Two-Buffer Simulation Games

Milka Hutagalung[1]   Norbert Hundeshagen[1]   Dietrich Kuske[2]
Martin Lange[1]   Etienne Lozes[3]

[1] School of Electrical Engineering and Computer Science
University of Kassel, Germany
[2] Technische Universität Ilmenau, Germany
[3] LSV, ENS Cachan, France

We consider simulation games played between Spoiler and Duplicator on two Büchi automata in which the choices made by Spoiler can be buffered by Duplicator in two different buffers before she executes them on her structure. Previous work on such games using a single buffer has shown that they are useful to approximate language inclusion problems. We study the decidability and complexity and show that games with two buffers can be used to approximate corresponding problems on finite transducers, i.e. the inclusion problem for rational relations over infinite words.

1 Introduction

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". Formally, it relates the states of the two transition systems such that each state t' in $\mathcal{T}'$ that is connected to some t in $\mathcal{T}$ can mimic the immediate behaviour of t, i.e. it carries the same label, and whenever t has a successor then t' has a matching one, too.

Simulation relations have become popular in the area of automata theory because they can be used to efficiently approximate language inclusion problems for automata on finite or infinite words and trees and to minimise such automata [9, 13, 14, 1]. Simulation relationship is usually 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. Approximation is to be understood in this context as follows: a positive instance of simulation yields a positive instance of language inclusion but not necessarily vice-versa.

Games played between two players – usually called Spoiler and Duplicator– on the state spaces of two automata yield a very intuitive characterisation which can be used to reason about simulations. It is very easy to construct examples of pairs of automata such that language inclusion holds but simulation does not, i.e. the game makes Duplicator too weak. This has led to the study of several extensions of simulation relations and games with the aim of making Duplicator stronger or Spoiler weaker whilst retaining a better complexity than language inclusion. Two examples in this context are *multi-pebble simulation* [12] and *multi-letter simulation* [18, 7].

- In *multi-pebble simulation* [12], Duplicator controls several pebbles and can therefore act in foresight of several of Spoiler's later moves. This kind of simulation is still computable in polynomial time for any fixed number of pebbles. It forms a hierarchy of more and more refined simulation relations that approximate language inclusion.

- In *multi-letter simulation* [18, 7], Spoiler is forced to reveal more than one transition of the run that he constructs in the limit. Again, these relations are computable in polynomial time for any fixed look-ahead, that is the number of symbols that Spoiler advances in the construction of his run ahead of Duplicator's. In fact, all are computable in time linear in the size of each underlying

automaton – unlike multi-pebble simulations –, and polynomial of a higher degree only in the size of the underlying alphabet. They also form a hierarchy between ordinary simulation and language inclusion.

In multi-letter simulations Spoiler is forced to reveal more than one transition in each round. This look-ahead can be realised using a bounded FIFO buffer which is filled by Spoiler and emptied by Duplicator. This has led to another extension of simulation.

- In *buffered simulations* [19], the letters chosen by Spoiler get stored in an unbounded FIFO buffer, and Duplicator consumes them to form her moves. These games show a limit of what is possible in terms of approximating language inclusion: buffered simulation is in general EXPTIME-complete [19], i.e. even harder than language inclusion, yet there are pairs of Büchi automata on which Spoiler wins the buffered simulation game even though language inclusion holds. This is only true for ω-languages, though. On finite words, buffered simulation captures language inclusion because Duplicator can simply wait for Spoiler to produce an entire word before she makes any moves.

  This may raise the question of why buffered simulation is EXPTIME-complete when it captures a PSPACE-complete problem in this case. There is in fact a natural restriction of buffered simulation which is only PSPACE-complete [19]; it requires Duplicator to always flush the entire buffer when she moves. Note that simulation games on automata on finite words can be played with this restriction since they degenerate to games with no proper alternation: first Spoiler produces a word, then Duplicator consumes it entirely from the buffer.

In this paper we consider a natural extension of simulation games played with FIFO buffers, namely *two-buffer* simulations. Again, Spoiler and Duplicator each move a pebble along the transitions of two Büchi automata, forming runs, and Duplicator wins a play if her run is accepting or Spoiler's is not. The letters chosen by Spoiler get put into one of two buffers from which Duplicator takes letters to form her run. Note that two-buffer games do *not* approximate language inclusion on Büchi automata because the availability of more than one buffer introduces a commutativity property between alphabet symbols. Since this commutativity is based on a partition of the alphabet, it matches the commutativity in the direct product of two free monoids. Consequently, as an application, we obtain polynomial-time computable approximation procedures for the otherwise undecidable inclusion problem between rational relations over infinite words.

The paper is organised as follows. Section 2 recalls necessary definitions etc. on Büchi automata, simulation games and buffered simulation games. In Section 3 we introduce and study two-buffer simulations. We show that they are potentially interesting from a point of efficiency: they can be decided in polynomial time for any fixed number of bounded buffers and any fixed buffer capacity. We also examine when undecidability occurs: it is not surprising that using two unbounded buffers leads to undecidability since problems involving two unbounded FIFO buffers are typically undecidable, for instance the reachability problem for a network of finite state machines communicating through unbounded perfect FIFO channels is undecidable for various set-ups [4, 6]. In Section 4 we present our application of two-buffer simulation games which features the aforementioned partial commutativity to the inclusion problem between rational relations over infinite words.

2 Preliminaries

Automata.

A *Büchi automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ where Q is a finite set of states with a designated starting state $q_I \in Q$, Σ is the underlying alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation and $F \subseteq Q$ is a designated set of accepting states.

The language $L(\mathcal{A})$ of a Büchi automaton is, as usual, the set of all infinite words for which there is a run starting in q_I that visits some states in F infinitely often. It is known that single-buffer simulation games, as recalled below, can be used to approximate Büchi language inclusion problems [19]. However, the simulation games introduced in Section 3 lead away from the problem of language inclusion between Büchi automata. Hence, we are not particularly concerned with Büchi automata as acceptors of ω-languages. Instead it suffices at this point to simply see them as directed graphs with labelled edges, a designated starting state and some marked states that will be used to define winning conditions in games played on these graphs. We will therefore rather write $q \xrightarrow{a} p$ than $(q, a, p) \in \delta$ when the underlying transition relation is clear from the context. We will also simply speak of *automata* rather than *Büchi automata* to take away the focus from plain language inclusion problems.

Buffered simulation.

The *buffered simulation* game [19] is played between SPOILER and DUPLICATOR on two automata $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, q_I, \delta^{\mathcal{A}}, F^{\mathcal{B}})$ and $\mathcal{B} = (Q^{\mathcal{B}}, \Sigma, p_I, \delta^{\mathcal{B}}, F^{\mathcal{B}})$ on configurations of the form (q, β, p) with $q \in Q^{\mathcal{A}}$, $p \in Q^{\mathcal{B}}$ and $\beta \in \Sigma^*$. This additional component acts like a buffer through which the alphabet symbols chosen by SPOILER get channelled before DUPLICATOR can use them. Such a buffer has a capacity $k \in \mathbb{N} \cup \{\omega\}$. If $k \in \mathbb{N}$ then the buffer is called *bounded*, for $k = \omega$ it is called *unbounded*.

Any play of that game starts in the configuration (q_I, ε, p_I). Any round that has reached a configuration (q, β, p) proceeds as follows.

1. SPOILER chooses a $q' \in Q^{\mathcal{A}}$ and $a \in \Sigma$ such that $q \xrightarrow{a} q'$. Let $\beta' := a\beta$.

2. DUPLICATOR can either skip her turn in which case the play proceeds in the configuration (q', β', p). Or we have $\beta' = \beta'' b$ for some $b \in \Sigma$. In this case she can choose a $p' \in Q^{\mathcal{B}}$ such that $p \xrightarrow{b} p'$, and the play proceeds with (q', β'', p').

A play $(q_0, \beta_0, p_0), (q_1, \beta_1, p_1), \ldots$ is won by DUPLICATOR iff

- $|\beta_i| \leq k$ for all $i \in \mathbb{N}$, i.e. the buffer never exceeds its capacity, and

- either
    - there are only finitely many i such that $q_i \in F^{\mathcal{A}}$, or
    - DUPLICATOR moves infinitely often, and there are infinitely many i such that $p_i \in F^{\mathcal{B}}$.

Otherwise SPOILER wins. We write $\mathcal{A} \sqsubseteq^k \mathcal{B}$ to state that duplicator has a winning strategy for the buffered simulation game on $\mathcal{A}$ and $\mathcal{B}$ with buffer capacity k.

Note that according to these winning conditions, the buffer capacity is only checked after DUPLICATOR's moves. Hence, it is possible to play on a buffer of capacity $k = 0$; this just means that DUPLICATOR has to consume the alphabet symbol chosen by SPOILER immediately. We also remark that in this definition of buffered simulation game, DUPLICATOR only ever consumes a single alphabet symbol in each of her moves. One could equally allow her to consume several symbols from the buffer. This has no immediate advantage for her; if she has a winning strategy then she also has one in which she only ever

makes moves on one symbol. The reason for this is the simple fact that the buffer gives her a look-ahead on the choices made by Spoiler in an ordinary simulation game, and winning is monotone in the amount of information available about the opponent's future moves.

For two automata $\mathcal{A}$ and $\mathcal{B}$, we write $\mathcal{A} \sqsubseteq \mathcal{B}$ if Duplicator has a winning strategy in the ordinary fair simulation game [13].

Proposition 1 ([18, 19]). *We have*

a) $\sqsubseteq = \sqsubseteq^0$,

b) $\sqsubseteq^k \subsetneq \sqsubseteq^{k+1}$ *for any* $k \in \mathbb{N}$,

c) $\bigcup_{k \geq 0} \sqsubseteq^k \subsetneq \sqsubseteq^\omega$,

d) $\mathcal{A} \sqsubseteq^\omega \mathcal{B}$ *implies* $L(\mathcal{A}) \subseteq L(\mathcal{B})$ *for any Büchi automata* $\mathcal{A}$ *and* $\mathcal{B}$. *The converse does not hold in general.*

3 Two-Buffer Simulations

3.1 Simulation Games Using Two Buffers

We fix two automata $\mathcal{A} = (Q^\mathcal{A}, \Sigma, q_I, \delta^\mathcal{A}, F^\mathcal{B})$ and $\mathcal{B} = (Q^\mathcal{B}, \Sigma, p_I, \delta^\mathcal{B}, F^\mathcal{B})$ and develop the theory of two-buffer simulation games with respect to these two automata.

We assume a mapping $\sigma : \Sigma \to \{1, 2\}$ which assigns a buffer index to each input symbol. Hence, the buffer that a symbol is put into is determined by the symbol itself, not by one of the players.

Definition 2. Let $(k_1, k_2) \in (\mathbb{N} \cup \{\omega\})^2$ be a pair of buffer capacities, i.e. buffers can have bounded or unbounded capacity. The *two-buffer simulation game with capacities* (k_1, k_2), denoted $\mathcal{G}^{k_1,k_2}(\mathcal{A}, \mathcal{B})$, is played between players Spoiler and Duplicator on $\mathcal{A}$ and $\mathcal{B}$ using two FIFO-buffers, of capacities k_1, k_2 and initially empty, and two tokens which are initially placed on q_I and p_I. A configuration is denoted by (q, β_1, β_2, p), consisting of the two states which currently carry the tokens on the left and right and the contents of the two buffers written as finite words over Σ in between. The initial configuration is $(q_I, \varepsilon, \varepsilon, p_I)$. A round consists of a move by player Spoiler followed by a move by Duplicator. In a configuration of the form (q, β_1, β_2, p),

1. Spoiler selects a $q' \in Q^\mathcal{A}$ and an $a \in \Sigma$ such that $q \xrightarrow{a} q'$. Let $i = \sigma(a)$ and update the buffers as follows: $\beta_i := a\beta_i$, i.e. a gets appended to the i-th buffer.

2. Duplicator repeats the following step k times for some k with $0 \leq k \leq |\beta_1| + |\beta_2|$.

   - She selects a $j \in \{1, 2\}$ such that $\beta_j = \gamma b$ for some $b \in \Sigma$, as well as a $p' \in Q^\mathcal{B}$ with $p \xrightarrow{b} p'$. Then she updates the buffers as follows: $\beta_j := \gamma$, i.e. b gets removed from the j-th buffer. She proceeds with $(q', \beta_1, \beta_2, p')$, i.e. with the new state p' instead of p.

A play is a sequence $(q_0, \beta_1^0, \beta_2^0, p_0), (q_1 \beta_1^1, \beta_2^1, p_1), \ldots$ of configurations. It is won by Duplicator iff

- $|\beta_j^i| \leq k_j$ for all $i \in \mathbb{N}$ and $j \in \{1, 2\}$, i.e. no buffer ever exceeds its capacity, and

- either
  - there are only finitely many i such that $q_i \in F^\mathcal{A}$, or
  - every alphabet symbol that gets put into one of the buffers also gets removed from it eventually, and there are infinitely many i such that $p_i \in F^\mathcal{B}$.

Otherwise Spoiler wins.

Definition 3. Given k_1, k_2 as above, we say that $\mathcal{B}$ (k_1, k_2)-*simulates* $\mathcal{A}$, written $\mathcal{A} \sqsubseteq^{k_1, k_2} \mathcal{B}$ if player DUPLICATOR has a winning strategy for the game $\mathcal{G}^{k_1, k_2}(\mathcal{A}, \mathcal{B})$.

Example 4. Consider the following two NBA $\mathcal{A}$ (left) and $\mathcal{B}$ (right) over the alphabet $\Sigma = \{a, b, c, d\}$ with $\sigma(a) = 1$, $\sigma(b) = \sigma(c) = \sigma(d) = 2$.
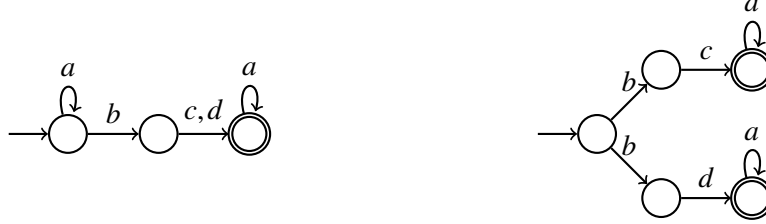
We have $\mathcal{A} \sqsubseteq^{\omega, 1} \mathcal{B}$, since DUPLICATOR has a winning strategy: skipping her turn until SPOILER reads c or d. DUPLICATOR then consumes the entire content of the two buffers and reaches the accepting loop.

We describe a second example in order to explain a possibly apparent difference in the definitions of the single- and the two-buffer games: in single-buffer games we let DUPLICATOR consume at most one symbol per round only because allowing her to consume more than one does not make her stronger. There are two-buffer games, though, which she can only win when she is allowed to consume more than one symbol at a time. In such games at least one buffer must be bounded; if both buffers are unbounded then DUPLICATOR can defer her moves for any finite number of times at any point and can never be forced by SPOILER to consume a letter in order to not exceed a buffer capacity. Now if there are two buffers then in order to consume an element from one that is close to overflow she may have to first consume one from the other buffer, hence consume more than one letter in a round.

Example 5. Consider the following two NBA $\mathcal{A}$ (left) and $\mathcal{B}$ (right) over the alphabet $\Sigma = \{a, b\}$ with $\sigma(a) = 1$, $\sigma(b) = 2$, and the play $\mathcal{G}^{\omega, 1}(\mathcal{A}, \mathcal{B})$.
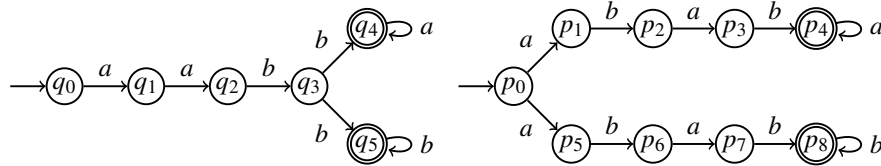

DUPLICATOR's winning strategy for the first three rounds is to store the three symbols chosen by SPOILER resulting in buffer contents (aa, b). Note that then the second buffer is full. Hence, when SPOILER chooses a b now in state q_3, DUPLICATOR must consume it in this round but in her current state p_0 she only has a-transitions available. Hence, she must consume the first a, advance to p_1 or p_5 depending on whether SPOILER has moved to q_4 or q_5. Say it was q_4 so DUPLICATOR has moved to p_1 and the current buffer content is (a, bb). She then needs to move over to p_2 creating the buffer content (a, b) and can end the round. She could also consume the next buffered a and move to p_3 with buffers (ε, b) but this does not help her, neither in this case nor in general.

The hierarchy of buffered simulations stated in Proposition 1 easily carries over to two-buffer simulations. The total order $\leq$ on $\mathbb{N} \cup \{\omega\}$ extends to the usual partial order of point-wise comparisons on pairs from $\mathbb{N} \cup \{\omega\}$. I.e. we have $(78, 6) \leq (\omega, 6)$ but $(3, 4) \not\leq (5, 2)$.

Theorem 6. *For any $k_1, k_2, \ell_1, \ell_2 \in \mathbb{N} \cup \{\omega\}$ with $(k_1, k_2) \leq (\ell_1, \ell_2)$ we have $\sqsubseteq^{k_1, k_2} \subseteq \sqsubseteq^{\ell_1, \ell_2}$. Moreover, if $(\ell_1, \ell_2) \not\leq (k_1, k_2)$ then there are automata $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \not\sqsubseteq^{k_1, k_2} \mathcal{B}$ but $\mathcal{A} \sqsubseteq^{\ell_1, \ell_2} \mathcal{B}$.*

Proof. We immediately get $\sqsubseteq^{k_1,k_2} \subseteq \sqsubseteq^{\ell_1,\ell_2}$ for $(k_1,k_2) \le (\ell_1,\ell_2)$ since any winning strategy for DUPLICATOR in $\mathcal{G}^{k_1,k_2}(\mathcal{A},\mathcal{B})$ is also a winning strategy for her in $\mathcal{G}^{\ell_1,\ell_2}(\mathcal{A},\mathcal{B})$. Note that she is not required to use the additional buffer capacities.

For the strictness part suppose w.l.o.g. that $\ell_1 > k_1$ which implies $k_1 \in \mathbb{N}$. Then consider the two NBA $\mathcal{A}$ (left) and $\mathcal{B}$ (right) over Σ as follows, and let $\Sigma_1 = \{a \in \Sigma \mid \sigma(a) = 1\}$.
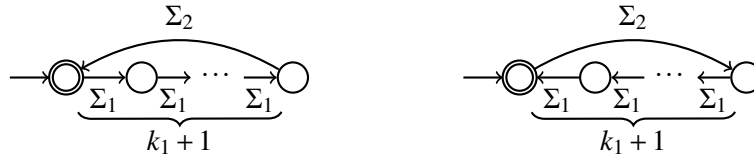
It should be clear that we have $\mathcal{A} \sqsubseteq^{\ell_1,\ell_2} \mathcal{B}$ but $\mathcal{A} \not\sqsubseteq^{k_1,k_2} \mathcal{B}$.   □

3.2   Reductions to Ordinary Simulation

The following theorem shows that bounded buffers can be eliminated at the cost of a blow-up in DUPLI-CATOR's state space. In fact, the buffer can be incorporated into DUPLICATOR's automaton, resulting in an effective capacity of 0 for this buffer. The structure needs to be defined such that DUPLICATOR can react immediately to any alphabet symbol that would have been put into that buffer, rather than consuming the older buffer content first and thus advancing in the automaton to a different state.

Theorem 7. *Let $k_1 \in \mathbb{N} \cup \{\omega\}$ and $k_2 \in \mathbb{N}$. For any automaton $\mathcal{B}$ of size n there is an automaton $\mathcal{B}'$ of size $\le 2n \cdot (|\Sigma|^{k_2+1} - 1)$ such that for any automaton $\mathcal{A}$ we have that $\mathcal{A} \sqsubseteq^{k_1,k_2} \mathcal{B}$ iff $\mathcal{A} \sqsubseteq^{k_1,0} \mathcal{B}'$.*

Proof. Intuitively, we save the content of the buffers in the state space of $\mathcal{B}'$. Formally, let $\Delta := \{a \in \Sigma \mid \sigma(a) = 2\}$ be the set of all letters that get stored in the buffer of capacity k_2 which is to be reduced down to size 0. We write $\Delta^{\le k}$ to denote $\{\varepsilon\} \cup \Delta^1 \cup \ldots \cup \Delta^k$. Let $\mathcal{B} = (Q, \Sigma, q_I, \delta, F)$. Then define $\mathcal{B}' := (Q \times \Delta^{\le k_2} \times \{0,1\}, \Sigma, (q_I, \varepsilon, 0), \delta', F \times \Delta^{\le k_2} \times \{0\})$ with

$$\delta'((q,w,d),a) = \begin{cases} \{(q,aw,1)\} \cup \{(p,v,0) \mid aw = vb, p \in \delta(q,b)\} & \text{, if } a \in \Delta \text{ and } |w| < k_2, \\ \{(p,av,0) \mid w = vb, p \in \delta(q,b)\} & \text{, if } a \in \Delta \text{ and } |w| = k_2, \\ \{(p,w,d) \mid p \in \delta(q,a)\} & \text{, if } a \in \Sigma \setminus \Delta. \end{cases}$$

The third component in the states of $\mathcal{B}'$ is used to indicate whether or not something has been put into the buffer. It is not difficult to transform DUPLICATOR's winning strategies between the games $\mathcal{G}^{k_1,k_2}(\mathcal{A},\mathcal{B})$ and $\mathcal{G}^{k_1,0}(\mathcal{A},\mathcal{B}')$. Suppose $\mathcal{A} \sqsubseteq^{k_1,k_2} \mathcal{B}$. Then, by [16], DUPLICATOR has a positional winning strategy σ in the game $\mathcal{G}^{k_1,k_2}(\mathcal{A},\mathcal{B})$. Then we can define a positional winning strategy σ' for her in the latter via $\sigma'(q,(\beta_1,\varepsilon),(p,w,d)) := \sigma(q,(\beta_1,w),p)$ in case of $|w| = k_2$. If $|w| < k_2$ then she simply follows the deterministic transitions in $\mathcal{B}'$ which amount to waiting for the buffer to be filled. The transformation of positional winning strategies in the other direction works in the same way. Note that there is no choice for her with respect to the value of d; its value is determined by the value of the other components and the history of a play.   □

Clearly, the same argument can be used to reduce (k_1,k_2)-simulation to $(0,k_2)$-simulation when $k_1 \in \mathbb{N}$.

The following result should be obvious given that two buffers of capacity 0 do not introduce any partial commutativity between input symbols since they always have to be consumed by DUPLICATOR
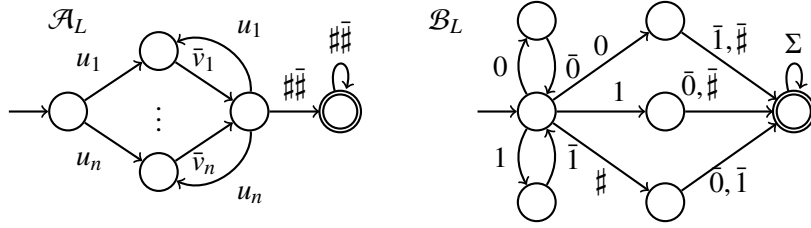
Figure 1: Automata $\mathcal{A}_L$ and $\mathcal{B}_L$ used in the proof of Theorem 10.

immediately after SPOILER has produced them. Hence, the order of consumption remains the same as the order in which they are produced.

Lemma 8. $\sqsubseteq^{0,0} = \sqsubseteq^0$.

3.3 Decidability and Complexity

By applying Theorem 7 twice we can transform the input automata for a two-buffer simulation game with bounded buffers into a pair of automata on which we need to check the relation $\sqsubseteq^{0,0}$ which, by Lemma 8 and Proposition 1 (a) is the same as ordinary fair simulation. It is known [13] that fair simulation games can be solved in polynomial time because they are special cases of parity games of index 3, and parity games of fixed index can be solved in polynomial time [11, 20].

Corollary 9. *For every fixed* $k_1, k_2 \in \mathbb{N}$, *the relation* $\sqsubseteq^{k_1, k_2}$ *is decidable in polynomial time.*

For unbounded buffers the situation is different. Decision problems involving two unbounded buffers typically become undecidable as stated in the introduction. Here we adapt the argument used for the reachability problem for communicating finite state machines [4, 6], and show the undecidability of $\sqsubseteq^{\omega, \omega}$ by a reduction from Post's Correspondence Problem [22]. The reduction basically constructs a pair of automata, such that to win the game SPOILER is required to produce a possible solution for the PCP and store it in the two unbounded buffers. DUPLICATOR's role is to check whether this is indeed a correct solution.

Theorem 10. *The relation* $\sqsubseteq^{\omega, \omega}$ *is undecidable.*

Proof. Let $L = \{(u_1, v_1), \ldots, (u_n, v_n)\}$ be an input for the PCP where u_i and v_i are non-empty finite words over $\{0, 1\}$. We construct two automata $\mathcal{A}_L$ and $\mathcal{B}_L$ over $\Sigma = \{0, 1, \sharp, \bar{0}, \bar{1}, \bar{\sharp}\}$. We write $\bar{w}$ to denote $\bar{a}_1 \ldots \bar{a}_m \in \{\bar{0}, \bar{1}, \bar{\sharp}\}^*$ for $w = a_1 \ldots a_m$. We define $\sigma : \Sigma \to \{1, 2\}$, where $\sigma(x) = 1$ for $x \in \{0, 1, \sharp\}$, and $\sigma(x) = 2$ otherwise. Let $\mathcal{A}_L$ and $\mathcal{B}_L$ be the automata depicted in Figure 1. We abbreviate a sequence of transitions with letters $a_1, \ldots, a_k$ in $\mathcal{A}_L$ with a single arrow.

An infinite word is accepted by $\mathcal{A}_L$ iff it is of the form $u_1 \bar{v}_1 \ldots u_n \bar{v}_n (\sharp \bar{\sharp})^\omega$. An infinite word is accepted by $\mathcal{B}_L$ if it starts with balanced pairs $(a\bar{a})^*$, $a \in \{0, 1\}$, and at one point contains an unbalanced pair $x\bar{y}$ with $x, y \in \{0, 1, \sharp\}$ and $x \neq y$.

We claim that $\mathcal{A}_L \not\sqsubseteq^{\omega, \omega} \mathcal{B}_L$ iff there is a solution for L. Suppose such a solution $i_1, \ldots, i_m$ exists. Then Spoiler wins by producing an accepting word $u_{i_1} \bar{v}_{i_1} \ldots u_{i_m} \bar{v}_{i_m} (\sharp \bar{\sharp})^\omega$. By Lemma 11, DUPLICATOR has to find a run on some word of the form $x_1 \bar{y}_1 x_2 \bar{y}_2 x_3 \bar{y}_3 \ldots$ with $x_1 x_2 x_3 \cdots = u_{i_1} u_{i_2} \ldots u_{i_m} \sharp^\omega = v_{i_1} v_{i_2} \ldots v_{i_m} \sharp^\omega = y_1 y_2 y_3 \ldots$, i.e., without a mismatch. Hence she cannot win the play.

On the other hand, if there is no solution for L, then no matter which path SPOILER chooses in his automaton it will either not reach the accepting loop or it will have to contain a mismatch in the sense

that the concatenation of the *u*-parts and the concatenation of the *v*-parts differ in one digit at some position. DUPLICATOR's winning strategy consists of skipping her turn until such a mismatch is being produced which enables her to go to the accepting loop. Otherwise she waits forever but SPOILER does not produce an accepting run so she also wins such plays. $\qquad\square$

3.4 The Relationship to Language-Like Inclusion Problems

Note that so far we have considered automata as finite-state devices with no particular semantics other than that provided by the two-buffer games. Clearly, these automata are Büchi automata but we have avoided this analogy because two-buffer games, unlike single-buffer games, lead away from the problem of language inclusion. We end this section on two-buffer games with a lemma that relates winning strategies in two-buffer games with a language-theoretic problem that boils down to language inclusion in the case of a single buffer only. It is the analogy of case (d) of Proposition 1 for games with two buffers.

Suppose the alphabet Σ and a distribution function $\sigma : \Sigma \to \{1,2\}$ is given. Let $\Sigma_i := \{a \in \Sigma \mid \sigma(a) = i\}$ for any $i \in \{1,2\}$. For a word $w \in \Sigma^\omega$ and an $i \in \{1,2\}$ we write $w\!\downarrow_i$ for the projection of w onto Σ_i. Note that $w\!\downarrow_i \in \Sigma_i^* \cup \Sigma_i^\omega$.

Remember that $L(\mathcal{A})$ is used to denote the language of $\mathcal{A}$ seen as a Büchi automaton, i.e. the set of all words $w \in \Sigma^\omega$ for which there is an infinite path labelled by w that visits final states infinitely often.

Lemma 11. *Let $\mathcal{A}, \mathcal{B}$ be two automata over an alphabet Σ with distribution function $\sigma : \Sigma \to \{1,2\}$. Let $k_1, k_2 \in (\mathbb{N} \cup \{\omega\})$. If $\mathcal{A} \sqsubseteq^{k_1,k_2} \mathcal{B}$ then for every word $w \in L(\mathcal{A})$ there is a $v \in L(\mathcal{B})$ such that for all $i \in \{1,2\}$ we have $w\!\downarrow_i = v\!\downarrow_i$.*

Proof. Let $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, q_I^{\mathcal{A}}, \delta^{\mathcal{A}}, F^{\mathcal{B}})$ and $\mathcal{B} = (Q^{\mathcal{A}}, \Sigma, q_I^{\mathcal{A}}, \delta^{\mathcal{A}}, F^{\mathcal{B}})$. Suppose there is a word $w = a_0 a_1 \ldots \in L(\mathcal{A})$, i.e. there is an infinite path $\rho = q_0, a_0, q_1, a_1, \ldots$ such that $q_i \in F^{\mathcal{A}}$ for infinitely many i. Suppose furthermore that we have $\mathcal{A} \sqsubseteq^{k_1,k_2} \mathcal{B}$, i.e. player DUPLICATOR has a winning strategy ζ for the game $\mathcal{G}^{k_1,k_2}(\mathcal{A}, \mathcal{B})$. We remark that the values of k_1 and k_2 are irrelevant for what follows; only their existence is needed.

So suppose that SPOILER chooses the run ρ. Following ζ, player DUPLICATOR will construct – possibly in chunks – an infinite path $\rho' = p_0, b_0, p_1, b_1, \ldots$ through $\mathcal{B}$. Since the resulting play is winning for DUPLICATOR, this path must contain infinitely many states in $F^{\mathcal{B}}$. Thus, we have $v := b_0 b_1 \ldots \in L(\mathcal{B})$. It remains to be seen that for every $i \in \{1,2\}$ we have $w\!\downarrow_i = v\!\downarrow_i$. This is a simple consequence of three facts.

1. DUPLICATOR can only choose transitions with symbols that have been put into one of the buffers by SPOILER. Hence, for every j there is a j' such that $b_j = a_{j'}$.

2. Every symbol that gets put into the buffer is eventually removed from it. Hence, for every j there is a j' such that $a_j = b_{j'}$.

3. The buffers make sure that the order of two symbols from the same Σ_i is being preserved.

Thus, we get $w\!\downarrow_i = v\!\downarrow_i$ for every i. $\qquad\square$

Note that in the case of the single buffer simulation game, we necessarily have $v = w$ in the formulation of this lemma which says nothing more than $L(\mathcal{A}) \subseteq L(\mathcal{B})$. In cases with two buffers, this lemma predicts language inclusion of the two automata modulo partial commutativity between alphabet symbols of different σ-index. The following section shows cases of problems in which such partial commutativity occurs naturally and shows how two-buffer games can be used to characterise such problems.

4 Application: Approximating Inclusion of ω-Rational Relations

The main motivation for studying single-buffer games played on two Büchi automata is derived from the fact that such enhanced simulations approximate language inclusion between Büchi automata, an important problem in the specification and verification of reactive systems [10]. Remember that simulations based on a single bounded buffer ($\sqsubseteq^k$ for some $k \in \mathbb{N}$) provide polynomial-time approximations to a problem that is PSPACE-complete. Using an unbounded buffer defeats the purpose here because $\sqsubseteq^\omega$ still only provides an approximation to language inclusion but additionally it is EXPTIME-complete [19], i.e. even harder than Büchi inclusion.

We consider the situation for rational relations[1]. Let Σ_{in} and Σ_{out} be finite alphabets, usually referred to as *input* and *output* alphabets. We write Σ^∞ for $\Sigma^* \cup \Sigma^\omega$. An *infinitary rational relation* is an $R \subseteq \Sigma_{\mathsf{in}}^\infty \times \Sigma_{\mathsf{out}}^\infty$ that is recognised in the following sense.

Definition 12 (see, e.g., [15]). A 2-head Büchi transducer is a $\mathcal{T} = (Q, \Sigma_{\mathsf{in}}, \Sigma_{\mathsf{out}}, q_I, \delta, F)$ where Q is a finite set of states with a designated starting state $q_I \in Q$ and a designated set of *final* states $F \subseteq Q$; Σ_{in} and Σ_{out} are two finite alphabets and $\delta \subseteq Q \times \Sigma_{\mathsf{in}}^* \times \Sigma_{\mathsf{out}}^* \times Q$ is a finite set of transitions.

A *run* is an infinite sequence $q_0, u_0, v_0, q_1, u_1, v_1, \ldots$ over $Q \times \Sigma_{\mathsf{in}}^* \times \Sigma_{\mathsf{out}}^*$ such that for all $i \in \mathbb{N}$ we have $(q_i, u_i, v_i, q_{i+1}) \in \delta$. The run is accepting if $q_0 = q_I$ and there is some $q \in F$ such that $q = q_i$ for infinitely many i. In that case, we say that the pair $(u, v) \in \Sigma_{\mathsf{in}}^\infty \times \Sigma_{\mathsf{out}}^\infty$ with $u = u_0 u_1 u_2 \ldots$ and $v = v_0 v_1 v_2 \ldots$ is *accepted* or *recognised* by $\mathcal{T}$.

The relation recognised by $\mathcal{T}$ is $R(\mathcal{T}) := \{(u, v) \mid (u, v) \text{ is recognised by } \mathcal{T}\}$.

The equivalence problem for infinitary rational relations – given $\mathcal{T}, \mathcal{T}'$, decide whether or not $R(\mathcal{T}) = R(\mathcal{T}')$ – and, hence, the inclusion problem is undecidable [2], even for deterministic transducers in the sense above. The latter can easily be shown by a reduction from the infinitary PCP. Equivalence becomes decidable for Büchi transducers that read a pair of exactly one input and one output symbol in each step and, hence, can be seen as Büchi-automata over the alphabet $\Sigma_{\mathsf{in}} \times \Sigma_{\mathsf{out}}$.

Transducers operating on infinite words have important applications. For instance, they are used to represent components of infinite structures, namely ω-automatic ones [3] where decidability results are obtained for model-checking like problems; they are used in concurrency to specify the synchronisation behaviour of parallel processes [21]; they can represent functions and relations on real numbers [5]; etc.

Given the undecidability of the inclusion problem for infinitary rational relations on one hand and their applications on the other, it is fair to ask whether or not there are possibilities to approximate relation inclusion in the form of algorithms that are sound but incomplete for instance. Such a possibility is given by the two-buffer simulations developed in the previous section: just like single-buffer simulation games approximate language inclusion between Büchi automata, two-buffer games approximate inclusion between infinitary rational relations, as is shown in the following.

Definition 13. A 2-head Büchi transducer $\mathcal{T} = (Q, \Sigma_{\mathsf{in}}, \Sigma_{\mathsf{out}}, q_I, \delta, F)$ is *normalised* if its transition relation is of the form

$$\delta \;\subseteq\; (Q \times \Sigma_{\mathsf{in}} \times \{\varepsilon\} \times Q) \cup (Q \times \{\varepsilon\} \times \Sigma_{\mathsf{out}} \times Q) \,.$$

It should be clear that every 2-head Büchi transducer can be normalised preserving the relation that it recognises, and that this involves a linear blow-up at most: every transition that consumes the input-output pair $(a_1 \ldots a_n, b_1 \ldots b_m)$ can be simulated by $n + m$ transitions, each of which consumes exactly one letter from either input or output. Moreover, w.l.o.g. we can assume $\Sigma_{\mathsf{in}} \cap \Sigma_{\mathsf{out}} = \emptyset$, i.e. every symbol is

---

[1]Here we only consider the case of binary relations. The generalisation to relations between a larger number of words is straight-forward.

either input or output but not both. Then the transition relation of such a normalised transducer can be seen as of type $Q \times (\Sigma_{\text{in}} \cup \Sigma_{\text{out}}) \times Q$ and, syntactically, this is nothing more than a Büchi automaton as used in Section 2.

Theorem 14. *Let $\mathcal{T}$ and $\mathcal{T}'$ be two normalised 2-head Büchi transducers over inputs Σ_{in} and outputs Σ_{out} with $\Sigma_{\text{in}} \cap \Sigma_{\text{out}} = \emptyset$. Let $\Sigma := \Sigma_{\text{in}} \cup \Sigma_{\text{out}}$ with the alphabet mapping function $\sigma(a) = 1$ if $a \in \Sigma_{\text{in}}$ and $\sigma(a) = 2$ otherwise. Then we have: if $\mathcal{T} \sqsubseteq^{\omega,\omega} \mathcal{T}'$ then $R(\mathcal{T}) \subseteq R(\mathcal{T}')$.*

Proof. This follows immediately from Lemma 11 with the observation that $R(\mathcal{T}) \subseteq R(\mathcal{T}')$ iff for every word $w \in \Sigma^\omega$: if $(w{\downarrow}_{\text{in}}, w{\downarrow}_{\text{out}}) \in R(\mathcal{T})$ then $(w{\downarrow}_{\text{in}}, w{\downarrow}_{\text{out}}) \in R(\mathcal{T}')$.   □

Combining this with Thm. 6 and Cor. 9 we obtain polynomial-time computable approximations for the inclusion problem between infinitary recognisable relations.

Corollary 15. *Let $k_1, k_2 \in \mathbb{N}$. We have that*

- *it is decidable in polynomial time whether or not $\mathcal{T} \sqsubseteq^{k_1,k_2} \mathcal{T}'$ holds for arbitrary 2-head Büchi transducers, and*

- *if $\mathcal{T} \sqsubseteq^{k_1,k_2} \mathcal{T}'$ holds then we have $R(\mathcal{T}) \subseteq R(\mathcal{T}')$.*

The approximation is indeed not complete as the following example shows.

Example 16. Consider the normalised transducers $\mathcal{T}$ (left) and $\mathcal{T}'$ (right) over the alphabets $\Sigma_{\text{in}} = \{a\}$ and $\Sigma_{\text{out}} = \{b, c\}$.
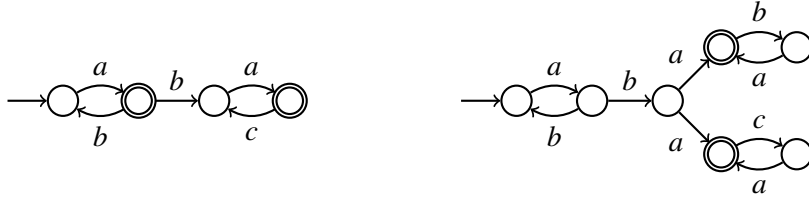
Both recognise the infinitary relation described by $(a^\omega, b^\omega) \cup (a^\omega, b^* c^\omega)$. Thus, relation inclusion is given between them. On the other hand, it is not difficult to see that SPOILER has a winning strategy for $\mathcal{G}^{k_1,k_2}(\mathcal{T}, \mathcal{T}')$ for all $k_1, k_2 \in \mathbb{N} \cup \{\omega\}$: he cycles on the left loop until DUPLICATOR leaves her left cycle (which she has to eventually since it contains no final states) and then, depending on where DUPLICATOR went, either continues cycling there or moves over to his right cycle so that he produces symbols that DUPLICATOR cannot consume anymore.

5   Conclusion and Further Work

We have studied simulation games that use two FIFO buffers to store SPOILER's choices for a limited amount of time before DUPLICATOR has to respond to them. We have shown how the correspondence between (single-)buffered simulation games and the language inclusion problem for Büchi automata naturally extends to two-buffer simulations and inclusion problems for rational relations over infinite words. The fact that games with bounded capacities can be solved in polynomial time, and that higher buffer capacities do not make DUPLICATOR weaker yields polynomial-time approximations to such inclusion problems with underlying partial commutativity.

Our results are closely related to the theory of (infinite) Mazurkiewicz traces (see [8] for a general overview on this theory). The reason is that relations in $\Sigma_1^\omega \times \Sigma_2^\omega$ can be seen as languages of real traces

over particular independence alphabets (namely, complete bipartite ones). More precisely: Fix a mapping $\sigma : \Sigma \rightarrow \{1, 2\}$ and let $\Sigma_i = \{a \in \Sigma \mid \sigma(a) = i\}$. Furthermore, let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi-automata with $\mathcal{A} \sqsubseteq^{k_1,k_2} \mathcal{B}$. Then, for any word $w \in L(\mathcal{A})$, there exists a word $v \in L(\mathcal{B})$ with $w\downarrow_i = v\downarrow_i$ for $i = 1, 2$. In trace-theoretic terms, this means that the trace closure of $L(\mathcal{A})$ is contained in the trace closure of $L(\mathcal{B})$ (where the underlying independence relation is $I = (\Sigma_1 \times \Sigma_2) \cup (\Sigma_2 \times \Sigma_1)$, i.e., complete bipartite). Hence, for such special independence relations, we obtain polynomial-time computable approximations for the inclusion problem of trace closures of ω-regular languages (which is undecidable). It is possible to extend the results of this paper to $m \geq 2$ buffers, which allows similar polynomial-time computable approximations to be obtained in case of complete m-partite independence relations. In our ongoing work [17], we extend σ and the multi-buffer simulation game in a way that allows arbitrary independence relations to be captured (not just complete m-partite ones).

Regarding decidability, we show that $\sqsubseteq^{\omega,0}$ is highly undecidable (i.e., not arithmetical) [17]. This is in sharp contrast with the decidability result for $\sqsubseteq^{\omega}$ from [19] and the fact that bounded buffers can be encoded in the control state.

An idea for further work is the following. Recall that $\sqsubseteq^{\omega}$ is EXPTIME-complete and it has a PSPACE-complete variant in which DUPLICATOR may never consume an element from the buffer *and* leave others in, i.e. she always has to flush the buffer whenever she moves [19]. One can study such a variant for two-buffer simulations as well. This becomes technically a little bit more tedious but not conceptually problematic; one could require her to flush *both* buffers or just *one*, etc. The same reduction of PCP shows that the simulation remains undecidable if two buffers are unbounded. However, it would be interesting to determine the status of $\sqsubseteq^{\omega,k}$: is it highly undecidable as $\sqsubseteq^{\omega,0}$ [17], arithmetical or even decidable and, if so, what is its precise complexity.

References

[1] P. A. Abdulla, A. Bouajjani, L. Holík, L. Kaati & T. Vojnar (2008): *Computing Simulations over Tree Automata*. In: *TACAS'08*, *Lecture Notes in Computer Science* 4963, Springer, pp. 93–108, doi:10.1007/978-3-540-78800-3_8.

[2] J. Berstel (1979): *Transductions and Context-Free Languages*. Leitfäden der angewandten Mathematik und Mechanik, Teubner, doi:10.1007/978-3-663-09367-1.

[3] A. Blumensath & E. Grädel (2000): *Automatic Structures*. In: *Proc. 15th IEEE Symp. on Logic in Computer Science, LICS'00*, IEEE, pp. 51–62, doi:10.1109/LICS.2000.855755.

[4] D. Brand & P. Zafiropulo (1983): *On Communicating Finite-State Machines*. *J. ACM* 30(2), pp. 323–342, doi:10.1145/322374.322380.

[5] O. Carton (2010): *Right-Sequential Functions on Infinite Words*. In: *Proc. 5th Int. Conf. on Computer Science in Russia, CSR'10, Lecture Notes in Computer Science* 6072, Springer, pp. 96–106, doi:10.1007/978-3-642-13182-0.

[6] P. Chambart & P. Schnoebelen (2008): *Mixing Lossy and Perfect Fifo Channels*. In: *CONCUR'08*, *LNCS* 5201, Springer, pp. 340–355, doi:10.1007/978-3-540-85361-9_28.

[7] L. Clemente & R. Mayr (2013): *Advanced automata minimization*. In: *POPL'13*, ACM, pp. 63–74, doi:10.1145/2429069.2429079.

[8] V. Diekert & Y. Métivier (1997): *Partial Commutation and Traces*. In G. Rozenberg & A. Salomaa, editors: *Handbook of Formal Languages*, 3, Springer, pp. 457–533, doi:10.1007/978-3-642-59126-6_8.

[9] D. L. Dill, A. J. Hu & H. Wong-Toi (1992): *Checking for Language Inclusion Using Simulation Preorders*. In: *CAV'91*, *LNCS* 575, Springer, pp. 255–265, doi:10.1007/3-540-55179-4_25.

[10] E. A. Emerson (1996): *Automated Temporal Reasoning about Reactive Systems*, pp. 41–101. *LNCS* 1043, Springer, New York, NY, USA, doi:10.1007/3-540-60915-6_3.

[11] E. A. Emerson & C. L. Lei (1986): *Efficient Model Checking in Fragments of the Propositional μ–Calculus*. In: *Symposion on Logic in Computer Science*, IEEE, Washington, D.C., USA, pp. 267–278.

[12] K. Etessami (2002): *A Hierarchy of Polynomial-Time Computable Simulations for Automata*. In: *CONCUR'02, Lecture Notes in Computer Science* 2421, Springer, pp. 131–144, doi:10.1007/3-540-45694-5_10.

[13] K. Etessami, T. Wilke & R. A. Schuller (2001): *Fair Simulation Relations, Parity Games, and State Space Reduction for Büchi Automata*. In: *ICALP'01, LNCS* 2076, Springer, pp. 694–707, doi:10.1007/3-540-48224-5_57.

[14] C. Fritz & T. Wilke (2005): *Simulation relations for alternating Büchi automata*. *Theor. Comput. Sci* 338(1-3), pp. 275–314, doi:10.1016/j.tcs.2005.01.016.

[15] F. Gire & M. Nivat (1984): *Relations rationelles infinitaires*. *Calcolo* 21, pp. 91–125, doi:10.1007/BF02575909.

[16] Y. Gurevich & L. Harrington (1982): *Trees, Automata, and Games*. In: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pp. 60–65, doi:10.1145/800070.802177.

[17] M. Hutagalung, N. Hundeshagen, D. Kuske, M. Lange & E. Lozes (2016): *Multi-Buffer Simulations for Trace Language Inclusion*. In preparation.

[18] M. Hutagalung, M. Lange & E. Lozes (2013): *Revealing vs. Concealing: More Simulation Games for Büchi Inclusion*. In: *LATA'2013, LNCS* 7810, Springer, pp. 347–358, doi:10.1007/978-3-642-37064-9_31.

[19] M. Hutagalung, M. Lange & E. Lozes (2014): *Buffered Simulation Games for Büchi Automata*. In: *AFL'14*, *EPTCS* 151, pp. 286–300, doi:10.4204/EPTCS.151.20.

[20] M. Jurdziński (2000): *Small progress measures for solving parity games*. In H. Reichel & S. Tison, editors: *STACS'00, LNCS* 1770, Springer, pp. 290–301, doi:10.1007/3-540-46541-3_24.

[21] M. Nivat (1981): *Infinitary Relations*. In: *Proc. 6th Coll. on Trees in Algebra and Programming, CAAP'81*, *Lecture Notes in Computer Science* 112, Springer, pp. 46–75, doi:10.1007/3-540-10828-9_54.

[22] E. Post (1946): *A variant of a recursively unsolvable problem*. *Bulletin of the American Mathematical Society* 53, pp. 264–268, doi:10.1090/S0002-9904-1946-08555-9.