Fair Equivalence Relations

Orna Kupferman[1] and Nir Piterman[2] and Moshe Y. Vardi[3]*

[1] Hebrew University, School of Engineering and Computer Science, Jerusalem 91904, Israel
Email: orna@cs.huji.ac.il, URL: http://www.cs.huji.ac.il/~orna
[2] Weizmann Institute of Science, Department of Computer Science, Rehovot 76100, Israel
Email: nirp@wisdom.weizmann.ac.il, URL: http://www.wisdom.weizmann.ac.il/~nirp
[3] Rice University, Department of Computer Science, Houston, TX 77251-1892, U.S.A.
Email: vardi@cs.rice.edu, URL: http://www.cs.rice.edu/~vardi

Abstract. Equivalence between designs is a fundamental notion in verification. The linear and branching approaches to verification induce different notions of equivalence. When the designs are modeled by fair state-transition systems, equivalence in the linear paradigm corresponds to fair trace equivalence, and in the branching paradigm corresponds to fair bisimulation.
In this work we study the expressive power of various types of fairness conditions. For the linear paradigm, it is known that the Büchi condition is sufficiently strong (that is, a fair system that uses Rabin or Streett fairness can be translated to an equivalent Büchi system). We show that in the branching paradigm the expressiveness hierarchy depends on the types of fair bisimulation one chooses to use. We consider three types of fair bisimulation studied in the literature: $\exists$-bisimulation, game-bisimulation, and $\forall$-bisimulation. We show that while game-bisimulation and $\forall$-bisimulation have the same expressiveness hierarchy as tree automata, $\exists$-bisimulation induces a different hierarchy. This hierarchy lies between the hierarchies of word and tree automata, and it collapses at Rabin conditions of index one, and Streett conditions of index two.

1 Introduction

In formal verification, we check that a system is correct with respect to a desired behavior by checking that a mathematical model of the system satisfies a formal specification of the behavior. In a concurrent setting, the system under consideration is a composition of many components, giving rise to state spaces of exceedingly large size. One of the ways to cope with this state-explosion problem is *abstraction* [BCG88,CFJ93,BG00]. By abstracting away parts of the system that are irrelevant for the specification being checked, we hope to end up with manageable state-spaces. Technically, abstraction may cause different states s and s' of the system to become equivalent. The abstract system then has as its state space the equivalence classes of the equivalence relation between the states. In particular, s and s' are merged into the same state.

We distinguish between two types of equivalence relations between states. In the *linear* approach, we require s and s' to agree on linear behaviors (i.e., properties satisfied by all the computations that start in s and s'). In the *branching* approach, we require

* Supported in part by NSF grants CCR-9700061 and CCR-9988322, and by a grant from the Intel Corporation.

s and s' to agree on branching behaviors (i.e., properties satisfied by the computation trees whose roots are s and s'). When we model systems by *state-transition systems*, two states are equivalent in the linear approach iff they are *trace equivalent*, and they are equivalent in the branching approach iff they are *bisimilar* [Mil71]. The branching approach is stronger, in the sense that bisimulation implies trace equivalence but not vice versa [Mil71,Pnu85].

Of independent interest are the one-way versions of trace equivalence and bisimulation, namely *trace containment* and *simulation*. There, we want to make sure that s does not have more behaviors than s'. This corresponds to the basic notion of verification, where an implementation cannot have more behaviors than its specification [AL91]. In the *hierarchical refinement* top-down methodology for design development, we start with a highly abstract specification, and we construct a sequence of "behavior descriptions". Each description refers to its predecessor as a specification, and the last description is sufficiently concrete to constitute the implementation (cf. [LT87,Kur94]).

The theory behind trace equivalence and bisimulation is well known. We know that two states are trace equivalent iff they agree on all LTL specifications, and the problem of deciding whether two states are trace equivalent is PSPACE-complete [MS72,KV98b]. In the branching approach, two states are bisimilar iff they agree on all CTL* formulas, which turned out to be equivalent to agreement on all CTL and μ-calculus formulas [BCG88,JW96]. The problem of deciding whether two states are bisimilar is PTIME-complete [Mil80,BGS92], and a witnessing relation for bisimulation can be computed using a symbolic fixpoint procedure [McM93,HHK95]. Similar results hold for trace containment and simulation. The computational advantage of simulation makes it a useful precondition to trace containment [CPS93].

State-transition systems describe only the *safe* behaviors of systems. In order to model *live* behaviors, we have to augment systems with *fairness conditions*, which partition the infinite computations of a system into fair and unfair computations [MP92,Fra86]. It is not hard to extend the linear approach to account for fairness: s and s' are equivalent if every sequence of observations that is generated along a fair computation that starts in s can also be generated along a fair computation that starts in s', and vice versa. Robustness with respect to LTL, and PSPACE-completeness extend to the fair case. It is less obvious how to generalize the branching approach to account for fairness. Several proposals for *fair bisimulation* can be found in the literature. We consider here three: $\exists$-*bisimulation* [GL94], *game-bisimulation* [HKR97,HR00], and $\forall$-*bisimulation* [LT87]. In a bisimulation relation between $\mathcal{S}$ and $\mathcal{S}'$ with no fairness, two related states s and s' agree on their observable variables, every successor of s is related to some successor of s', and every successor of s' is related to some successor of s. In all the definitions of fair bisimulation, we require related states to agree on their observable variables. In $\exists$-bisimulation, we also require every fair computation starting at s to have a related fair computation starting at s', and vice versa. In game-bisimulation, the related fair computations should be generated by strategies that depend on the states visited so far, and in $\forall$-bisimulation, the relation is a bisimulation in which related computations agree on their fairness (we review the formal definitions in Section 2).

The different definitions induce different relations: $\forall$-bisimulation implies game-bisimulation, which implies $\exists$-bisimulation, but the other direction does not hold [HKR97].

The difference in the distinguishing power of the definitions is also reflected in their logical characterization: while $\exists$-bisimulation corresponds to fair-$\mathrm{CTL}^\star$ (that is, two systems are $\exists$-bisimilar iff they agree on all fair-$\mathrm{CTL}^\star$ formulas, where path quantifiers range over fair computations only [CES86]), game-bisimulation corresponds to fair-alternation-free μ-calculus[1]. Thus, unlike the non-fair case, where almost all modal logics corresponds to bisimulation, here different relations correspond to different logics [ASB+94] [2]. Finally, the different definitions induce different computational costs. The exact complexity depends on the fairness condition being used. For the case of the Büchi fairness condition, for example, the problem of checking whether two systems are bisimilar is PSPACE-complete for $\exists$-bisimulation [KV98b], NP-complete for $\forall$-bisimulation [Hoj96], and PTIME-complete for game-bisimulation [HKR97,HR00].

There are various types of fairness conditions with which we can augment labeled state-transition systems [MP92]. Our work here relates fair transition systems and automata on infinite objects, and we use the types and names of fairness conditions that are common in the latter framework [Tho90]. The simplest condition is *Büchi* (also known as *unconditional* or *impartial* fairness), which specifies a set of states that should be visited infinitely often along fair computations. In its dual condition, *co-Büchi*, the specified set should be visited only finitely often. More involved are *Streett* (also known as *strong* fairness or *compassion*), *Rabin* (Streett's dual), and *parity* conditions, which can restrict both the set of states visited infinitely often and the set of states visited finitely often. Rabin and parity conditions were introduced for automata and are less frequent in the context of state-transition systems. Rabin conditions were introduced by Rabin and were used to prove that the logic S2S is decidable [Rab69]. Parity conditions can be easily translated to both Rabin and Streett conditions. They have gained their popularity as they are suitable for modeling behaviors that are given by means of fixed-points [EJ91]. As we formally define in Section 2, Rabin, Streett, and parity conditions are characterized by their *index*, which is the number of pairs (in the case of Rabin and Streett) or sets (in the case of parity) they contain. When we talk about a *type* of a system, we refer to its fairness condition and, in the case of Rabin, Streett, and parity, also to its index. For example, a Rabin[1] system is a system whose fairness condition is a Rabin condition with a single pair.

The relations between the various types of fairness conditions are well known in the linear paradigm. There, we can regard fair transition systems as a notational variant of automata on infinite words, and adopt known results about translations among the various types and about the complexity of the trace-equivalence and the trace-containment problems [Tho90]. In particular, it is known that the Büchi fairness condition is sufficiently strong, in the sense that every system can be translated to an equivalent Büchi system, where equivalence here means that the systems are trace equivalent.

In the branching paradigm, tight complexity bounds are known for the fair-bisimulation problem with respect to the three definitions of fair bisimulation and the various types of fairness conditions [Hoj96,HKR97,KV98b], but nothing is known about their expressive power, and about the possibilities of translations among them. For example,

[1] A semantics of fair-alternation-free μ-calculus is given in [HR00].

[2] As shown in [ASB+94], the logic CTL induces yet another definition, strictly weaker than $\exists$-bisimulation. Also, no logical characterization is known for $\forall$-bisimulation.

it is not known whether every system can be translated to an equivalent Büchi system, where now equivalence means fair bisimulation. In particular, it is not clear whether one can directly apply results from the theory of *automata on infinite trees* in order to study fair-bisimulation, and whether the different definitions of fair bisimulation induce different expressiveness hierarchies.

In this paper, we study the expressive power of the various types of fairness conditions in the context of fair bisimulation. For each of the three definitions of fair bisimulation, we consider the following question: given types γ and γ' of fairness conditions, is it possible to translate every γ-system to a fair-bisimilar γ'-system? If this is indeed the case, we say that γ' is *at least as strong as* γ. Then, γ is *stronger than* γ' if γ is at least as strong as γ', but γ' is not at least as strong as γ. When γ is stronger than γ', we also say that γ' is *weaker than* γ. We show that the expressiveness hierarchy for game-bisimulation and $\forall$-bisimulation is strict, and it coincides with the expressiveness hierarchy of tree automata. Thus, Büchi and co-Büchi systems are incomparable and are the weakest, and for all $i \geq 1$, Rabin$[i+1]$, Streett$[i+1]$, and parity$[i+1]$, are stronger than Rabin$[i]$, Streett$[i]$, and parity$[i]$, respectively [Rab70,DJW97,Niw97,NW98]. In contrast, the expressiveness hierarchy for $\exists$-bisimulation is different, and it is not strict. We show that Büchi and co-Büchi systems are incomparable, and they are both weaker than Streett[1] systems. Streett[1] systems are in turn weaker than Streett[2] and Rabin[1] systems, which are both at least as strong as Rabin$[i]$ and Streett$[i]$, for all $i \geq 1$.

Our results imply that the different definitions of fair bisimulation induce different expressiveness relations between the various types of fairness conditions. These relations are different than those known for the linear paradigm, and, unlike the case there, they do not necessarily coincide with the relations that exist in the context of automata on infinite trees. A decision of which fairness condition and which type of fair-bisimulation relation to use in a modeling and verificatiuon process should take into an account all the characteristics of these types, and it cannot be assumed that what is well known for one type is true for another.

Due to space limitations, most of the proofs are omitted. A full version can be found in the homepages of the authors.

2 Definitions

A *fair state-transition system* (*system*, for short) $S = \langle \Sigma, W, R, W_0, L, \alpha \rangle$ consists of an alphabet Σ, a finite set W of states, a total transition relation $R \subseteq W \times W$ (i.e., for every $w \in W$ there exists $w' \in W$ such that $R(w, w')$), a set W_0 of initial states, a labeling function $L : W \to \Sigma$, and a fairness condition α. We will define several types of fairness conditions shortly. A *computation* of S is a sequence $\pi = w_0, w_1, w_2, \ldots$ of states such that for every $i \geq 0$, we have $R(w_i, w_{i+1})$. Each computation $\pi = w_0, w_1, w_2, \ldots$ induces the word $L(\pi) = L(w_0) \cdot L(w_1) \cdot L(w_2) \cdots \in \Sigma^\omega$. In order to determine whether a computation is *fair*, we refer to the set $inf(\pi)$ of states that π visits infinitely often. Formally, $inf(\pi) = \{w \in W : \text{for infinitely many } i \geq 0, \text{ we have } w_i = w\}$. The way we refer to $inf(\pi)$ depends on the fairness condition of S. Several types of fairness conditions are studied in the literature:

- *Büchi* (*unconditional* or *impartial*), where $\alpha \subseteq W$, and π is fair iff $inf(\pi) \cap \alpha \neq \emptyset$.

– *co-Büchi*, where $\alpha \subseteq W$, and π is fair iff $inf(\pi) \cap \alpha = \emptyset$.
– *Parity*, where α is a partition of W, and π is fair in $\alpha = \{F_1, F_2, \ldots, F_k\}$ if the minimal index i for which $inf(r) \cap F_i \neq \emptyset$ exists and is even.
– *Rabin*, where $\alpha \subseteq 2^W \times 2^W$, and π is fair in $\alpha = \{\langle G_1, B_1\rangle, \ldots, \langle G_k, B_k\rangle\}$ if there is a $1 \leq i \leq k$ such that $inf(\pi) \cap G_i \neq \emptyset$ and $inf(\pi) \cap B_i = \emptyset$.
– *Streett* (*compassion* or *strong fairness*), where $\alpha \subseteq 2^W \times 2^W$, and π is fair in $\alpha = \{\langle G_1, B_1\rangle, \ldots, \langle G_k, B_k\rangle\}$ if for all $1 \leq i \leq k$, we have that $inf(\pi) \cap G_i \neq \emptyset$ implies $inf(\pi) \cap B_i \neq \emptyset$.

The number k of sets in a parity fairness condition or of pairs in a Rabin or Streett fairness condition is the *index* of α. When we talk about the *type* of a system, we refer to its fairness condition and, in the case of Rabin, Streett, and parity, also to its index. For example, a Rabin[1] system is a system whose fairness condition is a Rabin condition with a single pair. For a state w, a w-computation is a computation $w_0, w_1, w_2, \ldots$ with $w_0 = w$. We use $\mathcal{T}(S^w)$ to denote the set of all traces $\sigma_0 \cdot \sigma_1 \cdots \in \Sigma^\omega$ for which there exists a fair w-computation $w_0, w_1, \ldots$ in S with $L(w_i) = \sigma_i$ for all $i \geq 0$. The *trace set* $\mathcal{T}(S)$ of S is then defined as $\bigcup_{w \in W_0} \mathcal{T}(S^w)$.

We now formalize what it means for two systems (or two states of the same system) to be equivalent. We give the definitions with respect to two systems $S = \langle \Sigma, W, R, W_0, L, \alpha\rangle$ and $S' = \langle \Sigma, W', R', W_0', L', \alpha'\rangle$, with the same alphabet.[3] We consider two equivalence criteria: *trace equivalence* and *bisimulation*. While the first criterion is clear ($\mathcal{T}(S) = \mathcal{T}(S')$), several proposals are suggested in the literature for bisimulation in the case of systems with fairness. Before we define them, let us first recall the definition of bisimulation for the non-fair case.

Bisimulation [Mil71] A relation $H \subseteq W \times W'$ is a *bisimulation relation* between S and S' iff the following conditions hold for all $\langle w, w'\rangle \in H$.

1. $L(w) = L'(w')$.
2. For all $s \in W$ with $R(w, s)$, there is $s' \in W'$ such that $R'(w', s')$ and $H(s, s')$.
3. For all $s' \in W$ with $R'(w', s')$, there is $s \in W$ such that $R(w, s)$ and $H(s, s')$.

We now describe three extensions of bisimulation relations to the fair case. In all definitions, we extend a relation $H \subseteq W \times W'$, over the states of S and S', to a relation over infinite computations of S and S': for two computations $\pi = w_0, w_1, \ldots$ in S, and $\pi' = w_0', w_1', \ldots$ in S', we have $H(\pi, \pi')$ iff $H(w_i, w_i')$, for all $i \geq 0$.

$\exists$-bisimulation [GL94] A relation $H \subseteq W \times W'$ is an *$\exists$-bisimulation relation* between S and S' iff the following conditions hold for all $\langle w, w'\rangle \in H$.

1. $L(w) = L'(w')$.
2. Each fair w-computations π in S has a fair w'-computation π' in S' with $H(\pi, \pi')$.
3. Each fair w'-computations π' in S' has a fair w-computation π in S with $H(\pi, \pi')$.

Game bisimulation [HKR97,HR00] Game bisimulation is defined by means of a game between a protagonist against an adversary. The positions of the game are pairs

[3] In practice, S and S' are given as systems over alphabets 2^{AP} and $2^{AP'}$, when AP and AP' are the sets of atomic propositions used in S and S', and possibly $AP \neq AP'$. When we compare S with S', we refer only to the common atomic propositions, thus $\Sigma = 2^{AP \cap AP'}$.

in $W \times W'$. A *strategy* τ for the protagonist is a partial function from $(W \times W')^* \times (W \cup W')$ to $(W' \cup W)$, such that for all $\rho \in (W \times W')^*$, $w \in W$, and $w' \in W'$, we have that $\tau(\rho \cdot w) \in W'$ and $\tau(\rho \cdot w') \in W$. Thus, if the game so far has produced the sequence ρ of positions, and the adversary moves to w in S, then the strategy τ instructs the protagonist to move to $w' = \tau(\pi \cdot w)$, resulting in the new position $\langle w, w' \rangle$. If the adversary chooses to move to w' in S', then τ instructs the protagonist to move to $w = \tau(\pi \cdot w')$, resulting in the new position $\langle w, w' \rangle$. A sequence $\overline{w} = \langle w_0, w'_0 \rangle \cdot \langle w_1, w'_1 \rangle \cdots \in (W \times W')^\omega$ is an *outcome* of the strategy τ if for all $i \geq 0$, either $w'_{i+1} = \tau(\langle w_0, w'_0 \rangle \cdots \langle w_i, w'_i \rangle \cdot w_{i+1})$, or $w_{i+1} = \tau(\langle w_0, w'_0 \rangle \cdots \langle w_i, w'_i \rangle \cdot w'_{i+1})$.

A binary relation $H \subseteq W \times W'$ is a *game bisimulation relation* between S and S' if there exists a strategy τ such that the following conditions hold for all $\langle w, w' \rangle$ in H.

1. $L(w) = L(w')$.
2. Every outcome $\overline{w} = \langle w_0, w'_0 \rangle \cdot \langle w_1, w'_1 \rangle \cdots$ of τ with $w_0 = w$ and $w'_0 = w'$ has the following two properties: (1) for all $i \geq 0$, we have $\langle w_i, w'_i \rangle \in H$, and (2) the projection $w_0 \cdot w_1 \cdots$ of $\overline{w}$ to W is a fair w_0-computation of S iff the projection $w'_0 \cdot w'_1 \cdots$ of $\overline{w}$ to W' is a fair w'_0-computation of S'.

$\forall$-bisimulation [LT87,DHW91] A binary relation $H \subseteq W \times W'$ is a *$\forall$-bisimulation relation* between S and S' if the following conditions hold:

1. H is a bisimulation relation between S and S'.
2. If $H(w, w')$, then for every fair w-computation π of S and for every w'-computation π' of S', if $H(\pi, \pi')$, then π' is fair.
3. If $H(w, w')$, then for every fair w'-computation π' of S' and for every w-computation π of S, if $H(\pi, \pi')$, then π is fair.

It is not hard to see that if H is a $\forall$-bisimulation relation, then H is also a game-bisimulation relation. Also, if H is a game-bisimulation relation, then H is also an $\exists$-bisimulation relation. As demonstrated in [HKR97], the other direction is not true.

For all types β of bisimulation relations (that is $\beta \in \{\exists, game, \forall\}$), a β-bisimulation relation H is a *β-bisimulation between S and S'* if for every $w \in W_0$ there exists $w' \in W'_0$ such that $H(w, w')$, and for every $w' \in W'_0$ there exists $w \in W_0$ such that $H(w, w')$. If there is a β-bisimulation between S and S', we say that S and S' are *β-bisimilar*. Intuitively, bisimulation implies that S and S' have the same behaviors. Formally, two bisimilar systems with no fairness agree on the satisfaction of all branching properties that can be specifi ed in a conventional temporal logic (in particular, CTL* and μ-calculus) [BCG88,JW96]. When we add fairness, the logical characterization becomes less robust: $\exists$-simulation corresponds to fair-CTL*, and game-simulation corresponds to fair-alternation-free μ-calculus [ASB$^+$94,GL94,HKR97,HR00].

For $\exists$-bisimulation and $\forall$-bisimulation, a relation $H \subseteq W \times W'$ is a *β-simulation relation* from S to S' if conditions 1 and 2 for H being a β-bisimulation relation hold. For game-bisimulation, a relation H is a *game-simulation relation* from S to S' if we restrict the moves of the adversary to choose only states from S. A β-simulation relation H is a *β-simulation from S to S'* iff for every $w \in W_0$ there exists $w' \in W'_0$ such that $H(w, w')$. If there is a β-simulation from S to S', we say that S' *β-simulates* S, and we write $S \leq_\beta S'$. Intuitively, while bisimulation implies that S and S' have the same behaviors, simulation implies that S has less behaviors than S'.

It is easy to see that bisimulation implies trace equivalence. The other direction, however, is not true [Mil71]. Hence, our equivalence criteria induce different equivalence relations. When attention is restricted to trace equivalence, it is known how to translate all fair systems to an equivalent Büchi system. In this paper we consider the problem of translations among systems that preserve bisimilarity.

3 Expressiveness with ∃-bisimulation

In the linear case, it follows from automata theory that co-Büchi systems are weaker than Büchi systems, which are as strong as parity, Rabin, and Streett systems. In the branching case, nondeterministic Büchi and co-Büchi tree automata are both weaker than Rabin tree automata, and, for all $i \geq 1$, parity$[i]$, Rabin$[i]$, and Streett$[i]$ are weaker than parity$[i+1]$, Rabin$[i+1]$, and Streett$[i+1]$, respectively [Rab70,DJW97,Niw97,NW98]. In this section we show that the expressiveness hierarchy in the context of ∃-bisimulation is located between the hierarchies of word and tree automata.[4]

We first show that Büchi and co-Büchi systems are weak. The arguments we use are similar to these used by Rabin in the context of tree automata [Rab70]. Our proofs use the notion of maximal models [GL94,KV98c]. A system M_ψ is a *maximal model* for an ∀CTL* formula ψ if $M_\psi \models \psi$ and for every module M we have that $M \leq_\exists M_\psi$ iff $M \models \psi$. It can be shown that there is no Büchi system that is ∃-bisimilar to the maximal model of the formula $\forall\Diamond\Box p$ and that there is no co-Büchi system that is ∃-bisimilar to the maximal model of the formula $\forall\Box\Diamond p$. Hence, we have:

Theorem 1. *Büchi is not at least as ∃-strong as co-Büchi and co-Büchi is not at least as ∃-strong as Büchi.*

Note that Theorem 1 implies that the Büchi condition is too weak for defining maximal models for ∀CTL* formulas. On the other hand, the Büchi condition is sufficiently strong for defining maximal models for ∀CTL formulas [GL94,KV98a]. Since parity, Rabin, and Streett are at least as ∃-strong as Büchi and co-Büchi, it follows from Theorem 1 that parity, Rabin, and Streett are all ∃-stronger than Büchi and co-Büchi.

So far things seem to be very similar to tree automata, where Büchi and co-Büchi conditions are incomparable [Rab70]. In particular, the ability of the Büchi condition to define maximal models for ∀CTL and its inability to define maximal models for ∀CTL* seems related to the ability to translate CTL formulas to Büchi tree automata and the inability to translate CTL* formulas to Büchi tree automata (as follows from Rabin's result [Rab70]). In tree automata, the hierarchy of expressive power stays strict also when we proceed to parity (or Rabin or Streett) fairness condition with increasing indices [DJW97,Niw97,NW98]. We now show that, surprisingly, in the context of ∃-bisimulation, Rabin conditions of index one are at least as strong as parity, Rabin, and Streett conditions with an unbounded index. In particular, it follows that maximal models for ∀CTL* can be defined with Rabin[1] fairness. The idea behind the construction is similar to the conversion of Rabin and Streett automata on infinite words to Büchi automata on infinite words.

[4] Here and in the sequel, we use terms like γ is ∃-stronger than γ' to indicate that γ is stronger than γ' in the context of ∃-bisimulation.

Lemma 1. *Every Rabin system with n states and index k has an $\exists$-bisimilar Rabin system with $O(nk)$ states and index* 1.

Proof: Let $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ be a Rabin system with $\alpha = \{\langle G_1, B_1 \rangle, \ldots, \langle G_k, B_k \rangle\}$. We define $S' = \langle \Sigma, W', W'_0, R', L', \alpha' \rangle$ as follows.

- For every $1 \leq i \leq k$, let $W_i = (W \setminus B_i) \times \{i\}$. Then, $W' = (W \times \{0\}) \cup \bigcup_{1 \leq i \leq k} W_i$, and $W'_0 = W_0 \times \{0\}$.
- $R' = \bigcup_{0 \leq i \leq k} \{\langle (w,0), (w',i) \rangle, \langle (w,i), (w',0) \rangle, \langle (w,i), (w',i) \rangle : \langle w, w' \rangle \in R\} \cap (W' \times W')$. Note that R' is total.
- For all $w \in W$ and $0 \leq i \leq k$, we have $L'((w,i)) = L(w)$.
- $\alpha' = \{\langle \bigcup_{1 \leq i \leq k} G_i \times \{i\}, W \times \{0\} \rangle\}$.

Thus, S' consists of $k+1$ copies of S. One copy ("the idle copy") contains all the states in W, marked with 0. Then, k copies are partial: every such copy is associated with a pair $\langle G_i, B_i \rangle$, its states are marked with i, and it contains all the states in $W \setminus B_i$. A computation of S' can return to the idle copy from all copies, where it can choose between staying in the idle copy or moving to one of the other k copies. The acceptance condition forces a fair computation to visit the idle copy only finitely often, forcing the computation to eventually get trapped in a copy associated with some pair $\langle G_i, B_i \rangle$. There, the computation cannot visit states from B_i (indeed, W_i does not contain such states), and it has to visit infinitely many states from G_i. It is not hard to see that the relation $H = \{\langle w, (w,i) \rangle : w \in W \text{ and } 0 \leq i \leq k\}$ is an $\exists$-bisimulation between S and S', thus S and S' are $\exists$-bisimilar. □

In the case of transforming Rabin[k] word automata to Rabin[1] (or Büchi) automata, runs of the automaton on different computations are independent of each other, so there is no need for the automaton to "change its mind" about the pair in α with respect to which the computation is fair. Accordingly, there is no need to return to an idle copy. In the case of tree automata, runs on different computations of the tree depend on each other, and the run of the automaton along a computation may need to postpone its choice of a suitable pair in α ad infinitum, which cannot be captured with a Rabin[1] condition. The crucial observation about $\exists$-bisimulation is that here, if π_1 and π_2 are different fair w-computations, then the fair computations π'_1 and π'_2 for which $H(\pi_1, \pi'_1)$ and $H(\pi_2, \pi'_2)$ are independent. Thus, each computation eventually reaches a state where it can stick to its suitable pair in α. Accordingly, a computation needs to change its mind only finitely often. A visit to the idle copy corresponds to the computation changing its mind, and the fairness condition guarantees that there are only finitely many visits to the idle copy.

We now describe a similar transformation for Streett systems. While in Rabin systems each copy of the original system corresponds to a guess of a pair $\langle G_i, B_i \rangle$ for which G_i is visited infinitely often and B_i is visited only finitely often, here each copy would correspond to a subset $I \subseteq \{1, \ldots, k\}$ of pairs, where the copy associated with I corresponds to a guess that B_i and G_i are visited infinitely often for all $i \in I$, and G_i is visited only finitely often for all $i \notin I$.

Lemma 2. *Every Streett system with n states and index k has an $\exists$-bisimilar Rabin system with $O(n \cdot 2^{O(k)})$ states and index* 1.

Note that while the blow up in the construction in Lemma 1 is linear in the index of the Rabin system, the blow up in the construction in Lemma 2 is exponential in the index of the Streett system. The above blow ups are tight for the linear paradigm [SV89][5]. Since $\exists$-bisimulation implies trace equivalence, it follows that these blow ups are tight also for the $\exists$-bisimulation case.

Since the parity condition is a special case of Rabin, Lemma 1 also implies a translation of parity systems to $\exists$-bisimilar Rabin[1] systems. Also, a Rabin[1] condition $\{\langle G, B\rangle\}$ can be viewed as a parity condition $\{B, G \setminus B, W \setminus (G \cup B)\}$. Hence, parity[3] is as $\exists$-strong as Rabin[1] [6]. A Rabin[1] condition $\{\langle G, B\rangle\}$ is equivalent to the Streett[2] condition $\{\langle W, G\rangle, \langle B, \emptyset\rangle\}$. So, Streett[2] is also as $\exists$-strong as Rabin[1]. It turns out that we can combine the arguments for Büchi and co-Büchi in Theorem 1 to prove that Streett[1] is $\exists$-weaker than Streett[2]. To sum up, we have the following.

Theorem 2. *For every fairness type γ, the types Rabin[1], Streett[2], and parity[3] are all at least as $\exists$-strong as γ.*

Note that the types described in Theorem 2 are tight, in the sense that, as discussed above, Büchi, co-Büchi, Streett[1], and parity[2] may be $\exists$-weaker than γ.

In the full version, we also show that a system with a *generalized Büchi condition* or with a *justice condition* [MP92] can be translated to an $\exists$-bisimilar Büchi system, implying that generalized Büchi and justice conditions are also too weak.

4 Expressiveness with Game-bisimulation and $\forall$-bisimulation

We now study the expressiveness hierarchy for game-bisimulation and $\forall$-bisimulation. We show that unlike $\exists$-simulation, here the hierarchy coincides with the hierarchy of tree automata. Thus, Rabin[i+1] is stronger than Rabin[i], and similarly for Streett and parity. In order to do so, we define game-bisimulation between tree automata, and define transformations preserving game-bisimulation between tree automata and fair systems. We show that game-bisimilar tree automata agree on their languages (of trees), which enables us to relate the expressiveness hierarchies in the two frameworks.

Due to lack of space we only give an outline of the proof. We define a special type of tree automata, called *loose tree automata*. Unlike conventional tree automata [Tho90], the transition function of loose tree automata does not distinguish between the successors of a node, it does not force states to be visited, and it only restricts the set of states that each of the successors may visit. When $\mathcal{A}$ runs on a labeled tree $\langle T, V\rangle$ and it visits a node x with label σ at state q, then $\delta(q, \sigma) = S$ (where S is a subset of the states of $\mathcal{A}$) means that $\mathcal{A}$ should send to all the successors of x copies in states in S. Loose tree automata can use all types of fairness. A run of a loose tree automaton is accepting if all the infinite paths of the run tree satisfy the fairness condition.

[5] [SV89] shows that the transition from Streett word automata to Büchi word automata is exponential in the index of the Streett automaton. Since the transition from Rabin[1] to Büchi is linear, a lower bound for the transition from Streett to Rabin[1] follows.

[6] Recall that a parity fairness condition is a partition of the state set. Hence, a parity[2] condition can be translated to an equivalent co-Büchi fairness condition and vice versa, implying that Rabin[1] is $\exists$-stronger than parity[2].

We can define game-bisimulation for loose tree automata. Given two loose tree automata, we define a game whose positions are pairs of states. A strategy for the game is similar to the strategy defined for systems, but this time the adversary gets to choose an alphabet letter and a successor corresponding to this letter. The protagonist has to follow with a successor corresponding to the same letter in the other automaton. A relation is a game-bisimulation relation if all the outcomes of such plays starting at related states have both projections fair or have both projections unfair. Two loose tree automata are game-bisimilar if there exists a game-bisimulation between them that relates the starting states of each one of the automata to starting states of the other.

Recall that game-bisimulation between systems implies trace equivalence. Game-bisimulation between loose tree automata implies not only agreement on traces that may label paths of accepted trees, but also agreement on the accepted trees! The idea is that given an accepting run tree of one automaton, we use the strategy to build an accepting run tree of its game-bisimilar counterpart. This property of game-bisimulation between loose tree automata enables us to relate the hierarchy of loose tree automata with that of game-bisimulation. Formally, we have the following.

Theorem 3. *Let γ and γ' be two types of fairness conditions. If γ is at least as strong as γ' in the context of game-bisimulation or $\forall$-bisimulation, then γ is at least as strong as γ' also in the context of loose tree automata.*

While loose tree automata are weaker than conventional tree automata [Tho90],the expressiveness hierarchy of loose tree automata coincides with that of tree automata (this is beacause the latter coincides with the hierarchy of deterministic word automata [Wag79,Kam85], and is proven in [KSV96,DJW97,Niw97,NW98] by means of languages that can be recognized by loose tree automata). It follows that the expressiveness hierarchy in the context of game-bisimulation and $\forall$-bisimulation coincides with that of tree automata.

5 Discussion

We considered two equivalence criteria — bisimulation and trace equivalence — between fair state-transition systems. We studied the expressive power of various fairness conditions in the context of fair bisimulation. We showed that while the hierarchy in the context of trace equivalence coincides with the one of nondeterministic word automata, the hierarchy in the context of bisimulation depends on the exact definition of fair bisimulation, and it does not necessarily coincide with the hierarchy of tree automata. In particular, we showed that Rabin[1] systems are sufficiently strong to model all systems up to $\exists$-bisimilarity.

There is an intermediate equivalence criterion: *two-way simulation* (that is $S \leq S'$ and $S' \leq S$) is implied by bisimulation, it implies trace equivalence, and it is equal to neither of the two [Mil71]. Two-way simulation is a useful criterion: S and S' are two-way similar iff for every system S'' we have $S'' \leq S$ iff $S'' \leq S'$ and $S \leq S''$ iff $S' \leq S''$. Hence, in hierarchical refinement, or when defining maximal models for universal formulas, we can replace S with S'. A careful reading through our proofs shows that all the results described in the paper for bisimulation hold also for two-way simulation.

Finally, the study of $\exists$-bisimulation in Section 3 has led to a simple definition of parallel compositions for Rabin and parity systems, required for modular verification of concurrent systems. In the linear paradigm, the composition $S = S_1 \| S_2$ of S_1 and S_2 is defined so that $\mathcal{T}(S) = \mathcal{T}(S_1) \cap \mathcal{T}(S_2)$ (cf. [Kur94]). In the branching paradigm [GL94], Grumberg and Long defined the parallel compositions of two Streett systems. As studied in [GL94,KV98a], in order to be used in modular verification, a definition of composition has to satisfy the following two conditions, for all systems S, S', and S''. First, if $S' \leq_\exists S''$, then $S \| S' \leq_\exists S \| S''$. Second, $S \leq_\exists S' \| S''$ iff $S \leq_\exists S'$ and $S \leq_\exists S''$. In particular, it follows that $S \| S' \leq_\exists S'$, thus every universal formula that is satisfied by a component of a parallel composition, is satisfied also by the composition. When S_1 and S_2 are Streett systems, the definition of $S_1 \| S_2$ is straightforward, and is similar to the product of two Streett word automata. When, however, S_1 and S_2 are Rabin systems, the definition of product of word automata cannot be applied, and a definition that follows the ideas behind a product of tree automata is very complicated and complex. In the full paper we show that the fact that $\exists$-bisimulation is located between word and tree automata enables a simple definition of parallel composition that obeys the two conditions above.

References

[AL91] M. Abadi and L. Lamport. The existence of refinement mappings. *TCS*, 82(2):253–284, 1991.

[ASB+94] A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A.L. Sangiovanni-Vincentelli. Equivalences for fair kripke structures. In *Proc. 21st ICALP*, Jerusalem, Israel, July 1994.

[BCG88] M.C. Browne, E.M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *TCS*, 59:115–131, 1988.

[BG00] D. Bustan and O. Grumberg. Simulation based minimization. In *Proc. 17th ICAD*, Pittsburgh, PA, June 2000.

[BGS92] J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4(6):638–648, 1992.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, January 1986.

[CFJ93] E.M. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. In *Proc. 5th CAV*, LNCS 697, 1993.

[CPS93] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Trans. on Programming Languages and Systems*, 15:36–72, 1993.

[DHW91] D.L. Dill, A.J. Hu, and H. Wong-Toi. Checking for language inclusion using simulation relations. In *Proc. 3rd CAV*, LNCS 575, pp. 255–265, 1991.

[DJW97] S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games. In *Proc. 12th LICS*, pp. 99–110, 1997.

[EJ91] E.A. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd FOCS*, pp. 368–377, 1991.

[Fra86] N. Francez. *Fairness*. Texts and Monographs in Computer Science. Springer-Verlag, 1986.

[GL94] O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Trans. on Programming Languages and Systems*, 16(3):843–871, 1994.

[HHK95] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. 36th FOCS*, pp. 453–462, 1995.

[HKR97] T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *Proc. 8th Conference on Concurrency Theory*, LNCS 1243, pp. 273–287, 1997.

[Hoj96] R. Hojati. *A BDD-based Environment for Formal Verification of Hardware Systems*. PhD thesis, University of California at Berkeley, 1996.

[HR00] T. Henzinger and S. Rajamani. Fair bisimulation. In *Proc. 4th TACAS*, LNCS 1785, pp. 299–314, 2000.

[JW96] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to the monadic second order logic. In *Proc. 7th Conference on Concurrency Theory*, LNCS 1119, pp. 263–277, 1996.

[Kam85] M. Kaminski. A classification of ω-regular languages. *TCS*, 36:217–229, 1985.

[KSV96] O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. In *Proc. 11th LICS*, pp. 322–333, 1996.

[Kur94] R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.

[KV98a] O. Kupferman and M.Y. Vardi. Modular model checking. In *Proc. Compositionality Workshop*, LNCS 1536, pp. 381–401, 1998.

[KV98b] O. Kupferman and M.Y. Vardi. Verification of fair transition systems. *Chicago Journal of TCS*, 1998(2).

[KV98c] O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proc. 30th STOC*, pp. 224–233, 1998.

[LT87] N. A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proc. 6th PODC*, pp. 137–151, 1987.

[McM93] K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd International Joint Conference on Artificial Intelligence*, pp. 481–489, 1971.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, LNCS 92, 1980.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, Berlin, January 1992.

[MS72] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proc. 13th SWAT*, pp. 125–129, 1972.

[Niw97] D. Niwiński. Fixed point characterization of infinite behavior of finite-state systems. *TCS*, 189(1–2):1–69, December 1997.

[NW98] D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Symposium on Theoretical Aspects in Computer Science*, LNCS 1373, 1998.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th ICALP*, LNCS 194 pp. 15–32, 1985.

[Rab69] M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.

[Rab70] M.O. Rabin. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, pp. 1–23. North Holland, 1970.

[SV89] S. Safra and M.Y. Vardi. On ω-automata and temporal logic. In *Proc. 21st STOC*, pp. 127–137, 1989.

[Tho90] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pp. 165–191, 1990.

[Wag79] K. Wagner. On ω-regular sets. *Information and Control*, 43:123–177, 1979.