Buffered Simulation Games for Büchi Automata

Milka Hutagalung, Martin Lange and Etienne Lozes

School of Electr. Eng. and Computer Science, University of Kassel, Germany*

Simulation relations are an important tool in automata theory because they provide efficiently computable approximations to language inclusion. In recent years, extensions of ordinary simulations have been studied, for instance multi-pebble and multi-letter simulations which yield better approximations and are still polynomial-time computable.

In this paper we study the limitations of approximating language inclusion in this way: we introduce a natural extension of multi-letter simulations called buffered simulations. They are based on a simulation game in which the two players share a FIFO buffer of unbounded size. We consider two variants of these buffered games called continuous and look-ahead simulation which differ in how elements can be removed from the FIFO buffer. We show that look-ahead simulation, the simpler one, is already PSPACE-hard, i.e. computationally as hard as language inclusion itself. Continuous simulation is even EXPTIME-hard. We also provide matching upper bounds for solving these games with infinite state spaces.

1   Introduction

Nondeterministic Büchi automata (NBA) are an important formalism for the specification and verification of reactive systems. While they have originally been introduced as an auxiliary device in the quest for a decision procedure for Monadic Second-Order Logic [4] they are by now commonly used in such applications as LTL software model-checking [13, 21], or size-change termination analysis for recursive programs [24, 17]. Typical decision procedures from these domains then reduce to automata-theoretic decision problems like emptiness or inclusion for instance [30].

While emptiness for Büchi automata is NLOGSPACE-complete, deciding inclusion between two nondeterministic finite automata is already more difficult, namely PSPACE-complete [25]. This is also the complexity of inclusion for NBA. Thus, it is – given current knowledge – exponential in the size of the involved NBA regardless of whether it is solved using explicit complementation [27, 29, 23] or other means [1, 16]. One major issue of automata manipulation is therefore to keep the number of states as small as possible.

Since the early works of Dill et al [11], simulations have been intensively used in automata-based verification. Unlike the PSPACE-hard problems like inclusion, simulation between two NBA is cheap to compute. Simulations are interesting with respect to several aspects. On the one hand, they offer a sound, but incomplete, approximation of language inclusion that may be sufficient in many practical cases. On the other hand, simulations can be used for quotienting automata [5, 18, 14], for pruning transitions [1, 2], or for improving existing decision procedures on NBA like the Ramsey-based [17] or the antichain algorithm for inclusion, resp. universality checking [12].

There is a simple game-theoretic characterisation of simulation between two NBA: two players called Spoiler and Duplicator move two pebbles on the transition graph of the NBA, each of them controls one pebble. In order to decide whether or not an NBA $\mathscr{A}$ is simulated by an NBA $\mathscr{B}$, Spoiler starts with his

---

pebble on the initial state of $\mathscr{A}$ and moves it along a transition labeled with some alphabet symbol a. Duplicator starts with her pebble on the initial state of $\mathscr{B}$ and responds with a move along a transition labeled with the same letter. This proceeds ad infinitum. There are different kinds of simulation depending on the winning conditions in these games. For instance, fair simulation models the Büchi acceptance condition and requires Duplicator to have visited infinitely often accepting states if Spoiler has done so. While it is close to the actual condition on inclusion between these two automata, quotienting automata with respect to fair simulation does not preserve the automaton's language.

It is therefore that different winning conditions like delayed simulation have been invented which require Duplicator to eventually visit an accepting states whenever Spoiler has visited one [14]. They, however, do not necessarily provide better approximations to language inclusion. Extensions of the plain simulation relation have been considered since, in particular multi-pebble [15] and multi-letter simulations [9, 22]. Both try to alleviate the gap between simulation and language inclusion which shows up in the game-theoretic characterisation as Spoiler being too strong: language inclusion would correspond to a game in which player chooses an entire run in $\mathscr{A}$ and then Duplicator produces one in $\mathscr{B}$ on the same word. In the simulation game, Spoiler reveals his run step-wise and can therefore dupe Duplicator into positions from which she cannot win anymore even though language inclusion holds.

The two extensions – multi-pebble and multi-letter simulation – use different approaches to approximate language inclusion better: multi-pebble simulation add a certain degree of imperfectness to these games by allowing Duplicator to be in several positions at the same time. Multi-letter simulation forces Spoiler to reveal more of his runs and therefore allows Duplicator to delay her choices for a few rounds and therefore benefit from additional information she gained about Spoiler's moves. The complexity of computing these extended simulations has been studied before: both are polynomial for a fixed number of pebbles, respectively a fixed look-ahead in the multi-letter games. However, nothing is known about the complexity of these simulations if the number of pebbles/letters is not fixed.

Contribution. This paper studies a natural extension of multi-letter games to unbounded look-aheads. We introduce a new family of simulation relations for Büchi automata, called buffered simulations. In a buffered simulation, Spoiler and Duplicator move two pebbles along automata transitions, but unlike in standard simulations, Spoiler and Duplicator's moves do not always alternate. Indeed, Duplicator can "skip her turn" and wait to see Spoiler's next moves before responding. Spoiler and Duplicator share a first-in first-out buffer: every time Spoiler moves along an a-labelled transition, he adds an a into the buffer, whereas every time Duplicator makes a step along a b-labelled transition, she removes a b from the buffer. Since Duplicator has more chances to defeat Spoiler than in standard simulations, buffered simulations better approximate language inclusion. They also improve multi-letter simulations, and it is thus a natural question to ask if they are polynomial time decidable and could be used in practice.

We study two notions of buffered simulation games, called continuous and look-ahead simulation games, respectively. Their rules only differ in the way that Duplicator must use the buffer: in look-ahead simulations, Duplicator is forced to flush the buffer, so that she "catches up" with Spoiler every time she decides to make a move. Thus, the buffer is flushed completely with each of Duplicator's moves. In the continuous case, Duplicator can choose to only consume a part of the buffer with every move, and it need not ever be flushed.

We show that these unbounded buffer simulation games – whilst naturally extending the "easy" multi-letter simulations – provide in a sense a limit to the efficient approximability of language inclusion: we show that look-ahead simulations are already PSPACE-hard, i.e. as difficult as language inclusion itself, while continuous simulations are even worse: they are EXPTIME-hard, i.e. presumably even more

difficult than language inclusion.

We also provide matching upper bounds in order to show that these lower bounds are tight, i.e. these simulations problems are not worse than that. In particular, look-ahead simulation is therefore as difficult as language inclusion, and continuous simulation is "only" slightly more difficult. Decidability of these simulations is not obvious. In the finitary cases, it is provided by a rather straight-forward reduction to parity games but games with unbounded buffers would yield parity games of infinite size. Moreover, questions about systems with unbounded FIFO buffers are often undecidable; for instance, linear-time properties of a system of two machines and one buffer are known to be undecidable [6]. Decidability of these simulation relations may therefore be seen as surprising, and it is also not inconceivable that the decidability results for these unbounded FIFO buffer simulations may lead to developments in other areas, for instance reachability in infinite-state systems etc.

Outline.   Section 2 first recalls Büchi automata and ordinary simulation relations. It then introduces continuous simulation as a simulation game extended with an unbounded buffer. Look-ahead simulation is obtained by restricting the use of the buffer in a natural way. Section 3 contains the most important results in these relations: it shows that look-ahead simulation is already as hard as language inclusion whereas continuous simulation is even harder. Section 4 shows that these bounds are tight by introducing a suitable abstraction called quotient game which yields corresponding upper bounds. Finally, Section 5 collects further interesting results on these simulation relations like topological characterisations for instance and concludes with comments on their use in automata minimisation.

2   Extended Simulation Relations

2.1   Background

Nondeterministic Büchi Automata.   A non-deterministic Büchi automaton (NBA) is a tuple $\mathscr{A} = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states with q_0 being a designated starting state, $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $F \subseteq Q$ is a set of accepting states. A state $q \in Q$ is called a *dead end* when there is no $a \in \Sigma$ and $q' \in Q$ such that $(q, a, q') \in \delta$. If $w = a_1 \ldots a_n$, a sequence $q_0 a_1 q_1 \ldots q_n$ is called a w-path from q_0 to q_n if $(q_i, a_{i+1}, q_{i+1}) \in \delta$ for all $i \in \{0, \ldots, n-1\}$. It is an *accepting w-path* if there is some $i \in \{1, \ldots, n\}$ such that $q_i \in F$. We write $q_0 \xrightarrow{w} q_n$ to state that there is a w-path from q_0 to q_n, and $q_0 \xrightarrow{w}\!\!\!\!{}_\spadesuit q_n$ to state that there is an accepting one.

A *run* of $\mathscr{A}$ on a word $w = a_1 a_2 \cdots \in \Sigma^\omega$ is an infinite sequence $\rho = q_0 q_1 \ldots$ such that $(q_i, a_{i+1}, q_{i+1}) \in \delta$ for all $i \geq 0$. The run is *accepting* if there is some $q \in F$ such that $q = q_i$ for infinitely many i. The language of $\mathscr{A}$ is the set $L(\mathscr{A})$ of infinite words for which there exists an accepting run.

Fair Simulation.

Fair simulation [19] is an extension of standard simulation to Büchi automata. The easiest way of defining fair simulation is by means of a game between two players called *Spoiler* and *Duplicator*. Let us fix two NBA $\mathscr{A} = (Q, \Sigma, \delta, q_I, F)$ and $\mathscr{B} = (Q', \Sigma, \delta', q'_I, F')$. Spoiler and Duplicator are each given a pebble that is initially placed on $q_0 := q_I$ for Spoiler and $q'_0 := q'_I$ for Duplicator. Then, on each round $i \geq 1$,

1. Spoiler chooses a letter $a_i \in \Sigma$ and a transition $(q_{i-1}, a_i, q_i) \in \delta$, and moves his pebble to q_i;

2. Duplicator responds by choosing a transition $(q'_{i-1}, a_i, q'_i) \in \delta'$ and moves his pebble to q'_i.

Either the play terminates because one player reaches a dead end, and then the opponent wins the play. Or the game produces two infinite runs $\rho = q_0 a_1 q_1, \ldots$ and $\rho' = q_0' a_1 q_1' \ldots$, in which case Duplicator is declared the winner of the play if ρ is not accepting or ρ' is accepting. Otherwise Spoiler wins this play.

We say that $\mathscr{A}$ is *fairly simulated* by $\mathscr{B}$, written $\mathscr{A} \sqsubseteq^{\mathsf{f}} \mathscr{B}$, if Duplicator has a winning strategy for this game. Clearly, $\mathscr{A} \sqsubseteq^{\mathsf{f}} \mathscr{B}$ implies $L(\mathscr{A}) \subseteq L(\mathscr{B})$, but the converse does not hold in general.

Remark 2.1. Notice that standard simulation, as defined for labelled transition systems, is a special case of fair simulation. Indeed, for a given labelled transition system (Q, Σ, δ), and a given state q, we can define the NBA $\mathscr{A}(q)$ with $q_I := q$ as the initial state, and $F := Q$ as the set of accepting states. Then q' simulates q in the standard sense (without taking care of fairness) if and only if $\mathscr{A}(q) \sqsubseteq^{\mathsf{f}} \mathscr{A}(q')$. We write $q \sqsubseteq q'$ when q' simulates q in the standard sense.

2.2 Continuous Simulation

Continuous simulations are defined by games in which Duplicator is allowed to see in advance some finite but *unbounded* number of Spoiler's moves. This naturally extends recent work on extensions of fair simulation called *multi-letter* or *look-ahead simulations* in which Duplicator is allowed to see a number of Spoiler's moves that is *bounded* by a constant [22, 9].

Let $\mathscr{A} = (Q, \Sigma, \delta, q_I, F)$ and $\mathscr{B} = (Q', \Sigma, \delta', q_I', F')$ be two NBA. In the continuous fair simulation game, Spoiler and Duplicator now share a FIFO buffer β and move two pebbles through the automata's state spaces. The positions of the pebbles form a word w and two runs ρ and ρ', obtained by successively extending sequences ρ_i and ρ_i' in each round i with zero or more states. At the beginning we have $\rho_0 := q_I$ and $\rho_0' := q_I'$, i.e. Spoiler's pebble is on q_I and Duplicator's pebble is on q_I'. Initially, both word and buffer are empty, i.e. we have $w_0 := \varepsilon$ and $\beta_0 := \varepsilon$.

For the m-th round, with $m \geq 1$ suppose that $w_{m-1} = a_1, \ldots, a_{m-1}$, $\rho_{m-1} = q_0, \ldots, q_{m-1}$, $\rho_{m-1}' = q_0', \ldots, q_{m'}'$ and β_{m-1} have been created already. Duplicator's run in $\mathscr{B}$ is shorter than Spoiler's run, i.e. $m' \leq m$. Furthermore, the buffer β contains the suffix $a_{m'+1}, \ldots, a_m$ of w_{m-1} that Duplicator has not mimicked yet. The m-th round then proceeds as follows.

1. Spoiler chooses a letter $a_m \in \Sigma$ and a transition $(q_{m-1}, a_m, q_m) \in \delta$ and moves the pebble to q_m, i.e. we get $w_m := w_{m-1} a_m$ and $\rho_m := \rho_{m-1} q_{m+1}$. The letter a_m is added to the buffer, i.e. $\beta' := \beta, a_m$.

2. Suppose we now have $\beta' = b_1, \ldots, b_k$. Duplicator picks some r with $0 \leq r \leq k$ as well as states $q_{m'}', \ldots, q_{m'+r-1}'$ such that $(q_{m'+i-1}', b_i, q_{m'+i}') \in \delta'$ for all $i = 1, \ldots, r$. Then we get $\rho_m' := \rho_{m-1}', q_{m'+1}', \ldots, q_{m'+r}'$. The letters get flushed from the buffer, i.e. $\beta_i := b_{r+1}, \ldots, b_k$.

   Note that we have $\rho_m' = \rho_{m-1}'$ if Duplicator chooses $r = 0$. In this case we also say that she *skips her turn*.

A play of this game defines a finite or infinite run $\rho = q_0, q_1, \ldots$ for Spoiler (finite if Spoiler reaches a dead end), and a finite or infinite run $\rho' = q_0', q_1', \ldots$ for Duplicator (finite if Duplicator eventually always skips her turn) on the finite or infinite word $w = a_1 a_2 \ldots$.

Duplicator is declared the winner of the play if

- ρ is finite, or

- ρ is infinite (and w is necessarily infinite as well) and

  - ρ is not an accepting run on w, or

  - ρ' is infinite and an accepting run on w.

In all other cases, Spoiler wins the play.

We say that $\mathscr{B}$ *continuously fairly simulates* $\mathscr{A}$, written $\mathscr{A} \sqsubseteq_{\mathsf{co}}^{\mathsf{f}} \mathscr{B}$, if Duplicator has a winning strategy for the continuous fair simulation game on $\mathscr{A}$ and $\mathscr{B}$. We also consider the (unfair) continuous simulation $\sqsubseteq_{\mathsf{co}}$ for pairs of LTS states by considering an LTS with a distinguished state as an NBA where all states are accepting.

Example 2.2. Consider the following two NBA $\mathscr{A}$ (left) and $\mathscr{B}$ (right) over the alphabet $\Sigma = \{a,b,c\}$.
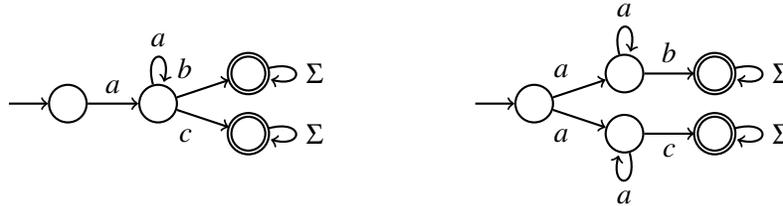
Clearly, we have $L(\mathscr{A}) \subseteq L(\mathscr{B})$.

Duplicator has a winning strategy for the continuous fair simulation game on this pair of automata: she skips her turns until Spoiler follows either the b- or the c-transition. However, if we ignore the accepting states and consider these automata as a transition system, then Spoiler has a winning strategy for the continuous simulation: he iterates the a-loop, and then either Duplicator waits forever and loses the play, or she makes a move and it is then easy for Spoiler to defeat her.

This example also shows that continuous fair simulation strictly extends multi-letter fair simulation which can be seen as the restriction of the former to a bounded buffer. I.e. in these games, Duplicator can only benefit from a fixed look-ahead of at most k letters for some k. It is not hard to see that Spoiler wins the game with a bounded buffer of length k for any k on these two automata: he simply takes k turns on the a-loop in $\mathscr{A}$ which forces Duplicator to choose a transition out of the initial state in $\mathscr{B}$. After doing so, Spoiler can choose the b- or c-transition that is not present for Duplicator anymore and make her get stuck.

2.3 Look-Ahead Simulations

We now consider a variant of the continuous simulation games called *look-ahead* simulation games (the terminology follows [9]). Look-ahead simulation games proceed exactly like the continuous ones, except that now Duplicator has only two possibilities: either she skips her turn, or she flushes the entire buffer. Formally, the definition of the game only differs from the one of Section 2.2 in that the number r of letters removed by Duplicator in a round is either 0 or the size $|\beta|$ of the current buffer β, whereas continuous simulation allowed any $r \in \{0,\ldots,|\beta|\}$.

We write $\mathscr{A} \sqsubseteq_{\mathsf{la}}^{\mathsf{f}} \mathscr{B}$ if Duplicator has a winning strategy for the look-ahead fair simulation on the two automata $\mathscr{A}, \mathscr{B}$. Similarly, we define the look-ahead fair simulation for LTS, $\sqsubseteq_{\mathsf{la}}$.

Example 2.3. Consider again $\mathscr{A}$ and $\mathscr{B}$ as in Example 2.2. It holds that $\mathscr{A} \sqsubseteq_{\mathsf{la}}^{\mathsf{f}} \mathscr{B}$, because Duplicator can flush the buffer once she has seen the first b or c.

Clearly, look-ahead simulation implies continuous simulation but the converse does not hold.

Example 2.4. Consider the following two NBA $\mathscr{A}$ (left) and $\mathscr{B}$ (right) over the alphabet $\Sigma = \{a,b,c\}$.

Duplicator wins the continuous fair simulation: a winning strategy for Duplicator is to skip her first turn, and then to remove one letter at a time during the rest of the play. Thus, after each round, the buffer always contains exactly one element.

On the other hand, Spoiler wins the look-ahead simulation, because the first time Duplicator flushes the buffer, she has committed to a choice between the two right states and thus makes a prediction about the next letter that Spoiler will play.

Remark 2.5. Multi-pebble simulations [15] are another notion of simulation in which duplicator is given more than just one pebble, which she can move, duplicate, and drop during the game. If the number of such pebbles is not bounded, multi-pebble simulations better approximate language inclusion than continuous and look-ahead simulation; in particular, the look-ahead simulation game corresponds to the multi-pebble simulation game in which duplicator is required to drop all but one pebble infinitely often.

3 Lower Bounds: The Complexity of Buffered Simulations

The difficulty of deciding continuous and look-ahead simulation is shown by reduction from suitable tiling problems.

Definition 3.1. A tiling system is a tuple $\mathscr{T} = (T, H, V, t_I, t_F)$, where T is a set of tiles, $H, V \subseteq T \times T$ are the horizontal and vertical compatibility relations, $t_I, t_F \in T$ are the initial and final tiles.

Let n, m be two natural numbers. A tiling with n columns and m rows according to $\mathscr{T}$ is a function $t : \{1, \ldots, n\} \times \{1, \ldots, m\} \to T$; the tiling is valid if (1) $t_{1,1} = t_I$ and $t_{n,m} = t_F$, (2) for all $i = 1, \ldots, n-1$ and all $j = 1, \ldots, m$ we have $(t_{i,j}, t_{i+1,j}) \in H$, (3) for all $i = 1, \ldots, n$, for all $j = 1, \ldots, m-1$ we have $(t_{i,j}, t_{i,j+1}) \in V$.
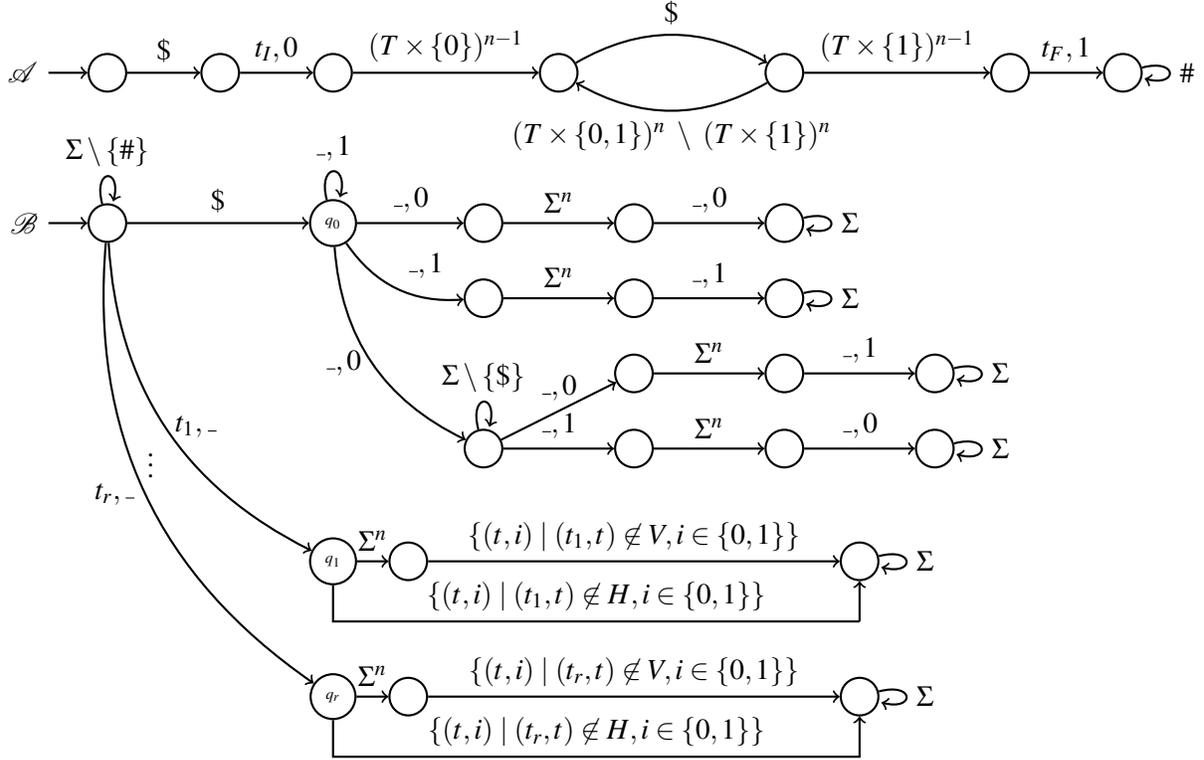
The problem of deciding whether there exists a valid tiling with n columns and 2^n rows, for a given n in unary and a tiling system $\mathscr{T}$, is known to be PSPACE-hard [3].[0] Clearly, the problem to decide whether there is no such tiling is equally PSPACE-hard. We reduce the complement of the tiling problem to look-ahead buffered simulation.

Theorem 3.2. *Deciding $\sqsubseteq_{\mathsf{la}}$ (resp. $\sqsubseteq_{\mathsf{la}}^{\mathsf{f}}$) is PSPACE-hard.*

Proof. Given a tiling system $\mathscr{T} = (T, H, V, t_I, t_F)$ and an $n \in \mathbb{N}$, we consider the alphabet $\Sigma := (T \times \{0,1\}) \cup \{\$, \#\}$. We define the two automata $\mathscr{A}, \mathscr{B}$ as depicted on Figure 1, where all states are accepting. The sizes of $\mathscr{A}$, $\mathscr{B}$ are polynomial in $|T| + n$. Let us consider first the automaton $\mathscr{A}$. A word accepted by $\mathscr{A}$ is composed of blocks of n tiles separated by the $\$$ symbol, such that each block is tagged with the binary representation of a number in $\{0, \ldots, 2^n - 1\}$. We take as a convention that the first bit is the least significant one. Either the word contains finitely many blocks, in which case, the word ends with the symbol $\#$ repeated infinitely often, or it contains infinitely many blocks. Moreover, the first block is tagged with 0, and the last one, if it exists, is tagged with $2^n - 1$ and it is the only one that may be tagged with $2^n - 1$.

Consider now the automaton $\mathscr{B}$. From state q_0, the automaton accepts a word if the two first blocks are not tagged with consecutive numbers. From the state q_i, the automaton accepts a word if either it starts with a tile that is not horizontally compatible with t_i, or if after n symbols it contains a tile that is not vertically compatible with t_i.

---

[0]The requirement on the final tile for instance is not needed for PSPACE-hardness but this variant of the tiling problem is most convenient for the reductions presented here.

Figure 1: Automata $\mathscr{A}$ and $\mathscr{B}$ used in the proof of Theorem 3.2.

The claim is that $\mathscr{A} \sqsubseteq_{\text{la}} \mathscr{B}$ (resp. $\mathscr{A} \sqsubseteq^{\text{f}}_{\text{la}} \mathscr{B}$) if and only if there is no valid $n \times 2^n$ tiling. Assume first that a valid tiling exists. Then Spoiler wins if he plays the word that contains in the i-th block the i-th row of the tiling tagged with the binary representation of i. Note that Duplicator cannot loop forever in the initial state because she cannot read the # symbol. Conversely, assume there is no valid tiling. Then Duplicator wins if she waits until she has seen at most $2^n + 1$ blocks: either two blocks are not tagged with consecutive numbers, or Spoiler played exactly 2^n blocks but these do not code a valid tiling. In the former, Duplicator then accepts by moving to q_0 at the beginning of the first ill-tagged block, and in the later, she wins by moving to q_i after having read a tile t_i whose horizontal or vertical successor does not match.   □

In order to establish an even higher lower bound for the continuous game we consider an EXPTIME-hard game-theoretic variant of the tiling problem on some tiling system $\mathscr{T}$. The game is played by two players: *Starter* and *Completer*. The task for Completer is to produce a valid tiling, whereas Starter's goal is to make it impossible. On every round $i \geq 1$,

1. Starter selects the tile $t_{1,i}$ starting the i-th row; if $i = 1$, then $t_{1,i} = t_I$, otherwise $(t_{1,i-1}, t_{1,i}) \in V$.

2. Completer selects the tiles $t_{2,i}, \dots t_{n,i}$ completing the i-th row; $(t_{1,i}, t_{2,i}), \dots, (t_{n-1,i}, t_{n,i}) \in H$, and $(t_{2,i-1}, t_{2,i}), \dots, (t_{n,i-1}, t_{n,i}) \in V$.

If one of the players gets stuck, the opponent wins. Otherwise Completer wins iff there are i, j such that $t_{i,j} = t_F$. The problem of deciding whether there exists a winning strategy for Starter in this tiling game is known to be EXPTIME-hard [7, 3]. Equally, deciding whether there is no winning strategy for
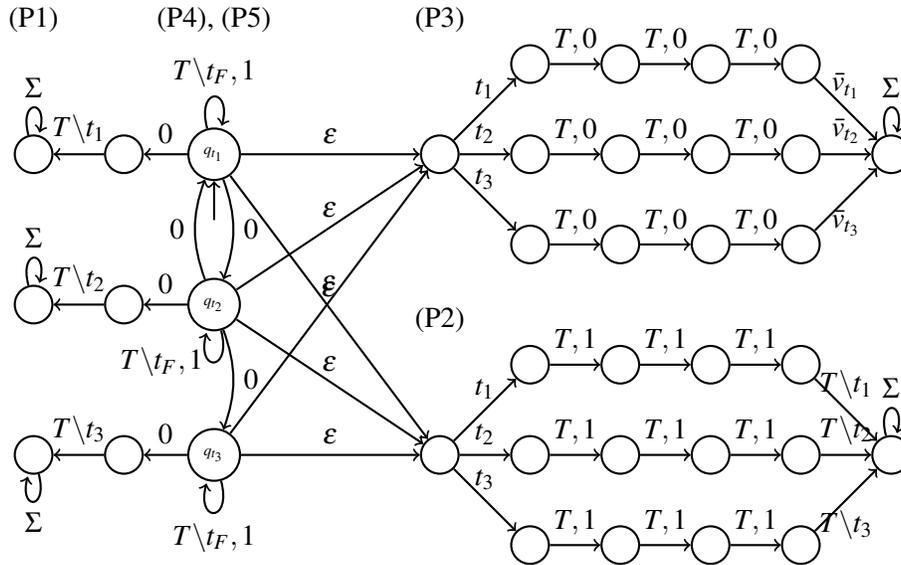
Figure 2: The NBA $\mathscr{B}$ from the construction in the proof of Thm. 3.3 for $m = 3$ and the tiling system $\mathscr{T} = (T, H, V, t_1, t_3)$ where $T = \{t_1, t_2, t_3\}$, $H = \{(t_1, t_1), (t_1, t_3), (t_3, t_3)\}$, and $V = \{(t_1, t_2), (t_2, t_1), (t_2, t_3)\}$. $\bar{v}_t$ denotes the set of tiles that are not vertically compatible with t.

him is EXPTIME-hard and – since the games are easily seen to be determined – so is the problem of deciding whether or not Completer has a winning strategy. This distinction is important because, as in the previous construction, we will reduce the complement of the tiling game problem to continuous buffered simulation. In other words, we present a reduction from one game to another in which the players' roles are inverted. Thus, Starter in the tiling game corresponds to Duplicator in the simulation game, and so do Completer and Spoiler.

Theorem 3.3. *Deciding $\sqsubseteq_{co}$ (resp. $\sqsubseteq_{co}^{f}$) is EXPTIME-hard.*

Proof. Given a tiling system $\mathscr{T} = (T, H, V, t_0, t_F)$, we construct two NBA $\mathscr{A}$, $\mathscr{B}$ of polynomial size, that only contain accepting states, such that there is a winning strategy for Starter in the tiling game if and only if there is a winning strategy for Duplicator in the continuous simulation game ($\mathscr{A} \sqsubseteq_{co} \mathscr{B}$, resp $\mathscr{A} \sqsubseteq_{co}^{f} \mathscr{B}$).

We consider the alphabet $T \uplus \{0, 1\}$. Spoiler's automaton $\mathscr{A}$ is defined such that an infinite word w is accepted by $\mathscr{A}$ if and only if it is of the form $b_0 w_0 b_1 w_1 b_2 w_2 \ldots$, where for all $i \geq 0$, $b_i \in \{0, 1\}$, $w_i \in T^n$, and two consecutive tiles in w_i are in the horizontal relation.

Duplicator's automaton does several things. It forces Spoiler to repeat the previous row when bit 1 occurs, i.e. if Spoiler plays $w_i 1 w_{i+1}$, then $w_i = w_{i+1}$. Duplicator also forces Spoiler to provide a vertically matching row when bit 0 occurs, i.e. if Spoiler plays $w_i 0 w_{i+1}$, then w_i and w_{i+1} must be vertically compatible consecutive rows. However, Duplicator does more: she always forces Spoiler to start the row with a given tile t; this tile is determined by the state q_t in which Duplicator currently is. Informally, the states q_t of Duplicator's automaton $\mathscr{B}$ are such that (1) q_{t_I} is the initial state of $\mathscr{B}$, and (2) if one starts reading from q_t, the following holds:

(P1) for an infinite word starting with $0t' \ldots$, with $t \neq t'$, one can pick an accepting run that does not depend on the infinite suffix;

(P2) for an infinite word starting with $bv1v'\dots$, $b \in \{0,1\}$, $v, v' \in T^n$, and $v \neq v'$, one can pick an accepting run that does not depend on the infinite suffix;

(P3) for an infinite word starting with $bv0v'\dots$, $b \in \{0,1\}$, $v, v' \in T^n$, if there is $i \in \{1,\dots,n\}$ such that the i-th letters of v and v' are not vertically compatible, then one can pick an accepting run that does not depend on the infinite suffix;

(P4) if $v \in T^n$ does not contain t_F, then $q_t \xrightarrow{1v} q_t$;

(P5) if $v \in T^n$ does not contain t_F, then $q_t \xrightarrow{0v} q_{t'}$ for all t' such that $(t,t') \in V$.

We illustrate the construction of $\mathcal{B}$ in Figure 2.

The main component is formed by the states q_{t_i} for $t_i \in T$. Each q_{t_i} is connected to another component that can detect a vertical mismatch (P3) and a non-proper repetition (P2). Each state q_{t_i} is also connected to a component that can detect when Spoiler does not respect Duplicator's choice of the first tile (P1). Each state q_{t_i} has a self-loop by reading $T \setminus t_F$ or 1 (P4) to consume the buffer and form an accepting run if one of Spoiler's mistakes is detected. Moreover, the automaton $\mathcal{B}$ encodes vertical compatibility for Duplicator's choice of the first tile by having edges $(q_{t_i}, 0, q_{t_{i'}}) \in \delta$ if and only if $(t_i, t_{i'}) \in V$ (P5).

We first show that if Completer has a winning strategy in the tiling game, then Spoiler has a winning strategy in the continuous fair simulation game on $\mathcal{A}$ and $\mathcal{B}$. Spoiler plays as follows: first, he moves along $0v_1$, where v_1 is the first row of the tiling. Then he iterates $1v_1$ for a while. This forces Duplicator to eventually remove $0v_1$ from the buffer, and commit to choosing some q_t, due to (P4) and (P5). Spoiler then considers the second row v_2 that Completer would answer if Starter would put t at the beginning of the second row. Spoiler picks this row v_2, and plays $0v_2$, followed by iterations of $1v_2$, and repeats the same principle.

Now we show that if Starter has a winning strategy then Duplicator has a winning strategy. Duplicator first waits for the $2n+2$ first letters of Spoiler. Because of (P1–P3), Spoiler has nothing better to do than to play $0v1v$ for some v encoding a valid first row of a tiling. Duplicator considers the tile t that would be played by Starter in the second row if Completer played v on the first row. Duplicator then removes $0v$ and ends in the state q_t. From there, she waits again for $n+1$ letters, so that the buffer now contains $1vbv'$ for some $b \in \{0,1\}$. Repeating the same process if $b = 1$, she can force Spoiler to eventually play $0v'$ where v' encodes a row vertically compatible with v and starting with t. Iterating this principle results in a play won by Duplicator, since either Completer never uses the final tile or Spoiler's move can always be mimicked by Duplicator due to (P4) and (P5) or, when Completer gets stuck on some row, Spoiler is forced to play a word with a vertical mismatch, and Duplicator wins by accepting the rest of the word. □

One may wonder why the EXPTIME-hardness proof for continuous simulation does not need the machinery of the binary counter as used in the PSPACE-hardness proof for look-ahead simulation. The reason is the following. In the look-ahead game Duplicator always has to flush the buffer entirely. Thus, she has to wait for the entire row-by-row tiling to be produced by Spoiler before she can point out a mistake. Thus, her best strategy is to wait for as long as possible but this would make her lose ultimately. The integrated counter forces Spoiler to get closer and closer to the moment when he has to play the final tile, and Duplicator can therefore relax and wait for that moment before she flushes the entire buffer. In the continuous game, Duplicator's ability to consume parts of the buffer is enough to force Spoiler to not delay the production of a proper tiling forever.

4 Upper Bounds: Quotient Games

We now show that the bounds of the previous section are tight by establishing the decidability of buffered simulations with corresponding complexity bounds. For this, we define a "quotient game" that has a finite state space, and show that it is equivalent to the buffered simulation game.

Continuous Quotient Game. The quotient game is based on the congruence relation associated with the Ramsey-based algorithm for complementation. We briefly recall its definition. Let us fix two Büchi automata $\mathscr{A} = (Q, \Sigma, \delta, q_I, F)$ and $\mathscr{B} = (Q, \Sigma, \delta, q_I', F)$ – for simplicity we assume they share the same state space and only differ in their initial state. We introduce the function $f_w : Q^2 \to \{0, 1, 2\}$ defined as

$$f_w(q, q') = \begin{cases} 0 & \text{if } q \xrightarrow{w}_{\spadesuit} q' \\ 1 & \text{if } q \not\xrightarrow{w} q' \\ 2 & \text{otherwise} \end{cases}$$

We say that two finite words $w, w' \in \Sigma^*$ are equivalent, $w \sim w'$, if $f_w = f_{w'}$. Observe that $\sim$ is an equivalence relation, a congruence for word concatenation, and that the number $|\Sigma^*/\sim|$ of equivalence classes is bounded by $3^{|Q|^2}$. We write $[w]$ to denote the equivalence class of w with respect to $\sim$. We say a class $[w]$ is idempotent if $[ww] = [w]$.

Definition 4.1. The *continuous quotient game* is played between players Refuter and Prover[1] as follows. Initially, Refuter's pebble is on $q_0 := q_I$ and Prover's pebble is on $q_0' := q_I'$. The players use an abstraction by equivalence classes of a buffer that, initially, contains $[\varepsilon]$. On each round $i \geq 1$:

1. Refuter chooses two equivalence classes $[w_1], [w_2]$ and a state q_i, such that $q_{i-1} \xrightarrow{w_1} q_i \xrightarrow{w_2}_{\spadesuit} q_i$ and $[w_2]$ is idempotent

2. Prover chooses q_i' such that $q_{i-1}' \xrightarrow{\beta w_1 w_2} q_i' \xrightarrow{w_2}_{\spadesuit} q_i'$. The value β of the abstract buffer is set to $[w_2]$ for the next turn.

Prover wins the play if Refuter gets stuck or the play is infinitely long.

Proposition 4.2. *Whether Prover has a winning strategy for the continuous quotient game is decidable in EXPTIME.*

Proof. Observe first that the arena of the quotient game is finite and can be computed in exponential time. Indeed, a configuration of a quotient game is either a tuple $(q, q', [w])$ for Refuter's turn or a tuple $(q, q', [b], [w], [w'])$ for Prover's turn. The arena of the quotient game is thus finite and its size is bounded by $2|Q|^2 \cdot |\Sigma^*/\sim|^2 = 2|Q|^2 \cdot 3^{2|Q|^2} = 2^{\mathcal{O}(|Q|^2 \cdot \log |Q|)}$. The finite monoid $\Sigma^*/\sim$ can be computed in exponential time: starting from the set $\{[a] \mid a \in \Sigma\}$, compose any two classes until a fixpoint is reached. Composition of two equivalence classes given as functions of type $Q^2 \to \{0, 1, 2\}$ is not hard to compute [10].

Observe now that the quotient game is a reachability game from Refuter's point of view (he wins if he reaches a configuration in which Prover gets stuck), so once the arena is computed, one can decide the winner of the game in time polynomial in the size of the arena, which is exponential in $|Q|$. □

We show that quotient games characterise the relation $\sqsubseteq_{co}^f$.

---

[1]We use different player names on purpose to make an easy distinction between the original simulation game and the quotient game.

Lemma 4.3. $\mathscr{A} \sqsubseteq_{\mathsf{co}}^{\mathsf{f}} \mathscr{B}$ *only if Prover has a winning strategy for the continuous quotient game.*

Proof. Assume that Refuter has a winning strategy for the continuous quotient game. We want to show that then Spoiler has a winning strategy for the continuous fair simulation game. We actually consider a variant of the continuous fair simulation game in which Spoiler may add more than one letter in a round, and Duplicator only removes one letter in a round. Clearly, Spoiler has a winning strategy for this variant if and only if he has a winning strategy for the continuous fair simulation game as defined in Section 2.2. Spoiler's strategy basically follows the one of Refuter. In the first round, Spoiler adds into the buffer some representatives w_1, w_2 of the equivalence classes played by Refuter. Spoiler then adds w_2 into the buffer on every round until the answer of Duplicator can be identified as a Prover's move in the quotient game, i.e. if Duplicator does not get stuck, she will eventually produce a trace of the form $q_0' \xrightarrow{w_1 w_2^*} q_1' \xrightarrow{w_2^+} q_1'$, since there are only finitely many states in the automaton. Then Spoiler considers the state q_1' in which Duplicator is and looks at what Refuter would play if Prover would have picked q_1'. Iterating this principle, Spoiler mimics Refuter's winning strategy: eventually, since Prover gets stuck on some round i, Duplicator will get stuck when trying to mimic $w_1 w_2^+ \ldots w_i^+$, and then Spoiler wins by continuously adding w_i into the buffer for the rest of the play. $\square$

A key argument in the proof of the converse direction is the following lemma which is easily proved using Ramsey's Theorem [26].

Lemma 4.4. *Let $q_0, q_1, \ldots$ be an infinite accepting run on $a_1 a_2 \ldots$. Then there are i, j, k with $i < j < k$ such that $q_i = q_j = q_k$ is accepting and $a_{i+1} \ldots a_j \sim a_{j+1} \ldots a_k \sim a_{i+1} \ldots a_k$.*

Lemma 4.5. $\mathscr{A} \sqsubseteq_{\mathsf{co}}^{\mathsf{f}} \mathscr{B}$ *if Prover has a winning strategy for the continuous quotient game.*

Proof. When the continuous simulation game starts, Duplicator just skips his turn for a while. Then Spoiler starts providing an infinite accepting run $q_0 a_0 q_1 a_1 \ldots$ – if he does not, Duplicator waits forever and wins the play. At some point, Lemma 4.4 applies: the buffer contains $w_1 w_2 w_2'$ with $[w_2] = [w_2']$ being idempotent, and Spoiler is in a state q that admits a $[w_2]$-loop. Then Duplicator considers the state q' in which Prover would move if Refuter played $[w_1], [w_2], q$ in the first round. She removes $w_1 w_2$ from the buffer and moves to this state q'. Duplicator proceeds identically in the next rounds, and either Spoiler eventually gets stuck or he follows a non-accepting run or the play is infinite. $\square$

Lemmas 4.3 and 4.5 together with Prop. 4.2 yield an upper bound on the complexity of deciding continuous simulation. Together with the lower bound from Theorem 3.3 we get a complete characterisation of the complexity of continuous fair simulation.

Corollary 4.6. *Continuous fair simulation is EXPTIME-complete.*

Look-Ahead Quotient Game.   In order to establish the decidability of look-ahead simulations, we introduce a look-ahead quotient game. The game essentially differs from the continuous quotient game in that it does not use a buffer.

Definition 4.7. The look-ahead quotient game is played between Refuter and Prover. Initially, Refuter's pebble is on $q_0 := q_I$, Prover's pebble is on $q_0' := q_I'$, and the buffer β contains the equivalence class $[\varepsilon]$. On each round $i \geq 1$:

1. Refuter chooses two equivalence classes $[w_1]$, $[w_2]$ and a state q_i, such that $q_{i-1} \xrightarrow{w_1} q_i \xrightarrow{w_2} q_i$ and $[w_2]$ is idempotent.

2. Prover chooses q_i' such that there is a $q_{i-1}' \xrightarrow{w_1} q_i' \xdashrightarrow{w_2} q_i'$.

Prover wins the play if Refuter gets stuck or if the play is infinitely long.

Following the same kind of arguments we used for the continuous quotient game, the result below can be established.

Proposition 4.8. $\mathscr{A} \sqsubseteq_{la}^f \mathscr{B}$ *if and only if Prover has a winning strategy for the look-ahead quotient game.*

The size of the arena of a look-ahead quotient game is again exponential in the size of the automata; but there are only $|Q|^2$ positions for Refuter, so look-ahead quotient games can be solved slightly better than continuous ones.

Proposition 4.9. *Whether Prover has a winning strategy for the look-ahead quotient game can be decided in PSPACE.*

Proof. Consider the following non-deterministic algorithm that guesses the set W of all pairs (q_0, q_0') of initial configurations of the game such that Duplicator has a winning strategy. For all (q_0, q_0') in W, the following can then be checked in polynomial space: for all $[w_1], [w_2]$, and q_1 that could be played by Spoiler, there is q_1' that can be played by Duplicator such that (q_1, q_1') is in W. Inclusion in PSPACE then follows from Savitch's Theorem [28]. □

Corollary 4.10. *Look-ahead fair simulation is PSPACE-complete.*

5 Properties of Buffered Simulations

In this section we investigate some fundamental properties of buffered simulations starting with a comparison to language inclusion. Remember that the main motivation for studying simulations is the approximation thereof.

Continuous Simulation vs. Language Inclusion. Continuous simulation is strictly smaller than language inclusion. It is not hard to see that continuous simulation implies language inclusion, so we focus on strictness.

The following example shows a case where language inclusion holds, indeed $L(\mathscr{A}) = L(\mathscr{B})$, but $\mathscr{A} \not\sqsubseteq_{co}^f \mathscr{B}$ since Spoiler can win the game by always producing a, whereas Duplicator has to keep the pebble on the initial state of $\mathscr{B}$ to be ready for a possible b.

Topological Characterisation. Consider a run of an NBA on some word $w = a_1 a_2 \ldots \in \Sigma^\omega$ to be an infinite sequence $q_0, a_1, q_1, \ldots$ with the usual properties, i.e. the word is actually listed in the run itself. We write $\mathsf{Runs}(\mathscr{A})$ for the set of runs of $\mathscr{A}$ in this respect, and $\mathsf{ARuns}(\mathscr{A})$ for the set of accepting runs.

Given a set Δ, the set Δ^ω is equipped with a standard structure of a metric space. The distance $d(x, y)$ between two infinite sequences $x_0 x_1 x_2 \ldots$ and $y_0 y_1 y_2 \ldots$ is the real $\frac{1}{2^i}$, where i is the first index for which $x_i \neq y_i$. Intuitively, two words are "significantly close" if they share a "significantly long" prefix. The sets $\mathsf{Runs}(\mathscr{A})$ and $\mathsf{ARuns}(\mathscr{A})$ are subsets of $(Q \cup \Sigma)^\omega$; $\mathsf{Runs}(\mathscr{A})$ has the particularity of being a closed subset, and it is thus a compact space, whereas $\mathsf{ARuns}(\mathscr{A})$ is not.

We call a function $f : \mathsf{ARuns}(\mathscr{A}) \to \mathsf{ARuns}(\mathscr{B})$ *word preserving* if for all $\rho \in \mathsf{ARuns}(\mathscr{A})$, $f(\rho)$ and ρ are labelled with the same word. It can be seen that $L(\mathscr{A}) \subseteq L(\mathscr{B})$ holds if and only if there is a word preserving function $f : \mathsf{ARuns}(\mathscr{A}) \to \mathsf{ARuns}(\mathscr{B})$.

Proposition 5.1. *Let $\mathscr{A}, \mathscr{B}$ be two NBA. The following holds: $\mathscr{A} \sqsubseteq^{\mathsf{f}}_{\mathsf{co}} \mathscr{B}$ if and only if there is a continuous word preserving function $f : \mathsf{ARuns}(\mathscr{A}) \to \mathsf{ARuns}(\mathscr{B})$.*

Proposition 5.1 has some interesting consequences. First, it shows again that $\mathscr{A} \sqsubseteq^{\mathsf{f}}_{\mathsf{co}} \mathscr{B}$ implies $L(\mathscr{A}) \subseteq L(\mathscr{B})$, and explain the difference between the two in terms of continuity. Second, it shows that $\sqsubseteq_{\mathsf{co}}$ and $\sqsubseteq^{\mathsf{f}}_{\mathsf{co}}$ are transitive relations, since the composition of two continuous functions is continuous. Another application of Proposition 5.1 is that $\sqsubseteq_{\mathsf{co}}$ (but not $\sqsubseteq^{\mathsf{f}}_{\mathsf{co}}$) is decidable in 2-EXPTIME using a result of Holtmann et al. [20]. This is of course not optimal as seen in the previous section.

Remark 5.2. It might be asked whether look-ahead simulation has a topological characterisation similar to this one. The answer is negative: if it had (a reasonable) one, it would entail that look-ahead simulation is a transitive relation. However, Mayr and Clemente [9] gave examples of automata that show that look-ahead simulation is not transitive in general.

Buffered Simulations in Automata Minimisation.   An important application of simulation relations in automata theory is automata minimisation. A preoder R over the set of states of an automaton $\mathscr{A}$ defines two new automata: its quotient $\mathscr{A}/R$, and its pruning $\mathsf{prune}(\mathscr{A}, R)$, c.f. Clemente's PhD thesis [8] for a formal definition of these notions. Intuitively, the quotient automaton is defined by merging states that are equivalent with respect to the preorder R, whereas pruning is obtained by removing a transition $q \xrightarrow{a} q_1$ if it is "subsumed" by a transition $q \xrightarrow{a} q_2$, where $q_1 \, R \, q_2$.

A preoder R is then said to be good for quotienting (GFQ) if $L(\mathscr{A}/R) = L(\mathscr{A})$, and good for pruning (GFP) if $L(\mathsf{prune}(\mathscr{A}, R)) = L(\mathscr{A})$. It can be checked that GFQ and GFP are antitone properties: if $R \supseteq R'$ and R is GFQ (resp. GFP), then so does R'.

Fair simulation is neither GFQ nor GFP; as a consequence, fair continuous and fair look-ahead simulations, which contain fair simulation, are not GFQ and GFP either. Simulation preorders that are used for automata minimisation rely on less permissive winning conditions than fairness. The *delayed* winning condition asserts that every round in which Spoiler visits an accepting state is (not necessarily immediately) succeeded by some round in which Duplicator also visits an accepting state. The *direct* winning condition imposes that, if Spoiler visits an accepting state in a given round, then in the same round Duplicator should visit an accepting state. Delayed simulation is known to be GFQ but not GFP, whereas direct simulation is known to be both GFP and GFQ. Since a play of a continuous/look-ahead simulation game yields a play of the standard simulation game, there is a natural buffered counterpart of delayed and direct simulation, obtained by changing the winning conditions accordingly.

Proposition 5.3. *Delayed continuous and delayed look-ahead simulation is GFQ but not GFP, and direct continuous as well as direct look-ahead simulation is GFP and GFQ.*

The proof is a rather straightforward consequence of similar results for multi-pebble simulations [15], and from the fact that these multi-pebble simulations subsume continuous simulations (provided the number of pebbles is larger than the number of states of duplicator's automaton).

Recall that bounded buffered simulation relations are polynomial time computable [22] and can be used to significantly improve language inclusion tests for NBA using automata minimisation [9]. We already showed that fair, unbounded, buffered simulation is not polynomial time computable, and thus cannot be used for improving language inclusion tests. We now extend this result to the delayed and direct buffered simulations.

Theorem 5.4. *The delayed (resp. direct) continuous simulation is EXPTIME hard, and the delayed (resp. direct) look-ahead simulation is PSPACE hard.*

This follows from a simple observation: the automata that were used in the hardness proofs had all states accepting, and in this case, fair, delayed and direct simulation coincide.

References

[1] P. A. Abdulla, Y.-F. Chen, L. Clemente, L. Holík, C.-D. Hong, R. Mayr & T. Vojnar (2010): *Simulation Subsumption in Ramsey-Based Büchi Automata Universality and Inclusion Testing*. In: *Proc. 22nd Int. Conf. on Computer-Aided Verification, CAV'10, LNCS* 6174, Springer, pp. 132–147, doi:10.1007/978-3-642-14295-6_14.

[2] P. Aziz Abdulla, Y.-F. Chen, L. Clemente, L. Holík, C.-D. Hong, R. Mayr & T. Vojnar (2011): *Advanced Ramsey-Based Büchi Automata Inclusion Testing*. In: *Proc. 22nd Int. Conf. on Concurrency Theory, CONCUR'11, LNCS* 6901, Springer, pp. 187–202, doi:10.1007/978-3-642-23217-6_13.

[3] Peter Van Emde Boas (1997): *The Convenience of Tilings*. In: *In Complexity, Logic, and Recursion Theory*, Marcel Dekker Inc, pp. 331–363, doi:10.1.1.38.763.

[4] J. R. Büchi (1962): *On a Decision Method in Restricted Second Order Arithmetic*. In: *Proc. Congress on Logic, Method, and Philosophy of Science*, Stanford University Press, Stanford, CA, USA, pp. 1–12, doi:10.1007/978-1-4613-8928-6_23.

[5] D. Bustan & O. Grumberg (2003): *Simulation-based minimization*. ACM Trans. Comput. Logic 4(2), pp. 181–206, doi:10.1145/635499.635502.

[6] G. Cécé & A. Finkel (2005): *Verification of programs with half-duplex communication*. Inf. Comput. 202(2), pp. 166–190, doi:10.1016/j.ic.2005.05.006.

[7] B. S. Chlebus (1986): *Domino-Tiling Games*. Journal of Computer and System Sciences 32, pp. 374–392, doi:10.1016/0022-0000(86)90036-X.

[8] L. Clemente (2012): *Generalized Simulation Relations with Applications in Automata Theory*. Ph.D. thesis, University of Edinburgh.

[9] Lorenzo Clemente & Richard Mayr (2013): *Advanced automata minimization*. In: *Proc. 40th Symp. on Principles of Programming Languages, POPL'13*, ACM, pp. 63–74, doi:10.1145/2429069.2429079.

[10] C. Dax, M. Hofmann & M. Lange (2006): *A proof system for the linear time μ-calculus*. In: *Proc. 26th Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06, LNCS* 4337, Springer, pp. 274–285, doi:10.1007/11944836_26.

[11] D. L. Dill, A. J. Hu & H. Wong-Toi (1991): *Checking for Language Inclusion Using Simulation Preorders*. In: *Proc. 3rd Int. Workshop on Computer-Aided Verification, CAV'91, LNCS* 575, Springer, pp. 255–265, doi:10.1007/3-540-55179-4_25.

[12] L. Doyen & J.-F. Raskin (2009): *Antichains for the Automata-Based Approach to Model-Checking*. Logical Methods in Computer Science 5(1), doi:10.2168/LMCS-5(1:5)2009.

[13] K. Etessami & G. J. Holzmann (2000): *Optimizing Büchi Automata*. In: *Proc. 11th Int. Conf. on Concurrency Theory, CONCUR'00, LNCS* 1877, Springer, pp. 153–167, doi:10.1007/3-540-44618-4_13.

[14] K. Etessami, T. Wilke & R. A. Schuller (2001): *Fair Simulation Relations, Parity Games, and State Space Reduction for Büchi Automata*. In: *Proc. 28th Int. Coll. on Algorithms, Languages and Programming, ICALP'01, LNCS* 2076, Springer, pp. 694–707, doi:10.1137/S0097539703420675.

[15] Kousha Etessami (2002): *A Hierarchy of Polynomial-Time Computable Simulations for Automata*. In Lubo Brim, Mojmr Ketnsk, Antonn Kuera & Petr Janar, editors: *CONCUR 2002 Concurrency Theory, Lecture Notes in Computer Science* 2421, Springer Berlin Heidelberg, pp. 131–144, doi:10.1007/3-540-45694-5_10.

[16] S. Fogarty & M. Y. Vardi (2010): *Efficient Büchi Universality Checking*. In: *Proc. 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'10, LNCS* 6015, Springer, pp. 205–220, doi:10.1007/978-3-642-12002-2_17.

[17] S. Fogarty & M. Y. Vardi (2012): *Büchi Complementation and Size-Change Termination*. *Logical Methods in Computer Science* 8(1), doi:10.2168/LMCS-8(1:13)2012.

[18] S. Gurumurthy, R. Bloem & F. Somenzi (2002): *Fair simulation minimization*. In: *Proc. 14th Int. Conf. on Computer-Aided Verification, CAV'02, LNCS* 2404, Springer, pp. 610–624, doi:10.1007/3-540-45657-0_51.

[19] T. A. Henzinger, O. Kupferman & S. K. Rajamani (2002): *Fair Simulation*. *Inf. Comput.* 173(1), pp. 64–81, doi:10.1006/inco.2001.3085.

[20] M. Holtmann, L. Kaiser & W. Thomas (2012): *Degrees of Lookahead in Regular Infinite Games*. *Logical Methods in Computer Science* 8(3), doi:10.2168/LMCS-8(3:24)2012.

[21] Gerard J. Holzmann (2004): *The SPIN Model Checker - primer and reference manual*. Addison-Wesley.

[22] M. Hutagalung, M. Lange & É. Lozes (2013): *Revealing vs. Concealing: More Simulation Games for Büchi Inclusion*. In: *Proc. 7th Int. Conf. on Language and Automata Theory and Applications, LATA'13, LNCS*, Springer, pp. 347–358, doi:10.1007/978-3-642-37064-9_31.

[23] O. Kupferman & M. Y. Vardi (2001): *Weak Alternating Automata Are Not That Weak*. *ACM Trans. on Comput. Logic* 2(3), pp. 408–429, doi:10.1145/377978.377993.

[24] C. S. Lee, N. D. Jones & A. M. Ben-Amram (2001): *The size-change principle for program termination*. In: *Proc. 28th Symp. on Principles of Programming Languages, POPL'01*, ACM, pp. 81–92, doi:10.1145/360204.360210.

[25] A. R. Meyer & L. J. Stockmeyer (1973): *Word problems requiring exponential time*. In: *Proc. 5th Symp. on Theory of Computing, STOC'73*, ACM, New York, pp. 1–9, doi:10.1145/800125.804029.

[26] F. P. Ramsey (1930): *On a problem in formal logic*. *Proc. London Math. Soc. (3)* 30, pp. 264–286, doi:10.1007/978-0-8176-4842-8_1.

[27] S. Safra (1988): *On the complexity of ω-automata*. In: *Proc. 29th Symp. on Foundations of Computer Science, FOCS'88*, IEEE, pp. 319–327, doi:10.1109/SFCS.1988.21948.

[28] W. J. Savitch (1970): *Relationships between nondeterministic and deterministic tape complexities*. *Journal of Computer and System Sciences* 4, pp. 177–192, doi:10.1016/S0022-0000(70)80006-X.

[29] W. Thomas (1999): *Complementation of Büchi automata revisited*. In J. Karhumäki et al., editor: *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, Springer, pp. 109–122, doi:10.1007/978-3-642-60207-8_10.

[30] M. Y. Vardi (1996): *An Automata-Theoretic Approach to Linear Temporal Logic*, pp. 238–266. *LNCS* 1043, Springer, New York, NY, USA, doi:10.1007/3-540-60915-6_6.