Applicability of Fair Simulation

Doron Bustan and Orna Grumberg

Computer Science Department
Technion, Haifa 32000, Israel
{orna,doron2}@cs.technion.ac.il

Abstract. In this paper we compare among four notions of fair simulation: direct [6], delay [7], game [10], and exists [9]. Our comparison refers to three main aspects: The time complexity of constructing the fair simulation, the ability to use it for minimization, and the relationship between the fair simulations and universal branching-time logics.
Based on our comparison we derive several practical implications: We develop an efficient approximated minimization algorithm for the direct/delay simulations. In addition, we suggest a new implementation for the assume-guarantee modular framework presented in [9]. The new implementation, significantly improves the complexity of the framework.

1 Introduction

Temporal logic model checking is a method for verifying finite-state systems with respect to propositional temporal logic specifications. The method is fully automatic and quite efficient in time, but is limited by its high space requirements. Many approaches for overcoming the *state explosion problem* of model checking have been suggested [4]. They are often based on the idea that the model of the verified system can be replaced by a more abstract model, which is smaller in size. The abstract and concrete models are sufficiently similar, so that properties that are verified on the abstract model can be concluded as true for the concrete one. This idea is often formalized by relating models with the *simulation preorder* [14] in which the greater, more abstract model has "more behaviors", and the verified properties are written in a universal branching time logic such as ACTL or ACTL* [9].

In order to avoid unrealistic behaviors introduced to the model by abstraction, it is common to add fairness constraint that distinguish between wanted (fair) and unwanted (unfair) behaviors and to exclude unfair behaviors from consideration. The simulation preorder does not distinguish between fair and unfair behaviors. It is therefore desirable to find an alternative definition that relates only fair behaviors of the two models. This task, however, is not uniquely defined. Indeed, several distinct notions of *fair simulation* have been suggested in the literature [6,7,10,9].

A question that naturally arises is, which notion of fair simulation is preferable. In [10] some of these notions are compared with respect to the complexity of checking for fair simulation. In [7] a different set of notions is compared with respect to two criteria: The complexity of constructing the preorder, and the ability to minimize a fair model by constructing a quotient model that is language equivalent to the original one.

In this paper we give a wider comparison among four notions of fair simulation: direct [6], delay [7], game [10], and exists [9]. We refer to several criteria, which emphasize the advantages of each of the notions. The results of the comparison are summarized in a table in Figure 1.

Based on our comparison we derive several practical implications. We develop an efficient approximated minimization algorithm for the delay, game and exists simulations. For these preorders, a unique equivalent smallest model does not exist. Therefore, an approximation is appropriate. In addition, we suggest a new implementation for the *assume-guarantee* [8,11,15,16] modular framework, presented in [9]. The new implementation, based on the game simulation rather than the exists simulation, significantly improves the complexity of the framework.

Our comparison refers to three main aspects of fair simulation. The first is the time complexity of constructing the preorder. There, we mainly summarize results of other works (see Figure 1). We see that constructing the direct, delay and game simulations is polynomial in the number of states n and the number of transitions m [7]. In contrast, constructing the exists simulation is PSPACE-complete [12] .

The second aspect that we consider is the ability to use the preorder for minimization. We say that two models are *equivalent* with respect to a preorder if each is smaller by the preorder than the other. The goal of minimization is to find the smallest in size model which is equivalent with respect to the preorder to the original one[1].

We examine for each of the fair simulation preorders the following three issues. Given a model M, 1. Is there a unique smallest in size model that is simulation equivalent to M. 2. Is the quotient model of M, simulation equivalent to M. 3. Is the result of disconnecting little brothers (to be explain in Sect. 3) in M, simulation equivalent to M.

Our examination (see Figure 1) leads to a new minimization algorithm that uses the direct and delay simulations as approximations for the game and exists simulations. The new algorithm obtains a better reduction than the algorithm suggested in [7].

The third aspect is the relationship between the simulation preorders and universal branching-time logics. A basic requirement is that the preorder preserves the specification logic, I.e. if $M_1 \leq M_2$ then, for every formula ϕ in the logic, $M_2 \models \phi$ implies $M_1 \models \phi$. Indeed all four notions of fair simulation satisfy this requirement. A stronger requirement is that the preorder has a *logical characterization* by some logic. This means that $M_1 \leq M_2$ iff for every formula ϕ in the logic, $M_2 \models \phi$ implies $M_1 \models \phi$.

Logical characterization is useful in determining if model M_2 is an abstraction of model M_1, when the logic $\mathcal{L}$ should be preserved. If the preorder $\leq$ is logically

[1] Note that this is a stronger criterion than the one used in [7], where only language equivalence is required.

characterized by $\mathcal{L}$ then checking $M_1 \leq M_2$ is a necessary and sufficient condition and will never give false negative result.

Another important relationship between a logic and a preorder is the existence of a *maximal model* $\mathcal{T}_\phi$ for a formula ϕ such that for every model M', $M' \leq \mathcal{T}_\phi$ if and only if $M' \models \phi$. Maximal models are used as tableaux in the framework described in [9] for the *assume-guarantee paradigm*.

In this work we show that there is a maximal model for ACTL formulas also with respect to the game simulation. In addition, we show that other conditions required for a sound implementation of the assume-guarantee paradigm hold for the game simulation.

The results of our comparison are presented in the table in Figure 1. The proofs of the claims that are not cited appear in the next sections. Due to lack of space some proofs are omitted. The rest of the paper is organized as follows:

		minimization			relation to logic	
simulation notion	time complexity of constructing the preorder	unique smallest model	quotient model	little brothers	has logical characterization	maximal model
Direct	$O(m \cdot n)$ [7]	*true*	true	*true*	*false*	*false*
Delay	$O(m \cdot n^3)$ [7]	*false*	true [2]	*false*	*false*	*false*
Game	$O(m \cdot n^3)$ [7]	*false*	false [7]	*false*	$\forall AFMC$ [10]	*true*
Exists	PSPACE complete [12]	*false*	*false*	*false*	$ACTL^*$	true [9]

Fig. 1. The properties of the different notions of fair simulation

In Section 2 we define the simulation preorder and the different notions of fair simulation. Section 3 investigates simulation minimization. Section 4 investigates the relationships of fair simulation with logic. In Section 5 we prove that the game simulation can replace the exists simulation in the implementation of the assume-guarantee paradigm. Finally, in Section 6 we discuss some conclusions.

2 Preliminaries

Let AP be a set of atomic propositions. We model systems by a *fair Kripke structure* M over AP, $M = (S, R, S_0, L, F)$, where S is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $R \subseteq S \times S$ is the transition relation, which must be *total*. That is, for every state $s \in S$ there is a state $s' \in S$ such that $(s, s') \in R$ (states which do not satisfy this condition are deleted). $L : S \to 2^{AP}$ is a function that labels each state with the set of atomic propositions true in that state, and $F \subseteq S$ is a set of fair states.

[2] In [7] it is shown that the quotient model is language equivalent to the original model. Here, we show that they are delay equivalent.

Let s be a state in a Kripke structure M. A *trace* in M starting from s is an infinite sequence of states $\rho = s_0 s_1 s_2 \ldots$ such that $s_0 = s$, and for every $i \geq 0$, $(s_i, s_{i+1}) \in R$. The i-th state of trace ρ is denoted ρ^i. In order to capture the infinite behavior of ρ, we define
$\inf(\rho) = \{\, s \mid s = \rho^i \text{ for infinitely many } i \,\}$.
We say that a trace ρ is *fair* according to the fair set F iff $\inf(\rho) \cap F \neq \emptyset$.

In this work we refer to two branching-time logics ACTL* and ACTL [9]. ACTL* is the universal fragment of the powerful branching-time logic, CTL*. ACTL* consists of the temporal operators **X** (next-time), **U** (until) and **R** (release), as well as the universal path quantifier **A** (for all paths). ACTL is a restricted sublogic of ACTL* in which every temporal operator is immediately preceded by a path quantifier. Due to lack of space we define precisely only ACTL. We define ACTL formulas in negation normal form, namely, negation is applied only to atomic propositions. ACTL is the set of formulas defined as follows:

- if $p \in AP$ then p and $\neg p$ are formulas.
- If ϕ and ψ are formulas, then $\phi \wedge \psi$ and $\phi \vee \psi$ are formulas.
- If ϕ and ψ are formulas, then $\mathbf{AX}\,\phi$, $\mathbf{A}[\phi\,\mathbf{U}\,\psi]$ and $\mathbf{A}[\phi\,\mathbf{R}\,\psi]$ are formulas.

An ACTL formula ϕ is interpreted in a state s, with respect to the fair traces which start at s. The formal definition of the semantics for ACTL can be found in [4]. We say that $M \models \phi$ iff for every initial state $s_0 \in S_0$, $M, s_0 \models \phi$.

2.1 Simulation and Fair Simulation

We start by defining simulation relation over Kripke structures with $F = S$ (Kripke structures with trivial fairness constraints).

Definition 1. *Given two structures M_1 and M_2 over AP, a relation $H \subseteq S_1 \times S_2$ is a simulation relation [14] over $M_1 \times M_2$ iff the following conditions hold:*

1. *For every $s_{01} \in S_{01}$ there exists $s_{02} \in S_{02}$ such that $(s_{01}, s_{02}) \in H$.*
2. *For all $(s_1, s_2) \in H$,*
   a) $L_1(s_1) = L_2(s_2)$ and
   b) $\forall s'_1[(s_1, s'_1) \in R_1 \rightarrow \exists s'_2[(s_2, s'_2) \in R_2 \wedge (s'_1, s'_2) \in H]]$.

M_2 simulates M_1 (denoted by $M_1 \leq M_2$) if there exists a simulation relation H over $M_1 \times M_2$. We say that M_1 and M_2 are *simulation equivalent if* $M_1 \leq M_2$ and $M_2 \leq M_1$. Similarly $(s_1, s_2) \in H$ is denoted $s_1 \leq s_2$ and s_1 and s_2 are equivalent if $s_1 \leq s_2$ and $s_2 \leq s_1$ and denoted $s_1 \equiv s_2$.

The relation $\leq$ is a preorder on the set of structures. That is, $\leq$ is reflexive and transitive. In [9,2] it is shown that $M_1 \leq M_2$ iff, for every ACTL* formula ψ (with atomic propositions in AP), $M_2 \models \psi$ implies $M_1 \models \psi$. Thus, simulation relation has logical characterization over structures with trivial fairness constraints.

Next, we define the different notions of fair simulation.

Definition 2. *$H \subseteq S_1 \times S_2$ is a* direct simulation relation *[6] ($\leq_{di}$) over $M_1 \times M_2$ iff it satisfies the conditions of Def. 1 except that 2a is replaced by:*
$2(a')$ $L_1(s_1) = L_2(s_2)$ and $s_1 \in F_1$ implies $s_2 \in F_2$.

Definition 3. *[9] $H \subseteq S_1 \times S_2$ is an* exists simulation *($\leq_{\exists}$) over $M_1 \times M_2$ iff it satisfies the conditions of Def. 1 except that 2b is replaced by:*
$2(b')$ for every fair trace ρ_1 from s_1 in M_1 there exists a fair trace ρ_2 from s_2 in M_2 such that for all $i \in I\!N$, $(\rho_1^i, \rho_2^i) \in H$.[3]

The next definitions are based on games. We start with a game that characterizes the simulation over structures with trivial fairness constraints. Given two Kripke structures M_1, M_2, we define a game of two players over M_1, M_2. The players are called the adversary and the protagonist, where the adversary plays on M_1 and the protagonist plays on M_2.

Definition 4. *Given two Kripke structures, M_1 and M_2, a* simulation game *consists of a finite or infinite number of rounds. At the beginning, the adversary selects an initial state s_{01} in M_1 to start from, then the protagonist responds by selecting an initial state s_{02} in M_2 such that $L_1(s_{01}) = L_2(s_{02})$. In each round, assume that the adversary is at s_1 and the protagonist is at s_2. The adversary then moves to a successor s_1' of s_1 on M_1, after which the protagonist moves to a successor s_2' of s_2 on M_2 such that $L_1(s_1') = L_2(s_2')$.*

If the protagonist does not have a matching state then the protagonist fails. Otherwise, if the protagonist always has a matching successor to move to, then the game proceeds ad infinitum for ω rounds and the protagonist wins. The adversary wins iff the protagonist fails.

Definition 5. *Given two Kripke structures M_1 and M_2, a* strategy *π of the protagonist is a function $\pi : (S_1 \times S_2 \to S_2) \cup (S_{01} \times \{\perp\} \to S_{02})$. The function π should satisfy the following: If $s_2' = \pi(s_1', s_2)$ then $(s_2, s_2') \in R_2$.*

The protagonist plays according to a strategy π if initially when the adversary selects $s_{0_1} \in S_{0_1}$ the protagonist selects $s_{0_2} = \pi(s_{0_1}, \perp)$ and, for every round i, when the adversary moves to s_1' and the protagonist is in s_2 then the protagonist moves to $s_2' = \pi(s_1', s_2)$. π is a winning strategy for the protagonist if the protagonist wins whenever it plays according to π. We can now present an alternative definition to the simulation preorder. This definition is equivalent to Def. 1 [10].

Definition 6. *Given two Kripke structures, M_1 and M_2, M_2 simulates M_1 ($M_1 \leq M_2$) iff the protagonist has a winning strategy in a simulation game over M_1, M_2.*

In order to extend the simulation game to fair simulation, we add a winning condition which refers to the infinite properties of the game. We then give two additional definitions of fair simulation, the delay ($\leq_{de}$) and the game ($\leq_g$) simulations.

[3] In such a case we use the notation $(\rho_1, \rho_2) \in H$.

Definition 7. *[7] The protagonist* delay wins *a game over two fair Kripke structures M_1 and M_2 iff the game is played for infinitely many rounds. Moreover, whenever the adversary reaches a fair state then the protagonist reaches a fair state within a finite number of rounds.*

Definition 8. *[10] The protagonist* game wins *a game over two fair Kripke structures M_1 and M_2, iff the game is played for infinitely many rounds. Moreover, if the adversary moves along a fair trace, then the protagonist moves along a fair trace as well.*

We say that π is a delay/game winning strategy for the protagonist if the protagonist delay/game wins whenever it plays according to π.

Definition 9. *[10,7] Given two fair Kripke structures, M_1 and M_2, M_2 delay/game simulates M_1 iff the protagonist has a delay/game winning strategy over M_1, M_2.*

Definitions 2,3,9 are extensions of Def.1 and its equivalent Def. 6. Consequently, on structures with trivial fairness constraints ($F = S$), all four definitions are equivalent. In [10,7] the following relationships over the fair simulation preorders are shown,
$M_1 \leq_{di} M_2 \;\Rightarrow\; M_1 \leq_{de} M_2 \;\Rightarrow\; M_1 \leq_{g} M_2 \;\Rightarrow\; M_1 \leq_{\exists} M_2$
Note that the definitions of game/exists simulation are not limited to specific types of fairness constraints. They hold even if M_1 and M_2 each has a different type of fairness constraints. Finally we extend the delay/game simulations for states.

Definition 10. *For all states s_1 and s_2 in a structure M, $s_1 \leq_{de/g} s_2$ if the protagonist has a winning delay/game strategy in a game over $M \times M$ where the adversary starts at s_1 and the protagonist starts at s_2.*

3 Simulation Minimization

For structures with trivial fairness constraints ($F = S$), two forms of redundancy are considered [3]. These redundancies are handled in [3], by first constructing a quotient structure which results in a structure without equivalent states and then disconnecting *little brothers* eliminating the other redundancy. For structures with trivial fairness constraints, the result of eliminating these redundancies is a unique, smallest in size structure which is simulation equivalent to the original structure [3].

Lemma 1. *For every structure, there exists a unique, smallest in size structure, which is direct simulation equivalent to it.*

The proof of Lemma 1 and the construction of the smallest structure can be obtained in the same manner as in [3]. Unfortunately, performing the same operations for the other notions of fair simulations might result in an inequivalent structure. In this section we investigate minimization with respect to each notion of fair simulation. We start by checking whether the quotient structure is equivalent to the original one. Next we check whether it is safe to disconnect little brothers. We then determine whether there exists a unique smallest in size equivalent structure. Finally we use the results of this section to suggest a new better minimizing algorithm.

Definition 11.

- *The language of s_1 is contained in the language of s_2 ($s_1 \subseteq s_2$) if for every fair trace ρ_1 from s_1 there is a fair trace ρ_2 from s_2 such that $\forall i \geq 0$, $L(\rho_1^i) = L(\rho_2^i)$.*
- *$M_1 \subseteq M_2$ if for every fair trace starting at an initial state $s_{01} \in S_{01}$ there is a fair trace starting at an initial state $s_{02} \in S_{02}$ such that $\forall i \geq 0$, $L_1(\rho_1^i) = L_2(\rho_2^i)$.*
- *M_1 is language equivalent to M_2 if $M_1 \subseteq M_2$ and $M_2 \subseteq M_1$.*

Clearly, all notions of fair simulation imply language containment.

Quotient Structure

In a quotient structure all equivalent states are unified into equivalence classes. The equivalence classes are the states of the quotient structure. There is a transition from one equivalence class to another iff there exists a transition from a state in the former to a state in the latter. An equivalence class is initial if it contains an initial state and is fair if it contains a fair state. For the delay simulation, we presents the following lemma.

Lemma 2. *Let M^Q be the quotient structure of a structure M. Then $M \equiv_{de} M^Q$.*

In [7] it is shown that the quotient structure with respect to game simulation is not equivalent to the original one. We show that for every preorder $\leq_\clubsuit$ that lies between game simulation and language containment, the quotient structure with respect to this preorder may not be equivalent to the original structure.

Lemma 3. *Let $\leq_\clubsuit$ be any preorder such that for every M_1, M_2,*
$M_1 \leq_g M_2 \Rightarrow M_1 \leq_\clubsuit M_2 \Rightarrow M_1 \subseteq M_2$.
Then there exists a structure M whose quotient structure with respect to $\leq_\clubsuit$ is not equivalent to M with respect to $\leq_\clubsuit$.

Proof sketch. Consider the structure M_1 in Figure 2. States s_0 and s_2 are equivalent with respect to game simulation. To see that, consider a strategy that instructs the protagonist to move to the same state the adversary moves to. This strategy proves both directions of the game equivalence. Since $M_1 \leq_g M_2 \Rightarrow M_1 \leq_\clubsuit M_2$, s_0 and s_2 are also equivalent with respect to $\leq_\clubsuit$. Since $M_1 \leq_\clubsuit M_2 \Rightarrow M_1 \subseteq M_2$, it is sufficient to prove that the result of unifying states s_0 and s_2 is not language equivalent to M_1. To see that, note that the language of M_1 consists of all words in which both a and b occur infinitely often. However, any structure with two states that contains this language, must also contain a word with a suffix of a's only (or b's only). $\square$

Corollary 1. *For exists/game simulation, the quotient structure is not necessarily equivalent to the original structure.*

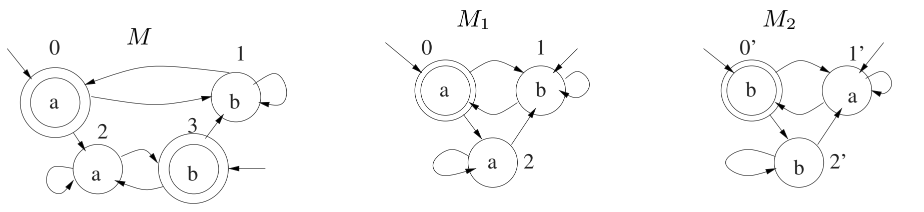

Fig. 2. The structures M_1 and M_2 are equivalent to M with respect to game/exists simulation, and they are both minimal. Note that states 0 and 2 ($0'$ and $2'$) are equivalent but cannot be unified. (double circles denote fair states)

Disconnecting Little Brothers

A state s_2 is a *little brother* of another state s_3 if both states are successors of the same state s_1, $s_2 \leq s_3$ and $s_3 \not\leq s_2$. Little brothers are disconnected by removing the transition (s_1, s_2) from R.

Lemma 4. *Let $\leq_\spadesuit$ be a preorder such that*
$M_1 \leq_{de} M_2 \Rightarrow M_1 \leq_\spadesuit M_2 \Rightarrow M_1 \subseteq M_2$.
The result of disconnecting little brothers with respect to $\leq_\spadesuit$ in a structure M might not be equivalent to M with respect to $\leq_\spadesuit$.

Proof sketch. Consider the structure M_1 in Figure 3. State s_2 is a little brother of state s_1 with respect to $\leq_\spadesuit$. To see that, note that $s_2 \leq_{de} s_1$ and therefore, $s_2 \leq_\spadesuit s_1$. Moreover, $s_1 \not\subseteq s_2$, and thus $s_1 \not\leq_\spadesuit s_2$.

Since $M_1 \leq_\spadesuit M_2 \Rightarrow M_1 \subseteq M_2$, it is sufficient to show that the result of disconnecting s_2 from s_0 is not language equivalent to M_1. But this is true since disconnecting s_2 results in a structure with no fair traces from s_1. □

Corollary 2. *The result of disconnecting little brothers with respect to delay/game/exists simulation might not be equivalent to the original structure with respect to delay/game/exists simulation.*

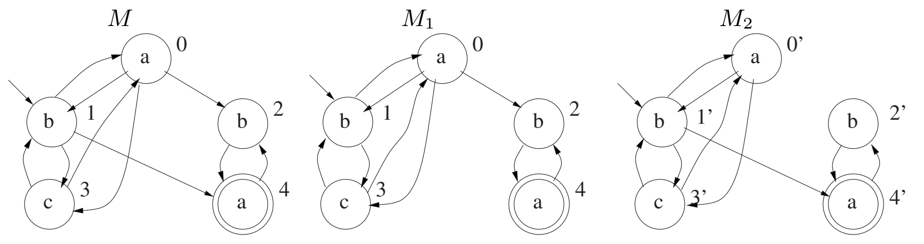

Fig. 3. The structures M_1 and M_2 are equivalent with respect to delay/game/exists simulation to M, and they are both minimal. Note that state 2 ($4'$) is a little brother of 1 ($0'$) but cannot be disconnected.

Unique Smallest in Size Structure

Lemma 5. *Let $\leq_\spadesuit$ be a preorder such that*
$M_1 \leq_{de} M_2 \Rightarrow M_1 \leq_\spadesuit M_2 \Rightarrow M_1 \subseteq M_2$.
There is no unique smallest in size structure with respect to $\leq_\spadesuit$.

Consider the structures in Figure 3. Structures M_1 and M_2 are equivalent with respect to $\leq_\spadesuit$ but are not isomorphic. Furthermore, there is no smaller structure that is equivalent to M_1 and M_2.

Corollary 3. *There is no unique smallest in size structure with respect to delay/game/exists simulation.*

An approximate minimization algorithm for delay/game/exists simulation. In [3] two efficient procedures for minimizing with respect to ordinary simulation are presented. In previous sections we have shown that when we consider game/exists simulation these procedures cannot be used. Furthermore, we have shown that there is no equivalent unique smallest in size structure with respect to these simulations. As a result we are suggesting an algorithm that performs some minimization but does not necessarily construct a minimal structure. Our algorithm uses the direct/delay simulations as an approximation of the game/exists simulation. The algorithm is presented in Figure 4. The first step

Given a structure M,

1. Construct a quotient structure M' with respect to delay simulation.
2. Construct M'' by disconnecting little brothers in M' with respect to direct simulation.

Fig. 4. Minimization algorithm for the delay/game/exists simulations.

results in $M' \equiv_{de} M$. The second step results in $M'' \equiv_{di} M'$. Since direct simulation implies delay simulation, $M'' \equiv_{de} M$. M'' is equivalent to M also with respect game/exists simulation. The complexity of the first step is $O(m \cdot n^3)$ [7], and of the second step $O(m \cdot n)$ [3]. Thus the total complexity of the algorithm is $O(m \cdot n^3)$.

4 Relating the Simulation Notions with Logics

In this section we check for each simulation notion whether it has a logical characterization. Then we check whether there exists a maximal structure for ACTL with respect to this notion.

4.1 Logical Characterization

Definition 12. *Logic $\mathcal{L}$ characterizes a preorder $\leq$ if for all structures M_1 and M_2, $M_1 \leq M_2$ if and only if for every formula ϕ in $\mathcal{L}$, $M_2 \models \phi$ implies $M_1 \models \phi$.*

[9] shows that, if $M_1 \leq_\exists M_2$ then the following property holds. $\forall \phi \in$ ACTL*, $M_2 \models \phi$ implies $M_1 \models \phi$. Since all other simulation notions imply the exists simulation, this property holds for all of these notions.

We now investigate which of the fair simulations satisfy the other direction of logical characterization. In [1] it is shown that **CTL*** characterizes the exists bisimulation. The proof that ACTL* characterizes the exists simulation is similar.

Unlike ACTL*, ACTL does not characterize the exists simulation. In [1] two structures M_1 and M_2 are given. It is shown in [1] that for every ϕ in ACTL, $M_2 \models \phi$ implies $M_1 \models \phi$. However, there exists an ACTL* formula φ such that $M_2 \models \varphi$ but $M_1 \not\models \varphi$. Since ACTL* characterizes the exists simulation, $M_1 \not\leq_\exists M_2$.

Unfortunately, the game, direct and delay simulations cannot be characterized by either ACTL* or ACTL. In [10] two structures M_1 and M_2 are given such that $M_1 \leq_\exists M_2$ but $M_1 \not\leq_g M_2$. Since ACTL* characterizes the exists simulation, for every ϕ in ACTL* (and therefore ACTL), $M_2 \models \phi$ implies $M_1 \models \phi$. Therefore, ACTL* (ACTL) does not characterizes the game simulation. Since the direct/delay simulation implies the game simulation, ACTL* (ACTL) does not characterize them as well.

The question that arises is, can the direct/delay/game simulation be characterized by any other logic. [10] shows that the game simulation can be characterized by the Universal Alternating Free μ-Calculus ($\forall$**AFMC**) logic when interpreted over fair structures.

We show that no reasonable logic that describes the fair branching behavior of a structure can characterize the direct/delay simulation. Consider structures M_1 and M_2 in Figure 5. M_1 and M_2 cannot be distinguished by a temporal logic formula. However, $M_1 \not\leq_{de} M_2$ and $M_1 \not\leq_{di} M_2$. Thus both simulation cannot be characterized by any such logic.

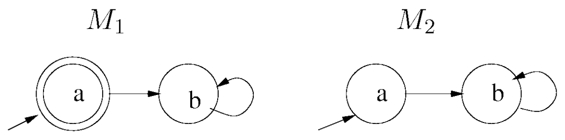

Fig. 5. The direct/delay simulations can not be characterized by temporal logics.

4.2 Maximal Structure

Next, we check for the existence of a maximal structure for a formula with respect to a preorder.

Definition 13. *A structure M_ϕ is maximal for formula ϕ with respect to preorder $\leq$ if for every structure M, $M \models \phi \Leftrightarrow M \leq M_\phi$.*

[9] presents a maximal structure for ACTL formulas with respect to $\leq_\exists$. Here we show that the same structure is maximal with respect to the game simulation. On the other hand we show that the formula $\mathbf{A}[a\,\mathbf{U}\,b]$ has no maximal structure with respect to the direct and delay simulations. This formula is contained in both ACTL and ACTL*.

A maximal structure for ACTL with respect to game simulation. We prove that for every ACTL formula, the tableau of the formula as defined in [9], is the maximal structure for the formula with respect to the game simulation. Before we prove that, we give the main details of the tableau construction. In [9] a different type of fairness constraints called generalized Büchi acceptance condition is used. A *generalized Büchi acceptance* condition is a set $F = \{f_1, f_2, \ldots f_n\}$ of subsets of S. A trace ρ is *fair* according to F iff for every $1 \leq i \leq n$, $\inf(\rho) \cap f_i \neq \emptyset$. Since the game simulation is not limited to a certain type of fairness constraints, we do not have to change anything in its definition.

For the remainder of this section, fix an ACTL formula ψ. Let AP_ψ be the set of atomic propositions in ψ. The tableau associated with ψ is a structure $\mathcal{T}_\psi = (S_T, R_T, S_{0T}, L_T, F_T)$.

We first define the set of *elementary formulas* $el(\psi)$ of ψ. This set consists of the atomic propositions in ψ, subformulas of ψ of the form $\mathbf{AX}\,\phi$, and $\mathbf{AX}\,\mathbf{A}[\phi_1 \mathbf{U}\,\phi_2]$, $\mathbf{AX}\,\mathbf{A}[\phi_1\,\mathbf{R}\,\phi_2]$, for every $\mathbf{A}[\phi_1\,\mathbf{U}\,\phi_2]$, $\mathbf{A}[\phi_1\,\mathbf{R}\,\phi_2]$ subformulas of ψ.

The set of tableau states is $S_T = \mathcal{P}(el(\psi))$[4]. The labeling function is $L_T(s_t) = s_t \cap AP_\psi$. In order to specify the set S_{0T} of initial states and the transition relation R_T, we need an additional function *sat* that associates with each sub-formula ϕ of ψ a set of states in S_T. Intuitively, $sat(\phi)$ will be the set of states that satisfy ϕ. The set of initial states of the tableau is $S_{0T} = sat(\psi)$. The transition relation is defined so that if $\mathbf{AX}\,\phi$ is included in some state then all its successors should satisfy ϕ.

$$R_T(s_1, s_2) = \bigwedge_{AX\,\phi \in el(\psi)} (\mathbf{AX}\,\phi) \in s_1 \Rightarrow s_2 \in sat(\phi).$$

The fairness constraint guarantees that *eventuality* properties are fulfilled.

$$F_T = \left\{ \big((S_T - sat(\mathbf{AX}\,\mathbf{A}[\phi\,\mathbf{U}\,\varphi])) \cup sat(\varphi)\big) \;\middle|\; \mathbf{AX}\,\mathbf{A}[\phi\,\mathbf{U}\,\varphi] \in el(\psi) \right\}.$$

The tableau is the maximal structure for game simulation. We now summarize the main steps in the proof that for every Kripke structure M, $M \models \psi$ iff $M \leq_g \mathcal{T}_\psi$. The steps included in Lemma 6 are proved in [9]. The other steps are different from [9] due to the change in the preorder.

Lemma 6. *[9]*

- *For all sub-formulas ϕ of ψ and $t \in S_T$, if $t \in sat(\phi)$, then $t \models \phi$.*
- *For every ACTL formula ψ, $\mathcal{T}_\psi \models \psi$.*
- *if $M \leq_g \mathcal{T}_\psi$, then $M \models \psi$.*

[4] Some of the states are deleted in order to keep R_T total

Our next step is to prove that $M \models \psi$ implies $M \leq_g \mathcal{T}_\psi$. We show that if $M \models \psi$ then the protagonist has a winning strategy function in a game over $M \times \mathcal{T}_\psi$. We define the strategy function π as follows: $\pi(s_0, \bot) = \{\, \phi \mid \phi \in el(\psi), s_0 \models \phi \,\}$ and $\pi(s', t) = \{\, \phi \mid \phi \in el(\psi), s' \models \phi \,\}$. Thus, whenever the adversary moves to a state s' the protagonist moves to $t' = \pi(s', t)$, such that both s', t' satisfy exactly the same set of elementary formulas of ψ. It can also be shown that s' and t' agree on every subformula of ψ.

Lemma 7. *π is a winning strategy.*

Corollary 4. *For any structure M, $M \models \psi$ iff $M \leq_g \mathcal{T}_\psi$. Thus, $\mathcal{T}_\psi$ is the maximal structure for ψ with respect to game simulation.*

A Maximal Structure for Direct/Delay Simulation

We now show that it is impossible to construct a maximal structure for the formula $\phi = \mathbf{A}[a\,\mathbf{U}\,b]$ with respect to the direct/delay simulations. Thus, any logic that contains this formula or an equivalent formula, in particular ACTL and ACTL*, does not have a maximal structure with respect to these simulations. Since the direct simulation implies the delay simulation, it is sufficient to prove this result for the delay simulation. Consider the structures $M_0, M_1, \ldots$ in Figure 6, each of which satisfies $\mathbf{A}[a\,\mathbf{U}\,b]$. The following lemma shows that no finite structure can be greater by the delay simulation than all of these structures.

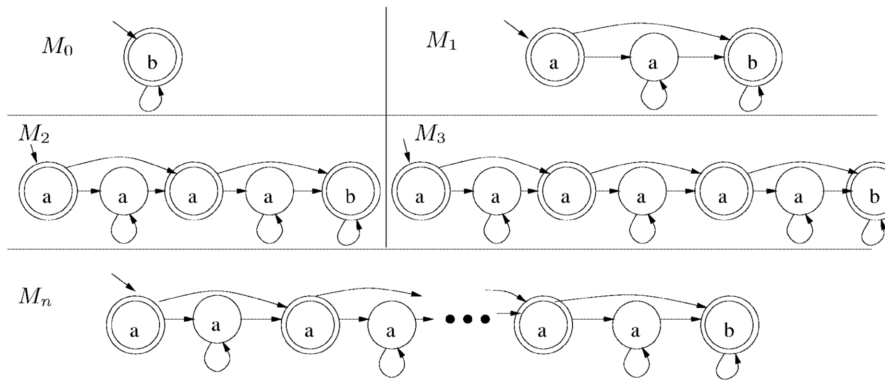

Fig. 6. There is no finite structure M' such that for every n in $I\!N$, M' is greater by direct/delay simulation than M_n, and $M' \models \mathbf{A}[a\,\mathbf{U}\,b]$

Lemma 8. *For every $n > 0$ and every structure M', if $M_n \leq_{de} M'$ and $M' \models \mathbf{A}[a\,\mathbf{U}\,b]$ then $|M'| \geq n$.*

5 A New Implementation for the Assume-Guarantee Framework

This section shows that the game simulation can replace the exists simulation in the implementation of the assume-guarantee paradigm [8,11,15,16] as suggested in [9].

[9] suggests a framework that uses the assume-guarantee paradigm for semi-automatic verification. It presents a general method that uses models as assumptions; the models are either generated from a formula as a *tableau*, or are abstract models which are given by the user. The proof of $\psi M \phi$ meaning that, if ψ is true in the environment then $M \models \phi$, is done automatically by verifying that the composition of the tableau for ψ with M satisfies ϕ. The method requires a preorder $\leq$, a composition operator $||$ and a specification language $\mathcal{L}$. In [9] an implementation for this framework is presented. The implementation uses the $ACTL$ logic as the specification language, the exists simulation preorder and a composition operator. The ability to replace the exists simulation by the game simulation is implied by Lemma 9.

Lemma 9. *1. For every two structures M_1, M_2, if $M_1 \leq_g M_2$ then for every formula ψ in $\mathcal{L}$, $M_2 \models \psi$ implies $M_1 \models \psi$.*
2. For every two structures M_1, M_2, $M_1 \| M_2 \leq_g M_1$.
3. For every three structures M_1, M_2, M_3, $M_1 \leq_g M_2$ implies $M_1 \| M_3 \leq_g M_2 \| M_3$.
4. Let ψ be a formula in $\mathcal{L}$ and $\mathcal{T}_\psi$ be a tableau for ψ, then $\mathcal{T}_\psi$ is the maximal structure with respect to the preorder $\leq_g$.

Complexity. Verifying a formula of the form $\psi M \varphi$ is PSPACE-complete in the size of ψ [13]. However, the real bottleneck of this framework is checking for fair simulation between models, which for the exists simulation is PSPACE complete in the size of the models (typically models are much larger than formulas). Thus replacing the exists simulation by the game simulation reduces this complexity to polynomial and eliminates the bottleneck of the framework. However, the algorithm for game simulation, presented in [7], refers to Kripke structures with regular Büchi constraints, and the implementation presented in [9] refers to Kripke structures with generalized Büchi constraints. In order to apply the algorithm suggested in [7] in the assume-guarantee framework, we need a translation between these types of fairness constraints.

[5] defines a transformation of a Büchi automaton with generalized fairness constraints into a Büchi automaton with regular fairness constraints. The result of the transformation is game equivalent to the original structure, thus can replace it. The translation effects the size of the structure and thus the complexity of the construction of the preorder. The sizes of S and R are multiplied by $|F|$, where $|F|$ is the number of sets in F. Thus the complexity of constructing the preorder is $|F| \cdot |R| \cdot (|S| \cdot |F|)^3 = |R| \cdot |S|^3 \cdot |F|^4$. Note that in the tableau for a formula, $|F|$ is bounded by the size of the formula and the size of the tableau is exponential in the size of the formula, thus, the transformation of the tableau to regular fairness constraints is logarithmic in its size.

6 Conclusion

The main consequence of this work is that there is no notion of fair simulation which has all the desired advantages. However, it is clear that the exists and game simulations have advantages in the relationship with the logics over the delay and direct simulations. On the other hand, the delay and direct simulations are better for minimization. Thus, it is advantageous to refer to the delay and direct simulations as approximations of the game/exists simulations. These approximations enable some minimization with respect to the exists and game simulations. Out of the four notions, we consider the game simulation the best due to its complexity and its applicability for modular verification.

References

1. A. Aziz, V. Singhal, T.R. Shiple, A.L. Sangiovanni-Vincentelli, F. Balarin, and R.K. Brayton. Equivalences for fair kripke structures. In *ICALP*, LNCS 840, pages 364–375, 1994.
2. S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis. Property preserving simulation. In *Computer-aided Verification*, volume 663 LNCS, pages 260–273, 1981.
3. D. Bustan and O. Grumberg. Simulation based minimization. In *Conference on Automated Deduction*, volume 17, pages 255–270, 2000.
4. E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking.* MIT Press, 1999.
5. C. Courcoubetis, M. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. In *Proceedings of Computer-Aided Verification*, volume 531 of *LNCS*, pages 233– 242, 1991.
6. D.L. Dill, A.J. Hu, and H. Wong-Toi. Checking for language inclusion using simulation relation. In *Computer-Aided Verification*, LNCS 575, pages 255–265, 1991.
7. K. Etessami, Th. Wilke, and R. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. In *Automata, Languages and Programming, 28th international collquium*, LNCS 2076, pages 694–707, 2001.
8. N. Francez. *The Analysis of Cyclic Programs.* PhD thesis, Weizmann Institute of Science, 1976.
9. O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Trans. on Programming Languages and Systems (TOPLAS)*, 16(3):843–871, 1994.
10. T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *Proc. 8th Conference on Concurrency Theory*, LNCS 1234, 1997.
11. C. B. Jones. Specification and design of (parallel) programs. In *In International Federation for Information Processing (IFIP)*, pages 321–332, 1983.
12. O. Kupferman and M.Y. Vardi. Verification of fair transition systems. In *Computer Aided Verification (CAV'96)*, LNCS 1102, pages 372–382, 1996.
13. O. Kupferman and M.Y. Vardi. Modular model checking. In *Proc. Compositionality Workshop*, LNCS 1536. Springer-Verlag, 1998.
14. R. Milner. An algebraic definition of simulation between programs. In *Proc. of the 2nd International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 481–489, London, UK, 1971.
15. J. Misra and K.M. Chandy. Proofs of networks of processes. *IEEE Transactions on Software Engineering*, 7(7):417–426, 1981.
16. A. Pnueli. In transition for global to modular temporal reasoning about programs. In K. R. Apt, editor, *Logics and Models of Concurrent Systems*, volume 13 of *NATO ASI series F*. sv, 1984.