Computing (bi)simulation relations preserving CTL^*_X for ordinary and fair Kripke structures[1]

P. E. Bulychev , I. V. Konnov , V. A. Zakharov

Abstract. The main goal of model checking is to verify whether a model of a given program satisfies some given specification. In this paper models are regarded as fair or ordinary Kripke structures whereas specifications are represented by formulae branching-time temporal logics (CTL^*_X or $ACTL^*_X$). Model checking can be substantially enhanced by reducing the size of models under consideration. Usually this is achieved by using binary relations on the set of Kripke structures that preserve the satisfiability of temporal formulae. It is known that stuttering bisimulation (simulation) relation preserves CTL^*_X (respectively $ACTL^*_X$) for ordinary Kripke structures. In this paper we present a fair game stuttering bisimulation(simulation) relation which preserves CTL^*_X ($ACTL^*_X$) for fair Kripke structures, and algorithms for computing stuttering (bi)simulation which utilize usual and parity games. If n is the number of states and m the number of transitions in a finite states transition system then our algorithms for computing stuttering simulation and bisimulation for ordinary Kripke structures is proved to have $O(m^2)$ time and space complexity, and our algorithms for computing the same relations for fair Kripke structures appear to have $O(m^2n^2)$ time and $O(m^2)$ space complexity. Thus the verification of CTL^*_X-formulae on a model M (ordinary or fair) can be reduced to the verification of these formulae on a smaller model.

1. Introduction

The main goal of model checking ([6]) is to verify whether the model of the program satisfies some specification. The model of the program M can be presented in the form of the ordinary (nonfair) Kripke structure, and specification ϕ can be given in the form of the formula of the branching time logic CTL^* (as well as its restricted subsets - $ACTL^*$, CTL^*_X, ...). The Kripke structures with fairness constraints are used in some special cases, for example, for modeling asynchronous parallel

[1]This paper is supported by the grants RFBR 06-01-00106 and INTAS 05-1000008-8144.

composition of programs.

In model checking, usual bisimulation(simulation) relations over the set of Kripke structures preserving CTL^*($ACTL^*$) logic are used. These relations make it possible to check some property expressed in CTL^*($ACTL^*$) logic in the abstracted (small) model M_2 rather than in the original (large) model M_1.

Sometimes, however, the weaker relations are to be considered. For instance, logic CTL^*_X (CTL^* without the neXttime operator) is sufficient for abstraction method ([6]). Stuttering bisimulation preserves this logic for ordinary Kripke structures ([1], [3]).

In this paper the game approach is used to compute stuttering (bi)simulation. Let n be the number of states and let m be the number of transitions. We give algorithms for computing ordinary(nonfair) stuttering (bi) simulation over Kripke structures with $O(m^2)$ time and space complexity. To the extent of our knowledge no effective algorithm for computing stuttering simulation have been found till now whereas the best discovered algorithm for computing stuttering bisimulation has $O(mn)$ time and $O(m)$ space complexity([8]). Computing simulation is harder than computing bisimulation ([10]) and so the suggested algorithm for computing stuttering simulation relation over ordinary Kripke structures can be considered as sufficiently effective.

We define the fair game stuttering bisimulation (simulation) relation which preserves CTL^*_X ($ACTL^*_X$) logic for fair Kripke structures. Algorithms for computing these types of fair (bi)simulation are introduced which are based on [2] and in which parity games are used in Buchi automata reduction. Algorithms for computing fair game stuttering simulation and bisimulation have $O(m^2n^2)$ time and $O(m^2)$ space complexity.

Thus the verification of the formula expressed in CTL^*_X logic on the model M (ordinary or fair) can be reduced to the verification of this formula on a smaller model.

2. Preliminaries

Definition 1 (Fair Kripke structure)**.** *Let AP be a set of atomic propositions. A fair Kripke structure M over AP is a 5-tuple $M = (S, R, S_0, L, F)$, where*

- *S - nonempty set of states,*
- *$R \subseteq S \times S$ - total transition relation (a transition relation R is total if for every state $s \in S$ there exists a state $s' \in S$ such that $(s, s') \in R$),*

- $S_0 \subseteq S$ - *nonempty set of initial states,*
- $L : S \to 2^{AP}$ - *a function that labels each state with the set of atomic propositions,*
- $F \subseteq S$ - *nonempty set of fair states.*

Definition 2 (Ordinary Kripke structure)**.** *Ordinary (nonfair) Kripke structure is a Kripke structure $M = (S, R, S_0, L, F)$, where $S = F$. I shall denote this structure $M = (S, R, S_0, L)$, for short.*

A branching time temporal logic CTL^* ([6]) is widely used for expressing specifications. There are two trace quantifiers A ($\forall$) and E ($\exists$), and four temporal operators X (neXt), U (Until), F (Follow), G (Global) in CTL^* . The syntax and the semantics of this logic is defined in many textbooks; for the sake of space the corresponding definitions are skipped in this paper.

The logic $ACTL^*$ is the restricted subset of CTL^* which does not allow E trace quantification, and negation in $ACTL^*$ can be applied only to subformulas which do not contain modalities.

In reasoning about concurrent systems the neXttime operator refers to the global next state rather then to the local next state ([9]), so this operator is useless and can be omitted. The logic CTL^*_X ($ACTL^*_X$) is the restricted subset of CTL^* ($ACTL^*$) without the operator X.

Let $\Im$ be some set of fair Kripke structures. Let us say that a relation $R \subseteq \Im \times \Im$ preserves a logic Λ for $\Im$ iff $M_2 \models \phi$ implies $M_1 \models \phi$ for every $\phi \in \Lambda$ and for every two structures M_1 and M_2 such that $(M_1, M_2) \in R$.

3. Stuttering simulation and bisimulation without fairness constraints

We start with defining (bi)simulation relation for ordinary Kripke structures.

Definition 3 (Divergence-blind stuttering simulation [3], [1])**.** *Given $M_1 = (S_1, R_1, S_{01}, L_1)$ and $M_2 = (S_2, R_2, S_{02}, L_2)$ over AP, a relation $H \subseteq S_1 \times S_2$ is a divergence-blind stuttering simulation (dbs-simulation) relation over (M_1, 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) *if $(s_1, s'_1) \in R_1$ then there exists a sequence $t_0 t_1 \dots t_n$ ($n \geq 0$) such that $t_0 = s_2$ and for all $i < n$, $(t_i, t_{i+1}) \in R_2 \wedge (s_1, t_i) \in H$ and $(s'_1, t_n) \in H$.*

A relation $H \in S_1 \times S_2$ is a dbs-bisimulation relation iff it is a dbs-simulation relation over (M_1, M_2) and H^T is a dbs-simulation relation over (M_2, M_1).

M_2 dbs-simulates M_1 (denoted $M_1 \leq_{dbs} M_2$) iff there exists a dbs-simulation relation $H \in S_1 \times S_2$ over (M_1, M_2). The structures M_2 and M_1 are dbs-bisimilar (denoted $M_1 \approx_{dbs} M_2$) iff there exists a dbs-bisimulation relation $H \in S_1 \times S_2$ over (M_1, M_2).

Definition 4. *The sequence $s_0 s_1 \dots s_n$ such that $L(s_i) = L(s_j)$, $(s_i, s_{i+1}) \in R$ and $s_n = s_0$ is called a cycle of identically labeled states.*

Dbs-bisimulation (simulation) preserves CTL^*_X ($ACTL^*_X$) logic only for structures without cycles of identically labeled states. In the Appendix we consider dbs-(bi)simulation over slightly modified Kripke structures called divergence-sensitive stuttering bisimulation (simulation) which preserves CTL^*_X ($ACTL^*_X$) for Kripke structures with cycles of identically labeled states ([3]).

Let's consider Kripke structures without cycles of identically labeled states. Then the logical properties of stuttering simulation are ([3]):

1. Dbs-bisimulation preserves CTL^*_X logic.
2. Dbs-simulation preserves $ACTL^*_X$ logic.

3.1. Algorithms for computing stuttering simulation and bisimulation

Let $M_1 = (S_1, R_1, S_{01}, L_1)$ and $M_2 = (S_2, R_2, S_{02}, L_2)$ be two ordinary Kripke structures, and $m = |R_1| + |R_2|$, $n = |S_1| + |S_2|$. We will focus on the following questions:

- Does there exist a stuttering simulation between M_1 and M_2? To the best of my knowledge this problem was not considered in the literature yet. In this work we present a simulation checking algorithm whose time and space complexity is $O(m^2)$.
- Does there exist a stuttering bisimulation between M_1 and M_2? The best

algorithm requires $O(mn)$ time and $O(m)$ space([8]). We adduce $O(m^2)$ algorithm.

The analogous problems for usual (bi)simulation were discussed in the literature: [2], [7], [12], [11].

Computing of usual (bi)simulation can be reduced to finding a winning strategy in the game of two players. For instance, an effective algorithm ($O(mn)$) for computing usual simulation by means of this method is given in [2]. Following this approach we reduce the computation of a stuttering simulation to the search of a winning strategy in the game of two players.

There are two players in the game - the player D (duplicator) who tries to show that the chosen relation (simulation or bisimulation) is fulfilled, and the player S (spoiler) who tries to stuck the player D.

Let's describe the principles of the game for finding the stuttering simulation. Let $M_1 = (S_1, R_1, S_{01}, L), S_{01} = \{s_{01}\}$ and $M_2 = (S_2, R_2, S_{02}, L_2), S_{02} = \{s_{02}\}$, $L(s_{01}) = L(s_{02})$. Each player has a pebble, the player S moves his pebble on the graph (S_1, R_1) and the player D moves his pebble on the graph (S_2, R_2) correspondingly. Initially the pebbles of the players S and D are placed on the states s_{01} and s_{02} respectively and the turn belongs to the player S. Then the game runs in the following way: if the turn belongs to the player S and his pebble is placed on s_1, then he chooses the state s'_1 such that $(s_1, s'_1) \in R_1$, reports this state to the player D and passes the turn to the player D. If the turn belongs to D, the pebbles of the players are placed on states s_1 and s_2, and S reported s'_1 at the previous turn, then D can perform one of the actions:

- If $L(s'_1) = L(s_2)$ then he can move the pebble of the player S to s'_1,
- if there exists s'_2 such that $(s_2, s'_2) \in R_2$ and $L(s'_2) = L(s_1)$, then he can move his pebble to s'_2,
- if there exists s'_2 such that $(s_2, s'_2) \in R_2$ and $L(s'_2) = L(s'_1)$, then he can move his pebble to s'_2 and move the pebble of the player S to s'_1.

Then the turn passes to S.

If one of the players can't make a move during the run of the game then another player wins. If the run of the game is infinite then D wins. It will be shown that $M_1 \leq_{dbs} M_2$ iff there exists a winning strategy for player D.

Formally, a game is a triple $G = (V_D, V_S, E)$ where V_D and V_S are the sets of the game states in which the turn belongs to D and S respectively, and $E \subseteq (V_D \cup$

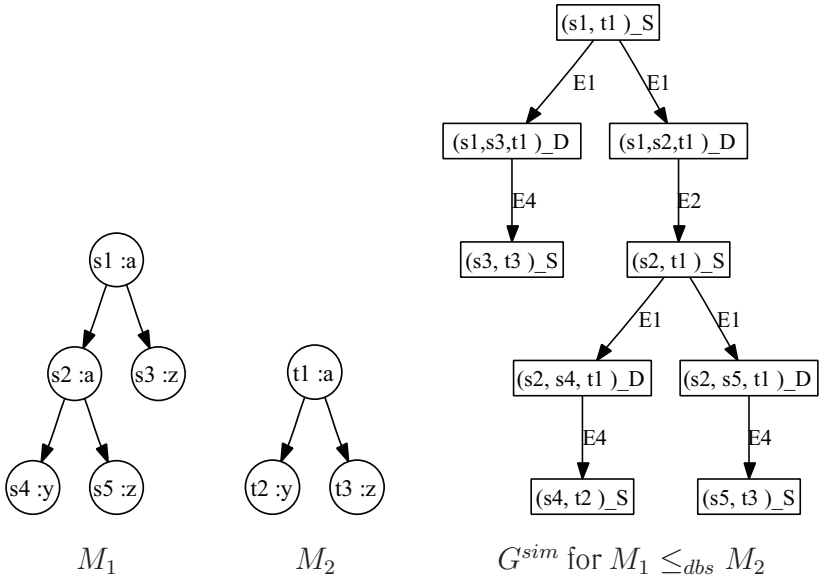
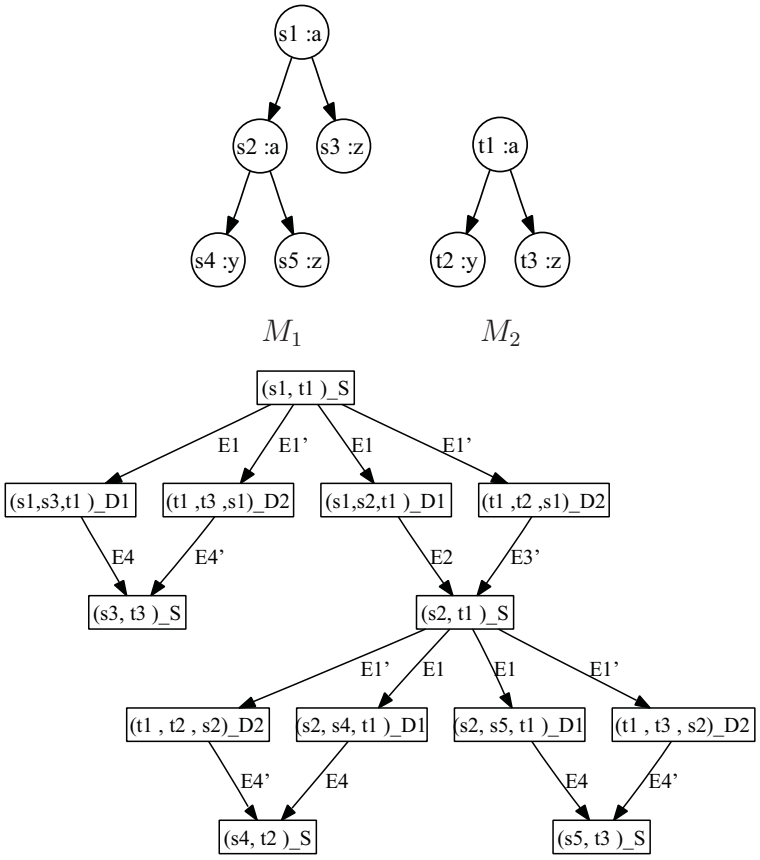


Figure 1: G^{sim}

$V_S) \times (V_D \cup V_S)$ is a set of permitted moves. A run of the game $G = (V_D, V_S, E)$ is a sequence $v_0 v_1 v_2 \ldots$ (finite or infinite) of states such that $(v_i, v_{i+1}) \in E$. A strategy of the player D (player S) is a function $W : V_D \to V_S \cup \{halt\}$ ($W : V_S \to V_D \cup \{halt\}$). A run of the game defined by the strategy W of the player D (player S) is a run $v_0 v_1 \ldots v_{n-1} v_n \ldots$ (finite or infinite) such that:

- $v_i \in V_D \Longrightarrow v_{i+1} = W(v_i)$
  ($v_i \in V_S \Longrightarrow v_{i+1} = W(v_i)$),
- if the run is finite $v_0 v_1 \ldots v_{n-1} v_n$ and $v_n \in V_D$, then $W(v_n) = halt$
  (if the run is finite $v_0 v_1 \ldots v_{n-1} v_n$ and $v_n \in V_S$, then $W(v_n) = halt$)

A strategy W of the player D is winning with initial state v_0 iff $v_n \in V_S$ for every finite run $v_0 v_1 \ldots v_{n-1} v_n$ defined by W.



G^{bis} for $M_1 \approx_{dbs} M_2$

Figure 2: G^{bis}

3.1.1. Stuttering simulation

Formally, the game graph $G^{sim} = (V_D, V_S, E)$ intended for checking a stuttering simulation is defined as follows:

- $V_S = \{(s_1, s_2)_S | s_1 \in S_1 \wedge s_2 \in S_2 \wedge L(s_1) = L(s_2)\}$,
- $V_D = \{(s_1, s_1', s_2)_D | s_1 \in S_1 \wedge s_1' \in S_1 \wedge s_2 \in S_2 \wedge L(s_1) = L(s_2) \wedge (s_1, s_1') \in R_1\}$,
- $E = E_1 \cup E_2 \cup E_3 \cup E_4$ where:
  - $E_1 = \{((s_1, s_2)_S, (s_1, s_1', s_2)_D) | (s_1, s_1') \in R_1\}$,
  - $E_2 = \{((s_1, s_1', s_2)_D, (s_1', s_2)_S) | L(s_2) = L(s_1')\}$,
  - $E_3 = \{((s_1, s_1', s_2)_D, (s_1, s_2')_S) | (s_2, s_2') \in R_2 \wedge L(s_2') = L(s_1)\}$,
  - $E_4 = \{((s_1, s_1', s_2)_D, (s_1', s_2')_S) | (s_2, s_2') \in R_2 \wedge L(s_2') = L(s_1')\}$

Definition 5 (Game stuttering simulation)**.** *Let $M_1 = (S_1, R_1, S_{01}, L)$ and $M_2 = (S_2, R_2, S_{02}, L_2)$ be two ordinary Kripke structures over AP. We say that M_2 gs-simulates M_1 (denoted $M_1 \leq_{gs} M_2$) iff for every $s_{01} \in S_{01}$ there exists $s_{02} \in S_{02}$ such that $L(s_{01}) = L(s_{02})$ and the player D has a winning strategy in the game G^{sim} with the initial state $(s_{01}, s_{02})_S$.*

Theorem 1. *Let M_1 and M_2 be two ordinary Kripke structures without cycles of identically labeled states. Then $M_1 \leq_{dbs} M_2$ iff $M_1 \leq_{gs} M_2$*

The proof of the theorem 1 is given in the Appendix.

Two Kripke structures M_1 and M_2 over $AP = \{a, y, z\}$ and the game graph for G^{sim} are depicted on the Figure 1. M_1 and M_2 are not total, and only winning subgraph of G^{sim} is presented on this Figure, for simple.

3.1.2. Stuttering bisimulation

The player S can move either of two pebbles, and the player D should move then only the pebble different from the pebble moved by S, in the game for computing stuttering bisimulation.

Formally, the game graph $G^{bis} = (V_D, V_S, E)$ corresponding to stuttering bisimulation between $M_1 = (S_1, R_1, S_{01}, L_1)$ and $M_2 = (S_2, R_2, S_{02}, L_2)$ is defined by:

- $V_S = \{(s_1, s_2)_S | s_1 \in S_1 \wedge s_2 \in S_2 \wedge L(s_1) = L(s_2)\}$,
- $V_D = V_{D1} \cup V_{D2}$ where:
  - $V_{D1} = \{(s_1, s'_1, s_2)_{D1} | s_1 \in S_1 \wedge s'_1 \in S_1 \wedge s_2 \in S_2 \wedge L(s_1) = L(s_2) \wedge (s_1, s'_1) \in R_1\}$
  - $V_{D2} = \{(s_2, s'_2, s_1)_{D2} | s_2 \in S_2 \wedge s'_2 \in S_2 \wedge s_1 \in S_1 \wedge L(s_1) = L(s_2) \wedge (s_2, s'_2) \in R_2\}$
- $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E'_1 \cup E'_2 \cup E'_3 \cup E'_4$ where:
  - $E_1 = \{((s_1, s_2)_S, (s_1, s'_1, s_2)_{D1}) | (s_1, s'_1) \in R_1\}$,
  - $E_2 = \{((s_1, s'_1, s_2)_{D1}, (s'_1, s_2)_S) | L(s_2) = L(s'_1)\}$,
  - $E_3 = \{((s_1, s'_1, s_2)_{D1}, (s_1, s'_2)_S) | (s_2, s'_2) \in R_2 \wedge L(s'_2) = L(s_1)\}$,
  - $E_4 = \{((s_1, s'_1, s_2)_{D1}, (s'_1, s'_2)_S) | (s_2, s'_2) \in R_2 \wedge L(s'_2) = L(s'_1)\}$
  - $E'_1 = \{((s_1, s_2)_S, (s_2, s'_2, s_1)_{D2}) | (s_2, s'_2) \in R_2\}$,
  - $E'_2 = \{((s_2, s'_2, s_1)_{D2}, (s_1, s'_2)_S) | L(s'_2) = L(s_1)\}$,
  - $E'_3 = \{((s_2, s'_2, s_1)_{D2}, (s'_1, s_2)_S) | (s_1, s'_1) \in R_1 \wedge L(s_2) = L(s'_1)\}$,
  - $E'_4 = \{((s_2, s'_2, s_1)_{D2}, (s'_1, s'_2)_S) | (s_1, s'_1) \in R_1 \wedge L(s'_2) = L(s'_1)\}$

Definition 6 (Game stuttering bisimulation)**.** *Let M_1 and M_2 be two Kripke structures over AP. M_1 and M_2 are gs-bisimilar (denoted $M_1 \approx_{gs} M_2$) iff for all $s_{01} \in S_{01}$ there exists $s_{02} \in S_{02}$ and for all $s'_{02} \in S_{02}$ there exists $s'_{01} \in S_{01}$ such that $L(s_{01}) = L(s_{02}) \wedge L(s'_{01}) = L(s'_{02})$ and player D has a winning strategy in game G^{bis} with initial states $(s_{01}, s_{02})_S$ and $(s'_{01}, s'_{02})_S$.*

Theorem 2. *Let M_1 and M_2 be two ordinary Kripke structures having T-property. Then $M_1 \approx_{dbs} M_2$ iff $M_1 \approx_{gs} M_2$.*

The game graph for stuttering bisimulation is shown in Figure 2.

3.1.3. Solution of games

Let M_1 and M_2 be two ordinary Kripke structures having a T-property. The problem of computing divergence-blind stuttering simulation (bisimulation) between M_1 and M_2 can be solved using theorems 1 and 2, by finding the winning strategy of the player D in the game G^{sim} (G^{bis}) or by proving that such the strategy does not exist.

	Time	Space
ordinary simulation	$O(mn)$ ([7])	—
ordinary bisimulation	$O(m \log n)$([12])	—
ordinary stuttering simulation	$O(m^2)$(*)	—
ordinary stuttering bisimulation	$O(mn)$ ([8])	$O(m)$ ([8])
fair game simulation	$O(mn^3)$ ([2])	$O(mn)$ ([2])
fair game bisimulation	$O(mn^3)$ ([2])	$O(mn)$ ([2])
fair game stuttering simulation	$O(m^2n^2)$(*)	$O(m^2)$(*)
fair game stuttering bisimulation	$O(m^2n^2)$(*)	$O(m^2)$(*)

(*) - this paper.
Note: m is the number of transitions, n is the number of states, $m > n$.

Figure 3: Best complexity of algorithms for computing simulation relations

Let's consider the game $G(V_D, V_S, E)$ where $m' = |E|$ and $n' = |V_D| + |V_S|$. In [2, 14] it was proved that computing the winning set for the player D can be performed in time and space $O(m' + n')$. As it can be seen from the description of the games G^{sim} and G^{bis} we have $m' = O(m^2)$ and $n' = O(mn)$ where n is the number of states and m is the number of transitions in Kripke structures M_1 and M_2 to be analyzed. Thus, stuttering simulation and stuttering bisimulation can be checked in time and space $O(m^2)$.

The same approach to the problem of checking stuttering (bi)simulation for fair Kripke structures. The details can be found in the Appendix.

4. Conclusions

In this paper we developed a uniform approach to the problem of checking stuttering simulation (bisimulation) for fair and ordinary Kripke structures. It was demonstrated that these problems can be reduced to that of finding winning strategies for the corresponding parity games. Based on this approach a number of new efficient simulation and bisimulation checking algorithms are introduced. These algorithms can be easily adapted to the model checking techniques in the framework of the abstraction method.

Review of complexity of known algorithms for computing simulation relations is given in Figure 3.

References

[1] M.C. Browne, E.M. Clarke, O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. Theoretical Computer Science, vol. 59 , Issue 1-2, pp. 115 - 131, 1988. 60, 61, 71

[2] K. Etessami, T. Wilke, R.A. Schuller. Fair Simulation Relations, Parity Games, and State Space Reduction for Buchi Automata. SIAM Journal on Computing, vol. 34, No. 5, pp. 1159-1175, 2001. 60, 63, 68, 72, 73, 75

[3] R. De Nicola , F. Vaandrager. Three Logics for Branching Bisimulation. Journal of ACM, vol. 42, No. 2, pp. 458-487, 1995. 60, 61, 62, 71

[4] T.A. Henzinger, O. Kupferman, S.K. Rajamani. Fair Simulation. Proc. 8th Conferance on Concurrency Theory, 1997. 72, 73

[5] D. Bustan, O. Grumberg. Applicability of fair simulation. Information and Computation, vol. 194, Issue 1, pp. 1 - 18, 2004. 72

[6] E.M. Clarke, O. Grumberg and D.A. Peled. Model Checking. MIT Press, 1999. 59, 60, 61

[7] M. Henzinger, T. Henzinger, P.Kopke. Computing simulations on finite and infinite graphs. In: Proc. of 36th IEEE Symp. on Foundations of Comp. Sci., pp. 453-462, 1995. 63, 68

[8] J.F. Groote, F. Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. Proceedings of the 17th International Colloquium on Automata, Languages and Programming, pp.626-638, 1990. 60, 63, 68

[9] L. Lamport. What good is temporal logic? Proceedings of IFIP 83, pp. 657-668, 1983. 61

[10] A. Kucera, R. Mayr. Why Is Simulation Harder than Bisimulation? Lecture Notes in Computer Science, vol. 2421, pp. 594 - 609, 2002. 60

[11] R. Paige, R.E. Tarjan. Three partition refinement algorithms. SIAM Journal on Computing, vol. 16 , Issue 6, 1987. 63

[12] J.-C. Fernandez. An Implementation of an Efficient Algorithm for Bisimulation Equivalence. Science of Computer Programming, 1989. 63, 68

[13] M. Jurdzinski. Small Progress Measures for Solving Parity Games. In: Proceedings of 17th Annual Symposium on Theoretical Aspects of Computer Science, 2000. 73, 75

[14] H.R. Andersen. Model checking and boolean graphs. Theoretical Computer Science archive vol. 126 , Issue 1, 1994. 68

5. Appendix

5.1. Divergence-sensitive stuttering (bi)simulation over ordinary Kripke structures

Let's introduce auxiliary designations.

Let p' be the special atomic proposition. Let us say that Kripke structure $M = (S, R, S_0, L, F)$ has a T-property if there exists no more than one state s' such that $p' \in L(s')$, and if such the state exists then $(s', s') \in R$ and there are no outgoing edges from s' except the edge (s', s'), and there are no more cycles of identically labeled states in M except (possibly) this self-loop. It can be easily seen that if there are no cycles of identically labeled states in M then M has a T-property. Let us call (s', s') a T-cycle.

Theorem 3. *Let M_1 and M_2 be two ordinary Kripke structures having T-property. Then:*

- $M_1 \approx_{dbs} M_2$ *iff* $M_1 \approx_{gs} M_2$.
- $M_1 \leq_{dbs} M_2$ *iff* $M_1 \leq_{gs} M_2$

Let $M = (S, R, S_0, L, F)$ be a ordinary Kripke structure over AP, $p' \notin AP$, and $s' \notin S$ be a special state. Let introduce the equivalence relation over S in the

following way: $s_1 \sim s_2$ iff $s_1 = s_2$ or there exists a cycle of identically labeled states containing s_1 and s_2. Let $[S]$ be a set of equivalence classes over $\sim$ on S. Let $C = \{[s] \in [S] |$ there is a cycle of identically labeled states in M such that all of it's state belong to $[s]\}$. We consider a Kripke structure $[M] = ([S] \cup \{s'\}, [R], [S_0], [L], [F])$ over $AP \cup \{p'\}$, where:

- $[R] = [R]_1 \cup [R]_2 \cup [R]_3$, where
  - $[R]_1 = \{([s_1], [s_2]) | [s_1] \in [S] \wedge [s_2] \in [S] \wedge [s_1] \neq [s_2] \wedge \exists s_1 \in [s_1], s_2 \in [s_2] \cdot (s_1, s_2) \in R\}$,
  - $[R]_2 = \{(s', s')\}$,
  - $[R]_3 = \{([s], s') | [s] \in [S] \wedge [s] \cap F \neq \emptyset \wedge [s] \in C\}$
- $[S_0] = \{[s_0] | [s_0] \in [S] \wedge [s_0] \cap S_0 \neq \emptyset\}$
- $[L]([s]) = L(s)$ for $[s] \in [S] \wedge s \in [s]$ (recall that all states of $[s]$ are identically labeled).
  $[L](s') = \{p'\}$
- $[F] = \{[s] | [s] \in [S] \wedge [s] \cap F \neq \emptyset\} \cup \{s'\}$

It can be noted that $[M]$ has a T-property.

Definition 7 (divergence-sensitive stuttering (bi)simulation [3], [1]). *Let M_1 and M_2 be two ordinary(nonfair) Kripke structures over AP. Then*

- *M_2 dss-simulates M_1 (denoted $M_1 \leq_{dss} M_2$) iff $[M_1] \leq_{dbs} [M_2]$.*
- *M_2 dss-bisimulates M_1 (denoted $M_1 \approx_{dss} M_2$) iff $[M_1] \approx_{dbs} [M_2]$.*

Dss-(bi)simulation deals with structures with cycles of identically labeled states. The logical properties of dss-(bi)simulation are:

1. Dss-bisimulation preserves CTL^*_X logic.
2. Dss-simulation preserves $ACTL^*_X$ logic.

Computing $[M]$ can be performed in linear time, so computing dss-(bi)simulation using the theorem 3 has the same complexity as dbs-(bi)simulation.

5.2. Fair simulation and bisimulation

We define the fair game stuttering simulation(bisimulation) relation preserving $ACTL^*_X$ (CTL^*_X) logic for fair Kripke structures. Then following an approach which is analogous to that considered in [2] and we reduce the problem of checking fair (bi)simulation to that of computing a winning strategy in the parity game.

There exist several definitions of fair usual (bi)simulation: Direct, Delay, Game, Exists, the review of them is given in [5]. My definition of fair game stuttering simulation is based on the definition of fair game simulation ([4]).

Let $M_1 = (S_1, R_1, S_{01}, L_1, F_1)$ and $M_2 = (S_2, R_2, S_{02}, L_2, F_2)$ be two fair Kripke structures, and let's consider the relation of fair stuttering simulation. Players S and D move their pebbles during the game over graphs (S_1, R_1) and (S_2, R_2) according to the same rules as in ordinary simulation. If the run of the game is finite then the winner is determined according to the ordinary stuttering simulation game rules. Let now the run of the game be infinite. In this case the winner is determined by the following rule: if the player S's pebble appears in fair states for infinitely many times, while the player D's pebble appears not then the player S is consider winning, otherwise the player D is.

Let's describe the game formally. Let P be the run of the game (G^{sim} or G^{bis}) for which only game states $(s_1, s_2)_S \in V_S$ are retained. Considering the sequence of all the pairs $(s_1, s_2)_S$ in P, one can construct two sequences ρ_1 and ρ_2 consisting of states from the sets S_1 and S_2 correspondingly.

Definition 8 (Fair game stuttering simulation). *Given two fair Kripke structures $M_1 = (S_1, R_1, S_{01}, L_1, F_1)$ and $M_2 = (S_2, R_2, S_{02}, L_2, F_2)$ over AP, let's write $M_1 \leq_{fgs} M_2$ iff for every $s_{01} \in S_{01}$ there exists some $s_{02} \in S_{02}$ such that $L(s_{01}) = L(s_{02})$ and also exists the winning strategy W of the player D in the game G^{sim} with initial state $(s_{01}, s_{02})_S$ such that if $\inf(\rho_1) \cap F_1 \neq \emptyset$ is valid for some run of the game defined by W then $\inf(\rho_2) \cap F_2 \neq \emptyset$ is also valid for this run of the game.*

Definition 9 (Fair game stuttering bisimulation). *Let's write $M_1 \approx_{fgs} M_2$ iff for every $s_{01} \in S_{01}$ there exists $s_{02} \in S_{02}$ and for every $s'_{02} \in S_{02}$ there exists $s'_{01} \in S_{01}$ such that $L(s_{01}) = L(s_{02}) \wedge L(s'_{01}) = L(s'_{02})$ and there exists a winning strategy W of the player D in the game G^{bis} with initial states $(s_{01}, s_{02})_S$ and $(s'_{01}, s'_{02})_S$ such that for every run defined by W:*
$\big(\inf(\rho_1) \cap F_1 \neq \emptyset \wedge \inf(\rho_2) \cap F_2 \neq \emptyset\big) \bigvee \big(\inf(\rho_1) \cap F_1 = \emptyset \wedge \inf(\rho_2) \cap F_2 = \emptyset\big)$

The logical properties of fair game stuttering (bi)simulation are:

1. If $[M_1] \leq_{fgs} [M_2]$, $\phi \in ACTL^*_X$ and $M_2 \models \phi$ then $M_1 \models \phi$.
2. If $[M_1] \approx_{fgs} [M_2]$, $\phi \in CTL^*_X$ and $M_2 \models \phi$ then $M_1 \models \phi$.

These properties can be proved analogously to the appropriate theorem in [4] if one takes into account that there are no cycles of identically labeled states in $[M_1]$ and $[M_2]$, except T-cycles.

Let's reduce the fair stuttering (bi)simulation relation to the parity game ([13], [2]). Let $G = (V_D, V_S, E)$ be a usual game. Let d be some natural number. Parity game is a 4-tuple $G_p = (V_D, V_S, E, p)$ where $p : V_D \cup V_S \to \{0, 1, \ldots, d-1\}$ is a function that assigns a priority to each game state. If the run of the game is finite then the winning player is determined according to usual game rules. Assume that the run of the game $\{v_0 v_1 \ldots\}$ is infinite. Let suppose that $\rho = \{m | m = p(v_i) \text{ for infinitely many } i\}$. Then if $min(\rho)$ is odd, then the player S wins, otherwise the player D wins.

It can be noted that usual game is a particular case of parity game.

5.2.1. Fair game stuttering simulation

Let's define the game $G^{sim}_p = (V_D, V_S, E, p)$ for computing fair stuttering simulation between $M_1 = (S_1, R_1, S_{01}, L_1, F_1)$ and $M_2 = (S_2, R_2, S_{02}, L_2, F_2)$. Let's take for a basis the game $G^{sim} = (V_D, V_S, E)$ for computing stuttering simulation. Let's assign, in analogy with [2], a priority to each game state $V_S \cup V_D$ in such the way:

$$p(v) = \begin{cases} 0, \text{ if } v = (s_1, s_2)_S \in V_S \wedge s_2 \in F_2, \\ 1, \text{ if } v = (s_1, s_2)_S \in V_S \wedge s_2 \notin F_2 \wedge s_1 \in F_1, \\ 2, \text{ otherwise.} \end{cases}$$

It can be easily seen that the run of the game G^{sim}_p is winning for the player D iff the corresponding infinite run of the game used in the definition 8 is wining for the player D.

Thus one comes to equivalent definition of fair stuttering simulation:

Definition 10 (Fair fame stuttering simulation)**.** *Let write $M_1 \leq_{fgs} M_2$ iff for every $s_{01} \in S_{01}$ there exists $s_{02} \in S_{02}$ such that $L(s_{01}) = L(s_{02})$ and there exists a winning strategy of the player D in the game G^{sim}_p with initial state $(s_{01}, s_{02})_S$.*

5.2.2. Fair game stuttering bisimulation

The structure of the game state for fair stuttering bisimulation is more complicated than for fair stuttering simulation - it will be necessary to keep information which of two pebbles visited the fair state the last.

For $s_1 \in S_1$, $s_2 \in S_2$ and $b \in \{1, 2\}$ let define

$$new(b, s1, s2) \equiv \begin{cases} 1 \text{ if } s_1 \in F_1, \\ 2 \text{ if } s_1 \notin F_1 \wedge s_2 \in F_2, \\ b \text{ otherwise.} \end{cases}$$

Formally, the game graph $G^{bis}_p = (V_D, V_S, E, p)$ corresponding to fair stuttering bisimulation between $M_1 = (S_1, R_1, S_{01}, L_1, F_1)$ and $M_2 = (S_2, R_2, S_{02}, L_2, F_2)$ is defined by:

- $V_S = \{(s_1, s_2, b)_S | s_1 \in S_1 \wedge s_2 \in S_2 \wedge L(s_1) = L(s_2) \wedge b \in \{1, 2\}\}$,
- $V_D = V_{D1} \cup V_{D2}$ where:
  - $V_{D1} = \{(s_1, s'_1, s_2, b)_{D1} | s_1 \in S_1 \wedge s'_1 \in S_1 \wedge s_2 \in S_2 \wedge L(s_1) = L(s_2) \wedge (s_1, s'_1) \in R_1 \wedge b \in \{1, 2\}\}$
  - $V_{D2} = \{(s_2, s'_2, s_1, b)_{D2} | s_2 \in S_2 \wedge s'_2 \in S_2 \wedge s_1 \in S_1 \wedge L(s_1) = L(s_2) \wedge (s_2, s'_2) \in R_2 \wedge b \in \{1, 2\}\}$
- $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E'_1 \cup E'_2 \cup E'_3 \cup E'_4$ where:
  - $E_1 = \{((s_1, s_2, b)_S, (s_1, s'_1, s_2, b')_{D1}) | (s_1, s'_1) \in R_1 \wedge b' = new(b, s_1, s_2)\}$,
  - $E_2 = \{((s_1, s'_1, s_2, b)_{D1}, (s'_1, s_2, b)_S) | L(s_2) = L(s'_1)\}$,
  - $E_3 = \{((s_1, s'_1, s_2, b)_{D1}, (s_1, s'_2, b)_S) | (s_2, s'_2) \in R_2 \wedge L(s'_2) = L(s_1)\}$,
  - $E_4 = \{((s_1, s'_1, s_2, b)_{D1}, (s'_1, s'_2, b)_S) | (s_2, s'_2) \in R_2 \wedge L(s'_2) = L(s'_1)\}$
  - $E'_1 = \{((s_1, s_2, b)_S, (s_2, s'_2, s_1, b')_{D2}) | (s_2, s'_2) \in R_2 \wedge b' = new(b, s_1, s_2)\}$,
  - $E'_2 = \{((s_2, s'_2, s_1, b)_{D2}, (s_1, s'_2, b)_S) | L(s'_2) = L(s_1)\}$,
  - $E'_3 = \{((s_2, s'_2, s_1, b)_{D2}, (s'_1, s_2, b)_S) | (s_1, s'_1) \in R_1 \wedge L(s_2) = L(s'_1)\}$,
  - $E'_4 = \{((s_2, s'_2, s_1, b)_{D2}, (s'_1, s'_2, b)_S) | (s_1, s'_1) \in R_1 \wedge L(s'_2) = L(s'_1)\}$

$$p(v) = \begin{cases} 0 \text{ if } v = (s_1, s_2, b)_S \in V_S \wedge s_{(3-b)} \in F_{(3-b)}, \\ 1 \text{ if } v = (s_1, s_2, b)_S \in V_S \wedge s_{(3-b)} \notin F_{(3-b)} \wedge s_b \in F_b, \\ 2 \text{ otherwise.} \end{cases}$$

Definition 11 (Fair game stuttering bisimulation)**.** *Let M_1 and M_2 be two Kripke structures over AP. Let say that M_1 and M_2 are fgs-bisimilar (denoted $M_1 \approx_{fgs} M_2$) iff for all $s_{01} \in S_{01}$ there exists $s_{02} \in S_{02}$ and for every $s'_{02} \in S_{02}$ there exists $s'_{01} \in S_{01}$ such that $L(s_{01}) = L(s_{02}) \wedge L(s'_{01}) = L(s'_{02})$ and the player D has a winning strategy in the game G_p^{bis} with initial states $(s_{01}, s_{02}, 1)_S$ and $(s'_{01}, s'_{02}, 1)_S$.*

Definitions 9 and 11 are equivalent.

5.2.3. Solution of parity games

In [2] the modification of Jurdzinski's algorithm([13]) is given which makes it possible to compute the winning strategy of the parity game with $d = 3$ (i.e. the measure takes values 0,1,2) in time $O(m'n_1)$ and space $O(m')$ where $n_1 = |p^{-1}(1)|$, n' is a number of states in the game graph, m' is a number of transitions in the game graph. Thus the fair game (bi)simulation can be computed in time $O(m^2n^2)$ and space $O(m^2)$ by taking into account that $n_1 \leq |V_S|$, $|V_S| = O(n^2)$ and $m' = O(m^2)$.

5.3. The proof of the theorem 3.1

Let $M_1 = (S_1, R_1, S_{01}, L_1)$ and $M_2 = (S_2, R_2, S_{02}, L_2)$ be two ordinary Kripke structures over AP. There are no cycles of identically labeled states in M_1 and M_2.

1. If $M_1 \leq_{gs} M_2$ then $M_1 \leq_{dbs} M_2$

Let W_D be a winning strategy for the player D in the game G^{sim} and $V_{S0} \subseteq S_{01} \times S_{02}$ be a set of winning initial states for the strategy W_D such that for every $s_{01} \in S_{01}$ there exists $s_{02} \in S_{02}$ such that $(s_{01}, s_{02}) \in V_{S0}$. The existence of such the set V_{S0} is the corollary of the definition **3.3**. Let define the simulation relation $H \subseteq S_1 \times S_2$ in the following way:

$H = \{(s_1, s_2) |$ the game state $(s_1, s_2)_S$ appears in some game run of G^{sim} defined by W_D with initial state $(s_{01}, s_{02}) \in V_{S0})\}$. It can be easily seen that the first part of the definition **3.1** is valid. The we prove that the second part of this definition is also valid. Let $(s_1, s_2) \in H$. It can be noted that $L_1(s_1) = L_2(s_2)$. Let $(s_1, s'_1) \in R_1$. It is necessary to prove that there exists the sequence $t_0t_1 \dots t_n$ $(n \geq 0)$ such that $t_0 = s_2$, $(s'_1, t_n) \in H$ and, for every $i < n$, $(t_i, t_{i+1}) \in R_2$ and $(s_1, t_i) \in H$.

Let W_S be a strategy of the player S:

$$W_S((s,t)_S) = \begin{cases} (s_1, s'_1, t)_D, \text{ if } s = s_1, \\ halt, \text{ otherwise.} \end{cases}$$

The game state $(s_1, s_2)_S$ is a winning state for the player D. Thus, the player D can always move his pebble at his turn. Consider a game run defined by strategies W_S and W_D starting at the game state $(s_1, s_2)_S$. Assuming that $t_0 = s_2$ we have to consider the following 3 scenarios of runs:

1. $(s_1, t_0)_S \xrightarrow{E_1} (s_1, s'_1, t_0)_D \xrightarrow{E_3} (s_1, t_1)_S \xrightarrow{E_1} (s_1, s'_1, t_1)_D \xrightarrow{E_3} (s_1, t_2)_S \xrightarrow{E_1} \dots$
   (the run is infinite)
2. $(s_1, t_0)_S \xrightarrow{E_1} (s_1, s'_1, t_0)_D \xrightarrow{E_3} (s_1, t_1)_S \xrightarrow{E_1} \dots \xrightarrow{E_1} (s_1, s'_1, t_n)_D \xrightarrow{E_2} (s'_1, t_n)_S, n \geq 0,$
3. $(s_1, t_0)_S \xrightarrow{E_1} (s_1, s'_1, t_0)_D \xrightarrow{E_3} (s_1, t_1)_S \xrightarrow{E_1} \dots \xrightarrow{E_1} (s_1, s'_1, t_{n-1})_D \xrightarrow{E_4} (s'_1, t_n)_S, n \geq 1,$

Actually, the first case can not be put into effect because there are no cycles of identically labeled states in M_2.

It can be easily seen that if the game follows the second or the third scenarios then the requirements of the divergence-blind stuttering simulation definition are satisfied: $(s'_1, t_n) \in H$, and for every $i < n$ we have $(t_i, t_{i+1}) \in R_2$ and $(s_1, t_i) \in H$. Thus, there exists a divergence-blind stuttering simulation $H \subseteq S_1 \times S_2$ between M_1 and M_2. Therefore $M_1 \leq_{dbs} M_2$.

2. If $M_1 \leq_{dbs} M_2$ then $M_1 \leq_{gs} M_2$

Let $H \subseteq S_1 \times S_2$ be a stuttering simulation according to the definition **3.1** and let $(s_1, s_2) \in H$ and $(s_1, s'_1) \in R_1$. Then there exists a sequence $t_0t_1 \dots t_n$ $(n \geq 0)$ such that $t_0 = s_2$, $(s'_1, t_n) \in H$, and $(t_i, t_{i+1}) \in R_2 \wedge (s_1, t_i) \in H$ for every $i < n$. Let define

$$W_D((s_1, s'_1, s_2)_D) = \begin{cases} (s'_1, s_2)_S \text{ if } n = 0, \\ (s'_1, t_1)_S \text{ if } n = 1, \\ (s_1, t_1)_S \text{ if } n > 1. \end{cases}$$

Let's define additionally $W_D((s_1, s'_1, s_2)_D) = halt$ iff $(s_1, s_2) \notin H \vee (s_1, s'_1) \notin R_1$

It can be easily seen that if $(s_1, s_2) \in H$ then strategy W_D is a winning strategy of the player D for the initial state $(s_1, s_2)_S$ in the game G^{sim}. Therefore $M_1 \leq_{gs} M_2$.