Multi-Buffer Simulations for Trace Language Inclusion

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 several buffers before she executes them on her structure. We show that the simulation games are useful to approximate the inclusion of trace closures of languages accepted by finite-state automata, which is known to be undecidable. We study the decidability and complexity and show that the game with bounded buffers can be decided in polynomial time, whereas the game with one unbounded and one bounded buffer is highly undecidable. We also show some sufficient conditions on the automata for Duplicator to win the game (with unbounded buffers).

1 Introduction

Simulation is a pre-order between labeled transition systems $\mathcal{T}$ and $\mathcal{T}'$ that formalizes 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 under-approximate language inclusion problems for automata on finite or infinite words and trees and to minimise such automata [7, 9, 11, 1]. One advantage of these simulation relationships is that they are often computable in polynomial time whereas language inclusion problems are PSPACE-complete for typical (finite, Büchi, parity, etc.) automata on words and EXPTIME-complete for such automata on trees. To reason about simulation relations, one very often characterises them by the existence of winning strategies of the second player in certain two-player games. These are played on the state spaces of two automata where one player (SPOILER) reveals a run of the first automaton piece-wise and the second player (DUPLICATOR) has to produce a corresponding run of the second automaton (where "corresponding" often means "on the same word or tree"). The simplest such game requires SPOILER to produce one step of his run per round and DUPLICATOR to answer immediately by one step of her run. With this game, it is very easy to construct pairs of automata such that language inclusion holds but simulation does not (i.e., DUPLICATOR has no winning strategy). Intuitively, DUPLICATOR is too weak to capture language inclusion. This observation 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. Examples in this context are *multi-pebble simulation* [8], *multi-letter simulation* [13, 5], *buffered simulations* [14], and *delayed games* [12]. In all these contexts, the winning condition is a regular set of infinite words over the set of pairs of letters (this is explicit in [12] and implicit in [8, 13, 5, 14] where DUPLICATOR aims to produce the same word).

In this paper, we aim at approximating the inclusion of the Mazurkiewicz trace closure of two regular languages using simulation technology. More precisely, we are given two Büchi automata $\mathcal{A}$ and $\mathcal{B}$ and a trace alphabet and we ask whether, for every infinite word accepted by $\mathcal{A}$, there is a trace-equivalent word accepted by $\mathcal{B}$. This problem was shown to be undecidable by Sakarovitch [19] (and [10] can be used to prove that it is even highly undecidable). To approximate this problem, we use a game approach as indicated above, i.e., Spoiler and Duplicator reveal runs of $\mathcal{A}$ and $\mathcal{B}$ piece-wise producing, in the limit, a pair of runs. In doing so, Duplicator tries to produce a run on a trace-equivalent word. Since the set of pairs of trace-equivalent words is not regular, Duplicator's winning condition is not a regular set. Hence the results from [12, 8, 13, 5, 14] are not applicable.

To overcome this problem, we first restrict Duplicator's moves in such a way that she is forced to produce a prefix of some trace-equivalent word. This is done using several buffers, i.e., extending the idea of buffered simulation from [14]: instead of using only one buffer, there are several buffers of certain capacities and associated (not necessarily disjoint) alphabets. Whenever Spoiler chooses a letter, it is written to all those buffers whose alphabet contains that letter. Dually, Duplicator can only use those letters that are available at all the associated buffers. Duplicator can only win if she does not leave any letter in any of the buffers for ever. With this setup of game and winning condition, Duplicator effectively attempts to produce a trace-equivalent word. The second part of the winning condition is standard: if Spoiler produces an accepting run, then Duplicator's run has to be accepting as well.

Our main results in this context are the following:

- If Duplicator has a winning strategy, then the language of the first automaton is contained in the trace closure of the language of the second automaton (Thm. 3). While the latter property is undecidable, the existence of a winning strategy with buffers of finite capacities is decidable in polynomial time (provided the number and capacities of buffers are unchanged, Thm. 4).
- From [14], we know that buffered simulation (using a single unbounded buffer) is decidable. Section 4 proves that adding a single bounded buffer yields a highly undecidable simulation relation, as hard as recursive Büchi games and therefore hard for the class of all Boolean combinations of Σ^1_1-problems (Thm. 9).
- Section 5 describes the simulation relations in terms of continuous functions between the accepting runs of the two automata. This yields completeness results in the sense that multi-buffer simulation implies trace-closure inclusion in certain cases.

2 Büchi Automata and Trace Languages

Let Σ be an alphabet. Then Σ^* denotes the set of finite words over Σ, Σ^ω is the set of all infinite words over Σ, and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. For a natural number k, we set $[k] = \{1, 2, \ldots, k\}$.

A *nondeterministic Büchi automaton* or *NBA* is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ where Q is a finite set of *states*, Σ is an alphabet, $q_I \in Q$ is the *initial state*, $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$ is the *transition function*, and $F \subseteq Q$ is the set of *accepting states*.

Let $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ be an infinite word over Σ. A *run* of $\mathcal{A}$ on w is an alternating sequence of states and letters $\rho = (q_0, a_0, q_1, a_1, \ldots)$ with $q_0 = q_I$ and $q_{i+1} \in \delta(q_i, a_i)$ for all $i \geq 0$. This run is *accepting* if $q_i \in F$ for infinitely many $i \in \mathbb{N}$. The *language $L(\mathcal{A})$ of $\mathcal{A}$* is the set of infinite words that admit an accepting run.

The main motivation of this paper is to approximate inclusion of trace languages. Therefore we shortly introduce the notions of finite and infinite traces, for a detailed treatment see [6].

A *trace alphabet* is a tuple $\sigma = (\Sigma_i)_{i \in [k]}$ of not necessarily disjoint alphabets (note that k is arbitrary). Let $\Sigma = \bigcup_{i \in [k]} \Sigma_i$ and, for $a \in \Sigma$, let $\sigma(a) = \{i \in [k] \mid a \in \Sigma_i\}$ which is by construction nonempty. The idea is that the letter $a \in \Sigma$ denotes an action that is performed by the set of processes $\sigma(a)$. For $i \in [k]$,

$\pi_i \colon \Sigma^\infty \to \Sigma_i^\infty$ is the natural projection function that deletes from each word all letters that do not belong to Σ_i. We call two words $u, v \in \Sigma^\infty$ *σ-equivalent* if $\pi_i(u) = \pi_i(v)$ for all $i \in [k]$. In this case, we write $u \sim_\sigma v$. The relation $\sim_\sigma$ is called *trace equivalence*.

The restriction of $\sim_\sigma$ to Σ^* has an alternative characterisation (that is the traditional definition of trace equivalence): let $D = \bigcup_{i\in[k]} \Sigma_i \times \Sigma_i \subseteq \Sigma^2$ denote the set of pairs (a,b) with $\sigma(a) \cap \sigma(b) \neq \emptyset$. This reflexive and symmetric relation is called the *dependence relation* associated with σ. Then the restriction of $\sim_\sigma$ to Σ^* is the least congruence on the free monoid Σ^* with $ab \sim_\sigma ba$ for all $(a,b) \notin D$.[1] The quotient $\mathbb{M}(\sigma) = \Sigma^*/_{\sim_\sigma}$ is called *trace monoid*, its elements are *finite traces*. The quotient $\mathbb{R}(\sigma) = \Sigma^\omega/_{\sim_\sigma}$ is the set of *real* or *infinite traces*. The *trace closure* of a language $L \subseteq \Sigma^\infty$ w.r.t. σ is the language $[L]_\sigma = \{v \in \Sigma^\infty \mid \exists u \in L \colon u \sim_\sigma v\}$. The language L is *trace closed* if it equals its trace closure.

Example 1. Let $\Sigma = \{a,b,c\}$ with $\sigma(a) = \{1\}$, $\sigma(b) = \{1,2\}$, and $\sigma(c) = \{2\}$. Then $a^*(bc)^*$ is trace closed, the trace closure of a^*c^* is the language $\{a,c\}^*$ and the trace closure of $(ac)^*$ is the language of all words $u \in \{a,c\}^*$ with the same numbers of occurrences of a and c, resp.

Let the mapping σ be such that the induced independence relation $\Sigma^2 \setminus D$ is not transitive. Then, given a regular language $L \subseteq \Sigma^*$, it is undecidable whether its trace closure $[L]_\sigma$ is regular [19]. Even more, it is undecidable whether the closure is universal, i.e., equals Σ^*. Consequently, for two regular languages K and L, it is undecidable whether $K \subseteq [L]_\sigma$ (which is equivalent to $[K]_\sigma \subseteq [L]_\sigma$). These negative results also hold for languages of infinite words and their trace closures.

3 Multi-Buffer Simulations

Let $\sigma = (\Sigma_i)_{i\in[k]}$ be a trace alphabet and $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, p_I, \delta^{\mathcal{A}}, F^{\mathcal{A}})$ and $\mathcal{B} = (Q^{\mathcal{B}}, \Sigma, q_I, \delta^{\mathcal{B}}, F^{\mathcal{B}})$ be two automata over the alphabet Σ. We aim at finding an approximation to the undecidable question of $L(\mathcal{A}) \subseteq [L(\mathcal{B})]_\sigma$ via simulation relations. In these, we have k FIFO buffers and $\sigma(a) \subseteq [k]$ is interpreted as the set of buffers that are used to transmit the letter a. Let $\kappa \colon [k] \to \mathbb{N} \cup \{\omega\}$ be a function that assigns a *capacity* to each buffer, i.e. the maximum number of letters that this buffer can contain at any time. We will often write such a function as a tuple $(\kappa(1), \ldots, \kappa(k))$.

The multi-buffer game $\mathcal{G}^\kappa_\sigma(\mathcal{A},\mathcal{B})$ is played on these two automata and the k buffers between players Spoiler and Duplicator as follows. Configurations are tuples $(p, \beta_1, \beta_2, \ldots, \beta_k, q) \in Q^{\mathcal{A}} \times \Sigma_1^* \times \ldots \times \Sigma_k^* \times Q^{\mathcal{B}}$ with $|\beta_i| \leq \kappa(i)$ for all $i \in [k]$. The first and last component can be seen as the places of two tokens on the state spaces of $\mathcal{A}$ and $\mathcal{B}$ respectively; the others denote the current buffer contents. The initial configuration is $(p_I, \varepsilon, \ldots, \varepsilon, q_I)$. A round consists of a move by Spoiler followed by a move by Duplicator. Spoiler choses $a \in \Sigma$, moves the token in $\mathcal{A}$ forward along an a-transition of his choice, and pushes a copy of the a-symbol to each of the buffers from $\sigma(a)$. Then Duplicator either skips her turn or chooses a non-empty word $a_1 \ldots a_n \in \Sigma^+$ and moves the token in $\mathcal{B}$ along some $a_1 \ldots a_n$-labeled path. While doing so, for every i, she pops an a_i from each of the buffers from $\sigma(a_i)$. More formally, in a configuration of the form $(p, \beta_1, \ldots, \beta_k, q)$,

1. Spoiler picks a letter $a \in \Sigma$ and a state $p' \in Q^{\mathcal{A}}$ such that $p \xrightarrow{a} p'$, and outputs ap'.
2. Duplicator picks a finite run $qv_1q_1v_2q_2 \cdots v_nq'$ from q in the automaton $\mathcal{B}$ such that $\pi_i(a)\beta_i = \beta_i{}'\pi_i(v_1v_2\ldots v_n)$ for all $b \in [k]$. She outputs $v_1q_1v_2q_2 \cdots v_nq'$.

The play proceeds in the configuration $(p', \beta_1{}', \ldots, \beta_k{}', q')$.

[1]Given a reflexive and symmetric relation $D \subseteq \Sigma^2$, one can always find a tuple $(\Sigma_i)_{i\in[k]}$ that induces D (where k depends on D.)

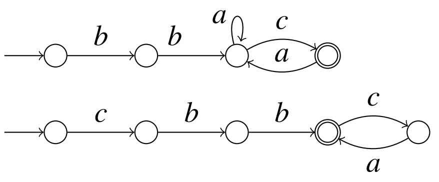
Since $(p', \beta_1', \ldots, \beta_k', q')$ is a configuration, we implicitely have $|\beta_i'| \leq \kappa(i)$, i.e., the size of the buffers is checked *after* the round. So SPOILER can write into a "full" buffer (i.e., with $|\beta_i| = \kappa(i)$) and it is DUPLICATOR's responsibility to shorten the buffer again. In particular, DUPLICATOR has to read all letters from buffers with capacity 0 in the very same round. Furthermore, if SPOILER uses buffers of finite capacity infinitely often, then DUPLICATOR cannot skip forever.

A finite play is *lost* by the player that got stuck (which, for SPOILER, means that he gets trapped in a sink of $\mathcal{A}$ while, for DUPLICATOR, it means that she should shorten a buffer but cannot do so). An infinite play produces an infinite run $\rho_{\mathcal{A}}$ of $\mathcal{A}$ over some infinite word $w_{\mathcal{A}} \in \Sigma^{\omega}$, and a finite or infinite run $\rho_{\mathcal{B}}$ of $\mathcal{B}$ over some word $w_{\mathcal{B}} \in \Sigma^{\infty}$. This play is *won* by DUPLICATOR iff

- $\rho_{\mathcal{A}}$ is not an accepting run, or
- $\rho_{\mathcal{B}}$ is an infinite accepting run and every letter written by SPOILER into a buffer will eventually be read by DUPLICATOR (formally: for every letter $a \in \Sigma$, the numbers of occurrences of a in $w_{\mathcal{A}}$ and in $w_{\mathcal{B}}$ are the same).

We write $\mathcal{A} \sqsubseteq_{\sigma}^{\kappa} \mathcal{B}$ if DUPLICATOR has a winning strategy for the game $\mathcal{G}_{\sigma}^{\kappa}(\mathcal{A}, \mathcal{B})$.

Example 2. Consider the trace alphabet σ with $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b\}$ and $\Sigma_3 = \{c\}$ and the following two NBA $\mathcal{A}$ (top) and $\mathcal{B}$ (below) over the alphabet Σ.



We have $\mathcal{A} \sqsubseteq_{\sigma}^{(\omega,2,0)} \mathcal{B}$. Note that in this game, a and b get put into an unbounded buffer, b also gets put into a buffer of capacity 2, and c gets put into a buffer of capacity 0, i.e. DUPLICATOR has to respond immediately to any c-move made by SPOILER. DUPLICATOR's winning strategy consists of skipping her turn until SPOILER produces a c. Note that he cannot produce more than 2 b's beforehand, hence he cannot win by exceeding the capacity of the second buffer. Note also that he cannot loop on the first a-loop for ever, otherwise he will lose for not producing an accepting run. Once SPOILER eventually produced a c, DUPLICATOR consumes it together with the entire content of the second buffer and moves to the accepting state in her automaton. After that she can immediately respond to every state-changing move by SPOILER.

The following theorem shows indeed that multi-buffer games approximate the inclusion between the trace closures of the languages of two NBA.

Theorem 3. *Let $\sigma = (\Sigma_i)_{i \in [k]}$ be a trace alphabet and let κ be a capacity function for k buffers. Let $\mathcal{A}$ and $\mathcal{B}$ be two NBA over Σ with $\mathcal{A} \sqsubseteq_{\sigma}^{\kappa} \mathcal{B}$. Then $L(\mathcal{A}) \subseteq [L(\mathcal{B})]_{\sigma}$.*

Proof. Let $w_{\mathcal{A}} = a_0 a_1 a_2 \cdots \in L(\mathcal{A})$ be arbitrary. Then SPOILER can play such that $\rho_{\mathcal{A}}$ is an accepting run over $w_{\mathcal{A}}$. Since DUPLICATOR has a winning strategy, she can play in such a way that also $\rho_{\mathcal{B}}$ is an accepting run and no letter remains in a buffer for ever. Now let $1 \leq i \leq k$. Then $\pi_i(w_{\mathcal{A}}) \in \Sigma_i^{\infty}$ is the sequence of letters that SPOILER writes into the buffer i during the play. Since DUPLICATOR can only execute letters that are available at the corresponding buffers, the word $\pi_i(w_{\mathcal{B}})$ is a prefix of $\pi_i(w_{\mathcal{A}})$. If it is a proper prefix, then DUPLICATOR failed to read all letters written into buffer i. As DUPLICATOR plays according to her winning strategy, this is not the case. Hence $\pi_i(w_{\mathcal{A}}) = \pi_i(w_{\mathcal{B}})$. Since this holds for all $i \in [k]$, we have $w_{\mathcal{A}} \sim_{\sigma} w_{\mathcal{B}}$ and therefore $L(\mathcal{A}) \subseteq [L(\mathcal{B})]_{\sigma}$. □

This yields, together with the following observation, a sound (but not necessarily complete) approximation procedure for trace language inclusion problems using bounded buffers.

Theorem 4. *Uniformly in the trace alphabet $\sigma = (\Sigma_i)_{i \in [k]}$ and the capacity function $\kappa \colon [k] \to \mathbb{N}$, the relation $\sqsubseteq_{\sigma}^{\kappa}$ is decidable on automata with m and n states, resp., in time $O((k+1) \cdot (mn|\Sigma|^{r+k}(k+1))^{2.5})$ where $r = \kappa(1) + \ldots + \kappa(k)$.*

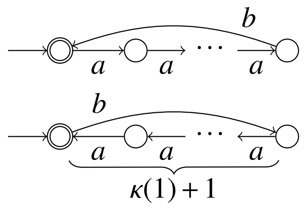
If we fix k and the capacity function κ, then this time bound reduces to a polynomial in $mn|\Sigma|$, i.e., in the size of the automata $\mathcal{A}$ and $\mathcal{B}$.

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be automata with m and n states, resp. Then $\mathcal{G}^{\kappa}_{\sigma}(\mathcal{A},\mathcal{B})$ can be understood as a game whose positions consist of a configuration and an element of $\{0,1,\dots,k\}$ to store one of the buffers DUPLICATOR used in her last move (0 stands for "DUPLICATOR skiped her move"), i.e., of finite size $\leq m \cdot n \cdot \prod_{i=1}^{k} |\Sigma_i|^{\kappa(i)+1} \cdot (k+1) \leq mn \cdot |\Sigma|^{r+k} \cdot (k+1)$. Its winning condition is a strong fairness condition ("*if* SPOILER visits final states infinitely often *then* so does DUPLICATOR") together with k Büchi-conditions ("infinitely often, DUPLICATOR reads some letter from buffer i or buffer i is empty"). By [4], such games can be solved in time $O((k+1)\cdot(mn|\Sigma|^{r+k}(k+1))^{2.5})$. □

Multi-buffer simulations form a hierarchy in the sense that DUPLICATOR's power strictly grows with the buffer capacities.

Theorem 5. *Let $\sigma = (\Sigma_i)_{i\in[k]}$ be a trace alphabet and let κ,κ' be capacity functions for k buffers.*
If $\kappa(i) \leq \kappa'(i)$ for all $i \in [k]$, then $\sqsubseteq^{\kappa}_{\sigma} \subseteq \sqsubseteq^{\kappa'}_{\sigma}$.
Moreover, if there are $a \in \Sigma_i$ and $b \in \Sigma$ with $\sigma(a) \cap \sigma(b) = \emptyset$ and $\kappa(i) < \kappa'(i)$, then there are automata $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \not\sqsubseteq^{\kappa}_{\sigma} \mathcal{B}$ but $\mathcal{A} \sqsubseteq^{\kappa'}_{\sigma} \mathcal{B}$.

Proof. We immediately get $\sqsubseteq^{\kappa}_{\sigma} \subseteq \sqsubseteq^{\kappa'}_{\sigma}$ for $\kappa \leq \kappa'$ since any winning strategy for DUPLICATOR in $\mathcal{G}^{\kappa}_{\sigma}(\mathcal{A},\mathcal{B})$ is also a winning strategy for her in $\mathcal{G}^{\kappa'}_{\sigma}(\mathcal{A},\mathcal{B})$.



For the strictness part suppose w.l.o.g. $a \in \Sigma_1$, $b \in \Sigma_2$, $\sigma(a) \cap \sigma(b) = \emptyset$, and $\kappa(1) < \kappa'(1)$. Then consider these two NBA $\mathcal{A}$ (top) and $\mathcal{B}$ (below) over Σ. DUPLICATOR wins the game $\mathcal{G}^{\kappa'}_{\sigma}(\mathcal{A},\mathcal{B})$ by simply choosing $ba^{\kappa(1)+1}$ every $\kappa(1)+1$ rounds (and skipping in the other rounds). SPOILER wins the game $\mathcal{G}^{\kappa}_{\sigma}(\mathcal{A},\mathcal{B})$ choosing a in the first $\kappa(1)+1$ rounds such that DUPLICATOR is forced to skip in the first $\kappa(1)+1$ rounds (since no b is available in the second buffer) which exceeds the capacity of the second buffer. □

Thms. 3, 4 and 5 can be used for an incremental inclusion test: suppose we want to check whether $L(\mathcal{A}) \subseteq [L(\mathcal{B})]_{\sigma}$ holds for the trace alphabet $\sigma = (\Sigma_i)_{i\in[k]}$. First consider κ_0 with $\kappa_0(i) = 0$ for all $i \in [k]$. If $\mathcal{A} \sqsubseteq^{\kappa_0}_{\sigma} \mathcal{B}$, then $L(\mathcal{A}) \subseteq L(\mathcal{B}) \subseteq [L(\mathcal{B})]_{\sigma}$. If this is not the case, chose κ_1 with $\kappa_0(i) \leq \kappa_1(i)$ for all i and $\kappa_0(i) < \kappa_1(i)$ for some i. If $\mathcal{A} \sqsubseteq^{\kappa_1}_{\sigma} \mathcal{B}$, then $L(\mathcal{A}) \subseteq [L(\mathcal{B})]_{\sigma}$. If, again, this fails, then extend the buffer capacities to some κ_2 etc. Sect. 5 analyses completeness of this procedure, i.e. the possibility for this to prove trace language non-inclusion.

4 Undecidability

It is not hard to show that multi-buffer simulation is in general undecidable by a reduction from Post's Correspondence Problem (PCP) adapting the argument used for the reachability problem for communicating finite state machines [2, 3] and yielding Π^0_1-hardness of $\sqsubseteq^{(\omega,\omega)}_{(\{a,b\},\{c,d\})}$. There is a variant of PCP called ωPCP(REG) that is known to be Σ^1_1-complete [10]. It asks for the existence of an infinite solution word that additionally belongs to some ω-regular language. It is not difficult to adjust the reduction to the $\sqsubseteq^{(\omega,\omega)}_{(\{a,b\},\{c,d\})}$-problem such that SPOILER's accepting runs correspond to valid solutions. This would yield Π^1_1-hardness of $\sqsubseteq^{(\omega,\omega)}_{(\{a,b\},\{c,d\})}$.

We do not give details of this reduction here because it is still possible to strengthen the undecidability result in two ways: (1) we will show that a single unbounded buffer suffices for undecidability. Note that $\sqsubseteq_{\Sigma}^{(\omega)}$ is decidable in EXPTIME [14]. Hence, the question is how many additional bounded buffers are needed to establish undecidability. We provide a tight result in this respect showing that $\sqsubseteq_{(\{a,b\},\{c\})}^{(\omega,0)}$ is undecidable already, i.e. the addition of a minimal number of buffers of minimal capacity (contrary to this, $\sqsubseteq_{(\{a\},\{c,d\})}^{(\omega,0)}$ is shown to be decidable). (2) We will show that the level of undecidability is genuinely higher than Π_1^1 by considering the problem of solving a Büchi game played on a recursive game graph.

A *recursive Büchi game* (RBG) is a graph $G = (V, E, Own, Fin, v_I)$ such that V is a decidable set of nodes, Own and Fin are decidable subsets of V, v_I is a designated starting node, and $E \subseteq V \times V$ is a decidable set of edges. The game is played between players 0 and 1 starting in $v_0 := v_I$. Whenever it reaches a node v_i and $v_i \in Own$ then player 0 chooses $v_{i+1} \in V$ such that $(v_i, v_{i+1}) \in E$ and the play continues with v_{i+1}. Otherwise player 1 chooses such a node v_{i+1}.

A player wins a play if the opponent is unable to choose a successor node. Moreover, player 0 wins an infinite play $v_0, v_1, \ldots$ if there are infinitely many i such that $v_i \in Fin$. The *recursive Büchi game problem* is to decide, given such a game represented using Turing machines, whether or not player 0 has a winning strategy for this game.

The existence of a winning strategy for player 0 in an RBG is a typical Σ_2^1-statement ("*there exists* a strategy for player 0 *such that all* plays $(v_i)_{i\geq 0}$ conforming to this strategy satisfy $\forall n \in \mathbb{N}\, \exists m \in \mathbb{N}\colon v_{m+n} \in Fin$"), i.e., the RBG problem belongs to Σ_2^1. By determinacy of Borel (and therefore of Büchi) games [16], the existence of a winning strategy for player 0 is equivalent to the non-existence of a winning strategy for player 1 (i.e., to "*for all* strategies of player 1 *there exists* a play conforming to this strategy satisfying $\forall n \in \mathbb{N}\, \exists m \in \mathbb{N}\colon v_{m+n} \in Fin$"). Hence the RBG problem also belongs to Π_2^1 and therefore to $\Sigma_2^1 \cap \Pi_2^1$. This class does not contain any complete problems [18, Thm. 16.1.X], but we can show the following lower bound for the RBG problem.

Theorem 6. *The recursive Büchi game problem is hard for the class $B\Sigma_1^1$ of all Boolean combinations of problems from Σ_1^1.*

Proof. To see this, recall that the set of (pairs of Turing machines accepting the nodes and edges of) recursive trees with an infinite branch is Σ_1^1-hard [15]. It follows that the class of tuples $(S_i, T_i)_{1\leq i\leq n}$ of recursive trees such that, for some $1 \leq i \leq n$, the tree S_i has an infinite branch while T_i does not, is complete for the class $B\Sigma_1^1$. We reduce this problem to the recursive Büchi game problem. So let $(S_i, T_i)_{1\leq i\leq n}$ be a tuple of recursive trees. We build a Büchi game as follows: First, to any tree S_i, we add a node g_i together with edges from all nodes (including g_i itself) to g_i. The set Own equals the set of nodes of S_i plus this additional node g_i. Next, we replace every edge by a path of length 2 (i.e., with two edges). The set Fin of winning nodes are the original nodes from S_i. Starting in v_0, player 0 has a winning strategy of this Büchi game G_i iff the tree S_i contains an infinite path.

Similarly, to any tree T_i, we add a node h_i together with edges from all nodes (including h_i itself) to h_i. Next, we replace every edge by a path of length 2. The set Own consists of these new nodes. The node h_i and the unique successor node h_i' (that originates from the replacement of the edge (h_i, h_i) by a path of length 2) are the only winning nodes from Fin. Starting in the root of T_i, player 0 has a winning strategy in this Büchi game H_i iff T_i does not contain any infinite path. Note that once a play enters a winning node it will continue with winning nodes *ad infinitum*.

For any i with $1 \leq i \leq n$, we construct the direct product of the two games G_i and H_i described above: Nodes are of the form (g, h) where g is a node from G_i and h a node from H_i with $g \in Own_{G_i} \iff h \in Own_{G_i}$. The set Own equals $Own_{G_i} \times Own_{H_i}$. There is an edge from (g, h) to (g', h') iff there are edges from g to g' in G_i and from h to h' in H_i. Finally, a node (g,h) belongs to Fin iff both g and h are winning. Clearly, any play in this game GH_i "consists" of two plays in G_i and in H_i, resp. Since a play in S_i cannot leave the set of winning nodes, a play in this game is won by player 0 if both component plays are won by player 0 in G_i and in H_i, resp. Hence player 0 has a winning strategy iff S_i contains an infinite branch while T_i does not.

Finally, we consider the disjoint union of all the games GH_i and add a node $v_I \in Own$. In addition, we add edges from v_I to the starting nodes of all the games GH_i. Now it is rather obvious that player 0 wins this game iff it wins one of the games GH_i and therefore iff, for some $1 \le i \le n$, the tree S_i contains an infinite branch and T_i does not.

From Turing machines that describe the trees S_i and T_i, we can construct Turing machines that describe this game. Hence, we reduced a $B\Sigma^1_1$-complete problem to the RBG problem. □

The rest of this section is devoted to showing that $\sqsubseteq^{(\omega,0)}_{(\{a,b\},\{c,d\})}$ is computationally at least as difficult as solving general RBGs. We present a reduction from the RBG problem to $\sqsubseteq^{(\omega,0)}_{(\{a,b\},\{c,d\})}$. Let G be a RBG. Using standard encoding tricks we can assume that its node set is $\{0,1\}^+$, $Own = 0\{0,1\}^*$, the initial node is 1, and $Fin = \{0,1\}^*1$. The edge relation of G is decided by a deterministic Turing Machine $\mathcal{M}$ with state set Q, tape alphabet Γ, and transition function $\delta\colon Q\times\Gamma \to Q\times\Gamma\times\{-1,0,1\}$. W.l.o.g., we can assume that $\mathcal{M}$ has designated initial / accepting / rejecting states init / acc / rej and that the tape alphabet Γ equals $\{0,1,\#,\triangleright,\triangleleft\}$ including a division symbol # and two end-of-tape markers $\triangleleft$ and $\triangleright$. Apart from the usual assumption that $\mathcal{M}$ uses the end-of-tape markers sensibly, we presume the following.

- There are two designated states acc and rej that the machine uses to signal acceptance and rejection.
- When started in the configuration $\triangleright w\#v\,\mathsf{init}\triangleleft$ with $v,w \in \{0,1\}^+$, the machine eventually halts in $\triangleright w\,\mathsf{acc}\triangleleft$ if w is a successor of v; otherwise it halts in $\triangleright w\,\mathsf{rej}\triangleleft$. Thus, we assume it to reproduce the name of the node that it checked for being a successor node to v. This helps a subsequent computation to be set up. Also note that we assume the machine's tape to be infinite to the left and that it starts reading its input from the right. This is purely done for presentational purposes since it better matches the use of buffers in the constructed multi-buffer games.

In order to ease the presentation we derive a function $\hat{\delta} : (\Gamma\cup Q)^4 \to (\Gamma\cup Q)^{\le 5}$ from the transition function δ such that, for any configuration $\triangleright a_1a_2\dots a_k\triangleleft$, the unique successor configuration equals

$$\hat{\delta}(\triangleright,a_1,a_2,a_3)\,\hat{\delta}(a_1,a_2,a_3,a_4)\,\hat{\delta}(a_2,a_3,a_4,a_5)\dots\hat{\delta}(a_{k-2},a_{k-1},a_k,\triangleleft)\,.$$

If $\triangleright,\triangleleft \notin \{b_1,b_2,b_3,b_4\}$, then we have $|\hat{\delta}(b_1,b_2,b_3,b_4)| = 1$, $|\hat{\delta}(\triangleright,b_1,b_2,b_3)| \in \{2,3\}$, $|\hat{\delta}(b_1,b_2,b_3,\triangleleft)| \in \{3,4\}$, and $|\hat{\delta}(\triangleright,b_1,b_2,\triangleleft)| \in \{4,5\}$.

The construction of two automata $\mathcal{A}$ and $\mathcal{B}$ from the RBG G hinges on a simple correspondence between winning strategies in G and those in $\mathcal{G}^{(\omega,0)}_{\sigma}(\mathcal{A},\mathcal{B})$: Spoiler and Duplicator simulate the RBG by players 1 and 0, respectively. The ω-buffer is used to name current nodes of the RBG. Its alphabet contains all symbols used to form configurations: $\Sigma_1 := \Gamma\cup Q$. The alphabet of letters that can be put into the capacity-0 buffer contains a special new symbol and a copy of every Σ_1-symbol: $\Sigma_2 := \{c\}\cup\{c_x \mid x \in \Sigma_1\}$.

We need to show how three aspects of the simulation can be realised:

1. The choice of a successor node by player 1 in G. This is easy since player 1 is simulated by Spoiler. It is easy to construct a $\mathcal{A}_{\mathsf{chs}}$ that allows Spoiler to choose a $v \in \{0,1\}^+$ and put $\triangleright v$ into

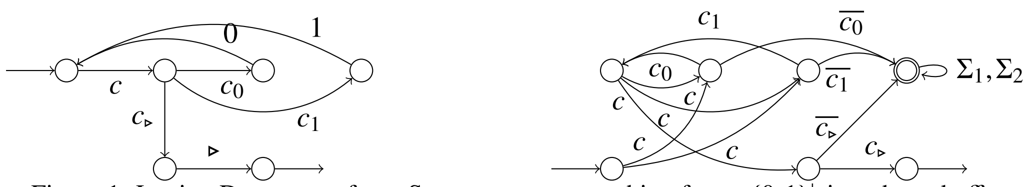
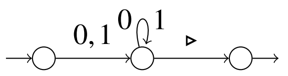


Figure 1: Letting Duplicator force Spoiler to put something from $\triangleright\{0,1\}^+$ into the ω-buffer.

the buffer. The automaton is shown on the right. It has two states marked with incoming and outgoing edges. These are used to indicate in which state a play should begin and where it should end according to the specification in these three cases. Later the constructed automata will be plugged together by merging such marked states forming NBA with infinite runs. We use final states at this point only in special sinks that make one of the players win, i.e. make the opponent lose immediately.



2. The choice of a successor node by player 0 in G. This is trickier, because we need to make Duplicator name a new node but it is only Spoiler who puts letters into the buffers. We will show in Lemma 7 below how the capacity-0 buffer can be used in order for Duplicator to force Spoiler to produce a certain content for the ω-buffer.

3. The check that a newly chosen node is indeed a successor of the current node. We make Spoiler produce a sequence of Turing machine configurations in the ω-buffer and Duplicator check that they form an accepting computation. This is where the assumption of $\mathcal{M}$ being deterministic is needed because it forces Spoiler to produce an accepting computation if one exists. Lemma 8 below shows how this can be done.

Lemma 7. *There are $\mathcal{A}_{\mathsf{frc}}$ and $\mathcal{B}_{\mathsf{frc}}$ with the following properties. Suppose the game $\mathcal{G}^{(\omega,0)}_{(\Sigma_1,\Sigma_2)}(\mathcal{A}_{\mathsf{frc}},\mathcal{B}_{\mathsf{frc}})$ is played with the initial content of the ω-buffer being $\#v\,\mathsf{init}\triangleleft$. For every $w \in \{0,1\}^+$,* Duplicator *has a strategy in the game $\mathcal{G}^{(\omega,0)}_{(\Sigma_1,\Sigma_2)}(\mathcal{A}_{\mathsf{frc}},\mathcal{B}_{\mathsf{frc}})$ to reach a configuration in which the buffer content is $\triangleright w\#v\,\mathsf{init}\triangleleft$.*

Proof. $\mathcal{A}_{\mathsf{frc}}$ is shown on the left of Fig. 1. $\mathcal{B}_{\mathsf{frc}}$ is shown on the right using the abbreviations $\overline{c_a} = \Sigma_2 \setminus \{c_a\}$.

Suppose the two players play on these automata starting in the states marked with incoming edges. Spoiler must open the game by playing c, and Duplicator can respond to this synchronisation move going to a state that has exactly one outgoing edge that does not lead to the accepting state, labeled with either c_0 or c_1. Spoiler is now forced to play this c_i for otherwise Duplicator will win by moving to the accepting state. In repsonse, Duplicator moves to the top left state and then Spoiler puts i into the ω-buffer. So, effectively, Duplicator has forced him to put $i \in \{0,1\}$ into the buffer with her choice in response to the c-move and the game proceeds with Spoiler in the initial state and Duplicator in the top left state. Note that here, the situation is similar, the only difference is that Duplicator now has the choice to go to three states as opposed to two before. If Duplicator (in repsonse to Spoiler's opening c-move) goes to his third option, then Spoiler is forced to put $\triangleright$ into the buffer and the play has reached the states marked with outgoing edges, and the content of the ω-buffer is of the form $\triangleright w\#v\,\mathsf{init}\triangleleft$ with a $w \in \{0,1\}^+$ chosen by Duplicator, if it was $\#v\,\mathsf{init}\triangleleft$ at the beginning. □

Lemma 8. *There are $\mathcal{A}_{\mathsf{chk}}$ and $\mathcal{B}_{\mathsf{chk}}$ with the following properties. Suppose the game $\mathcal{G}^{(\omega,0)}_{(\Sigma_1,\Sigma_2)}(\mathcal{A}_{\mathsf{chk}},\mathcal{B}_{\mathsf{chk}})$ is played on these automata, and the content of the ω-buffer is $C = \triangleright w\#v\,\mathsf{init}\triangleleft$. Then both players have a strategy to reach a configuration in which the buffer contains $\#w\,\mathsf{init}\triangleleft$ without losing in the meantime, iff $\mathcal{M}$ reaches an accepting configuration when started in C.*

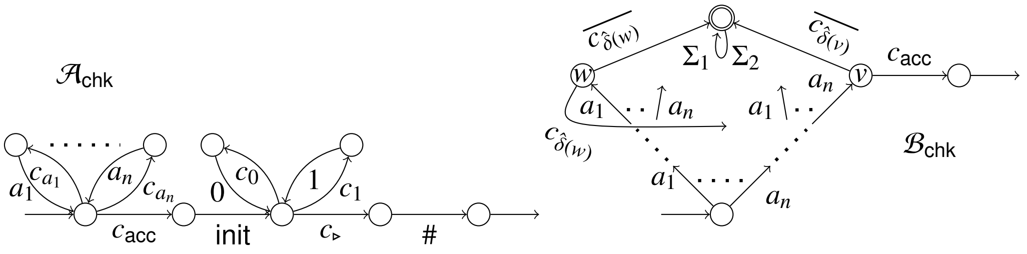


Figure 2: Two automata used to simulate computations of the Turing machine $\mathcal{M}$.

Proof. The automaton $\mathcal{A}_{\mathsf{chk}}$ is shown in Fig. 2 on the left, assuming that $\Sigma_1 \setminus \{\mathsf{acc}, \triangleleft, \triangleright, \#\} = \{a_1, \dots, a_{n'}\}$, $n = n' - 3$, $a_{n-2} = \triangleleft$, $a_{n-1} = \triangleright$, $a_n = \#$. Its structure forces Spoiler to do the following: if he wants to put a symbol from Σ_1 into the ω-buffer then he first has to announce this by playing the corresponding Σ_2-copy. This tells Duplicator immediately what the next symbol in the ω-buffer will be because of this synchronisation via the 0-buffer. Moreover, in order for Spoiler to form a run that passed through this automaton infinitely often and for this run to be accepting, Spoiler has to eventually announce via c_{acc} that he would put acc into the ω-buffer. However, at this point he actually puts init in instead and afterwards can only put in letters from Γ. This way he immediately sets up the buffer for the next simulation; remember that $\mathcal{M}$ is assumed to halt in $\triangleright w \,\mathsf{acc}\triangleleft$ when started in $\triangleright w\#v\triangleleft$. The same trick of announcing a symbol from a final configuration but putting a different one into the ω-buffer is used to turn the end-marker $\triangleright$ into # in order to set up the buffer for the start of the next simulation.

$\mathcal{B}_{\mathsf{chk}}$ is more difficult to depict. It is sketched on the right of Fig. 2. Its initial state is followed by a tree of depth 4 that is used to read words of the form $a_1a_2a_3a_4$ from the ω-buffer. This is used by Duplicator to remember the first 4 symbols from the ω-buffer which is supposed to be the beginning of a configuration of $\mathcal{M}$. Now she starts to accept synchronisation letters given by Spoiler who has begun to construct the next configuration. Remember that he can only play a synchronisation action c_a if he puts a into the ω-buffer right away. This way Duplicator can control that he does indeed construct the valid successor configuration.

Each state at depth 4 of this tree that can be reached by reading the word $w = a_1a_2a_3a_4 \in (\Sigma_1)^4$ has two successors: with $c_{\hat\delta(w)}$ it can reach the state corresponding to reading $a_2a_3a_4$ one level below in the tree. This is used when Spoiler correctly chooses the next symbol of the unique successor configuration and transmits this through $c_{\hat\delta(w)}$. This is shown on the leftmost state of the tree structure. The rightmost state corresponding to, say v, shows an exception: if $\hat\delta(v) = \mathsf{acc}$ then Spoiler is about to produce the last configuration of $\mathcal{M}$'s computation, then Duplicator can move to the right and finish the simulation. The ω-buffer is then set up for the next simulation already since Spoiler puts init instead of acc into the ω-buffer.

The other successor of the states corresponding to reading $w \in (\Sigma_1)^4$ from the ω-buffer is reached with $\Sigma_2 \setminus \{c_{\hat\delta(w)}\}$, here abbreviated as $\overline{c_{\hat\delta(w)}}$. This corresponds to Spoiler producing a symbol that is not the next one in the unique successor configuration, and this takes Duplicator to a state that makes Spoiler lose. Formally, $\mathcal{B}_{\mathsf{chk}}$ has states $(\Sigma_1)^{\le 4} \cup \{\mathsf{sink}, \mathsf{done}\}$ and the following transitions with initial state ε.

$w \xrightarrow{a} wa$, for $w \in (\Sigma_1)^{\le 3}, a \in \Sigma_1$ $\quad$ $w \xrightarrow{c_{\mathsf{acc}}} \mathsf{done}$, for $w \in (\Sigma_1)^4$ $\quad$ $aw \xrightarrow{c_{\hat\delta(aw)}} w$, for $w \in (\Sigma_1)^3$

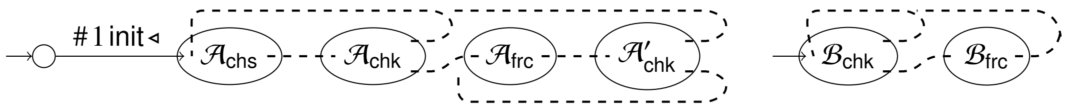
$w \xrightarrow{c} \mathsf{sink}$, for $w \in (\Sigma_1)^4, c \notin \{c_{\hat\delta(w)}, c_{\mathsf{acc}}\}$ $\quad$ $\mathsf{sink} \xrightarrow{x} \mathsf{sink}$, for every $x \in \Sigma_1 \cup \Sigma_2$ □

Theorem 9. *Given a RBG G, one can construct two NBA $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \sqsubseteq^{(\omega,0)}_{(\Sigma_1,\Sigma_2)} \mathcal{B}$ iff player 0 has a winning strategy for G.*

Proof. We construct $\mathcal{A}$ and $\mathcal{B}$ by plugging (variants of) $\mathcal{A}_{\mathsf{chs}}$, $\mathcal{A}_{\mathsf{frc}}$, $\mathcal{A}_{\mathsf{chk}}$, $\mathcal{B}_{\mathsf{frc}}$ and $\mathcal{B}_{\mathsf{chk}}$ together as follows. First of all, we modify $\mathcal{A}_{\mathsf{chk}}$ and $\mathcal{B}_{\mathsf{chk}}$ such that they remember if the buffer content that is produced in the end according to Lemma 8 is $\#1w\mathsf{init}\triangleleft$ or $\#0w\mathsf{init}\triangleleft$ for some $w \in \{0,1\}^*$. Thus, they have two different states each that would be marked with outgoing edges. Intuitively, they distinguish the case in which player 0, resp. player 1 is the owner of the current RBG node and therefore needs to perform a choice next.

Moreover, we use an automaton $\mathcal{A}'_{\mathsf{chk}}$ that is obtained from $\mathcal{A}_{\mathsf{chk}}$ by appending $\xrightarrow{c_{\mathsf{rej}}} \circledcirc \circlearrowleft \mathsf{rej}$ to its initial state. Remember that this automaton is used by Spoiler to produce a computation of the Turing machine $\mathcal{M}$ which is checked for being valid and accepting by Duplicator with $\mathcal{B}_{\mathsf{chk}}$. With $\mathcal{A}_{\mathsf{chk}}$, Spoiler has a strategy not to lose if in the game with initial buffer content $\triangleright w\#v\mathsf{init}\triangleleft$ if w is a valid successor of node v. Thus, it can be used to verify Spoiler's choices of such a successor node that were made with $\mathcal{A}_{\mathsf{chs}}$. However, if Duplicator proposed w instead, and it is not a valid successor of v in G, then Duplicator should lose. This happens using $\mathcal{A}'_{\mathsf{chk}}$: the simulation of $\mathcal{M}$'s computation will ultimately reach a rejecting configuration which allows Spoiler to move to the accepting sink which Duplicator cannot match.

The following picture shows how $\mathcal{A}$ (left) and $\mathcal{B}$ (right) are obtained. A dashed line is drawn in order to indicate that outgoing states (on the right in the automata) are merged with incoming states (on the left). For those automata that have two outgoing states we use the convention that the upper one is used when the next chosen node belongs to player 1 and the lower one otherwise.



$\#1\,\mathsf{init}\triangleleft$
$\mathcal{A}_{\mathsf{chs}}$
$\mathcal{A}_{\mathsf{chk}}$
$\mathcal{A}_{\mathsf{frc}}$
$\mathcal{A}'_{\mathsf{chk}}$
$\mathcal{B}_{\mathsf{chk}}$
$\mathcal{B}_{\mathsf{frc}}$

The part at the beginning in $\mathcal{A}$ ensures that the ω-buffer is filled with $\#1\triangleleft$, i.e. the initial game node which is owned by Spoiler. He then uses $\mathcal{A}_{\mathsf{chs}}$ to choose a successor, leading to a buffer content of the form $\triangleright w\#1\,\mathsf{init}\triangleleft$. The play then proceeds in $\mathcal{A}_{\mathsf{chk}}$ which requires Duplicator to make moves in $\mathcal{B}_{\mathsf{chk}}$, finally leading to the buffer content $\#w\mathsf{init}\triangleleft$ according to Lemma 8. Depending on whether w starts with 1 or 0, the corresponding player makes choices to fill the buffer to a content of $\triangleright v\#w\mathsf{init}\triangleleft$, either Spoiler using $\mathcal{A}_{\mathsf{chs}}$ or Duplicator using $\mathcal{B}_{\mathsf{frc}}$ with Spoiler executing her wishes in $\mathcal{A}_{\mathsf{frc}}$, according to Lemma 7.

Finally, we need to make sure that Duplicator wins iff the underlying play in G visits states of the form $w1$ infinitely often. We make the last states of $\mathcal{A}_{\mathsf{chs}}$ and $\mathcal{A}_{\mathsf{frc}}$ final. Then any run of Spoiler in which he produces finite simulations of the Turing machine and performs choices of finite nodes in the RBG, is accepting. Hence, we need to give Duplicator the ability to answer with an accepting run if it corresponds to a play in the RBG that was winning for her because it visits infinitely many states of the form $\{0,1\}^*1$. This can easily be done by letting her go through an accepting state in $\mathcal{B}_{\mathsf{chk}}$ only if Spoiler has signalled to her that the last configuration of the simulated computation is of the form $\triangleright w1\,\mathsf{acc}\triangleleft$ which can easily be checked by adding a few more states to this automaton. At last, we also need to give Spoiler the ability to win when Duplicator does not do her part in the simulation process properly. This can occur in $\mathcal{B}_{\mathsf{frc}}$ which Duplicator should use to force Spoiler to put a finite $\triangleright w$ for $w \in \{0,1\}^+$ into the buffer. So we need to make sure that she eventually terminates this forcing process. This is easily done by making the first state in $\mathcal{A}_{\mathsf{frc}}$ accepting for Spoiler. □

Remark 10. Given a RBG G, one can construct two NBA $\mathcal{A}$ and $\mathcal{B}$ over the alphabet $\{a,b,c\}$ such that $\mathcal{A} \sqsubseteq^{(\omega,0)}_{(\{a,b\},\{c\})} \mathcal{B}$ iff player 0 has a winning strategy for G.

Proof. Consider the automaton $\mathcal{A}$ constructed in the proof above. The crucial property is that this automaton can at most perform two *consecutive* transitions labeled in Σ_2. Now enumerate all nonempty words over Σ_2 of length at most 2 as $w_1, w_2, \ldots, w_n$. The automaton $\mathcal{A}'$ is obtained from $\mathcal{A}$ by deleting all transitions labeled in Σ_2 and replacing any path of labeled by a word w_i by a path labeled by c^i. Doing the analogous changes in $\mathcal{B}$, we obtain $\mathcal{B}'$. Now it should be clear that $\mathcal{A}' \sqsubseteq^{(\omega,0)}_{(\Sigma_1,\{c\})} \mathcal{B}'$ holds if and only if $\mathcal{A} \sqsubseteq^{(\omega,0)}_{(\Sigma_1,\Sigma_2)} \mathcal{B}$, i.e., if and only if player 0 wins G. □

Putting this together with the lower bound for solving recursive Büchi games established above, we obtain the following result that is in stark contrast to the EXPTIME-decidability of $\sqsubseteq^{\omega}_{\Sigma}$.

Corollary 11. *The relation* $\sqsubseteq^{(\omega,0)}_{(\{a,b\},\{c\})}$ *is* $B\Sigma^1_1$*-hard.*

Proof. All that is left is a coding of the alphabet Σ_1 by words over the binary alphabet $\{a,b\}$. □

Clearly, the same lower bound holds for any multi-buffer simulation involving at least two buffers of which at least one is unbounded such that $|\Sigma_1 \setminus \Sigma_2| \geq 2$ and $\Sigma_2 \setminus \Sigma_1 \neq \emptyset$.

However, if $|\Sigma_1 \setminus \Sigma_2| = 1$, then the multi-buffer game is decidable.

Theorem 12. *The relation* $\sqsubseteq^{(\omega,0)}_{(\{(a)\},\{c,d\})}$ *is decidable.*

Proof. Let $\mathcal{A} = (Q^{\mathcal{A}},\Sigma,p_I,\delta^{\mathcal{A}},F^{\mathcal{B}})$ and $\mathcal{B} = (Q^{\mathcal{B}},\Sigma,q_I,\delta^{\mathcal{B}},F^{\mathcal{B}})$ be two NBA over the alphabet $\{a,c,d\}$. Consider the following 1-counter automaton: The set of states equals $Q^{\mathcal{A}} \times Q^{\mathcal{B}} \times \{c,d,\varepsilon\} \times \{0,1\}$, and all transitions are ε-transitions. For all transitions $p \xrightarrow{a} p'$ in $\mathcal{A}$, the 1-counter automaton has an ε-transition from $(p,q,\varepsilon,0)$ to $(p',q,\varepsilon,1)$ that increments the counter. For all transitions $p \xrightarrow{c} p'$ in $\mathcal{A}$, the 1-counter automaton has an ε-transition from $(p,q,\varepsilon,0)$ to $(p',q,c,1)$ that leaves the counter unchanged (and similarly for transitions $p \xrightarrow{d} p'$). From any state $(p,q,\varepsilon,1)$, there is an ε-transition to $(p,q,\varepsilon,0)$ that does not change the content of the counter. For all transitions $q \xrightarrow{a} q'$ in $\mathcal{B}$, the 1-counter automaton has ε-transitions from $(p,q,\varepsilon,1)$ to $(p,q',\varepsilon,0)$ and to $(p,q',\varepsilon,1)$ that decrement the counter. For all transitions $q \xrightarrow{c} q'$ in $\mathcal{B}$, the 1-counter automaton has ε-transitions from $(p,q,c,1)$ to $(p,q',\varepsilon,0)$ and to $(p,q',\varepsilon,1)$ that leave the counter unchanged (and similarly for transitions $q \xrightarrow{d} q'$).

The set of configurations of this 1-counter automaton equals $Q^{\mathcal{A}} \times Q^{\mathcal{B}} \times \{c,d,\varepsilon\} \times \{0,1\} \times \mathbb{N}$. We now define a game whose nodes are the configurations: Configurations of the form $(p,q,x,0,n)$ belong to player 0, the other configurations belong to player 1. The moves are given by the transitions of the 1-counter automaton. A finite play is lost by the player that got stuck. An infinite play $(p_i,q_i,x_i,n_i)_{i\geq 0}$ is won by player 1 if $p_i \in F^{\mathcal{A}}$ for only finitely many $i \in \mathbb{N}$ or $q_i \in F^{\mathcal{B}}$ for infinitely many $i \in \mathbb{N}$ and the counter increases only finitely often or decreases infinitely often.

Then player 1 wins this game G iff Duplicator wins the game $\mathcal{G}^{(\omega,0)}_{(\{a\},\{c,d\})}(\mathcal{A},\mathcal{B})$. The existence of a winning strategy of player 1 can be expressed as a MSO-formula talking about the configuration graph of the 1-counter automaton. Since the MSO-theories of 1-counter automata are uniformly decidable [17], the existence of a winning strategy for player II in the game G can be decided. □

5 Completeness

In Section 3, we have shown that if Duplicator wins a multi-buffer game between $\mathcal{A},\mathcal{B}$ over the trace alphabet $\sigma = (\Sigma_i)_{i\in[k]}$, then $[L(\mathcal{A})]_\sigma \subseteq [L(\mathcal{B})]_\sigma$. In this section, we characterize the relation $\sqsubseteq^{\kappa}_{\sigma}$ for unbounded buffers analogously to the one-buffer case [14]. More precisely, we will show that unbounded multi-buffer simulation is equivalent to the existence of a continuous function that maps accepting runs of $\mathcal{A}$ to accepting runs of $\mathcal{B}$ with trace-equivalent words.

Notation. Throughout this section, let $k \in \mathbb{N}$ be fixed, let κ_ω be the capacity function for k buffers with $\kappa_\omega(i) = \omega$ for all $i \in [k]$, and let $\sigma = (\Sigma_i)_{i\in[k]}$ be an arbitrary trace alphabet. Furthermore, we fix two automata $\mathcal{A} = (Q^{\mathcal{A}},\Sigma,p_{\mathsf{I}},\delta^{\mathcal{A}},F^{\mathcal{A}})$ and $\mathcal{B} = (Q^{\mathcal{B}},\Sigma,q_{\mathsf{I}},\delta^{\mathcal{B}},F^{\mathcal{B}})$.

First recall that an infinite run of some NBA $\mathcal{A}$ is an infinite word over $Q^{\mathcal{A}} \cup \Sigma$. We denote the set of runs of $\mathcal{A}$ by $Run(\mathcal{A})$ and the set of accepting runs by $ARun(\mathcal{A})$.

Given a set Δ, the set Δ^{ω} of infinite words over Δ is equipped with the standard structure of a metric space. The distance $d(x,y)$ between two distinct infinite sequences $x_0x_1x_2\dots$ and $y_0y_1y_2\dots$ is $\inf\{2^{-i} \mid x_j = y_j \text{ for all } j < i\}$. Intuitively, two words are "significantly close" if they share a "significantly long" prefix.

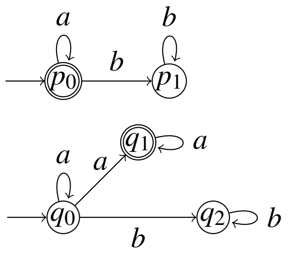
We call a function $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$ *trace-preserving* if for all accepting runs $\rho \in ARun(\mathcal{A})$, the run $f(\rho)$ is accepting and the words of these two runs are trace equivalent. Consequently, $[L(\mathcal{A})]_\sigma \subseteq [L(\mathcal{B})]_\sigma$ iff there exists a trace-preserving function $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$. Throughout this section, we will show that the existence of a continuous and trace-preserving function characterizes $\mathcal{A} \sqsubseteq_\sigma^{\kappa_\omega} \mathcal{B}$.

Lemma 13. *If $\mathcal{A} \sqsubseteq_\sigma^{\kappa_\omega} \mathcal{B}$, then there exists a continuous trace-preserving function $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$.*

Proof. Suppose Duplicator wins $\mathcal{G}_\sigma^{\kappa_\omega}(\mathcal{A},\mathcal{B})$ with some winning strategy θ. We define $f(\rho) = \rho'$ such that ρ,ρ' are the output of $\mathcal{G}_\sigma^{\kappa_\omega}(\mathcal{A},\mathcal{B})$, in which Spoiler plays ρ and Duplicator plays according to θ. The function f is trace-preserving since θ is a winning strategy for Duplicator.

Let $\rho_1 \in ARun(\mathcal{A})$ be an accepting run of $\mathcal{A}$ and let $n \in \mathbb{N}$. Since $f(\rho_1) \in ARun(\mathcal{B})$ is the output of Duplicator's moves according to the winning strategy θ, there is some round m such that Duplicator's output after that round has length at least n. Now let ρ_2 be another run of $\mathcal{A}$ that agrees with ρ_1 in the first m transitions, i.e., with $d(\rho_1,\rho_2) \le 2^{-(2m+3)}$. Then, in the first m rounds, Duplicator does not see any difference between Spoiler's runs ρ_1 and ρ_2. Hence Duplicator (playing according to the strategy θ) makes the same moves. This implies that $f(\rho_1)$ and $f(\rho_2)$ agree in the first n positions, i.e., $d(f(\rho_1), f(\rho_2)) < 2^{-n}$. Thus, we showed that f is continuous. □

Example 14. Under the assumption $\mathcal{A} \sqsubseteq_\sigma^{\kappa_\omega} \mathcal{B}$, of the lemma, there need not be a continuous function $f\colon Run(\mathcal{A}) \to Run(\mathcal{B})$ (let alone a continuous function from $(Q^{\mathcal{A}} \cup \Sigma)^\omega$ to $(Q^{\mathcal{B}} \cup \Sigma)^\omega$) that is trace-preserving and maps accepting runs to accepting runs.



Consider these two NBA $\mathcal{A}$ (above) and $\mathcal{B}$ (below), and the trace alphabet $\sigma = (\{a,b\})$. We have $\mathcal{A} \sqsubseteq_\sigma^{\kappa_\omega} \mathcal{B}$, since Duplicator has the following winning strategy: if, in state q_0 or q_1, Duplicator sees a in the buffer, then she moves to q_1. If, in state q_0 or q_2, Duplicator sees b in the buffer, then she moves to q_2. In all other cases, Duplicator skips her move. This is a winning strategy since the only accepting run $(p_0a)^\omega$ of $\mathcal{A}$ is answered by the accepting run $q_0a(q_1a)^\omega$ of $\mathcal{B}$.

Now let $f\colon Run(\mathcal{A}) \to Run(\mathcal{B})$ be a continuous and trace-preserving function that maps accepting runs of $\mathcal{A}$ to accepting runs of $\mathcal{B}$. Since f is trace-preserving, we get $f((p_0a)^m p_0b(p_1b)^\omega) = (q_0a)^m q_0b(q_2b)^\omega$ for all $m \in \mathbb{N}$. Note that, when m grows, the runs $(p_0a)^m p_0b(p_1b)^\omega$ converge to the run $(p_0a)^\omega$ and their f-images $(q_0a)^m q_0b(q_2b)^\omega$ converge to $(q_0a)^\omega$. Since f is continuous, this implies $f((p_0a)^\omega) = (q_0a)^\omega$, i.e., the accepting run $(p_0a)^\omega$ of $\mathcal{A}$ is mapped to the non-accepting run $(q_0a)^\omega$ of $\mathcal{B}$. Hence, indeed, there is no continuous and trace preserving function $f\colon Run(\mathcal{A}) \to Run(\mathcal{B})$ that maps accepting runs to accepting runs.

We are interested in the reverse direction of Lemma 13 because it shows that continuity is the weakest condition that implies multi-buffer simulation on unbounded buffers. We will use a *delay game* as in [12]. In this game, the winning condition is given by some function $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$, and Duplicator is allowed to form a run without considering any buffers.

Let $\mathcal{A},\mathcal{B}$ be NBAs and $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$ be a function that maps accepting runs of $\mathcal{A}$ to accepting runs of $\mathcal{B}$. The *delay game* $\mathcal{G}_{\mathsf{del}}^{f}(\mathcal{A},\mathcal{B})$ is played between players Spoiler and Duplicator on

$\mathcal{A}$ and $\mathcal{B}$, where in each round each player tries to extend a run. A configuration is a pair of finite runs $(r_\mathcal{A}, r_\mathcal{B})$ on $\mathcal{A}$ and $\mathcal{B}$, respectively (the pair (p_I, q_I) is the initial configuration). For every round $i > 0$, with configuration $(r_\mathcal{A}, r_\mathcal{B})$: Spoiler tries to extend $r_\mathcal{A}$ with one step: $r'_\mathcal{A} := r_\mathcal{A} a p$, and Duplicator tries to extend $r_\mathcal{B}$ with $n \geq 0$ steps: $r'_\mathcal{B} := r_\mathcal{B} b_1 q_1 \dots b_n p_n$. They continue to the next round with the configuration $(r'_\mathcal{A}, r'_\mathcal{B})$.

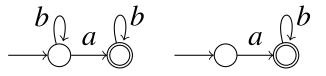
A play builds two runs: an infinite run ρ of $\mathcal{A}$ (chosen by Spoiler) and a finite or infinite run ρ' of $\mathcal{B}$ (chosen by Duplicator). We say that Duplicator wins the play iff ρ is not accepting or $f(\rho) = \rho'$. We write $\mathcal{A} \sqsubseteq^f_\mathsf{del} \mathcal{B}$ as shorthand for "Duplicator has a winning strategy in the delay game $\mathcal{G}^f_\mathsf{del}(\mathcal{A}, \mathcal{B})$".

Note that if $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$ is continuous, then $\mathcal{A} \sqsubseteq^f_\mathsf{del} \mathcal{B}$. The winning strategy for Duplicator is to move properly according to f: on configuration $(r_\mathcal{A}, r_\mathcal{B})$, if there exists $(b, q) \in \Sigma \times Q^\mathcal{B}$ such that for any $\rho \in ARun(\mathcal{A})$ with prefix $r_\mathcal{A}$, $r_\mathcal{B} b q$ is a prefix of $f(\rho)$, then Duplicator extends $r_\mathcal{B}$ to $r'_\mathcal{B} := r_\mathcal{B} b q$. Otherwise, Duplicator skips her move. In this way, Duplicator always forms a run that is the image of f.

Lemma 15. *Let $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$ be a continuous function. Then $\mathcal{A} \sqsubseteq^f_\mathsf{del} \mathcal{B}$.*

Suppose $k = 1$, i.e., we are in the one-buffer case and let $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$ be continuous and trace-preserving. Then, as we saw above, Duplicator has a winning strategy θ in the delay game $\mathcal{G}^f_\mathsf{del}(\mathcal{A}, \mathcal{B})$. In [14], it was shown that this strategy is also a winning strategy in the buffer game $\mathcal{G}^{\kappa_\omega}_{(\Sigma)}(\mathcal{A}, \mathcal{B})$. The following example shows that this is not the case in the multi-buffer game: θ may tell Duplicator to output a letter that has not already been played by Spoiler.

Example 16. Consider the following two NBA $\mathcal{A}$ (left) and $\mathcal{B}$ (right) and the trace alphabet $\sigma = (\{a\}, \{b\})$. Take the continuous function $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$ that maps all infinite runs of $\mathcal{A}$ to the unique infinite run of $\mathcal{B}$. Duplicator wins $\mathcal{G}^f_\mathsf{del}(\mathcal{A}, \mathcal{B})$ with strategy θ: in the first round she forms $q_0 a q_1$, in the second round she forms $q_0 a q_1 b q_1$, and so on. However, Duplicator cannot use θ to win $\mathcal{G}^{\kappa_\omega}_\sigma(\mathcal{A}, \mathcal{B})$, since Spoiler may not output a in the first round. Nevertheless, Duplicator wins $\mathcal{G}^{\kappa_\omega}_\sigma(\mathcal{A}, \mathcal{B})$ by simply waiting for the first a in buffer 1 and then emptying both buffers and, from then on, follows the strategy θ.



More generally, we can derive a winning strategy for Duplicator on $\mathcal{G}^{\kappa_\omega}_\sigma(\mathcal{A}, \mathcal{B})$, from some winning strategy θ on $\mathcal{G}^f_\mathsf{del}(\mathcal{A}, \mathcal{B})$, as stated in the following lemma.

Lemma 17. *Let $f\colon ARun(\mathcal{A}) \to ARun(\mathcal{B})$ be a continuous and trace-preserving function. Then $\mathcal{A} \sqsubseteq^{\kappa_\omega}_\sigma \mathcal{B}$.*

Proof. We naturally extend the projection functions $\pi_i\colon \Sigma^\infty \to \Sigma_i{}^\infty$ to $\pi_i\colon (Q \times \Sigma \times Q)^\infty \to \Sigma_i{}^\infty$ by setting $\pi_i(p, a, q) = \pi_i(a)$.

By Lemma 15, Duplicator has a winning strategy θ in the delay game $\mathcal{G}^f_\mathsf{del}(\mathcal{A}, \mathcal{B})$. To win the multi-buffer game $\mathcal{G}^\kappa_\sigma(\mathcal{A}, \mathcal{B})$, Duplicator tries to mimic this strategy θ. Her problem is that in some situation, θ tells her to play some transition, but the letter of this transition is not available in the buffers.

We next describe the modified strategy θ' of Duplicator: Suppose Spoiler has played the finite run $r_\mathcal{A}$ in the delay game $\mathcal{G}^f_\mathsf{del}(\mathcal{A}, \mathcal{B})$. Let r be Duplicator's answer according to her strategy θ. Let $r_\mathcal{B}$ be the maximal prefix of r such that, for all $i \in [k]$, the word $\pi_i(r_\mathcal{B})$ is a prefix of $\pi_i(r_\mathcal{A})$. Then Duplicator's strategy θ' shall ensure that she outputs this run $r_\mathcal{B}$ in response to Spoiler playing $r_\mathcal{A}$.

We first verify that Duplicator can play according to this strategy: so suppose Spoiler extends his run $r_\mathcal{A}$ to $r'_\mathcal{A} = r_\mathcal{A} a p$. Then Duplicator's answer r' in the delay game extends the run r. The maximal prefix $r'_\mathcal{B}$ of r' such that $\pi_i(r'_\mathcal{B})$ is a prefix of $\pi_i(r'_\mathcal{A}) = \pi_i(r_\mathcal{A})\pi_i(a)$ for $i \in [k]$ extends $r_\mathcal{B}$ by some word $x \in (Q^\mathcal{B} \cup \Sigma)^*$. Then, clearly Duplicator can play the difference x between $r_\mathcal{B}$ and $r'_\mathcal{B}$.

$$\begin{tikzcd}
L(\mathcal{A}) \subseteq [L(\mathcal{B})]_\sigma \arrow[r, Leftrightarrow, "\text{by definition of } f"] & \text{ex. trace-preserving function } f\colon \mathit{ARun}(\mathcal{A}) \to \mathit{ARun}(\mathcal{B}) \\
\mathcal{A} \sqsubseteq_\sigma^{(\omega,\omega,\dots,\omega)} \mathcal{B} \arrow[u, Rightarrow, "\text{Thm. 3}"] \arrow[r, Leftrightarrow, "\text{Thm. 18}"] & \text{ex. trace-preserving cont. function } f\colon \mathit{ARun}(\mathcal{A}) \to \mathit{ARun}(\mathcal{B}) \\
\mathcal{A} \sqsubseteq_\sigma^\kappa \mathcal{B} \arrow[u, Rightarrow, "\text{Thm. 5}"] &
\end{tikzcd}$$

Figure 3: Characterising trace inclusion and multi-buffer simulation.

It remains to be shown that θ' is a winning strategy. To this aim, let $\rho_{\mathcal{A}}$ be an accepting run of $\mathcal{A}$. Since θ is winning in the delay game $\mathcal{G}^f_{\mathsf{del}}(\mathcal{A},\mathcal{B})$, the accepting run $\rho = f(\rho_{\mathcal{A}})$ is Duplicator's answer in this game to Spoiler playing $\rho_{\mathcal{A}}$. Let $\rho_{\mathcal{B}}$ be Duplicator's answer according to the strategy θ' in the simulation game $\mathcal{G}^{\kappa_\omega}_\sigma(\mathcal{A},\mathcal{B})$. We show $\rho = \rho_{\mathcal{B}}$: Clearly, by the construction of θ', any finite prefix of $\rho_{\mathcal{B}}$ is a prefix of ρ. Conversely, let $r_{\mathcal{B}}$ be a finite prefix of $\rho_{\mathcal{B}}$. There is a finite prefix $r_{\mathcal{A}}$ of $\rho_{\mathcal{A}}$ such that, once Spoiler has played $r_{\mathcal{A}}$ in the simulation game, Duplicator's answer according to θ' is at least $r_{\mathcal{B}}$. The rules of the simulation game imply $\pi_i(r_{\mathcal{B}}) \le \pi_i(r_{\mathcal{A}})$ for all $i \in [k]$.

Since f is trace-preserving, we get $\pi_i(\rho_{\mathcal{A}}) = \pi_i(\rho)$ for all $i \in [k]$. Hence we have, for all $i \in [k]$, $\pi_i(r_{\mathcal{B}}) \le \pi_i(r_{\mathcal{A}}) \le \pi_i(\rho_{\mathcal{A}}) = \pi_i(\rho)$. It follows that there is a finite prefix r or ρ such that $\pi_i(r_{\mathcal{B}}) \le \pi_i(r)$ for all $i \in [k]$. Since both, $r_{\mathcal{B}}$ and r are prefixes of $\rho_{\mathcal{B}}$, this implies that $r_{\mathcal{B}}$ is a prefix of r and therefore of ρ. Consequently, $\rho_{\mathcal{B}} = \rho = f(\rho_{\mathcal{A}})$. Since f maps the accepting run $\rho_{\mathcal{A}}$ to an accepting run, $\rho_{\mathcal{B}}$ is accepting. Furthermore, since f is trace-preserving, the words of $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{B}}$ are equivalent. Hence θ' is a winning strategy for Duplicator. □

Putting Lemmas 13, 15 and 17 together we obtain the following characterisation of a case in which multi-buffer simulation is complete for trace inclusion, namely that in which there is not only a trace-preserving function between the runs but one that is additionally continuous.

Theorem 18. *Let $\mathcal{A}$, $\mathcal{B}$ be two NBA over the trace alphabet $\sigma = (\Sigma_i)_{i\in[k]}$. We have $\mathcal{A} \sqsubseteq^{\kappa_\omega}_\sigma \mathcal{B}$ if and only if there exists a continuous trace-preserving function $f\colon \mathit{ARun}(\mathcal{A}) \to \mathit{ARun}(\mathcal{B})$.*

6 Conclusion and Further Work

We have defined multi-buffer simulation relations on Büchi automata and analysed them with respect to their usability for inclusion problems between trace languages defined by NBA. Fig. 3 presents a picture of how these concepts are related. There are (at least) three ways for the work presented here to be continued.

1. As can be seen from Fig. 3, the question of whether there is a characterisation of bounded multi-buffer simulation is still open. For single-buffer simulations, a matching criterion is known, namely that of a Lipschitz continuous function between the runs of the automata. However, it is possible to give examples which show that Lipschitz continuity is neither sufficient nor necessary for bounded multi-buffer simulation. We suspect that an additional condition on the looping structure of the automata in terms of the underlying dependency relation needs to be given such that Lipschitz continuity captures bounded multi-buffer simulation.

2. In the whole of this article we have assumed that all the used concepts like automata and games are defined w.r.t. a fixed trace alphabet. It is possible to relax this and study the effect that varying the independence relation has on the results, e.g. whether or not this also induces strict hierarchies w.r.t. expressive power. This could be used to refine the approximation sketched at the end of Sect. 3. This may

not make sense for trace inclusion problems but may yield better approximations for related problems like transducer inclusion which feature a very restricted form of independence on their alphabets.

3. Finally, recall that $\sqsubseteq^{\omega}_{\Sigma}$ is EXPTIME-complete. There is a variant that is "only" PSPACE-complete [14]; it is obtained by requiring Duplicator to either skip turns or flush the entire buffer. It is not clear what the complexity of such a restricted multi-buffer game is. Note that the undecidability proof for $\sqsubseteq^{(\omega,0)}_{(\Sigma_1,\Sigma_2)}$ (Thm. 9) heavily relies on Duplicator's ability to constantly keep some content in the buffer, namely the last configuration of a Turing machine in its simulation.

References

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

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

[3] 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.

[4] K. Chatterjee, W. Dvořák, M. Henzinger & V. Loitzenbauer (2016): *Conditionally Optimal Algorithms for Generalized Büchi Games*. In: *MFCS'16*. To appear.

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

[6] V. Diekert & G. Rozenberg (1995): *The Book of Traces*. World Scientific Publ. Co., doi:10.1142/2563.

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

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

[9] 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.

[10] O. Finkel (2012): *Three Applications to Rational Relations of the High Undecidability of the Infinite Post Correspondence Problem in a Regular ω-Language*. *Int. J. Found. Comp. Sci* 23(7), pp. 1481–1498, doi:10.1142/S0129054112400606.

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

[12] 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.

[13] 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.

[14] 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.

[15] S.C. Kleene (1943): *Recursive predicates and quantifiers*. *Trans. Amer. Math. Soc.* 53, pp. 41–73, doi:10.1090/S0002-9947-1943-0007371-8.

[16] D. A. Martin (1975): *Borel determinacy*. *Ann. Math.* 102, pp. 363–371, doi:10.2307/1971035.

[17] D.E. Muller & P.E. Schupp (1985): *The theory of ends, pushdown automata, and second-order logic*. *Theoretical Computer Science* 37(1), pp. 51–75, doi:10.1016/0304-3975(85)90087-8.

[18] H. Rogers (1987): *Theory of recursive functions and effective computability (Reprint from 1967)*. MIT Press.

[19] J. Sakarovitch (1992): *The "last" decision problem for rational trace languages*. In: *LATIN'92*, LNCS 583, Springer, pp. 460–473, doi:10.1007/BFb0023848.