Information and Computation 262 (2018) 280–310

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Multi-buffer simulations: Decidability and complexity

Milka Hutagalung [a], Norbert Hundeshagen [a], Dietrich Kuske [b,*], Martin Lange [a,*], Étienne Lozes [c]

[a] University of Kassel, Germany
[b] Technische Universität Ilmenau, Germany
[c] ENS Paris–Saclay, France

ARTICLE INFO

Article history:
Received 24 April 2017
Available online 6 September 2018

Keywords:
Büchi automata
Simulation games
Mazurkiewicz traces

ABSTRACT

Multi-buffer simulation is a refinement of fair simulation between two nondeterministic Büchi automata (NBA). It is characterised by a game in which letters get pushed to and taken from FIFO buffers of bounded or unbounded capacity.
Games with a single buffer approximate the PSPACE-complete language inclusion problem for NBA. With multiple buffers and a fixed mapping of letters to buffers these games approximate the undecidable inclusion problem between Mazurkiewicz trace languages.
We study the decidability and complexity of multi-buffer simulations and obtain the following results: P-completeness for fixed bounded buffers, EXPTIME-completeness in case of a single unbounded buffer and high undecidability (in the analytic hierarchy) with two buffers of which at least one is unbounded. We also consider a variant in which the buffers are kept untouched or flushed and show PSPACE-completeness for the single-buffer case.

1. Introduction

Simulation relations on automata Simulation is a pre-order between labelled transition systems $\mathcal{T}$ and $\mathcal{T}'$ that formalises the idea that "$\mathcal{T}'$ can do everything that $\mathcal{T}$ can" [20]. Such relations have become popular in the area of automata theory because they can be used to under-approximate language inclusion problems for automata on finite or infinite words and trees and to minimise such automata [1,6,9,11]: if an automaton $\mathcal{B}$ can simulate an automaton $\mathcal{A}$, then the language of $\mathcal{A}$ is contained in the language of $\mathcal{B}$. On the other hand, inclusion of languages does not guarantee simulation.

Simulation relations are often computable in polynomial time whereas language inclusion problems are PSPACE-complete for typical (finite, Büchi, parity, etc.) automata on words and EXPTIME-complete for such automata on trees. In this paper we focus on nondeterministic Büchi automata (NBA) [2].

Simulation games To reason about simulation relations, one very often characterises them by the existence of winning strategies in two-player games played on the state spaces of two automata with each player moving a pebble along the transitions of their automaton. The game is strictly turn-based with the beginning player, called Spoiler, moving in the automaton to be simulated, and the responding player, called Duplicator moving in the automaton that should simulate the other. Both players construct runs of their automata piece-wise, and it is Duplicator's burden to make her run "cor-

* Corresponding authors.
E-mail addresses: dietrich.kuske@tu-ilmenau.de (D. Kuske), martin.lange@uni-kassel.de (M. Lange).

https://doi.org/10.1016/j.ic.2018.09.008
0890-5401/

respond" to Spoiler's run in the sense that both are runs over the same ω-word. Moreover, Duplicator's run must be an accepting one whenever Spoiler's is. This is also known as *fair* simulation. Other more refined winning conditions have been considered, mainly for the purpose of automata minimisation, known as *direct* and *delayed* simulation [9]. Here we are only concerned with cases of simulation games with winning condition as in fair simulation, simply because it is the most difficult case in terms of decidability and complexity concerns.

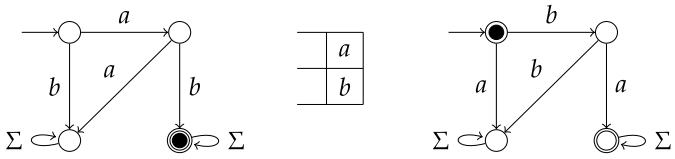
Refined simulation relations The fact that simulation is too weak to capture language inclusion in general has led to the study of extended simulation relations and games. We briefly list them here; a thorough study of the relationships amongst each other is beyond the scope of this paper.

- In *multi-pebble simulation* [8], Duplicator controls several pebbles which allows her to follow several runs of which only one has to be "corresponding" in the above sense. Hence Duplicator can act in foresight of several of Spoiler's later moves.
  Multi-pebble simulation with a fixed number k of pebbles, i.e. k-pebble simulation, is computable in polynomial time for any k [8, Thm. 4].
- *Multi-letter simulation* [4,17], is equally parametrised by some $k \in \mathbb{N}$. In the characterising game, Spoiler first reveals k steps of his run and Duplicator answers with the same number of steps in her run. Equally, k-letter simulation is computable in polynomial time for any fixed k [17, Thms. 4 and 8]. In fact, it is computable in time linear in the size of the underlying automata – unlike multi-pebble simulations –, and genuinely polynomial in the size of the underlying alphabet only.
- The idea underlying the dynamic multi-letter games is extended to form *buffered simulation games* [18]. Here Spoiler chooses one letter in each round, but Duplicator can store this letter in a FIFO buffer (bounded or unbounded) for as long as the buffer's capacity is not exceeded. When she decides to move her pebble, she consumes the first letter(s) from the buffer, thus delaying the construction of her run in comparison to Spoiler's. One can construct pairs of automata such that Duplicator wins the buffered simulation game with an unbounded buffer but not with any bounded buffer [18, Ex. 2.2].
- Simulation games can be seen as special Gale–Stewart games in which input and output words must be equal. As such, one can also consider *delay games* [14] as a form of refined simulation games. Here, one fixes a sequence $(n_i)_{i\geq 0}$ of positive integers and an ω-regular set L of pairs of ω-words as the winning condition. In round i, Spoiler produces n_i letters of his word and Duplicator answers with a single letter of her word. Duplicator wins if the pair of words constructed this way belongs to L.
  What distinguishes delay games from buffered simulation is the predescribed relative speed with which Spoiler advances. As a consequence, one obtains: if Duplicator can win then she can win with a fixed delay [14, Thm. 6.4].

Multi-buffer simulations In this paper, we study a further extension of buffered simulation called *multi-buffer* simulation. The characterising game is played on two NBA with several FIFO buffers – parametrised by their capacities – to store and delay Spoiler's moves.

With several FIFO buffers at hand, one clearly needs to determine which buffer(s) a letter gets stored in when Duplicator wants to delay her corresponding move. One can imagine several possibilities to do so, for instance giving one of the players control over this. Here we follow a different way by fixing, a priori, a mapping σ between letters and sets of buffers such that the letter a always gets put into all the buffers in $\sigma(a)$.

The motivation for having such a fixed mapping comes, again, from formal language theory. It is easy to see that having multiple buffers in a buffered simulation game leads to a relaxed notion of correspondence between the runs constructed by both players. For instance, Duplicator can win the game on the following two automata over letters $\Sigma = \{a, b\}$ that get mapped to two separate buffers.



The picture shows the buffer content after Spoiler has moved his pebble to the state at the bottom right in two rounds while Duplicator has skipped her turns so far; her pebble is still on the initial state. The two letters played by Spoiler have been stored in the two FIFO buffers whose heads are depicted on the right-hand side.

Duplicator can now move her pebble along the b- and then the a-transition, even though Spoiler has chosen these letters in the opposite order. The fact that they get stored in different buffers introduces an independence between the letters regarding the order in which they occur in a run, and this is why multi-buffer simulations approximate language inclusion problems modulo such independence between letters, also known by the name (Mazurkiewicz) *trace inclusion*, known to be (highly) undecidable [10,22].