond properties are proved for an LTS labelled with proofs [5].

4 Fairness for Asynchronous Transition Systems

Let S be a rooted ATS. A phenomenon exhibited by a non-interleaving semantic model, but not considered in the standard interleaving approaches,

Figure 2: Asymmetric confusion ($a \iota b$)



is that of *confusion*. Informally, confusion arises if there is an interference between choice and concurrency. To introduce confusion formally we shall require the notion of a *conflict set* of a transition. For $t \in T$ define $\mathit{cfl}(t)$ to be $\{t' \mid \mathrm{s}(t) = \mathrm{s}(t') \wedge \neg(t \iota t')\}$. We say a triple (q, t, t') is a confusion if $q \xrightarrow{t}, q \xrightarrow{t'}, t \iota t'$ and $\mathit{cfl}(t) \neq \mathit{cfl}(t / t')$. A triple (q, t, t') is a symmetric confusion iff (q, t', t) is also a confusion, and is asymmetric otherwise.

Figure 2 shows an example of confusion, where $a \iota b$ and for simplicity we have assumed that $a / b = a$, $b / a = b$. Note that $(ab)^\omega$ is fair (since c is never enabled), while $(ba)^\omega$ is *not* fair (since c is infinitely often enabled. Yet, the two sequences are equivalent. This raises the possibility of an undesirable situation, which would make the fairness or unfairness of a computation dependent on the particular choices made by the scheduler. Naturally, we would like to avoid this, and so would propose that fairness properties be closed with respect to equivalence $\equiv$. A similar point has also been made elsewhere, including [2, 12].

We find that we have to impose the following monotonicity condition on the enabledness predicate E. We define the predicate $\mathrm{E} \subseteq \omega \times \omega$ to be *monotone* iff

$$\forall f, g. f[k] \iota g[k] \Rightarrow (\mathrm{E}(f[k], i) \iff \mathrm{E}(g[k], i)).$$

where for finite sequences σ, τ we say $\sigma \iota \tau$ iff they are incomparable under $\preceq$ and there exist $t_1, ..., t_n$, $t'_1, ..., t'_n$, $t_i \iota t'_i$ all i, $\gamma \equiv \gamma'$, such that $\sigma = \gamma t_1 ... t_n$, $\tau = \gamma' t'_1 ... t'_n$.

The idea of the condition is that the enabledness of a transition is independent of the choice of a sequence within a $\equiv$-equivalence class, as shown by the lemma below.

Lemma 4.1 For all $f, g \in \mathit{Inf}(S)$, $f \equiv g$ implies $\forall i\, \forall k.(E(f[k], i) \iff E(g[k], i))$.

A recursive ATS is *finitely concurrent* iff the set $C(t) = \{t' \in T \mid t \,\iota\, t'\}$ is finite for all t. We have the following.

Proposition 4.2 For any rooted, recursive and finitely concurrent asynchronous transition system S the following are equivalent:

1. F^+ is the closure under $\equiv$ of a Π^0_3 subset of $\mathit{Inf}(S)$.
2. F^+ is the closure under $\equiv$ of the derived set of $\mathit{Fin}(S)$ for some Π^0_1 ultrametric refining the usual Baire metric δ.
3. $F^+ = \{f \mid \forall i\, \forall^\omega k.\neg \mathrm{E}^+(f[k], i)\}$ for some recursive monotone predicate $\mathrm{E}^+ \subseteq \omega \times \omega$.

Proof: ((3) $\Rightarrow$ (1)) That F^+ is Π^0_3 is immediate. The closure under $\equiv$ follows from Lemma 4.1.

To show (1) $\iff$ (2) observe that if a Π^0_3 set F coincides with the derived set for some metric (by Proposition 2.3), then the respective closures under the trace equivalence $\equiv$ also coincide.

To show (1) $\Rightarrow$ (3) assume F^+ is the closure under $\equiv$ of a Π^0_3 set F. Then $F = \{f \mid \forall i\, \exists j\, \forall k. R(f, i, j, k)\}$ for some recursive relation R. By Proposition 2.3 it follows that $F = \{f \mid \forall i\, \forall^\omega k. \neg E(f[k], i)\}$. Define $E^+(x, i)$ from $E(x, i)$ by:

$$E^+(x, i) \iff E(x, i) \text{ or } \exists y \,\iota\, x. E(y, i).$$

The restriction to finite concurrency is essential so that all sequences y independent of x can be enumerated in finite time. Finally, observe that E^+ satisfies the condition of monotonicity and by Lemma 4.1 $F^+ = \{f \mid \forall i\, \forall^\omega k. \neg E^+(f[k], i)\}$. $\square$

5 Conclusion and Further Work

Several authors have pointed out that fairness and unbounded non-determinism are connected, see *e.g.* [24, 21]. This has led some of them to reject fairness on the grounds that it is not computable. This is in contradiction with the practical knowledge, which provides ample evidence that fairness is an important property of hardware and software systems (such as communication protocols, distributed databases), which can be expressed in logics such as the temporal logic CTL*, mechanically verified by means of a model checker, and guaranteed by an algorithm. What we would like to argue is that there is really no contradiction, because unbounded non-determinism is a proof-theoretic phenomenon (a proof being a well-founded, ω-branching tree), while fairness is a phenomenon of programs (finitely branching trees with possibly infinite paths). Since there exists a one-one effective map between well-founded, ω-branching trees and finitely branching trees with only those infinite paths which satisfy a certain infinitary formula of a modal logic, it follows that fairness is provable, as well as computable in practice.

We have extended a connection between Π^0_3 sets of functions and fairness to cater for effective transition systems with concurrency structure given by an independency of transitions. Since there is a strong correspondence between such sets of functions and the Borel hierarchy – in fact, the following relationship holds ([16], page 799):

$$\mathbf{\Pi}^0_n = \bigcup_{\epsilon \in \omega^\omega} \Pi^0_n[\epsilon]$$

where boldface $\mathbf{\Pi}^0_n$ denotes the classical Borel hierarchy of sets of functions, and lightface $\Pi^0_n[\epsilon]$ is a *relativization* of Π^0_n with an *oracle* ϵ – one might ask whether this work helps to find the right topological conditions that one would like semantic fairness properties (viewed extensionally) to satisfy. Unfortunately, the answer is no because the link with the topology of the semantic domain is lost as soon as the reduction has been performed.

Effective transition graphs have been introduced by de Simone [8], who showed that every r.e. transition graph can be expressed in SCCS or Meije using a certain format of rules. It is an interesting problem if there is any format of rules which can be associated with r.e. asynchronous transition systems, and if similar expressiveness results can be obtained. Also, following the work of Badouel and Darondeau on extracting concurrency from a basic SOS format of rules [4], a generalization to more expressive formats would be desirable.

Acknowledgements

We would like to acknowledge Samson Abramsky, Iain Phillips, Mike Smyth, Philippe Darondeau and Lutz Priese for conversations on the subject. The author is also grateful to the Nuffield Science Foundation grant (SCI/124/528/G) for providing funds to visit the Theory and Formal Methods Section of the Department of Computing, Imperial College.

References

[1] Samson Abramsky. Private communication, 1993.

[2] Krzysztof Apt, Nissim Francez, and Shmuel Katz. Appraising fairness in languages for distributed systems. *Distributed Computing*, 2, 1988.

[3] Krzysztof Apt and Gordon Plotkin. Countable nondeterminism and random assignment. *Journal of the ACM*, 33(4), 1986.

[4] E. Badouel and P. Darondeau. Structural operational semantics and trace automata. In *CONCUR 92*, volume 630 of *Lecture Notes in Computer Science*, pages 302–316. Springer-Verlag, 1992.

[5] Gerard Boudol and Ilaria Castellani. Three equivalent semantics for CCS. In *Semantics of Systems for Concurrent Processes*, volume 469 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[6] P. Darondeau. Private communication., 1993.

[7] P. Darondeau, D. Nolte, L. Priese, and S. Yoccoz. Fairness, distances and degrees. *Theoretical Computer Science*. To appear.

[8] Robert de Simone. Higher-level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1987.

[9] Nissim Francez. *Fairness.* Springer-Verlag, 1986.

[10] D. Harel. Effective transformations on infinite trees with applications to high undecidability, dominoes and fairness. *Journal of the ACM*, 33:224–248, 1986.

[11] K. Kuratowski. *Topology.* Academic Press, 1966.

[12] Marta Z. Kwiatkowska. Event fairness and non-interleaving concurrency. *Formal Aspects of Computing*, 1(3):213–228, 1989.

[13] Marta Z. Kwiatkowska. Fairness for non-interleaving concurrency (PhD Thesis). Technical Report 22, University of Leicester, 1989.

[14] Marta Z. Kwiatkowska. Survey of fairness notions. *Information and Software Technology*, 31(7):371–386, 1989.

[15] Zohar Manna and Amir Pnueli. The anchored version of the temporal framework. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 112 of *Lecture Notes in Computer Science*, pages 201–284. Springer-Verlag, 1989.

[16] D. A. Martin. Descriptive set theory. In J. Barwise, editor, *Handbook of Mathamatical Logic.* North-Holland, 1977.

[17] Antoni Mazurkiewicz. Trace theory. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 279–324. Springer-Verlag, 1986.

[18] Antoni Mazurkiewicz. Basic notions of trace theory. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 25–34. Springer-Verlag, 1989.

[19] Robin Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.

[20] D. Nolte and L. Priese. Fairness in models with true concurrency. In *CONCUR '91*, Lecture Notes in Computer Science. Springer-Verlag, 1991.

[21] D. Park. On the semantics of fair parallelism. In D. Bjorner, editor, *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*, pages 504–526. Springer-Verlag, 1980.

[22] Gordon Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5:452–487, 1976.

[23] H. Rogers. *Theory of recursive functions and effective computability.* McGraw Hill, 1967.

[24] A. W. Roscoe. Two papers on CSP. Technical Report PRG-67, Oxford University, 1988.

[25] M.W. Shields. Deterministic asynchronous automata. In *Formal Methods in Programming.* North Holland, 1985.