Fair Equivalence Relations

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

[1] Hebrew University, School of Eng. and Computer Science, Jerusalem 91904, Israel
orna@cs.huji.ac.il, URL: http://www.cs.huji.ac.il/~orna
[2] Weizmann Institute of Science, Department of CS, Rehovot 76100, Israel
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.
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 [30, 31]. 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* [5, 9, 6]. 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

* This author is grateful to Zohar Manna for introducing him to logic in computer science. Supported in part by NSF grants CCR-9700061 and CCR-9988322, and by a grant from the Intel Corporation.

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 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* [34]. The branching approach is stronger, in the sense that bisimulation implies trace equivalence but not vice versa [34, 38]. There is a slightly weaker notion of equivalence in the branching approach. Instead of considering bisimulation, consider *two-way simulation*. Again the two states have the same set of behaviors. We can show a system with two states s and s' such that s and s' are not bisimilar but they are 2-way similar [34].

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 [1]. 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. [29, 28])[1].

The theory behind trace equivalence and bisimulation of finite state systems 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 [33, 26]. 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 [5, 21]. The problem of deciding whether two states are bisimilar is PTIME-complete [35, 4], and a witnessing relation for bisimulation can be computed using a symbolic fixpoint procedure [32, 17]. Similar results hold for trace containment and simulation. The computational advantage of simulation makes it a useful precondition to trace containment [10].

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 [30, 15]. It is not hard to extend the linear approach to account for

[1] Note that both the specification and the implementation describe the possible behaviors of the system, but the specification is more abstract than the implementation, which may have less behaviors. Sometimes we may want the implementation to have exactly all the behaviors of the specification. Then, we use trace equivalence or bisimulation relations, and the implementation being less abstract from the specification is reflected in things like a richer set of observable variables.

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* [16], *game-bisimulation* [18, 19], and $\forall$-*bisimulation* [29]. In a bisimulation relation between $\S$ and $\S'$ 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 [18]. The difference in the distinguishing power of the definitions is also reflected in their logical characterization: while $\exists$-bisimulation corresponds to fair-CTL* (that is, two systems are $\exists$-bisimilar iff they agree on all fair-CTL* formulas, where path quantifiers range over fair computations only [8]), game-bisimulation is 'stronger', that is, game-bisimulation can distinguish between systems that agree on all fair-CTL* formulas. Thus, unlike the non-fair case, where almost all modal logics corresponds to bisimulation, here different relations correspond to different logics [2] [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 [26], NP-complete for $\forall$-bisimulation [20], and PTIME-complete for game-bisimulation [18, 19].

There are various types of fairness conditions with which we can augment labeled state-transition systems [30]. 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 [42]. 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

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

context of state-transition systems. Rabin conditions were introduced by Rabin and were used to prove that the logic S2S is decidable [39]. 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 [14]. As we formally define in Section 2, Rabin, Streett, and parity conditions are characterized by their *index*, i.e. 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 [42]. 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 [20, 18, 26], but nothing is known about their expressive power, and about the possibilities of translations among them. For example, 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 [40, 13, 36, 37]. 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 verification 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.

2 Definitions

2.1 Fair State-Transition Systems

A *fair state-transition system* (*system*, for short) $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ 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 *trace* $L(\pi) = L(w_0) \cdot L(w_1) \cdot L(w_2) \cdots \in \Sigma^\omega$. 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 $inf(\pi) \cap \alpha = \emptyset$.
- *generalized Büchi* (*justice*), where $\alpha \subseteq 2^W$, and π is fair iff $inf(\pi) \cap F \neq \emptyset$ for every $F \in \alpha$.
- *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 or generalized Büchi 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, generalized Büchi, 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 $\bigcup_{w \in W_0} \mathcal{T}(S^w)$.

2.2 Equivalence between Systems

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 $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ and $S' = \langle \Sigma, W', W_0', R', L', \alpha' \rangle$, 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 [34] 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 $H(s, s')$.

A relation $H \subseteq W \times W'$ is a *simulation relation* from S to S' if for every pair $\langle w, w' \rangle \in H$ we require that only conditions 1 and 2 above hold.

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 [16] 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-computation π in S has a fair w'-computation π' in S' with $H(\pi, \pi')$.
3. Each fair w'-computation π' in S' has a fair w-computation π in S with $H(\pi, \pi')$.

[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 $\Sigma = 2^{AP \cap AP'}$.

Game-Bisimulation [18, 19] Game bisimulation is defined by means of a game between a protagonist against an antagonist. The *positions* of the game are pairs 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 antagonist 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 antagonist 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'.

Note that the antagonist can choose to move either in W or in W'. As we shall see, in the proofs in this paper we use only plays in which the antagonist chooses between W and W' in the first move and then continues with the same set ad infinitum.

We specialize the notion of outcome to our purposes as following. Given an infinite sequence $\pi = w_0 \cdot w_1 \cdot w_2 \cdots$ of states in W, the *outcome* of τ on π is the infinite sequence $\tau[\pi] = s'_0 \cdot s'_1 \cdot s'_2 \cdots$ of states in W', where $w'_i = \tau(w_0, s'_0, w_1, s'_1, \ldots, w_{i-1}, s'_{i-1}, w_i)$ for all $i \geq 0$. Similarly, the outcome of τ on a sequence $\pi' = w'_0 \cdot w'_1 \cdot w'_2 \cdots$ of states in W' is the infinite sequence $\pi = w_0 \cdot w_1 \cdot w_2 \cdots$ of states in W, where $w_i = \tau(s_0, w'_0, s_1, w'_1, \ldots, s_{i-1}, w'_{i-1}, w'_i)$ for all $i \geq 0$.

$\forall$-Bisimulation [29, 12] 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 [18], the other direction is not true. Furthermore, if we assume that every state has some fair computation starting from it then if H is an $\exists$-bisimulation relation, then H is also a bisimulation relation.

For all types β of bisimulation relations (that is $\beta \in \{\exists, \mathit{game}, \forall\}$), a β-bisimulation relation H is a *β-bisimulation between the systems $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ and $S' = \langle \Sigma, W', W'_0, R', L', \alpha' \rangle$* 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 specified in a conventional temporal logic (in particular, CTL* and μ-calculus) [5, 21]. When we add fairness, the logical characterization becomes less robust: $\exists$-simulation corresponds to fair-CTL*, and game-bisimulation can distinguish between systems even where fair-CTL* cannot [3, 16, 18, 19].

A relation $H \subseteq W \times W'$ is a *$\exists$-simulation relation* from S to S' if for every $\langle w, w' \rangle \in H$ we require only that conditions 1 and 2 for H being a $\exists$-bisimulation hold. A relation $H \subseteq W \times W'$ is a *$\forall$-simulation* from S to S' if H is a simulation relation from S to S' and for every $\langle w, w' \rangle \in H$ we require only that condition 2 for H being a $\forall$-bisimulation holds. A relation $H \subseteq W \times W'$ is a *game-simulation relation* from S to S' if we restrict the moves of the antagonist to choose only states from S and for every pair $\langle w, w' \rangle \in H$ we require that conditions 1 and 2 for H being a game-bisimulation hold (under the weaker interpretation of game). For $\beta \in \{\exists, \mathit{game}, \forall\}$, 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 all the behaviors of S.

Let γ and γ' be two fairness types and let β be a type of bisimulation. We say that γ is *at least as β-strong as* γ' (or γ is *at least as strong as* γ' in the context of β-bisimulation) if for every γ'-system S', there exists a γ-system S such that S and S' are β-bisimilar. We say that γ is *β-stronger* than γ' if γ is at least as β-strong as γ' but γ' is not at least as β-strong as γ. We also say that γ' is *β-weaker* than γ if γ is β-stronger than γ'.

As $\forall$-bisimulation implies game-bisimulation, and game-bisimulation implies $\exists$-bisimulation the following theorem is easy to prove.

Theorem 1. *Let γ_1 and γ_2 be two fairness types.*

1. *If γ_1 is at least as game-strong as γ_2, then γ_1 is at least as $\exists$-strong as γ_2.*
2. *If γ_1 is at least as $\forall$-strong as γ_2, then γ_1 is at least as game-strong as γ_2.*

Proof. Suppose γ_1 is at least as game-strong as γ_2, we have to show that γ_1 is at least as $\exists$-strong as γ_2. Given a γ_2-system S_2, we have to find a γ_1-system S_1 that is $\exists$-bisimilar to S_2. As γ_1 is at least as game-strong as γ_2 there exists a γ_1-system S_1 and a game-bisimulation H between S_1 and S_2. It follows that H

is also an $\exists$-bisimulation between S_1 and S_2. In particular, S_1 is $\exists$-bisimilar to S_2.

The proof that if γ_1 is at least as $\forall$-strong as γ_2, then it is also at least as game-strong as γ_2 follows the same lines.

It is easy to see that bisimulation implies trace equivalence. As mentioned every type β of bisimulation, implies bisimulation. Hence, a β-bisimulation relation between systems S and S' implies fair trace equivalence of S and S'. The other direction, however, is not true [34, 18]. There exist systems that have the same set of traces (fair traces) and are not bisimilar (β-bisimilar, for all types of bisimulation β). Formally, we have the following.

Theorem 2. [16, 18, 34] *For $\beta \in \{\exists, game, \forall\}$ if S and S' are β-bisimilar then $\mathcal{T}(S) = \mathcal{T}(S')$. The inverse implication does not hold.*

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 $\exists$-Bisimulation

When the notion of equivalence is trace equivalence, it follows from the theory of automata on infinite words (cf. [7]) that co-Büchi systems are weaker than Büchi systems, which are as strong as parity, Rabin, and Streett systems. When tree automata are considered, 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 [40, 13, 36, 37]. In this section we show that the expressiveness hierarchy in the context of $\exists$-bisimulation is located between the hierarchies of word and tree automata.

3.1 Büchi and Co-Büchi Are Weak

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 [40]. Our proofs use the notion of maximal models [16, 27]. A system M_ψ is a *maximal model* for an $\forall$CTL* formula ψ if $M_\psi \models \psi$ and for every system M we have that $M \leq_\exists M_\psi$ iff $M \models \psi$. We show that there is no Büchi system that is $\exists$-bisimilar to the maximal model of the formula $\forall\Diamond\Box p$. Hence, the following theorem.

Theorem 3. *Büchi is not at least as $\exists$-strong as co-Büchi.*

Proof. We show that there is a co-Büchi system that has no $\exists$-bisimilar Büchi system. Consider the co-Büchi system $M = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ appearing in Figure 1. The set $W_0 = \{q_0, q_1\}$ is the set of states marked by incoming arrows. A pair $(w, w') \in R$ iff there is an arrow from w to w'. The alphabet Σ is 2^{AP}

where $AP = \{p\}$. The labeling function L associates with every state q the letter $\sigma \in \Sigma$ such that proposition p is in σ iff proposition p marks state q. Finally, a state q is in the set of accepting states if it is marked by a double circle, i.e. the fairness condition of M is $\{q_0\}$ (a computation is fair iff it visits q_0 finitely often). We show that the system M is a maximal model for the $\forall$CTL* formula $\psi = \forall\Diamond\Box p$ ("in all computations, eventually always p"). It is easy to see that $M \models \psi$. Indeed, every fair computation of M eventually remains in state q_1 where p holds. We show that for every system $M' = \langle \Sigma, W', W'_0, R', L', \alpha' \rangle$, we have that $M' \leq_\exists M$ iff $M' \models \forall\Diamond\Box p$ [16, 25]. Given a system M' that satisfies ψ, it is easy to see that a relation $H \subseteq W' \times \{q_0, q_1\}$ that maps a reachable state w' with $p \in L'(w')$ to q_1 and maps a reachable state w' with $p \notin L'(w')$ to q_0 is a fair $\exists$-simulation from M' to M. Consider a system M' such that $M' \leq_\exists M$ with the simulation relation $H \subseteq W' \times \{q_0, q_1\}$. For every initial state $w'_0 \in W'_0$ it must be the case that $H(w'_0, q_0)$ or $H(w'_0, q_1)$. Consider a fair w'_0-computation $w'_0, w'_1, \ldots$ of M'. There exists a fair computation $q_{i_0}, q_{i_1}, \ldots$ of M such that forall j we have $H(w'_j, q_{i_j})$. A fair computation of M eventually remains in state q_1. As $H(w', q)$ implies $L'(w') = L(q)$, we conclude that every fair w'_0-computation satisfies ψ.

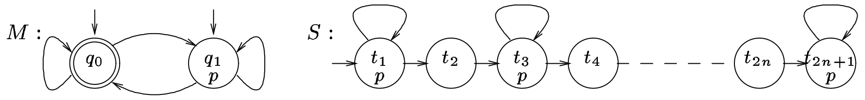

Fig. 1. Maximal co-Büchi model of $\psi = \forall\Diamond\Box p$ and a system satisfying ψ

We show that there is no Büchi system that is $\exists$-bisimilar to M. Assume, by way of contradiction, that such a system B exists. Note that then, $B \leq_\exists M$ and $M \leq_\exists B$. Let n be the number of states in B, and let α be its Büchi fairness condition. Consider the system S with trivial fairness (all computations are fair) appearing in Figure 1. It is easy to see that S satisfies ψ, thus $S \leq_\exists M$. Since $M \leq_\exists B$, then, from transitivity of $\exists$-simulation, $S \leq_\exists B$. Let H be an $\exists$-simulation from S to B. Consider the computation s_1^ω in S. Since s_1^ω is fair, there must be a fair computation $\pi^1 = t_1^1, t_2^1, \ldots$ in B such that $H(s_1, t_i^1)$ for all $i \geq 1$. Let $t_{i_1}^1 \in \alpha$ be the first visit of π^1 to α (since π^1 is fair, such a visit must exist). Consider now the fair computation s_1, s_2, s_3^ω in S. Since $H(s_1, t_{i_1}^1)$, there must be a fair computation $\pi^2 = t_{i_1}^1, t_1^2, t_2^2, \ldots$ in B such that $H(s_2, t_1^2)$ and $H(s_3, t_i^2)$ for all $i \geq 2$. Let $t_{i_2}^2 \in \alpha$ be the first visit of π^2 to α. In a similar way, we can generate a sequence $t_{i_1}^1, t_{i_2}^2, \ldots, t_{i_{n+1}}^{n+1}$ of states such that for all $j \geq 1$, the state $t_{i_{j+1}}^{j+1}$ is reachable in B from the state $t_{i_j}^j$, $H(s_{2j-1}, t_{i_j}^j)$ and the successor of $t_{i_j}^j$ on the computation to $t_{i_{j+1}}^{j+1}$ is associated by H to s_{2j}, where p does not hold. Since B has n states, there must be a state repeating twice in this sequence (that is, there must be j_1 and j_2 for which $t_{i_{j_1}}^{j_1} = t_{i_{j_2}}^{j_2}$). It follows that B contains a fair computation that visits infinitely many states in which p does not hold (indeed, between every two states in the sequence there is a state related by H to a state in which p does not hold). Hence. B does not satisfy ψ. Since M is a maximal model for ψ, if follows that $B \not\leq_\exists M$, contradicting the assumption that $B \leq_\exists M$.

Note that Theorem 3 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 [16, 24].

We now claim that co-Büchi is not at least as ∃-strong as Büchi.

Theorem 4. *Co-Büchi is not at least as ∃-strong as Büchi.*

Proof. We show that there is a Büchi system that has no ∃-bisimilar co-Büchi system. Consider the Büchi system $M = \langle 2^{\{p\}}, \{q_0, q_1\}, \{q_0, q_1\}, R, L\rangle$ appearing in Figure 2. We show that the system M is a *maximal model* for the ∀CTL* formula $\psi = \forall\Box\Diamond p$ ("in all computations, p holds infinitely often"). It is easy to see that $M \models \psi$. Indeed, every fair computation of M visits q_0, where p holds, infinitely often. We show that for every system $M' = \langle \Sigma, W', W'_0, R', L', \alpha'\rangle$, we have that $M' \leq_\exists M$ iff $M' \models \psi$ [16]. Given a system M' that satisfies ψ, it is easy to see that a relation $H \subseteq W' \times \{q_0, q_1\}$ that maps a reachable state w' with $p \in L'(w')$ to q_0 and maps reachable a state w' with $p \notin L'(w')$ to q_1 is a fair ∃-simulation from M' to M. Consider a system M' such that $M' \leq_\exists M$ with the simulation relation $H \subseteq W' \times \{q_0, q_1\}$. For every initial state $w'_0 \in W'_0$ it must be the case that $H(w'_0, q_0)$ or $H(w'_0, q_1)$. Consider a fair w'_0-computation $w'_0, w'_1, \ldots$ of M'. There exists a fair computation $q_{i_0}, q_{i_1}, \ldots$ of M such that forall j we have $H(w'_j, q_{i_j})$. A fair computation of M visits state q_0 infinitely often. As $H(w', q)$ implies $L'(w') = L(q)$, we conclude that every fair w'_0-computation satisfies ψ.

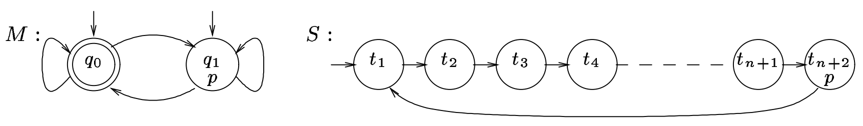

Fig. 2. Maximal Büchi model of $\psi = \forall\Box\Diamond p$ and a system satisfying ψ

We show that there is no co-Büchi system that is ∃-bisimilar to M. Assume, by way of contradiction, that such a system C exists. Let n be the number of states in C, and let α be its co-Büchi fairness condition. Consider the system S with trivial fairness appearing in Figure 2. Since S satisfies ψ, then $S \leq_\exists M$, implying that $S \leq_\exists C$. Let H be a ∃-simulation from S to C, and let t_1 be such that $H(s_1, t_1)$. Consider the computation $\pi = (s_1, s_2, \ldots, s_{n+2})^\omega$ in S . Since π is fair, there must be a fair computation $\pi' = t_1, t_2, \ldots$ in C such that $H(\pi, \pi')$. Let $t_j \in \alpha$ be the last visit of π to α. There is $l \geq j$ such that for all $1 \leq i \leq n+1$, we have $H(s_i, t_{l+i})$. Since C has n states, there must be a state repeating twice in $t_{l+1}, \ldots, t_{l+n+1}$, say $t_{l+j_1} = t_{l+j_2}$, with $j_1 < j_2$. Then, the computation $t_1, t_2, \ldots, t_l, t_{l+j_1}(t_{l+j_1+1}, \ldots, t_{l+j_2})^\omega$ is a fair computation in C in which p holds only finitely often (indeed, all the states in $t_{l+j_1+1}, \ldots, t_{l+j_2}$ are related by H to states in which p does not hold). Hence, C does not satisfy ψ. Since M is a maximal model for ψ, it follows that $C \not\leq_\exists M$, contradicting the assumption that $C \leq_\exists M$.

It follows that Büchi is not as $\exists$-strong as co-Büchi, and co-Büchi is not as $\exists$-strong as Büchi. Since parity, Rabin, and Streett are at least as $\exists$-strong as Büchi and co-Büchi, it follows from Theorems 3 and 4 that parity, Rabin, and Streett are all $\exists$-stronger than Büchi and co-Büchi.

3.2 Rabin[1] Is Strong

So far things seem to be very similar to tree automata, where Büchi and co-Büchi conditions are incomparable [40]. In particular, the ability of the Büchi condition to define maximal models for $\forall$CTL and its inability to define maximal models for $\forall$CTL* seems related to the ability to translate CTL formulas to Büchi tree automata [44] and the inability to translate CTL* formulas to Büchi tree automata (as follows from Rabin's result [40]). 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 [13, 36, 37]. We now show that, surprisingly, in the context of $\exists$-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 $\forall$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.

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. In order to see that $S \sim_\exists S'$, consider the relation $H = \{\langle w, (w,i) \rangle : w \in W \text{ and } 0 \leq i \leq k\}$. We prove that H is an $\exists$-bisimulation between S and S'.

Consider a pair $\langle w, (w,i) \rangle \in H$. By the definition of L', we have that $L(w) = L'((w,i))$. Given a fair w-computation $\pi = w_0, w_1, \ldots$ in S, let j be the pair in α for which $inf(\pi) \cap G_j \neq \emptyset$ and $inf(\pi) \cap B_j = \emptyset$. Let k be such that the set B_j is not visited in beyond w_k (that is, for all $k' \geq k$, we have $w_k \notin B_j$). Consider the path $\pi' = (w_0, i), (w_1, 0) \ldots, (w_k, 0), (w_{k+1}, j), (w_{k+2}, j), \ldots$. Since B_j is not visited beyond w_k this is a valid path of S'. Clearly, $H(\pi, \pi')$, and the idle copy is visited in π' only finitely often. Also, since π is fair, π' visits infinitely many states in $G_j \times \{j\}$, and hence, π' is fair. Consider now a fair path $\pi' = (w_0, i_0), (w_1, i_1), \ldots$ in S'. By similar considerations, it is not hard to see that the path $\pi = w_0, w_1, \ldots$, obtained from π' by replacing a state (w_j, i_j) by the state w_j is a valid and fair path in S, with $H(\pi, \pi')$. Hence, H is an $\exists$-bisimulation relation. Finally, since H is such that for all $w_0 \in W_0$, we have $H(w_0, (w_0, 0))$, the relation H is an $\exists$-bisimulation between S and S', and we are done.

Runs of a Rabin[k] automaton on different words are independent of each other: each run can chose a pair in α with respect to which the run is going to be fair. The choice of the pair is made when the run starts, and there is no need for the run to "change its mind" and switch to a different pair. This is why the transformation of Rabin[k] word automata to Rabin[1] (or Büchi) automata is rather simple and it involves a split to k copies of the automaton, each corresponding to a pair in α, with no need for an "idle copy" [7]. On the other hand, 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.*

Proof. Let $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ be a Streett 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 $I \subseteq \{1, \ldots, k\}$, let $W_I = (W \setminus \bigcup_{i \notin I} G_i) \times \{I\} \times I$. Then, $W' = W \cup \bigcup_{I \subseteq \{1,\ldots,k\}} W_I$.
- $W_0' = W_0$.
- In order to define R', we first define the partial function $f : W \times 2^{\{1,\ldots,k\}} \times \{1, \ldots, k\} \to \{1, \ldots, k\}$ below. For every $w \in W$, $I \subseteq \{1, \ldots, k\}$, and $j \in I$, let:

$$f(w, I, j) = \begin{cases} min(I) & j = max(I) \text{ and } w \in B_j \\ min(\{p \mid p \in I \text{ and } p > j\}) & j \neq max(I) \text{ and } w \in B_j \\ j & w \notin B_j \end{cases}$$

  Thus, $f(w, I, j)$ returns a value in I. If $w \notin B_j$, then f returns j. If $w \in B_j$, then f returns the minimal value in I that is greater than j. If j is maximal in I, f returns the minimal value in I.
  For every $I \subseteq \{1, \ldots, k\}$ and $j \in I$, we define

$$R_{I,j} = \left\{ \begin{array}{c} \langle w, (w', I, min(I)) \rangle, \langle (w, I, j), w' \rangle, \\ \langle (w, I, j), (w', I, f(w, I, j)) \rangle \end{array} \middle| \langle w, w' \rangle \in R \right\} \cap (W' \times W').$$

  Then, $R' = R \cup \bigcup_{I \subseteq \{1,\ldots,k\}} \bigcup_{j \in I} R_{I,j}$.
- For all $w \in W$, $I \subseteq \{1, \ldots, k\}$, and $j \in I$, we have $L'((w, I, j)) = L(w)$.
- $\alpha' = \{\langle (\bigcup_{I \subseteq \{1,\ldots,k\}} B_{min(I)} \times \{I\} \times \{min(I)\}) \cap W', W \rangle\}$.

The system S' consists of an "idle copy" of S that branches into 2^k "groups". The idle copy contains all the states in W. Each of the groups is associated with a set $I \subseteq \{1, \ldots, k\}$ of indices, and it contains $|I|$ copies of S. These copies do not contain states in G_i, for $i \notin I$, they are marked by I, and by an index j in I, which we refer to as the "stage" of the state. We log the visits to each of the B_i, for $i \in I$, according to the order on I. The stage is used to indicate the index of the pair $\langle G_i, B_i \rangle$ that has to be visited next. Once B_j is visited the stage is advanced. In case $j = max(I)$, after visiting B_j the stage is updated to $min(I)$ and the process is iterated ad infinitum. So far, the idea is similar to the translation of Streett systems to Büchi systems in the linear paradigm [41]. In order to fit the branching paradigm, we allow a computation of S' to return to the idle copy, where it can choose between staying in the idle copy or progressing to one of the other 2^k groups. The acceptance condition forces a fair computation to visit the idle copy only finitely often, forcing the computation to eventually get trapped in a group associated with some subset I of indices. There, the computation cannot visit states in the sets G_i, for $i \notin I$ (W_I does not contain these states), and it is forced by the acceptance condition to visit every set B_j, for $j \in I$, infinitely often.

In order to see that $S \sim_\exists S'$, consider the relation

$$H = \{\langle w, (w, I, j)\rangle : w \in W,\ I \subseteq \{1, \ldots, k\},\ \text{and } j \in I\} \cup \{\langle w, w\rangle : w \in W\}.$$

We prove that H is an $\exists$-bisimulation between the systems S and S'. Consider a pair $\langle w, (w, I, j)\rangle \in H$ (the arguments for a pair $\langle w, w\rangle \in H$ are similar). By the definition of L', we have that $L(w) = L'((w, I, j))$. Given a fair w-computation $\pi = w_0, w_1, ...$ in S, let M be the set of indices i in $\{1, \ldots, k\}$ such that B_i is visited infinitely often in π. Since π is fair, the states from G_i, for $i \notin M$, are visited only finitely often. Let l be such that states from G_i, for $i \notin M$, do not appear after l (that is, for all $l' \geq l$ and $i \notin M$, we have $w_{l'} \notin G_i$). Consider the computation $\pi' = (w_0, I, j)$, w_1, $\ldots$, w_l, $(w_{l+1}, M, min(M))$, $(w_{l+2}, M, f(w_{l+1}, M, min(M)))$, $\ldots$. Since G_i, for $i \notin M$, is not visited beyond w_l, the computation π' is a valid computation of S', satisfying $H(\pi, \pi')$. To see that π' is fair, note that states in W are visited in π' only finitely often. Furthermore, since π visits every set B_i, for $i \in M$, infinitely often, π' gets to stage $min(M)$ and visits there states from $B_{min(M)}$ infinitely often. Consider now a computation $\pi' = (w_0, I_0, j_0), (w_1, I_1, j_1), \ldots$ in S' (the case where π' contains states in W is similar). As in the Rabin case, the computation $\pi = w_0, w_1, ...$, obtained from π' by replacing a state (w_l, I_l, j_l) by the state w_l, is a valid and fair computation in S, with $H(\pi, \pi')$. We conclude that H is an $\exists$-bisimulation relation. Since for all $w_0 \in W_0$, we have $H(w_0, w_0)$, the relation H is an $\exists$-bisimulation between S and S'.

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 [41][4]. Since $\exists$-bisimulation implies trace equivalence, it follows that these blow ups are tight also for the $\exists$-bisimulation case.

3.3 Streett[1] Is Weak

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 as $\exists$-strong as Rabin[1]. The question arises then, is whether Streett[2] is stronger than Streett[1]. It turns out that we can combine the arguments in Theorems 3 and 4 to a proof that Streett[2] is indeed stronger than Streett[1]. Formally, we have the following theorem.

Theorem 5. *Streett*[1] *is not at least as $\exists$-strong as Streett*[2].

Proof. We combine the ideas of Theorems 3 and 4. We show that there is a Rabin[1] system that has no $\exists$-equivalent Streett[1] system. Consider the Rabin[1] system M appearing in Figure 3. The fairness condition of M is $\{\langle G, B\rangle\}$, where

[4] [41] 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.

$G = \{q_0\}$ and $B = \{q_1, q_3\}$ (i.e., a computation of M is fair iff q_0 is visited infinitely often and q_1 and q_3 are visited finitely often). We show that the system M is a maximal model for the $\forall$CTL* formula $\psi = \forall\Diamond\Box r \wedge \forall\Box\Diamond p$. It is easy to see that $M \models \psi$. Indeed, every fair computation of M eventually remains in states q_0 and q_2, thus it satisfies $\forall\Diamond\Box r$, it also visits state q_0 infinitely often, thus satisfying $\forall\Box\Diamond p$. We show that for every system $M' = \langle \Sigma, W', W'_0, R', L', \alpha' \rangle$ we have that $M' \leq_\exists M$ iff $M' \models \psi$. Given a system M' that satisfies ψ, it is easy to see that a relation $H \subseteq W' \times \{q_0, q_1, q_2, q_3\}$ that maps a reachable state w' as follows is a $\exists$-simulation from M' to M.

- If $p, r \in L'(w')$ then w' is mapped to q_0.
- If $p \in L'(w')$ and $r \notin L'(w')$ then w' is mapped to q_1.
- If $r \in L'(w')$ and $p \notin L'(w')$ then w' is mapped to q_2.
- If $p, r \notin L'(w')$ then w' is mapped to q_3.

Consider a system M' such that $M' \leq_\exists M$ with the simulation relation $H \subseteq W' \times \{q_0, q_1, q_2, q_3\}$. For every initial state $w'_0 \in W'_0$ there exists some state q_i such that $H(w'_0, q_i)$. Consider a fair w'_0-computation $w'_0, w'_1, \ldots$ of M'. There exists a fair computation $q_{i_0}, q_{i_1}, \ldots$ of M such that forall j we have $H(w'_j, q_{i_j})$. A fair computation of M eventually remains in states q_0 and q_2 and visits q_0 infinitely often. As $H(w', q)$ implies $L'(w') = L(q)$, we conclude that every fair w'_0-computation satisfies ψ.

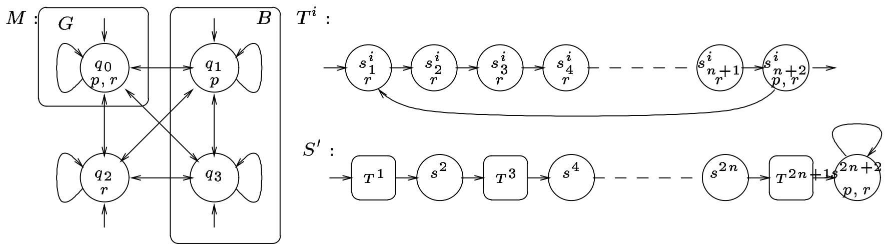

Fig. 3. Maximal Rabin[1] model of $\psi = \forall\Diamond\Box r \wedge \forall\Box\Diamond p$ and a system satisfying ψ

We show that there is no Streett[1] system that is $\exists$-equivalent to M. Assume, by way of contradiction, that such a system R exists. Let n be the number of states in R, and let $\alpha = \{\langle G', B' \rangle\}$ be its Streett fairness condition. Consider the system S' with trivial fairness appearing in Figure 3. It is easy to see that S' satisfies ψ, hence $S' \leq_\exists R$. Let H be an $\exists$-simulation from S' to R. Each block T^i is equivalent to the system S from Figure 2. We follow the proof of Theorem 4 and show that when R simulates the computation inside T^i, it must visit B' infinitely often. The structure of S' resembles that of the system S from Figure 1. We follow the proof of Theorem 3 and show that R contains a computation that visits B' infinitely often and violates $\Diamond\Box r$. Let t_1 be such that $H(s^1_1, t_1)$. Consider the computation $\pi = (s^1_1, s^1_2, \ldots, s^1_{n+2})^\omega$ in S'. Since π is fair, there must be a fair computation $\pi^1 = t_1^1, t_2^1, \ldots$ in R such that $H(\pi, \pi^1)$. As in the proof of Theorem 4, the computation π^1 cannot visit G' finitely often. So, π^1 visits B' infinitely often. Let $t_{i_1}^1 \in B'$ be the first visit to B'. Let s_p^1 be such that $H(s_p^1, t_{i_1}^1)$. Consider now the fair computation $\pi' = s_p^1, \ldots, s_{n+2}^1, s^2, (s_1^3, \ldots, s_{n+2}^3)^\omega$. Since $H(s_p^1, t_{i_1}^1)$ there must be a fair computation $\pi^2 = t_{i_1}^1, t_1^2, t_2^2, \ldots$ in R such that $H(\pi', \pi^2)$. Again π^2 must visit B' infinitely often. Let $t_{i_2}^2 \in B'$ be the first visit to B'. In a similar manner, we can generate the sequence $t_{i_1}^1, t_{i_2}^2, \ldots, t_{i_{n+1}}^{n+1}$ of states such that for all $j \geq 1$, the state $t_{i_{j+1}}^{j+1}$ is reachable from $t_{i_j}^j$, and on the computation between $t_{i_j}^j$ and $t_{i_{j+1}}^{j+1}$ there is a state associated by H with s^{2j}, where both p and r do not hold. Since R has n states, there must be a state repeating twice in this sequence. It follows that R contains a fair computation that visits infinitely many states in which r does not hold. Hence, R does not satisfy ψ, contradicting M's maximality.

3.4 Parity and Generalized Büchi

Before we summarize our results, let us refer also to the parity and the generalized Büchi fairness conditions. 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]. Also 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 condition and vice versa, implying that parity[2] is not at least as $\exists$-strong as parity[3].

On the other hand, generalized Büchi is as $\exists$-strong as Büchi. To prove this, we prove in Section 4 that every generalized Büchi system S has an $\forall$-bisimilar Büchi system S'. Since $\forall$-bisimulation is stronger than $\exists$-bisimulation, it follows that S and S' are also $\exists$-bisimilar, thus Büchi is at least as $\exists$-strong as generalized Büchi.

To sum up, we have the following.

Theorem 6. *For every fairness type γ, the types Rabin*[1]*, Streett*[2]*, and parity*[3] *are all at least as $\exists$-strong as γ, and are all $\exists$-stronger than Büchi, co-Büchi, Streett[1], and parity[2].*

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

In this section we 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.

4.1 Tree Automata

Let D be a finite set of directions. A D-tree is a set $T \subseteq D^*$ such that if $x \cdot d \in T$, for $x \in D^*$ and $d \in D$, then also $x \in T$. The elements of T are called *nodes*, and the empty word ϵ is the *root* of T. For every $x \in T$, the nodes $x \cdot d$, for $d \in D$, are the *successors* of x. We assume that each node has at least one successor. A *path* π of the tree T is a minimal set $\pi \subseteq T$ such that $\epsilon \in \pi$ and for every $x \in \pi$, exactly one successor of x is in π. Given an alphabet Σ, a *Σ-labeled D-tree* is a pair $\langle T, V \rangle$, where T is a D-tree and $V : T \to \Sigma$ maps each node of T to a letter in Σ. When Σ and D are not important or clear from the context, we call $\langle T, V \rangle$ a labeled tree. We sometimes extend V to paths and use $V(\pi)$ to denote the infinite word $V(\pi[0]) \cdot V(\pi[1]) \cdot V(\pi[2]) \cdots$.

Tree automata run on Σ-labeled trees. We define here a special type of tree automata, called *loose tree automata*. Unlike conventional tree automata, 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. Formally, a loose tree automaton is $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, where Σ is an alphabet, Q is a set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, and α is an acceptance condition. When $\mathcal{A}$ runs on $\langle T, V \rangle$ and it visits a node x with $V(x) = \sigma$ at state q, then $\delta(q, \sigma) = S$ means that $\mathcal{A}$ should send to all the successors of x copies in states in S. Formally, a *run* of $\mathcal{A}$ on $\langle T, V \rangle$ is a Q-labeled tree $\langle T, r \rangle$, where $r(\epsilon) \in Q_0$, and for all $x \in T$ and $d \in D$ with $x \cdot d \in T$, we have $r(x \cdot d) \in \delta(r(x), V(x))$.

Given a run $\langle T, r \rangle$ and a path $\pi \subseteq T$, we define $inf(r(\pi))$ as the set of states visited infinitely often along π. Formally, we define $inf(r(\pi)) = \{q :$ for infinitely many $x \in \pi$, we have $r(x) = q\}$. We say that π is *fair* in r if $r(\pi)$ satisfies the acceptance condition α, which is determined according to $inf(r(\pi))$. A run $\langle T, r \rangle$ is accepting if all the paths $\pi \subseteq T$, are fair in r. For example, a run $\langle T, r \rangle$ of a Büchi automaton is accepting if r visits infinitely many states in α along each path. An automaton $\mathcal{A}$ accepts $\langle T, V \rangle$ iff there exists an accepting run of $\mathcal{A}$ on $\langle T, V \rangle$. Given a loose tree automaton $\mathcal{A}$, the set $\mathcal{L}(\mathcal{A})$ is the set of all labeled trees accepted by $\mathcal{A}$. For a word $\rho = \sigma_0 \cdot \sigma_1 \cdots$ in Σ^ω and a state $q \in Q$, a sequence $\pi = q_0, q_1, \ldots$ in Q^ω is a (q, ρ)-computation of $\mathcal{A}$, if $q_0 = q$ and for all $i \geq 0$, we have $q_{i+1} \in \delta(q_i, \sigma_i)$.

4.2 Game-Bisimulation between Loose Tree Automata

We now show that the strength hierarchy of fairness conditions induced by game-bisimulation coincides with the one of loose tree automata. For that, we first define game-bisimulation between loose tree automata. Consider two loose tree automata $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ and $\mathcal{A}' = \langle \Sigma, Q', Q'_0, \delta', \alpha' \rangle$. As with fair state-transition systems, we define a game between a protagonist against an antagonist, whose positions are pairs in $Q \times Q'$. A *strategy* τ for the protagonist is a partial function from $(Q \times Q')^* \times \Sigma \times (Q \cup Q')$ to $(Q' \cup Q)$, such that for all $\rho = (q_0, q'_0), (q_1, q'_1), \ldots, (q_m, q'_m) \in (Q \times Q')^*$, $q \in Q$, $q' \in Q'$, and $\sigma \in \Sigma$, if $q \in \delta(q_m, \sigma)$, then $\tau(\rho \cdot (\sigma, q)) \in \delta'(q'_m, \sigma)$. Similarly, if $q' \in \delta'(q'_m, \sigma)$, then $\tau(\rho \cdot (\sigma, q')) \in \delta(q_m, \sigma)$. Thus, if the game so far has produced the sequence ρ of positions, ending in $\langle q_m, q'_m \rangle$, and the antagonist chooses the letter σ and a state q in $\delta(q_m, \sigma)$, then the strategy τ instructs the protagonist to move to a state in $\delta(q'_m, \sigma)$. If the antagonist chooses to move to q' in $\delta'(q'_m, \sigma)$, then τ instructs the protagonist to move to a state in $\delta(q_m, \sigma)$. Given an infinite sequence $\pi = q_0, q_1, q_2, \ldots$ of states in Q and an infinite word $\rho = \sigma_0 \cdot \sigma_1 \cdots$ in Σ^ω, the *outcome* of τ on π and ρ is the infinite sequence $\tau[\pi, \rho] = q'_0, q'_1, q'_2, \ldots$ of states in Q', where $q'_i = \tau(q_0, q'_0, q_1, q'_1, \ldots, q_{i-1}, q'_{i-1}, \sigma_i, q_i)$ for all $i \geq 0$. Similarly, the outcome of τ on a sequence $\pi' = q'_0, q'_1, q'_2, \ldots$ of states in Q' and ρ as above is the infinite sequence $\tau[\pi', \rho] = q_0, q_1, q_2, \ldots$ of states in Q, where $q_i = \tau(q_0, q'_0, q_1, q'_1, \ldots, q_{i-1}, q'_{i-1}, \sigma_i, q'_i)$ for all $i \geq 0$. Note that the outcome is not always defined.

A binary relation $H \subseteq Q \times Q'$ is a *game bisimulation relation* between $\mathcal{A}$ and $\mathcal{A}'$ if there exists a strategy τ such that the following two conditions hold for all $\langle q, q' \rangle$ in H.

1. For every word $\rho \in \Sigma^\omega$ and fair (q, ρ)-computation π of $\mathcal{A}$, the outcome $\tau[\pi, \rho]$ is a fair (q', ρ)-computation of $\mathcal{A}'$, and $H(\pi, \tau[\pi, \rho])$.
2. For every word $\rho \in \Sigma^\omega$ and fair (q', ρ)-computation π' of $\mathcal{A}'$, the outcome $\tau[\pi', \rho]$ is a fair (q, ρ)-computation of $\mathcal{A}$, and $H(\tau[\pi', \rho], \pi')$.

The relation H is a *game bisimulation* between $\mathcal{A}$ and $\mathcal{A}'$ iff for every initial state $q_0 \in Q_0$ there is an initial state $q'_0 \in Q'_0$ such that $H(q_0, q'_0)$, and for every initial state $q'_0 \in Q'_0$ there is an initial state $q_0 \in Q_0$ such that $H(q_0, q'_0)$.

We define a transformation from systems to loose tree automata as follows. Given $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$, let $Aut(S) = \langle \Sigma, W, W_0, \delta, \alpha \rangle$, where for all $w \in W$ and $\sigma \in \Sigma$, we have $\delta(w, L(w)) = \{s : R(w, s)\}$, and $\delta(w, \sigma) = \emptyset$, for $\sigma \neq L(w)$. As claimed below, the transformation preserves game-bisimulation.

Lemma 3. *If $S \sim_{game} S'$, then $Aut(S) \sim_{game} Aut(S')$.*

Proof. Let $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ and $S' = \langle \Sigma, W', W'_0, R', L', \alpha' \rangle$. Given a game-bisimulation H between S and S', let τ be the strategy that witnesses the bisimulation. We show that H is also a game-bisimulation between $Aut(S)$ and $Aut(S')$. We define a witnessing strategy τ' as follows. For all $\rho = (q_0, q'_0), (q_1, q'_1), \ldots, (q_m, q'_m) \in (Q \times Q')^*$, $q \in Q$, $q' \in Q'$, and $\sigma \in \Sigma$, we have $\tau'(\rho, \sigma, q) = \tau(\rho, q)$, if $L(q_m) = \sigma$, and is undefined otherwise. Similarly, $\tau'(\rho, \sigma, q') = \tau(\rho, q')$, if $L'(q'_m) = \sigma$, and is undefined otherwise.

Given a pair $\langle q, q' \rangle \in H$, a word $\rho = \sigma_0, \sigma_1, \ldots \in \Sigma^\omega$, and a fair (q, ρ)-computation $\pi = q_0, q_1, \ldots$ of $Aut(S)$, we show that the outcome $\tau'[\pi, \rho] = q'_0, q'_1, \ldots$ is defined and is equal to the outcome $\tau[\pi]$. Since $\langle q, q' \rangle \in H$ and $L(q) = \sigma_0$, then $\tau'(q, q', \sigma_0, q_1) = \tau(q, q', q_1)$ is defined. Consider a finite prefix $q'_0, \ldots, q'_m$ of $\tau'[\pi, \rho]$ such that $\langle q_j, q'_j \rangle \in H$ for all $0 \leq j \leq m$. Since π is a (q, ρ)-computation, thus $L(q_i) = \sigma_i$ for all $i \geq 0$, we know that $\tau'(q_0, q'_0, \ldots, q_m, q'_m, \sigma_m, q_{m+1}) = \tau(q_0, q'_0, \ldots, q_m, q'_m, q_{m+1})$ is defined. So, the outcome $\tau'[\pi, \rho]$ is equal to the outcome $\tau[\pi]$. Since π is fair in $Aut(S)$, it is fair also in S. Since H is a game-bisimulation relation between S and S', it must be that $\tau[\pi]$ is fair. Finally, for every state $q_0 \in W_0$, there exists a state $q'_0 \in W'_0$ such that $\langle q_0, q'_0 \rangle \in H$ and vice versa.

Recall that game-bisimulation between systems implies trace equivalence. As we claim below, game bisimulation between tree automata implies not only agreement on traces that may label paths of accepted trees, but agreement on the accepted trees! We show that given two game-bisimilar loose tree automata, $\mathcal{A}$ and $\mathcal{A}'$, we can use the strategy of the bisimulation to transform an accepting run of $\mathcal{A}$ into an accepting run of $\mathcal{A}'$ and vice versa. Formally we have the following Lemma:

Lemma 4. *If $\mathcal{A} \sim_{game} \mathcal{A}'$, then $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Proof. Let H be the game-bisimulation between $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ and $\mathcal{A}' = \langle \Sigma, Q', Q'_0, \delta', \alpha' \rangle$, and let τ be a strategy that witnesses the bisimulation. We prove that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$. The other direction is symmetrical. Consider a tree $\langle T, V \rangle$ accepted by $\mathcal{A}$, and let $\langle T, r \rangle$ be an accepting run of $\mathcal{A}$ on $\langle T, V \rangle$. We build an accepting run tree $\langle T, r' \rangle$ of $\mathcal{A}'$ on $\langle T, V \rangle$. Since $\langle T, r \rangle$ is a run of $\mathcal{A}$ on $\langle T, V \rangle$, there is an initial state $q_0 \in Q_0$ such that $r(\epsilon) = q_0$. Since H is a game-bisimulation between $\mathcal{A}$ and $\mathcal{A}'$, there is an initial state $q'_0 \in Q'_0$ such that $H(q_0, q'_0)$. We define $r'(\epsilon) = q'_0$. Consider now a node $x = d_1 \cdot d_2 \cdots d_m \in T$ for which $\langle r(x), r'(x) \rangle \in H$. The run $\langle T, r' \rangle$ continues to the level below x according to τ. For each $d \in D$, we define $r'(x \cdot d) = \tau(\ r(\epsilon),\ r'(\epsilon),\ r(d_1),\ r'(d_1),\ \ldots,\ r(x),\ r'(x),\ V(x),\ r(x \cdot d)\ )$. As τ is a winning strategy we are ensured that $r'(x \cdot d) \in \delta'(r'(x), V(x))$, hence $\langle T, r' \rangle$ is a legal run. To see that it is accepting, consider a path $\pi \subseteq T$. Since $\langle T, r \rangle$ is an accepting run, $r(\pi)$ is fair. Hence, $r'(\pi) = \tau[r(\pi), V(\pi)]$ must be fair too.

4.3 From Tree Automata to Game-Bisimulation and ∀-Bisimulation

The tight relation between game-bisimulation and language equivalence implies a tight relation also between the expressive power of fairness conditions used in game-bisimulation and tree automata. We note that Lemma 4 does not hold for ∃-bisimulation, and indeed, the expressiveness hierarchy there is different. Theorems 7 and 8 below formalize this relation.

Theorem 7. *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, then γ is at least as strong as γ' also in the context of loose tree automata (with respect to language equivalence).*

Proof. Given a loose tree automaton $\mathcal{A}$ of type γ', we construct a loose tree automaton of type γ with the same language. For that, we first define a transformation from loose tree automata to systems as follows. Given $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, let $Sys(\mathcal{A}) = \langle \Sigma, Q \times \Sigma, Q_0 \times \Sigma, R, L, \alpha' \rangle$, where for all $q, q' \in Q$ and $\sigma, \sigma' \in \Sigma$, we have $R((q, \sigma), (q', \sigma'))$ iff $q' \in \delta(q, \sigma)$. Also, $L((q, \sigma)) = \sigma$, and α' is obtained from α by replacing each set F participating in α by the set $F \times \Sigma$.

Consider the γ'-system $Sys(\mathcal{A})$ induced by $\mathcal{A}$. Since γ is at least as strong as γ' in the context of game-bisimulation, there is a γ-system S' that is game-bisimilar to $Sys(\mathcal{A})$. Consider the loose tree automaton $Aut(S')$ induced by S'. Clearly, $Aut(S')$ is of type γ. Since $Sys(\mathcal{A}) \sim_{game} S'$, then, by Lemma 3, we have that $Aut(Sys(\mathcal{A})) \sim_{game} Aut(S')$. Then, by Lemma 4, we have that $\mathcal{L}(Aut(Sys(\mathcal{A}))) = \mathcal{L}(Aut(S'))$. All is left to prove is that $\mathcal{L}(Aut(Sys(\mathcal{A}))) = \mathcal{L}(\mathcal{A})$.

By the definition, $Aut(Sys(\mathcal{A})) = \langle \Sigma, Q \times \Sigma, Q_0 \times \Sigma, \delta', \alpha' \rangle$ is such that $\delta'((q,\sigma),\varphi) = \delta(q,\varphi)$ if $\varphi = \sigma$, and $\delta'((q,\sigma),\varphi) = \emptyset$ otherwise. Also, the acceptance condition α' is obtained from α by replacing a set F in α by the set $F \times \Sigma$. Given an accepting run tree $\langle T, r\rangle$ of $\mathcal{A}$ on the labeled tree $\langle T, V\rangle$, we prove that the tree $\langle T, r'\rangle$, where $r'(x) = (r(x), V(x))$ is an accepting run tree of $Aut(Sys(\mathcal{A}))$ on $\langle T, V\rangle$. Indeed, $\langle T, r'\rangle$ is a valid run tree since $\delta'(r'(x), V(x)) = \delta(r(x), V(x)) \times \Sigma$. Also, $\langle T, r'\rangle$ is accepting since for every path $\pi \in T$, we have that $inf(r'|\pi) \subseteq inf(r|\pi) \times \Sigma$. Similarly, given an accepting run tree of $Aut(Sys(\mathcal{A}))$, its projection on Q is an accepting run of $\mathcal{A}$.

Theorem 8. *Let γ and γ' be two types of fairness conditions. If γ is at least as strong as γ' in the context of $\forall$-bisimulation, then γ is at least as strong as γ' also in the context of loose tree automata (with respect to language equivalence).*

Proof. We show that if γ is at least as strong as γ' in the context of $\forall$-bisimulation then γ is at least as strong as γ' in the context of game-bisimulation. Then we use Theorem 7 to complete the proof.

Given a system S of type γ', we construct a game-bisimilar system of type γ as follows. Since γ is at least as strong as γ' in the context of $\forall$-bisimulation, there is a γ-system S' that is $\forall$-bisimilar to S; i.e., $S \sim_\forall S'$. Since $\forall$-bisimulation implies game-bisimulation, then $S \sim_{game} S'$. Hence, γ is at least as strong as γ' in the context of game-bisimulation. By Theorem 7, it follows that γ is at least as strong as γ' also in the context of loose tree automata.

Theorems 7 and 8 refer to the expressive power of loose tree automata, which are weaker than conventional tree automata [42]. Nevertheless, as we argue below, the strictness of the expressiveness hierarchy for tree automata holds already for loose tree automata.

Theorem 9. *For all $k \geq 1$, we have that Rabin$[k{+}1]$ (Streett$[k{+}1]$, parity$[k{+}1]$) is more expressive than Rabin$[k]$ (Streett$[k]$, parity $[k]$) in the context of loose tree automata (with respect to language equivalence).*

Proof. As shown in [23, 37], the expressiveness hierarchy for tree automata coincides with that of deterministic word automata [45, 22], and it can be obtained from the hierarchy of word automata by means of *derivable languages*. More formally, given an ω-language L, the derivable language of L is the set $der(L) = \{\langle T, V\rangle : \text{for all paths } \pi \in T,\ V(\pi) \in L\}$ of trees all of whose paths are labeled by words in L. For every k there exists a language of infinite words L_{k+1}, such that L_{k+1} is recognizable by a deterministic Rabin[k+1] (Streett[k+1], parity[k+1]) word automaton, and L_{k+1} is not recognizable by a deterministic Rabin[k] (Streett[k], parity[k]) word automaton, and the tree language $der(L_{k+1})$ is recognizable by a Rabin[k+1] (Street[k+1], parity[k+1]) tree automaton, and is not recognizable by a Rabin[k] (Streett[k], parity[k]) tree automaton [23, 37].

We show that Rabin[k+1] loose tree automaton are more expressive than Rabin[k] loose tree automata. Let L_{k+1} be the language that is recognized by a deterministic Rabin[k+1] word automaton and $der(L_{k+1})$ is not recognizable by a Rabin[k] tree automaton, as above. Let $\mathcal{A}$ be a deterministic Rabin[k+1] word automaton recognizing L_{k+1}. As shown in [23, 37], when $\mathcal{A}$ is regarded as a loose tree automaton, it recognizes $der(L_{k+1})$. There does not exist a Rabin[k] tree automaton recognizing $der(L_{k+1})$, in particular there does not exists a Rabin[k] loose tree automaton (which is weaker) recognizing $der(L_{k+1})$. The proof for Streett and parity is similar.

Accordingly, we can sum up with the following.

Theorem 10. *The expressiveness hierarchy in the context of game-bisimulation and $\forall$-bisimulation coincides with that of tree automata. In particular, we have the following:*

- *For all $k \geq 1$, we have that Rabin$[k+1]$ (Streett$[k+1]$, parity$[k+1]$) is game-stronger and $\forall$-stronger than Rabin$[k]$ (Streett$[k]$, parity$[k]$).*
- *Rabin$[1]$ is game-stronger and $\forall$-stronger than Büchi and co-Büchi.*
- *Büchi is as game-strong and as $\forall$-strong as generalized Büchi.*

The proof of the last item is given in Appendix A.

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. The explanation to this collapse of the hierarchy is that $\exists$-bisimulation is very close in its nature to trace equivalence, where Büchi systems are as strong as other systems. In $\exists$-bisimulation, trace equivalence should hold for many pairs of states. Consequently, satisfaction of the Büchi condition that takes care of each of the trace equivalences should be combined with a requirement that trace equivalence is eventually checked, leading to Rabin[1] or Streett[2] systems.

There is an intermediate equivalence criterion: *two-way simulation* (that is $S_1 \leq S_2$ and $S_2 \leq S_1$) is implied by bisimulation, it implies trace equivalence, and it is equal to neither of the two [34]. Two-way simulation is a useful criterion: S_1 and S_2 are two-way similar iff for every system S we have $S \leq S_1$ iff $S \leq S_2$ and $S_1 \leq S$ iff $S_2 \leq S$. Hence, in hierarchical refinement, or when defining maximal models for universal formulas, we can replace $_1$ with S_2. A careful reading through our proofs shows that all the results described in the paper for bisimulation hold also for two-way simulation. Clearly, when two systems are bisimilar they are also two-way similar. Hence, whenever we say "equivalent", which means bisimilar it implies also two-way similar. Whenever we say "not equivalent", we prove the claims using two-way similarity. Thus, the systems are not two-way similar implying not bisimilar.

Recall that $\exists$-bisimulation corresponds to fair-CTL* (that is, two systems are $\exists$-bisimilar iff they agree on the satisfaction of all fair-CTL* formulas). Interestingly, the collapse of the expressiveness hierarchy in the context of $\exists$-bisimulation is to Rabin[1], which is exactly the fairness condition required from alternating tree automata in order to recognize CTL* formulas. This is a nice observation, but we still do not fully understand the exact relation between the expressive power required for a β-bisimulation and the power required from tree automata that recognize formulas of the logic that characterizes β. In particular, it is surprising that there are systems that are $\exists$-bisimilar, not game-bisimilar, and the μ-calculus formula that distinguishes between them is an alternation-free μ-calculus formula [18], which can be recognized by Büchi and co-Büchi tree automata. A better understanding of the relation can help us understanding the expressive power of tree automata, by studying the simpler framework of $\exists$-bisimulation (for example, we conjecture that Theorem 5 can be used to imply that Streett[1] tree automata are too weak for recognizing CTL* formulas).

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. [28]). In the branching paradigm [16], Grumberg and Long defined the parallel compositions of two Streett systems. As studied in [16, 24], in order to be used in modular verification, a definition of composition has to satisfy the following two conditions, for all systems S, S_1, and S_2. First, if $S_1 \leq_\exists S_2$, then $S \| S_1 \leq_\exists S \| S_2$. Second, $S \leq_\exists S_1 \| S_2$ iff $S \leq_\exists S_1$ and $S \leq_\exists S_2$. In particular, it follows that $S \| S_1 \leq_\exists S_1$, 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 Appendix B we show that the fact that $\exists$-bisimulation is located between word and tree automata enables a simple definition of parallel composition for Rabin systems that obeys the two conditions above.

References

1. M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.
2. A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. Equivalences for fair Kripke structures. In *Proc. 21st International Colloquium on Automata, Languages and Programming*, Jerusalem, Israel, July 1994.
3. A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. It usually works: the temporal logic of stochastic systems. In P. Wolper, editor, *Computer-Aided Verification, Proc. 7th International Conference*, volume 939 of *Lecture Notes in Computer Science*, pages 155–165. Springer-Verlag, Berlin, 1995.
4. J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4(6):638–648, 1992.
5. M. Browne, E. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.
6. D. Bustan and O. Grumberg. Simulation based minimization. In *Proc. of the 17th International Conference on Automated Deduction*, Pittsburgh, PA, June 2000.
7. Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.
8. E. Clarke, E. Emerson, and A. 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.
9. E. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. In *Proc. 5th Conference on Computer Aided Verification*, volume 697 of *Lecture Notes in Computer Science*. Springer-Verlag, June 1993.
10. 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.
11. W.-P. de Roever, H. Langmaack, and A. Pnueli, editors. *Compositionality: The Significant Difference. Proceedings of Compositionality Workshop*, volume 1536 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
12. D. Dill, A. Hu, and H. Wong-Toi. Checking for language inclusion using simulation relations. In *Proc. 3rd Conference on Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science*, pages 255–265, Aalborg, July 1991. Springer-Verlag.
13. S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games. In *Proc. 12th IEEE Symp. on Logic in Computer Science*, pages 99–110, 1997.
14. E. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 368–377, San Juan, October 1991.
15. N. Francez. *Fairness*. Texts and Monographs in Computer Science. Springer-Verlag, 1986.
16. O. Grumberg and D. Long. Model checking and modular verification. *ACM Trans. on Programming Languages and Systems*, 16(3):843–871, 1994.
17. M. Henzinger, T. Henzinger, and P. Kopke. Computing simulations on finite and infinite graphs. In *Proc. 36th Symp. on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.
18. T. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *Proc. 8th Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 273–287, Warsaw, July 1997. Springer-Verlag.

19. T. Henzinger and S. Rajamani. Fair bisimulation. In *Proc. 4th International Conference of Tools and Algorithms for Construction and Analysis of System*, volume 1785 of *Lecture Notes in Computer Science*, pages 299–314. Springer-Verlag, 2000.
20. R. Hojati. *A BDD-based Environment for Formal Verification of Hardware Systems.* PhD thesis, University of California at Berkeley, 1996.
21. 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*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer-Verlag, 1996.
22. M. Kaminski. A classification of ω-regular languages. *Theoretical Computer Science*, 36:217–229, 1985.
23. O. Kupferman, S. Safra, and M. Vardi. Relating word and tree automata. In *Proc. 11th IEEE Symp. on Logic in Computer Science*, pages 322–333, DIMACS, June 1996.
24. O. Kupferman and M. Vardi. Modular model checking. In *Proc. Compositionality Workshop*, volume 1536 of *Lecture Notes in Computer Science*, pages 381–401. Springer-Verlag, 1998.
25. O. Kupferman and M. Vardi. Relating linear and branching model checking. In *IFIP Working Conference on Programming Concepts and Methods*, pages 304 – 326, New York, June 1998. Chapman & Hall.
26. O. Kupferman and M. Vardi. Verification of fair transition systems. *Chicago Journal of Theoretical Computer Science*, 1998(2), March 1998.
27. O. Kupferman and M. Vardi. Weak alternating automata and tree automata emptiness. In *Proc. 30th ACM Symp. on Theory of Computing*, pages 224–233, Dallas, 1998.
28. R. Kurshan. *Computer Aided Verification of Coordinating Processes.* Princeton Univ. Press, 1994.
29. N. A. Lynch and M. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, pages 137–151, 1987.
30. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, Berlin, January 1992.
31. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Safety.* Springer-Verlag, New York, 1995.
32. K. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.
33. A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, pages 125–129, 1972.
34. R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd International Joint Conference on Artificial Intelligence*, pages 481–489. British Computer Society, September 1971.
35. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer Verlag, Berlin, 1980.
36. D. Niwiński. Fixed point characterization of infinite behavior of finite-state systems. *Theoretical Computer Science*, 189(1–2):1–69, December 1997.
37. D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Symposium on Theoretical Aspects in Computer Science*, volume 1373 of *Lecture Notes in Computer Science.* Springer Verlag, 1998.
38. A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th International Colloquium on Automata, Languages*

and Programming, volume 194, pages 15–32. Lecture Notes in Computer Science, Springer-Verlag, 1985.
39. M. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
40. M. Rabin. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, pages 1–23. North Holland, 1970.
41. S. Safra and M. Vardi. On ω-automata and temporal logic. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 127–137, Seattle, May 1989.
42. W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 165–191, 1990.
43. M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st Symp. on Logic in Computer Science*, pages 332–344, Cambridge, June 1986.
44. M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2):182–221, April 1986.
45. K. Wagner. On ω-regular sets. *Information and Control*, 43:123–177, 1979.

A Büchi Is as $\forall$-Strong as Generalized Büchi

A richer version of the Büchi condition is the *generalized Büchi* condition. It is well known that in the context of tree automata, Büchi conditions are as strong as generalized Büchi conditions [43]. We show that the classical conversion from generalized Büchi to Büchi works also in the context of $\forall$-bisimulation.

Theorem 11. *Every generalized Büchi system with n states and index k has an $\forall$-bisimilar Büchi system with $O(nk)$ states.*

Proof. Given a generalized Büchi system $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ where $\alpha = \{F_1, \ldots, F_k\}$, we define a Büchi system $S' = \langle \Sigma, W', W_0', R', L', \alpha' \rangle$ as follows:

- $W' = W \times \{1, \ldots, k\}$
- $W_0' = W_0 \times \{1\}$
- In order to define R', we first define the function $f : W \times \{1, \ldots, k\} \to \{1, \ldots, k\}$.
  For every $w \in W$ and $i \in \{1, \ldots, k\}$, let:

$$f(w,i) = \begin{cases} i & w \notin F_i \\ i+1 & w \in F_i \text{ and } i < k \\ 1 & w \in F_k \text{ and } i = k \end{cases}$$

  Thus, $f(w,i)$ advances i whenever $w \in F_i$. After reaching copy k, $f(w,k)$ returns to 1.

$$R' = \{\langle (w,i), (w', f(w,i)) \rangle : \langle w, w' \rangle \in R \text{ and } i \in \{1, \ldots, k\}$$

- For all $w \in W$ and $i \in \{1, \ldots, k\}$, we have $L'((w,i)) = L(w)$.
- $\alpha' = F_1 \times \{1\}$

The system S' consists of k copies of the system S. Each copy is marked by an index $i \in \{1,\ldots,k\}$. In copy i we are waiting for a visit to the set F_i. Once F_i is visited we proceed from copy i copy $i+1$. In case we are in copy k we proceed to copy 1, and the process is iterated ad infinitum. The acceptance condition forces a fair computation to visit the set F_1 in the first copy infinitely often, thus forcing the system to visit every set F_i.

In order to see that $S \sim_\forall S'$, consider the relation

$$H = \{\langle w,(w,i)\rangle : i \in \{1,\ldots,k\}\}$$

We prove that H is an $\forall$-bisimulation between S and S'. Consider a pair $\langle w,(w,i)\rangle \in H$. By the definition of L', we have that $L(w) = L'((w,i))$. Every successor w' of w in S has a related successor $(w', f(w,i))$ in S'. Every successor (w',j) of (w,i) in S' has a related successor w' of w in S.

A fair w-computation in S has a single computation of S' related to it by H (f is a function and not a relation). This computation is clearly fair. Similarly, a fair (w,i)-computation in S' has a single computation of S related to it by H, again this computation is fair.

As $\forall$-bisimulation implies game-bisimulation and $\exists$-bisimulation the following corollary follows.

Corollary 1. *Every generalized Büchi system with n states and index k has an game-bisimilar ($\exists$-bisimilar Büchi system with $O(nk)$ states.*

B Parallel Composition

The advent of concurrent programming has made formal verification significantly more necessary and difficult. Modular verification is one possible way to address the state-explosion problem [11]. In modular verification, one uses proof rules of the following form:

$$\left.\begin{array}{l} S_1 \models \psi_1 \\ S_2 \models \psi_2 \\ C(\psi_1,\psi_2,\psi) \end{array}\right\} S_1 \| S_2 \models \psi$$

Here, the symbol "$\|$" denotes parallel composition, and $C(\psi_1,\psi_2,\psi)$ is some logical condition relating ψ_1, ψ_2, and ψ. Using modular proof rules it is possible to apply model checking only to the underlying components, which have much smaller state spaces.

In the linear paradigm, the composition $S = S_1 \| S_2$ is defined so that $\mathcal{T}(S) = \mathcal{T}(S_1) \cap \mathcal{T}(S_2)$ (cf. [28]). Thus, S contains exactly all the behaviors that are common to S_1 and S_2, or equivalently, S satisfies exactly all the linear properties that are satisfied by both S_1 and S_2. In the branching paradigms, things are less clear: if we require the parallel composition of S_1 and S_2 to satisfy exactly all the properties that are satisfied by both S_1 and S_2, then, as branching temporal logic can characterize systems up to bisimulation, parallel composition cannot be defined, unless S_1 and S_2 are bisimilar. So, the branching paradigm requires a different definition of parallel composition. In [16], Grumberg and Long study modular verification of branching properties, and defined the parallel compositions of two Streett systems. The composition defined there is such that for all Streett systems S, S_1, and S_2, the following hold.

1. If $S_1 \leq_\exists S_2$ then $S \| S_1 \leq_\exists S \| S_2$.
2. $S \leq_\exists S_1 \| S_2$ iff $S \leq_\exists S_1$ and $S \leq_\exists S_2$.

In particular, it follows that $S \| S_1 \leq_\exists S_1$, thus every universal formula that is satisfied by a component of a parallel composition, is satisfied also by the composition. As studied in [16, 24], conditions 1 and 2 above are essential and sufficient for modular verification in the branching paradigm.

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. We show that the fact that $\exists$-bisimulation is located between word and tree automata enables a simple definition of parallel composition that obeys conditions 1 and 2.

Given two Rabin systems $S_1 = \langle \Sigma, W_1, W_0^1, R_1, L_1, \alpha_1 \rangle$ and $S_2 = \langle \Sigma, W_2, W_0^2, R_2, L_2, \alpha_2 \rangle$ with $\alpha_1 = \{\langle G_1^1, B_1^1 \rangle, \ldots, \langle G_n^1, B_n^1 \rangle\}$ and $\alpha_2 = \{ \langle G_1^2, B_1^2 \rangle, \ldots, \langle G_m^2, G_m^2 \rangle \}$, we define $S_1 \| S_2 = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ as follows.

- Let $W' = \{(w_1, w_2) : w_1 \in W_1,\ w_2 \in W_2 \text{ and } L_1(w_1) = L_2(w_2)\}$. For every $1 \leq i \leq n$ and $1 \leq j \leq m$, let $W_{i,j} = (W' \setminus (B_i^1 \times W_2 \cup W_1 \times B_j^2)) \times \{i\} \times \{j\} \times \{\perp, \top\}$. Then $W = W' \cup \bigcup_{1 \leq i \leq n} \bigcup_{1 \leq j \leq m} W_{i,j}$
- $W_0 = (W_0^1 \times W_0^2) \cap W'$
- In order to define R, we first define the function $f : W \to \{\perp, \top\}$.

$$f(w_1, w_2) = \perp$$

$$f((w_1, w_2, i, j, \perp)) = \begin{cases} \perp & w_1 \notin G_i^1 \\ \top & w_1 \in G_i^1 \end{cases}$$

$$f((w_1, w_2, i, j, \top)) = \begin{cases} \top & w_2 \notin G_j^2 \\ \perp & w_2 \in G_j^2 \end{cases}$$

Thus, the label $\perp$ changes to $\top$ whenever a state from G_i^1 is encountered, and the label $\top$ changes to $\perp$ whenever a state from G_j^2 is encountered. For every $1 \leq i \leq n$ and $1 \leq j \leq m$, we define

$$R_{i,j} = \left\{ \begin{array}{l|c} \langle \langle w_1, w_2 \rangle, \langle w_1', w_2', i, j, \perp \rangle \rangle & \\ \langle \langle w_1, w_2, i, j, \varphi \rangle, \langle w_1', w_2', i, j, f(w_1, w_2, i, j, \varphi) \rangle \rangle & \langle w_1, w_1' \rangle \in R_1 \\ \langle \langle w_1, w_2, i, j, \varphi \rangle, \langle w_1', w_2' \rangle \rangle & \text{and} \\ \langle \langle w_1, w_2 \rangle, \langle w_1', w_2' \rangle \rangle & \langle w_2, w_2' \rangle \in R_2 \end{array} \right\}$$

Then,

$$R = \bigcup_{1 \leq i \leq n} \bigcup_{1 \leq j \leq m} R_{i,j} \cap (W \times W).$$

– $\alpha = \{\langle \bigcup_{1\leq i \leq n} G_i \times W_2 \times \{i\} \times \{1,\ldots,m\} \times \{\bot\} \cap W, W'\rangle\}$.

The system W consists of copies of the product of the state sets of W_1 and W_2. There is an "idle copy" of the product, that branches into $n \cdot m$ pairs. Every pair consists of two copies of the product, and is associated with the pairs $\langle G^1_i, B^1_i\rangle$ and $\langle G^2_j, B^2_j\rangle$. As such, it does not contain states from the two bad sets. Whenever one of the good sets is visited, the tag ($\bot$ or $\top$) is flipped. If the tag is flipped infinitely often then both sets are visited infinitely often. As before we allow the system to change is guess, by returning to the idle copy and then going into a new copy. The acceptance condition forces a fair computation to visit the idle copy only finitely often. Thus, the computation eventually gets trapped in $W_{i,j}$, for some $1 \leq i \leq n$, and $1 \leq j \leq m$. There, the computation cannot visit states from the bad sets, and it is forced by the acceptance condition to visit the good sets.

Lemma 5. *For every three Rabin systems S, S_1 and S_2, If $S_1 \leq_\exists S_2$ then $S\|S_1 \leq_\exists S\|S_2$*

Proof. Let $S_1 = \langle \Sigma, W_1, W^1_0, R_1, L_1, \alpha_1\rangle$, $S_2 = \langle \Sigma, W_2, W^2_0, R_2, L_2, \alpha_2\rangle$, $S = \langle \Sigma, W, W_0, R, L, \alpha\rangle$ with $\alpha_1 = \{\langle G^1_1, B^1_1\rangle, \ldots, \langle G^1_n, B^1_n\rangle\}$, $\alpha_2 = \{ \langle G^2_1, B^2_1\rangle, \ldots, \langle G^2_m, G^2_m\rangle \}$, and $\alpha = \{\langle G_1, B_1\rangle, \ldots, \langle G_k, B_k\rangle\}$. Let H be the $\exists$-simulation of S_1 by S_2. Also, let $S\|S_1 = \langle \Sigma, W', W'_0, R', L', \alpha'\rangle$ and $S\|S_2 = \langle \Sigma, W'', W''_0, R'', L'', \alpha''\rangle$.

In order to see that $S\|S_1 \leq_\exists S\|S_2$, consider the following relation:

$$H' = \left\{ \begin{array}{l|l} \langle (w, w_1, i, j, \varphi), (w, w_2, i', j', \varphi')\rangle & \langle w_1, w_2\rangle \in H,\ w \in W, \\ \langle (w, w_1), (w, w_2, i', j', \varphi')\rangle & i, i' \in \{1, \ldots, k\}, \\ \langle (w, w_1, i, j, \varphi), (w, w_2)\rangle & j \in \{1, \ldots, n\}, \\ \langle (w, w_1), (w, w_2)\rangle & j' \in \{1, \ldots, m\} \text{ and} \\ & \varphi, \varphi' \in \{\bot, \top\} \end{array} \right\} \cap W' \times W''$$

We prove that H' is an $\exists$-simulation of $S\|S_1$ by $S\|S_2$. Consider a pair $\langle (w, w_1, i, j, \varphi), (w, w_2, i', j', \varphi')\rangle \in H'$. By the definition of L' and L'' and the fact that $\langle w_1, w_2\rangle \in H$, we have that $L'((w, w_1, i, j, \varphi)) = L_1(w_1) = L_2(w_2) = L''((w, w_2, i', j', \varphi'))$. Consider a fair (w, w_1, i, j, φ)-computation π in $S\|S_1$. Let $\pi \Downarrow_1$ ($\pi \Downarrow_2$) denote the sequence of first (second) elements in the tuples in π. Clearly, $\pi \Downarrow_1$ is a fair w-computation in S. Hence there is some i'' such that $\pi \Downarrow_1 = w_0, w_1, \ldots$ is fair according to $\langle G_{i''}, B_{i''}\rangle$ and for some l forall $l' > l$ we have $w_{l'} \notin B_{i''}$. Also, $\pi \Downarrow_2$ is a fair w_1-computation. Since $\langle w_1, w_2\rangle \in H$ there exists a fair w_2-computation $\pi_2 = w^2_0, w^2_1, \ldots$ in S_2 such that $H(\pi \Downarrow_2, \pi_2)$. There is some j'' such that π_2 is fair according to $\langle G^2_{j''}, B^2_{j''}\rangle$ and for some r, forall $r' > r$ we have $w^2_{r'} \notin B_{j''}$. Let $p = max(r, l)$, it follows that the combination of $\pi \Downarrow_1$ with π_2, namely, $(w_0, w^2_0, i', j', \varphi), (w_1, w^2_1), \ldots, (w_p, w^2_p), (w_{p+1}, w^2_{p+1}, i'', j'', \bot), \ldots$ is a fair $(w, w_2, i', j', \varphi')$-computation in $S\|S_2$. Finally, since for every $(w, w_1) \in W'_0$ there exists some $w_2 \in W^2_0$ such that $H(w_1, w_2)$ we have that $(w, w_2) \in W''_0$, $H'((w, w_1), (w, w_2))$ and H' is an $\exists$-simulation of $S\|S_1$ by $S\|S_2$

Lemma 6. *For every three Rabin systems S, S_1, and S_2, we have that $S \leq_\exists S_1 \| S_2$ iff $S \leq_\exists S_1$ and $S \leq_\exists S_2$,*

Proof. Let $S_1 = \langle \Sigma, W_1, W_0^1, R_1, L_1, \alpha_1 \rangle$, $S_2 = \langle \Sigma, W_2, W_0^2, R_2, L_2, \alpha_2 \rangle$, $S = \langle \Sigma, W, W_0, R, L, \alpha \rangle$ and $S_1 \| S_2 = \langle \Sigma, W', W_0', R', L', \alpha' \rangle$. Let n be the index of S_1 and m the index of S_2.

Assume first that $S \leq_\exists S_1 \| S_2$. Let H be the $\exists$-simulation of S by $S_1 \| S_2$. We show that $S \leq_\exists S_1$ (the proof of $S \leq_\exists S_2$ is similar). Consider the relation:

$$H' = \left\{ \langle w, w_1 \rangle \;\middle|\; \begin{array}{l} \exists w_2 \in W_2,\ i \in \{1, \ldots, n\},\ j \in \{1, \ldots, m\}, \text{ and } \varphi \in \{\bot, \top\} \\ \text{such that } \langle w, (w_1, w_2, i, j, \varphi) \rangle \in H \text{ or } \langle w, (w_1, w_2) \rangle \in H \end{array} \right\}.$$

We prove that H' is an $\exists$-simulation of S by S_1. Consider a pair $\langle w, w_1 \rangle \in H'$. There is some state $(w_1, w_2, i, k, \varphi) \in W'$ such that $\langle w, (w_1, w_2, i, j, \varphi) \rangle \in H$ (the case that $\langle w, (w_1, w_2) \rangle \in H$ is similar). By the definition of L and the fact that $\langle w, (w_1, w_2, i, j, \varphi) \rangle \in H$, we have that $L_1(w_1) = L(w)$. Given a fair w-computation π in S. Since $\langle w, (w_1, w_2, i, j, \varphi) \rangle \in H$ there exists a fair $(w_1, w_2, i, j, \varphi)$-computation π' in $S_1 \| S_2$ such that $H(\pi, \pi')$. Let $\pi \Downarrow_1$ be the projection of π' on its first elements. It is easy to see that $\pi \Downarrow_1$ is a fair w_1 computation with $H(\pi', \pi \Downarrow_1)$. Finally for $w_0 \in W_0$ there exists some $w_0' \in W_0^1$ such that $H'(w_0, w_0')$.

Assume that $S \leq_\exists S_1$ and $S \leq_\exists S_2$. Let H_1 be the $\exists$-simulation of S by S_1 and let H_2 be the $\exists$-simulation of S by S_2. We show that $S \leq_\exists S_1 \| S_2$. Consider the following relation:

$$H = \left\{ \begin{array}{l} \langle w, (w_1, w_2, i, j, \varphi) \rangle, \\ \langle w, (w_1, w_2) \rangle \end{array} \;\middle|\; \langle w, w_1 \rangle \in H_1 \text{ and } \langle w, w_2 \rangle \in H_2 \right\} \cap W \times W'.$$

We show now that H is an $\exists$-simulation of S by $S_1 \| S_2$. Consider a pair $\langle w, (w_1, w_2, i, j, \varphi) \rangle \in H$. From the definition of the labeling function in $S_1 \| S_2$ and the fact that $\langle w, w_1 \rangle \in H_1$ and $\langle w, w_2 \rangle \in H_2$, we have that the states w and $(w_1, w_2, i, j, \varphi)$ agree on their label. Given a fair w-computation π in S, there exists a fair w_1-computation π_1 in S_1 and a fair w_2-computation π_2 in S_2. We combine π_1 and π_2 into a computation π' in $S_1 \| S_2$ that goes back to the idle copy and eventually stays indefinitely in the copy that matches the pair according to which π_1 is winning and the pair according to which π_2 is winning. It is simple to see that π' is a fair $(w_1, w_2, i, j, \varphi)$-computation such that $H'(\pi, \pi')$. The argument for a pair of the form $\langle w, (w_1, w_2) \rangle \in H$ is similar. Finally, since H_1 and H_2 are $\exists$-bisimulations, for every $w_0 \in W_0$ there exists at least one initial state (w_1, w_2) of $S_1 \| S_2$ such that $H(w_0, (w_1, w_2))$.

Note that in order to define a composition of two Rabin[1] systems, only two copies of the product of the two systems is needed. Note also that given a Rabin[n] and a Rabin[m] system, transforming them first into bisimilar Rabin[1] systems and then composing the two Rabin[1] systems results in a system that is very similar to the composition as defined above.

Finally, note that while the blow-up in the transition to Rabin[1] systems is linear in the index for Rabin systems and is exponential in the index for Streett systems, parallel composition is easier for Streett systems. The reason is the conjunctive nature of the Streett condition (a computation is fair if it satisfies the conditions imposed by all pairs), which corresponds to the conjunctive nature of parallel composition. Rabin conditions, on the other hand, have a disjunctive nature, and the blow-up in a parallel composition is required to handle the disjunctive constraints. As the obvious way to combine two Büchi systems is by using the generalized Büchi condition, and as we discuss in Section 3, a generalized Büchi system can be easily translated to an $\exists$-bisimilar Büchi system, the parallel composition of two Büchi systems is also easy to define.