Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information and Computation 194 (2004) 1–18
Information and Computation
www.elsevier.com/locate/ic

Applicability of fair simulation

Doron Bustan*, Orna Grumberg*

Computer Science Department Technion, Haifa 32000, Israel

Received 29 September 2002; revised 5 November 2003

Abstract

In this paper we compare four notions of fair simulation: direct [9], delay [12], game [19], and exists [16]. 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. We developed a practical application that is based on this comparison. The application is a new implementation for the assume-guarantee modular framework presented By Grumberg at al. in [ACM Transactions on Programming Languages and Systems (TOPLAS), 16 (1994) 843]. The new implementation significantly improves the complexity of the framework.
Published by Elsevier Inc.

Keywords: Fair simulation; Minimization; Branching time logics; Preorder; Assume-guarantee

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, including abstraction, partial order reduction,

* Corresponding authors. Fax: +972 4 829 4353 (D. Bustan).
E-mail addresses: doron_b@cs.rice.edu (D. Bustan), orna@cs.technion.ac.il (O. Grumberg).

0890-5401/$ - see front matter. Published by Elsevier Inc.
doi:10.1016/j.ic.2004.02.005

modular methods, and symmetry [7]. These approaches are often based on the idea that the model of the verified system can be replaced by a more abstract model, smaller in size. The abstract and concrete models are sufficiently similar so that properties that are verified on the abstract model can be considered true for the concrete one. This idea is often formalized by relating models with the *simulation preorder* [27], 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* [16].

It often happens that during the construction of a reduced abstract model some unrealistic infinite behaviors are added. A common way to avoid these behaviors is to add fairness constraints to 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 [9,12,19,16].

Researchers have addressed the question of which notion of fair simulation is preferable. In [19], some of these notions are compared with respect to the complexity of checking for fair simulation. In [12], 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 [15,14] the definitions of three of these logics are extended for alternating Büchi automata, and are compered with respect to their applicability to minimization.

This paper is an extension of [6], in the paper we make a broader comparison of four notions of fair simulation: direct [9], delay [12], game [19], and exists [16]. We refer to several criteria that emphasize the advantages of each of the notions. The results of the comparison are summarized in Table 1.

In addition, we suggest a new implementation for the *assume-guarantee* [13,20,28,29] modular framework presented in [16]. 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 Table 1). We see that constructing the direct, delay, and game simulations is polynomial in the number of states n and

Table 1
The properties of the different notions of fair simulation

Notion	Time complexity of constructing the preorder	Minimization			Relation to logic	
		Unique smallest model	Quotient model	Little brothers	Has logical characterization	Max model
Direct	$O(m \cdot n)$ [12]	*true* [11,5,30]	*true* [11,5,30]	*true*	*false*	*false*
Delay	$O(m \cdot n^3)$ [12]	*false*	*true*[a]	*false*	*false*	*false*
Game	$O(m \cdot n^3)$ [12]	*false*	*false* [12]	*false*	$\forall AFMC$ [19]	*true*
Exists	PSPACE complete [23]	*false*	*false*	*false*	$ACTL^*$	*true* [16]

[a] In [12] it is shown that the quotient model is *language equivalent* to the original model. Here, we show that they are delay equivalent.

the number of transitions m [12]. In contrast, constructing the exists simulation is PSPACE-complete [23], which is a great disadvantage.

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 that is equivalent with respect to the preorder to the original one.[1]

For models with no fairness constraints there exists a unique smallest in size model which is simulation equivalent to them [5]. This unique and smallest in size model is the result of eliminating two types of redundancies. One is the existence of equivalent states. This redundancy is eliminated by constructing a *quotient model*. The other is the existence of a successor of a state whose behavior is contained in the behavior of another successor of the same state. Such a state is called *a little brother*. This redundancy is eliminated by *disconnecting little brothers*. For each of the fair simulation preorders, we check the following:

(1) Is there a unique smallest in size model with respect to simulation equivalent?
(2) Is the quotient model of model M, simulation equivalent to M?
(3) Is the result of disconnecting little brothers in a model M, simulation equivalent to M?

The third aspect that we investigate is the relationship between the simulation preorders and universal branching-time logics. A basic requirement of using a preorder in verification is that it preserves the specification logic, i.e., if $M_1 \leqslant M_2$ 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 have a *logical characterization* by some logic. This means that $M_1 \leqslant M_2$ **if and only if** for every formula ϕ in the logic, $M_2 \models \phi$ implies $M_1 \models \phi$.

Logical characterization of different specification logics is well studied [18,3,2,1,19]. For abstraction, this definition is useful in determining if model M_2 can be used as an abstraction for is an abstraction of model M_1, when the logic L should be preserved. If the preorder $\leqslant$ is logically characterized by L then checking $M_1 \leqslant M_2$ is a necessary and sufficient condition and will never give a false negative result.

Another important relationship between a logic and a preorder is the existence of a *maximal model* T_ϕ for a formula ϕ with respect to the preorder. The maximal model T_ϕ for a formula ϕ is such that for every model M', $M' \leqslant T_\phi$ if and only if $M' \models \phi$. Maximal structure of formula ψ can be used for abstracting an environment of a verified system, where we assume that the environment satisfies some requirements specified by ψ.

The results of our comparison direct us towards an improvement of a framework described in [16] for the *assume-guarantee paradigm*. The assume-guarantee is an inductive modular verification paradigm in which the environment of the verified part can be represented by a formula. The paradigm is used to create a proof schema which is based on the modular structure of the system. In [16], a semi-automatic framework for the assume-guarantee paradigm is presented. The framework uses the exists preorder and is defined with respect to the logic ACTL. We suggest that the game

[1] Note that this is a stronger criterion than the one used in [12], where only language equivalence is required.

simulation should replace the exists simulation in the framework, thus reducing its complexity dramatically.

The results of our comparison are presented in the Table 1. The rest of the paper is organized as follows: In Section 2, we define the simulation preorder and the different notions of fair simulation. Section 3 investigates simulation minimization. In Section 4, each notion is checked for logical characterization and for the existence of a maximal structure. 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, $M = \langle S, R, S_0, L, F\rangle$, where S is a finite set of states, $S_0 \subseteq S$ is a set of initial states, and $R \subseteq S \times S$ is the transition relation, which must be *total*. This means that 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.

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 $i \geqslant 0$, $(s_i, s_{i+1}) \in R$. The ith state of trace ρ is denoted $\rho[i]$, and the suffix of ρ starts at $\rho[i]$ is denoted ρ^i. 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 [16]. First, we define CTL* formulas in negation normal form, namely, negation is applied only to atomic propositions. CTL* contains trace formulas and state formulas and is defined inductively:

- Let p be an atomic proposition, then p and $\neg p$ are both state formulas and trace formulas.
- Let φ and ψ be trace formulas, then
  - $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$ are trace formulas.
  - $\mathbf{X}\varphi$, $(\varphi \mathbf{U} \psi)$, and $(\varphi \mathbf{R} \psi)$ are trace formulas.
  - $A\varphi$ and $E\varphi$ are state formulas.
- Let φ and ψ be state formulas, then
  - $(\varphi \vee \psi)$ and $(\varphi \wedge \psi)$ are state formulas.
  - φ and ψ are trace formulas.

Next we define the semantics of CTL* with respect to fair Kripke structures. A state formula ϕ is satisfied by a structure M at state s, denoted $M, s \models \phi$, if the following holds (M is omitted if clear from the context):

- For $p \in AP$, $s \models p$ iff $p \in L(s)$; $s \models \neg p$ iff $p \notin L(s)$.
- $s \models \phi \wedge \psi$ iff $s \models \phi$ and $s \models \psi$; $s \models \phi \vee \psi$ iff $s \models \phi$ or $s \models \psi$.

- $s \models \mathbf{A}\varphi$ iff for every fair trace ρ from s, $\rho \models \varphi$.
- $s \models \mathbf{E}\varphi$ iff there exists a fair trace ρ from s, such that $\rho \models \varphi$.

The conditions that a trace satisfies a trace formula φ, denoted $\rho \models \varphi$, are:

- $\rho \models \mathbf{X}\varphi$ iff $\rho^1 \models \varphi$.
- $\rho \models \varphi \mathbf{U} \psi$ iff for some $i \geqslant 0$, $\rho^i \models \psi$ and for all $j < i$, $\rho^j \models \varphi$.
- $\rho \models \varphi \mathbf{R} \psi$ iff for all $i \geqslant 0$, if for every $j < i$, $\rho^j \not\models \varphi$ then $\rho^i \models \psi$.

ACTL* is the universal fragment of CTL* where the only trace quantifier allowed is **A**. ACTL is a subset of ACTL* where every temporal operator is immediately preceded by the **A** quantifier. The modal logic as defined in [18] is equivalent to a subset of CTL* where the only temporal operator is **X** and every **X** operator is immediately preceded by a quantifier. The universal fragment of the modal logic is a restriction to the **A** quantifier only. 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 the simulation relation over Kripke structures with $F = S$ (Kripke structures with trivial fairness constraints).

Definition 2.1. Given two structures M_1 and M_2 over AP, a relation $H \subseteq S_1 \times S_2$ is a simulation relation [27] over $M_1 \times M_2$ iff the following holds:

(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 $M_1 \leqslant M_2$) 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* $M_1 \leqslant M_2$ and $M_2 \leqslant M_1$. Similarly, $(s_1, s_2) \in H$, is denoted $s_1 \leqslant s_2$ and s_1 and s_2 are equivalent if $s_1 \leqslant s_2$ and $s_2 \leqslant s_1$. This equivalence is denoted $s_1 \equiv s_2$. The relation $\leqslant$ is a preorder on the set of structures. This means that it is reflexive and transitive. Next, we define the different notions of fair simulation. The first notion is the direct simulation, which is the most straightforward extension of the ordinary simulation.

Definition 2.2. $H \subseteq S_1 \times S_2$ is a *direct simulation relation* [9] ($\leqslant_{di}$) over $M_1 \times M_2$ iff it satisfies the conditions of Definition 2.1, except that here 2a is replaced by:

$2(a')L_1(s_1) = L_2(s_2)$ and $s_1 \in F_1$ implies $s_2 \in F_2$.

We now define the exists simulation:

Definition 2.3 ([16])**.** $H \subseteq S_1 \times S_2$ is an *exists simulation* ($\leqslant_{\exists}$) over $M_1 \times M_2$ iff it satisfies the conditions of Definition 2.1, except that here 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 $i \in \mathbb{N}$, $(\rho_1[i], \rho_2[i]) \in H$.[2]

The next definitions are based on games over Kripke structures. We start with a game for ordinary simulation. A game is played by 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 2.4. 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, and 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, the game terminates and the protagonist fails. Otherwise, if the protagonist always has a matching successor to move to, the game proceeds ad infinitum for ω rounds and the protagonist wins. The adversary wins iff the protagonist fails.

Definition 2.5. 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 when the adversary initially 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, 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 Definition 2.1 [19].

Definition 2.6. Given two Kripke structures, M_1 and M_2, M_2 simulates M_1 ($M_1 \leqslant M_2$) iff the protagonist has a winning strategy in a simulation gameover M_1, M_2.

To extend the simulation gameto fair simulation, we add a winning condition which refers to the infinite properties of the game. We now give the definitions of the delay ($\leqslant_{de}$) and the game ($\leqslant_g$) simulations.

Definition 2.7 ([12])**.** 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 thereafter.

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

Definition 2.8 ([19]). 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 2.9 ([19,12]). 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 2.2, 2.3, and 2.9 are extensions of Definition 2.1 and its equivalent Definition 2.6. Consequently, on structures with trivial fairness constraints ($F = S$), all four definitions are equivalent. In [19,12] the following relationships over the fair simulation preorders are shown:

$$M_1 \leqslant_{di} M_2 \;\Rightarrow\; M_1 \leqslant_{de} M_2 \;\Rightarrow\; M_1 \leqslant_{g} M_2 \;\Rightarrow\; M_1 \leqslant_{\exists} M_2.$$

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 have different types of fairness constraints. Finally, we extend the delay/game simulations for states.

Definition 2.10. For all states s_1 and s_2 in a structure M, $s_1 \leqslant_{de/g} s_2$ 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.

Other relations we use are, language equivalence and language containment.

Definition 2.11.

- 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 $\forall i \geqslant 0, L(\rho_1[i]) = L(\rho_2[i])$.
- $M_1 \subseteq M_2$ if for every fair trace ρ_1 starting at an initial state of M_1, there is a fair trace ρ_2 starting at an initial state of M_2 such that $\forall i \geqslant 0, L_1(\rho_1[i]) = L_2(\rho_2[i])$.
- 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.

3. Simulation minimization

In this section we compare the applicability of the different notions of fair simulation for state space reduction. In [5], the use of ordinary simulation, for state space reduction is investigated thoroughly. It is shown in [5] that for ordinary simulation and structures with trivial fairness constraints, there are two forms of redundancy. The first is the existence of simulation equivalent states. This redundancy is being eliminated by unifying all equivalent states. The result of unifying all equivalent states is a quotient structure. The other redundancy is the existence of transition to *little brothers*, meaning successors that are not maximal with respect to the simulation preorder. This redundancy

is eliminated simply by removing these transitions (also called *disconnecting little brothers*). For a structure M, with trivial fairness constraints, eliminating these redundancies and removing unreachable states, results in a unique, smallest in size structure that is simulation equivalent to M [5]. In the following, we determine for every simulation notion, whether or not these useful properties of the ordinary simulation hold.

We start with direct simulation. In [11,30], reduction algorithms that uses the direct simulation are presented. These algorithms eliminate both redundancies. Then, it is shown that the result is language equivalent to the original. Lemma 3.1 can be easily deduced from [11,30].

Lemma 3.1. *For every structure M, the following holds*:

(1) *The quotient structure of M with respect to direct simulation is direct simulation equivalent to M.*
(2) *The result of removing transitions to little brothers in M using direct simulation is direct simulation equivalent to M.*

The direct simulation can also be considered as an ordinary simulation over structures with additional proposition F that is true in the fair states. Thus, Lemma 3.1 is also a direct consequence of the result in [5]. Another direct consequence of [5] is the following lemma:

Lemma 3.2. *For every structure, there exists a unique, smallest in size structure that is direct simulation equivalent to it.*

Unfortunately, performing the same operations for the other notions of fair simulations might result in an inequivalent structure. We continue our investigation by checking for the delay/game/exists simulations whether the quotient structure is equivalent to the original one. The *quotient structure* is the result of unifying all equivalent states into equivalence classes. Recall that states s_1 and s_2 are equivalent if $s_1 \leqslant s_2$ and $s_2 \leqslant s_1$. 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.

In [12], it is shown that for delay simulation the quotient structure is language equivalent to the original structure. The proof of the following lemma is similar to the proof in [12], and thus omitted. The full proof is presented in [4].

Lemma 3.3. *Let M^Q be the quotient structure of a structure M. Then $M \equiv_{de} M^Q$.*

Etessami et al. [12] shows a Büchi automaton M_n with n states that its quotient automaton with respect to game simulation has one state, and there is no automaton with fewer than n states that is language equivalent to M_n. Lemma 3.4 extends this result for all the preorders $\leqslant_\clubsuit$ that lie between game simulation and language containment.

Lemma 3.4. *Let $\leqslant_\clubsuit$ be any preorder such that for every M_1, M_2,*

$$M_1 \leqslant_g M_2 \Rightarrow M_1 \leqslant_\clubsuit M_2 \Rightarrow M_1 \subseteq M_2.$$

Then the quotient structure of M_n with respect to $\leqslant_\clubsuit$ is not equivalent to M_n with respect to $\leqslant_\clubsuit$.

Proof. Let $\leqslant_{\clubsuit}$ be a preorder that lies between language equivalent and game simulation. Since $\leqslant_g \Rightarrow \leqslant_{\clubsuit}$, the quotient structure M_n^Q of M_n with respect to $\leqslant_{\clubsuit}$ has only one state. This implies that $L(M_n) \neq L(M_n^Q)$. Since $\leqslant_{\clubsuit}$ implies language containment, M_n is not equivalent to M_n^Q with respect to $\leqslant_{\clubsuit}$. □

Corollary 3.5. *For the game and exists simulations, the quotient structure is not necessarily equivalent to the original structure.*

Next, we show that for the delay/game/exists simulations, disconnecting little brothers may not result in an equivalent structure. Formally, a state s_2 is a *little brother* of another state s_3 if both states are successors of the same state $s_1, s_2 \leqslant s_3$, and $s_3 \not\leqslant s_2$. A little brother s_2 is being disconnected by removing the transition (s_1, s_2) from R.

Lemma 3.6. *Let $\leqslant_{\spadesuit}$ be a preorder such that*

$$M_1 \leqslant_{de} M_2 \Rightarrow M_1 \leqslant_{\spadesuit} M_2 \Rightarrow M_1 \subseteq M_2.$$

Let M' be the result of disconnecting little brothers in structure M with respect to $\leqslant_{\spadesuit}$. M' might not be equivalent to M with respect to $\leqslant_{\spadesuit}$.

As an example observe the structure M_1 in Fig. 1. State 2 is a little brother of state 1 with respect to delay/game/exists simulation. However, the result of removing the transition $(0, 2)$ is not delay/game/exists simulation equivalent to M_1. This is because disconnecting state 2 results in a structure with no fair traces from state 0. The full proof is presented in [4]. A generalization of this lemma has been independently developed in [15].

Corollary 3.7. *The structure that results when little brothers are disconnected with respect to delay/game/exists simulation might not be equivalent to the original structure with respect to delay/ game/ exists simulation.*

The example in Fig. 1 also demonstrates the following result:

Lemma 3.8. *Let $\leqslant_{\spadesuit}$ be a preorder such that*

$$M_1 \leqslant_{de} M_2 \Rightarrow M_1 \leqslant_{\spadesuit} M_2 \Rightarrow M_1 \subseteq M_2.$$

Then there exists a structure M that has no unique smallest in size structure with respect to $\leqslant_{\spadesuit}$.

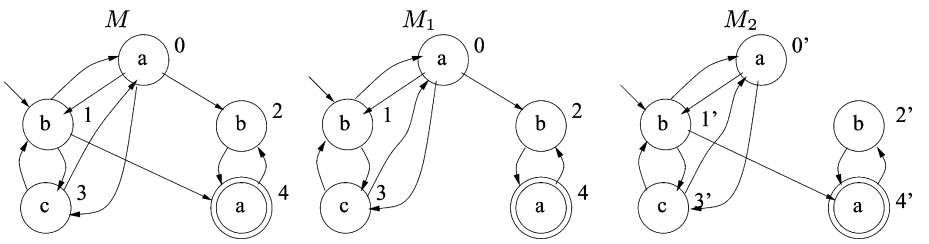

Fig. 1. The structures M_1 and M_2 are equivalent with respect to delay/game/exists simulation to M, and they are both minimal. Note that state 2 (4′) is a little brother of 1 (0′) but cannot be disconnected.

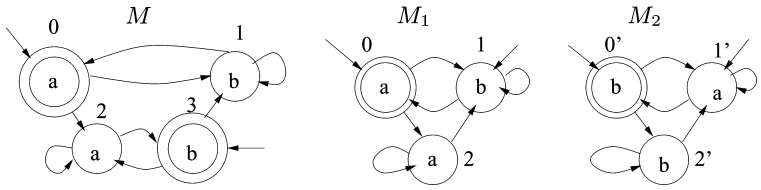

Fig. 2. The structures M_1 and M_2 are equivalent to M with respect to game/exists simulation, and they are both minimal. Note that states 0 and 2 ($0'$ and $2'$) are equivalent but cannot be unified. (Double circles denote fair states.)

Corollary 3.9. *There is no unique smallest in size structure with respect to delay/game/exists simulation.*

An interesting observation is that the minimization operations are not *independent* [3] [21]. For example, in structure M in Fig. 2, states 0 and 1 are game/exists equivalent to states 2 and 3, respectively. Unifying states 0 and 2 results in structure M_2. Unifying states 1 and 3 results in structure M_1. Both structures are equivalent to M and neither can be further minimized. A similar phenomenon occurs in structure M of Fig. 2: for delay/exists/game simulation, states 4 and 2 are little brothers of states 0 and 1, respectively. Disconnecting state 4 from state 1 results in M_1, and disconnecting state 2 from state 0 results in M_2. Again, both structures are equivalent to M, and neither structure can be further minimized. In [5], two efficient procedures for minimizing with respect to ordinary simulation are presented. In the above, we have shown that these procedures cannot be used for delay/game/exists simulation. Furthermore, we have shown that there is no equivalent unique smallest in size structure with respect to these simulations. These results implies that heuristics for finding minimal but not necessarily smallest structure are appropriate. An example for an algorithm that uses such heuristics is presented in [17]. The algorithm constructs a reduced automaton that is game simulation equivalent to the original one and is minimal in the sense that no equivalent state can be unified and no little brothers can be disconnected.

4. Relating the simulation notions to logics

In this section we investigate the relationships between the different notions of fair simulation and specification languages. First, we check for each notion whether it has a logical characterization. Then, we check whether there exists a maximal structure for ACTL with respect to each notion.

4.1. Logical characterization

Equivalence relations and preorders over Kripke structures can be naturally expressed using specification languages in the following way: Two structures M_2 and M_1 are equivalent with respect

[3] Operations are not independent if one operation disables another.

to a specification language L ($M_1 \equiv_L M_2$), if for every formula ψ in L we have $M_1 \models \psi$ iff $M_2 \models \psi$. A structure M_1 is smaller than structure M_2 with respect to specification language L ($M_1 \leqslant_L M_2$), iff for every formula ψ in L we have that $M_2 \models \psi$ implies $M_1 \models \psi$. Thus, the specification language L determines whether an abstract model can replace an equivalent or smaller w.r.t. L, concrete model. However, this natural definition, does not implies an algorithm for checking whether $M_1 \leqslant_L M_2$. Obviously we cannot check for every formula in L that $M_2 \models \psi$ implies $M_1 \models \psi$. Fortunately, for some specification languages, $\equiv_L$ and $\leqslant_L$ coincides with different notions of bisimulation and simulation respectively. We say that a specification language L characterizes a bisimulation relation $\equiv_*$ (simulation relation $\leqslant_*$), if $\equiv_L$ and $\equiv_*$ ($\leqslant_L$ and $\leqslant_*$) are the same relations [18].

In [18], it is shown that the modal logic characterizes the ordinary bisimulation, and that universal modal logic characterizes the ordinary simulation. In [3,2,26], it is shown that if a structure M_1 is smaller than M_2 by the ordinary simulation, then for every $\forall$-MC/$\forall$-AFMC/ACTL*/ACTL [4] formula ψ we have $M_2 \models \psi$ implies $M_1 \models \psi$. Since $\forall$-MC, $\forall$-AFMC, ACTL*, and ACTL are more expressive than the universal modal logic, this implies that $\forall$-MC, $\forall$-AFMC, ACTL* and ACTL characterize the ordinary simulation. Thus, for structures without fairness constraints the preorders that are implied by the $\forall$-MC, $\forall$-AFMC, ACTL*, ACTL and the universal modal logic are identical.

This however, is not true for structures with fairness constraints. In [1,19], it is shown that $\leqslant_{\forall AFMC} \subset \leqslant_{ACTL^*} \subset \leqslant_{ACTL}$, where, the containments are strict. Thus, each specification language induces different preorder. Lemma 4.1 is proven in [19].

Lemma 4.1. *The game simulation is characterized by the* $\forall$*-AFMC logic.*

In [16], it is shown that if $M_1 \leqslant_\exists M_2$, then $M_1 \leqslant_{ACTL^*} M_2$. Lemma 4.2 implies that the exists simulation is characterized by the ACTL* logic.

Lemma 4.2. *If* $M \leqslant_{ACTL^*} M'$, *then* $M \leqslant_\exists M'$.

The proof of Lemma 4.2 is an adaptation of the proof for the exists bisimulation [1] and thus omitted. The full proof is presented in [4]. In [1], a bisimulation relation that is characterized by the CTL logic is shown. We conjecture that their definition can be adapted to characterize the $\leqslant_{ACTL}$ preorder, however, this is still an open question.

The question arises whether the direct/delay simulation can be characterized by any specification language. We show that no reasonable logic that describes the fair branching behavior of a structure characterizes the direct/delay simulation. Consider the structures M_1 and M_2 in Fig. 3. M_1 and M_2 cannot be distinguished by a temporal logic formula. This is because they have computation trees, with exactly the same fair traces. However, $M_1 \not\leqslant_{de} M_2$ and therefore, $M_1 \not\leqslant_{di} M_2$. To see that $M_1 \not\leqslant_{de} M_2$ note that if the adversary chooses the path 123^∞ the protagonist must choose the path $1'2'3'^\infty$. However, 2 is a fair state while $2'$ and $3'$ are not. Thus neither simulation can be characterized by any such logic.

[4] The $\forall$-MC and $\forall$-AFMC are the universal fragments of the μ-calculus logic and the alternating free μ-calculus logic, respectively [22,10].

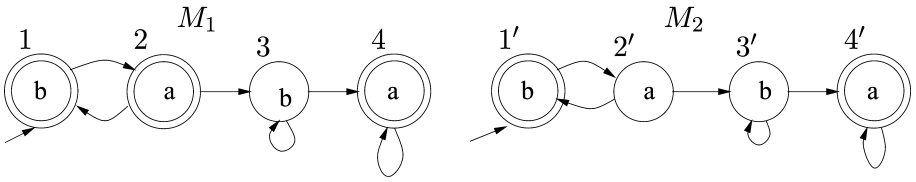

Fig. 3. No temporal logic characterizes the direct/delay simulations.

4.2. Maximal structures

Another important relationship between preorders and specification languages is the existence of maximal structures. A maximal structure M_ψ with respect to formula ψ is considered as an abstraction of the set of structures that satisfy ψ. Usually, we use a maximal structure of formula ψ for abstracting the environment of the verified structure, when the environment is assumed to satisfies ψ. Formally:

Definition 4.3. A structure M_ϕ is maximal for formula ϕ with respect to preorder $\leqslant$ if for every structure M, $M \models \phi \Leftrightarrow M \leqslant M_\phi$.

Given a specification language L and a simulation relation $\leqslant_*$, we would like to know whether for every formula ψ in L there exists a maximal structure of ψ with respect to $\leqslant_*$. Lemma 4.4 is presented in [25].

Lemma 4.4. *For every ACTL* formula ψ there exists a maximal structure for ψ with respect to $\leqslant_\exists$.*

Lemma 4.5 is presented in [16].

Lemma 4.5. *For every ACTL formula ψ there exists a maximal structure for ψ with respect to $\leqslant_\exists$.*

Note that Lemma 4.5 is implied by Lemma 4.4. Nevertheless, the construction of the maximal structure for ACTL formulas as presented in [16] is exponential in the size of the formula, while the construction of the maximal structure for ACTL* formulas as presented in [25] is double exponential in the size of the formula. Lemma 4.6 implies that there exists a maximal structure with respect to ACTL and game simulation. The lemma is proved by showing that the maximal structure which is defined in [16] is also a maximal structure with respect to game simulation. The proof is similar to the proof Lemma 4.5 in [16], and is presented in [4].

Lemma 4.6. *For every ACTL formula ψ there exists a maximal structure for ψ with respect to $\leqslant_g$.*

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. More specifically, we show that there is no finite structure $\mathcal{T}_\phi$ such that $\mathcal{T}_\phi \models \phi$ and $\mathcal{T}_\phi$ is greater by the direct/delay simulation than any structure that satisfies ϕ. Since the direct

simulation implies the delay simulation, it is sufficient to prove this result for the delay simulation. In Fig. 4 we present a sequence of structures $M_0, M_1, \ldots$ such that for every n in $\mathbb{N}$, $M_n \models A(aUb)$. Lemma 4.7 implies that every structure that satisfies $\mathbf{A}[a\ \mathbf{U}\ b]$ and is greater by the delay simulation than all the structures in the sequence has to be infinite.

Lemma 4.7. *For every $n > 0$ and every structure M', if $M_n \leqslant_{de} M'$ and $M' \models \mathbf{A}[a\ \mathbf{U}\ b]$, then $|M'| \geqslant n$.*

The proof of Lemma 4.7 is presented in Appendix.

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 [13,20,28,29], as suggested in [16].

In the assume-guarantee paradigm, properties of different parts of the system are verified separately. The environment of the verified part is represented by a formula that describes its properties. The formula either has been verified or is given by the user. The method proves assertions of the form $\psi M \phi$, meaning that if the environment satisfies ψ then the composition of M with the environment satisfies ϕ. The method enables the creation of a proof schema which is based on the structure of the system. [16] 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 given by the user. The proof of $\psi M \phi$ is done automatically by verifying that the composition of the tableau for ψ with M satisfies ϕ. The method requires a preorder $\leqslant$, a composition operator $||$, and a specification language $\mathcal{L}$ which satisfy the following properties:

(1) For every two structures M_1, M_2, if $M_1 \leqslant M_2$, 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, $M_1 \| M_2 \leqslant M_1$.
(3) For every three structures M_1, M_2, M_3, $M_1 \leqslant M_2$ implies $M_1 \| M_3 \leqslant M_2 \| M_3$.

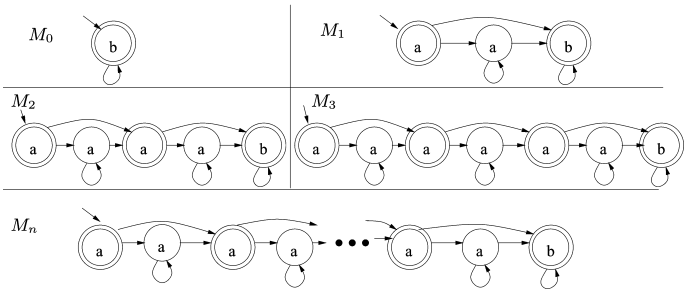

Fig. 4. There is no finite structure M' such that for every n in $\mathbb{N}$, M' is greater by direct/delay simulation than M_n, and $M' \models \mathbf{A}[a\ \mathbf{U}\ b]$.

(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 for ψ with respect to the preorder $\leqslant$.
(5) For every structure M, $M \leqslant M \| M$.

An implementation for this framework was presented in [16]. The implementation uses the *ACTL* logic as the specification language, the exists simulation preorder, and a composition operator which satisfy the properties above. We now show that the game simulation can replace the exists simulation in the framework presented in [16]. As we have stated, the game simulation preserves the ACTL logic, and thus property 1 is satisfied. Lemma 4.6 proves that the game simulation satisfies property 4. Thus, it is left to show that the game simulation preorder and the composition operator as defined in [16] satisfy properties 2, 3 and 5. Lemma 5.2, Lemma 5.3, and Lemma 5.4 prove these properties. The proofs for these lemmas are simple, thus we only give some intuition for them. The full proofs are presented in [4].

In [16], a different type of fairness constraint, the *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 $1 \leqslant i \leqslant n$, $\inf(\rho) \cap f_i \neq \emptyset$. Since the definition of game simulation does not depend on the fairness condition, no changes in its definition are needed. Before we present these lemmas, we define the composition operator $\|$.

Definition 5.1. Let M_1, M_2 be Kripke structures. The parallel composition of M_1 and M_2, denoted $M_1 \| M_2$, is the structure M defined as follows.

- $AP = AP_1 \cup AP_2$.
- $S = \{(s_1, s_2) | L_1(s_1) \cap AP_2 = L_2(s_2) \cap AP_1\}$.[5]
- $R = \{((s_1, s_2), (t_1, t_2)) | (s_1, t_1) \in R_1 \wedge (s_2, t_2) \in R_2\}$.
- $S_0 = (S_{0_1} \times S_{0_2}) \cap S$.
- $L((s_1, s_2)) = L_1(s_1) \cup L_2(s_2)$.
- $F = \{(f_i \times S_2) \cap S | f_i \in F_1\} \cup \{(S_1 \times f_i) \cap S | f_i \in F_2\}$.

Lemma 5.2 (Property 2)**.** *For all Kripke structures* M_1, M_2, $M_1 \| M_2 \leqslant_g M_1$.

It is easy to see that the following strategy is a winning strategy for the protagonist: Whenever the adversary moves to a state (s_1, s_2) in $M_1 \| M_2$, the protagonist moves to s_1 in M_1.

Lemma 5.3 (Property 3)**.** *Let* M_1, M_2, M_3 *be Kripke structures. Then* $M_1 \leqslant_g M_2$ *implies* $M_1 \| M_3 \leqslant_g M_2 \| M_3$.

To see why the lemma is true, suppose that π is a winning strategy for the protagonist in a game over $M_1 \times M_2$, then the following strategy is a winning strategy for the protagonist: Whenever the adversary moves to a state (s'_1, s'_3) in $M_1 \| M_3$, the protagonist moves from (s_2, s_3) to $(\pi(s_2, s'_1), s'_3)$ in $M_2 \| M_3$.

[5] Some of the states might have to be deleted in order to keep R total.

Remark. Note that all notions of simulation presented in this paper include the requirement that for $s \in M$ and $s' \in M'$, if $s \leqslant s'$ then $L(s) = L'(s')$. In many cases, however, we would like to compare structures defined over different sets of atomic propositions. Lemmas 5.2 and 5.3 are examples of such cases. We thus change the definition of simulation so that instead of $L(s) = L'(s')$ we require $L(s) \cap AP' = L'(s') \cap AP$.

Lemma 5.4 (Property 5)**.** *For every structure M, $M \leqslant_g M \| M$.*

It easy to see that the strategy, which instructs the protagonist to move to s whenever the adversary moves to (s, s), is a winning strategy. This completes the proof that game simulation can replace the exists simulation in the assume-guarantee framework presented in [16].

5.1. Complexity

Verifying an assertion of the form $\psi M \varphi$ is PSPACE-complete in the size of ψ [24]. 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 with the game simulation reduces this complexity to polynomial and eliminates the bottleneck of the framework. There is however, one gap that is needed to be closed. The algorithm for game simulation presented in [12] refers to Kripke structures with regular Büchi constraints, while the implementation presented in [16] refers to Kripke structures with generalized Büchi constraints both in the tableau construction and in the composition operator. In order to apply the algorithm suggested in [12] within the assume-guarantee framework, we need a translation between these types of fairness constraints.

Courcoubetis et al. [8] defines a transformation of a Büchi automaton with generalized fairness constraints into a Büchi automaton with regular fairness constraints. Lemma 5.5 implies that it is safe to check for game simulation between the transformed structures that have ordinary Büchi fairness constraints.

Lemma 5.5. *Let M be a fair Kripke structure with generalized Büchi fairness constraints, and let M_r be the result of transforming M into a Kripke structure with ordinary Büchi fairness constraints, using the transformation defined in [8]. Then, $M \equiv_g M_r$.*

A complete description of the transformation and the proof of Lemma 5.5 is presented in [4]. The translation affects 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 result in a structure that is logarithmic bigger than the original one.

6. Conclusion

This work shows that there is no notion of fair simulation which has all the desired advantages. However, it is clear that their relationship with the logics gives the exists and game simulations several advantages over the delay and direct simulations. On the other hand, the delay and direct simulations are better for minimization. Since this research is motivated by usefulness to model checking, relationships with a logic are important. 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 to be the best. This is due to its complexity and its applicability in modular verification.

Appendix

Proof for maximal structures

Proof of Lemma 4.7

The lemma claims: *For every* $n > 0$ *and every structure* M', *if* $M_n \leqslant_{de} M'$ *and* $M' \models \mathbf{A}[a\ \mathbf{U}\ b]$, *then* $|M'| \geqslant n$.

Proof. Let $n \in I\!N$ be a natural number and M' be a structure such that $M' \models \mathbf{A}[a\ \mathbf{U}\ b]$ and $M_n \leqslant_{de} M'$. In a game over $M_n \times M'$ the protagonist has a winning strategy and thus it wins in every game no matter how the adversary plays. Consider the following strategy of the adversary. It starts from the initial state. As long as the protagonist moves to a fair state the adversary moves to the next fair state (until it reaches the last one). If the protagonist moves to a state that is not fair, then the adversary moves to the successor which is not fair in M_n and stays there until the protagonist moves to a fair state in M'. We distinguish between two cases:

(1) The suffix of the game is an infinite sequence of unfair states in both structures. In this case the adversary is the last player who was in a fair state. Thus it wins the game. This means that M' is not greater than M_n by the delay simulation, a contradiction.
(2) Otherwise, the adversary moves through n fair states in M_n that are labelled a to the state labelled b. Since the adversary moves to a fair state only when the protagonist is in a fair state, the protagonist has been in n fair states that are labelled a. Since $M' \models \mathbf{A}[a\ \mathbf{U}\ b]$, these states must be different (otherwise there would be an infinite fair trace which is labelled a). Thus the size of M' is at least n. □

References

[1] A. Aziz, V. Singhal, T.R. Shiple, A.L. Sangiovanni-Vincentelli, F. Balarin, R.K. Brayton, Equivalences for fairkripke structures, in: ICALP, volume 840 of LNCS, 1994, pp. 364–375.

[2] S. Bensalem, A. Bouajjani, C. Loiseaux, J. Sifakis, Property preserving simulations, in: Proceedings of the 4th Conference on Computer Aided Verification, volume 663 of Lecture Notes in Computer Science, Montreal, Springer-Verlag, Berlin, 1992, pp. 260–273.
[3] M.C. Browne, E.M. Clarke, O. Grumberg, Characterizing finite kripke structures in propositional temporal logic, Theoretical Computer Science 59 (1988) 115–131.
[4] D. Bustan, Equivalence-based reductions andchecking for preorders, PhD thesis, Technion, Haifa, Israel, 2002.
[5] D. Bustan, O. Grumberg, Simulation based minimization, Conference on Automated Deduction 17 (2000) 255–270.
[6] D. Bustan, O. Grumberg, Applicability of fair simulation, in: Tools and Algorithms for Construction and Analysis of Systems (TACAS), volume 17 of LNCS, 2002, pp. 401–414.
[7] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, Cambridge, MA, 1999.
[8] C. Courcoubetis, M. Vardi, P. Wolper, M. Yannakakis, Memory efficient algorithms for the verification of temporal properties, in: Proceedings of Computer-Aided Verification, volume 531 of LNCS, 2002, pp. 233–242.
[9] D.L. Dill, A.J. Hu, H. Wong-Toi, Checking for language inclusion using simulation relation, in: Computer-Aided Verification, volume 575 of LNCS, 1991, pp. 255–265.
[10] E.A. Emerson, C.-L. Lei, Efficient model checking in fragments of the propositional μ-calculus, in: Proceedings of the 1st Symposium on Logic in Computer Science, Cambridge, 1986, pp. 267–278.
[11] K. Etessami, G.J. Holzmann, Optimizing Büchi automata, in: Proceedings of the CONCUR 2000, volume 1877 of LNCS, Springer-Verlag, Berlin, 2000, pp. 153–167.
[12] K. Etessami, T. Wilke, R. Schuller, Fair simulation relations, parity games, and state space reduction for Büchi automata, in: Automata, Languages and Programming, 28th International colloquium, volume 2076 of LNCS, 2001, pp. 694–707.
[13] N. Francez, The analysis of cyclic programs, PhD thesis, Weizmann Institute of Science, 1976.
[14] C. Fritz, Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata, in: O.H. Ibarra, Z. Dang (Eds.), Implementation and Application of Automata. Eighth International Conference, volume 2759 of LNCS, Santa Barbara, CA, USA, 2003, pp. 35–48.
[15] C. Fritz, T. Wilke, State space reductions for alternating büchi automata: quotienting by simulation equivalences, in: M. Agrawal, A. Seth (Eds.), Foundations of Software Technology and Theoretical Computer Science: 22nd Conference, volume 2556 of LNCS, Kanpur, India, 2002, pp. 157–168.
[16] O. Grumberg, D.E. Long, Model checking and modular verification, ACM Transactions on Programming Languages and Systems (TOPLAS) 16 (3) (1994) 843–871.
[17] S. Gurumurthy, R. Bloem, F. Somenzi, Fair simulation minimization, in: Proceedings of the 14th International Conference on Computer Aided Verification, Springer-Verlag, Berlin, 2002, pp. 610–624.
[18] M. Hennessy, R. Milner, Algebraic laws for nondeterminism and concurrency, Journal of ACM 32 (1985) 137–161.
[19] T.A. Henzinger, O. Kupferman, S. Rajamani, Fair simulation, in: Proceedings of the eighth Conference on Concurrency Theory, volume 1234 of LNCS, 1997.
[20] C.B. Jones, Specification and design of (parallel) programs, International Federation for Information Processing (IFIP) (1983) 321–332.
[21] Shmuel Katz, Doron Peled, Defining conditional independence using collapses, Theoretical Computer Science 101 (2) (1992) 337–359.
[22] D. Kozen, Results on the propositional μ-calculus, Theoretical Computer Science 27 (1983) 348–359.
[23] O. Kupferman, M.Y. Vardi, Verification of fair transition systems, in: Computer Aided Verification (CAV'96), volume 1102 of LNCS, 1996, pp. 372–382.
[24] O. Kupferman, M.Y. Vardi, Modular model checking, in: Proceedings of the Compositionality Workshop, volume 1536 of LNCS, Springer-Verlag, Berlin, 1998.
[25] O. Kupferman, M.Y. Vardi, An automata-theoretic approach to modular model checking, ACM Transactions on Programming Languages and Systems 22 (2000) 87–128.
[26] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, S. Bensalem, Property preserving abstractions for the verification of concurrent systems, Formal Methods in System Design 6 (1) (1995).
[27] R. Milner, An algebraic definition of simulation between programs, in: Proceedings of the 2nd International Joint Conferences on Artificial Intelligence (IJCAI), London, UK, 1971, pp. 481–489.

[28] J. Misra, K.M. Chandy, Proofs of networks of processes, IEEE Transactions on Software Engineering 7 (1981) 417–426.
[29] A. Pnueli, In transition from global to modular temporal reasoning about programs, in: K.R. Apt (Ed.), Logics and Models of Concurrent Systems, volume 13 of NATO ASI series F, Springer-Verlag, Berlin, 1984.
[30] F. Somenzi, R. Bloem, Efficient Büchi automata from ltl formulae, in: Twelfth Conference on Computer Aided Verification (CAV'00), volume 1633 of LNCS, Springer-Verlag, Berlin, 2000, pp. 247–263.