Multi-buffer simulations: Decidability and complexity

Milka Hutagalung, Norbert Hundeshagen, Dietrich Kuske, Martin Lange, Etienne Lozes

Multi-Buffer Simulations: Decidability and Complexity

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

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

Abstract

Multi-buffer simulation is a refinement of fair simulation between two nondeterministic Büchi automata (NBA). It is characterised by a game in which letters get pushed to and taken from FIFO buffers of bounded or unbounded capacity.

Games with a single buffer approximate the PSPACE-complete language inclusion problem for NBA. With multiple buffers and a fixed mapping of letters to buffers these games approximate the undecidable inclusion problem between Mazurkiewicz trace languages.

We study the decidability and complexity of multi-buffer simulations and obtain the following results: P-completeness for fixed bounded buffers, EXPTIME-completeness in case of a single unbounded buffer and high undecidability (in the analytic hierarchy) with two buffers of which at least one is unbounded. We also consider a variant in which the buffers are kept untouched or flushed and show PSPACE-completeness for the single-buffer case.

Keywords: Büchi automata, simulation games, Mazurkiewicz traces

1. Introduction

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

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

Simulation Games. To reason about simulation relations, one very often characterises them by the existence of winning strategies in two-player games played on the state spaces of two automata with each player moving a pebble along the transitions of their automaton. The game is strictly turn-based with the beginning player, called Spoiler, moving in the automaton to be simulated, and the responding player, called Duplicator moving in the automaton that should simulate the other. Both players construct runs of their automata piece-wise, and it is Duplicator's burden to make her run "correspond" to Spoiler's run in the sense that both are runs over the same ω-word. Moreover, Duplicator's run must be be an accepting one whenever Spoiler's is. This is also known as *fair* simulation. Other more refined winning conditions have been considered, mainly for the purpose of automata minimisation, known as *direct* and *delayed* simulation [9]. Here we are only concerned with cases of simulation games with winning condition as in fair simulation, simply because it is the most difficult case in terms of decidability and complexity concerns.

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

- In *multi-pebble simulation* [8], Duplicator controls several pebbles which allows her to follow several runs of which only one has to be "corresponding" in the above sense. Hence Duplicator can act in foresight of several of Spoiler's later moves.

  Multi-pebble simulation with a fixed number k of pebbles, i.e. k-pebble simulation, is computable in polynomial time for any k [8, Thm. 4].

- *Multi-letter simulation* [4, 17], is equally parametrised by some $k \in \mathbb{N}$. In the characterising game, Spoiler first reveals k steps of his run and Duplicator answers with the same number of steps in her run. Equally, k-letter simulation is computable in polynomial time for any fixed k [17, Thms. 4 and 8]. In fact, it is computable in time linear in the size of the underlying automata – unlike multi-pebble simulations –, and genuinely polynomial in the size of the underlying alphabet only.

- The idea underlying the dynamic multi-letter games is extended to form *buffered simulation games* [18]. Here Spoiler chooses one letter in each round, but Duplicator can store this letter in a FIFO buffer (bounded or unbounded) for as long as the buffer's capacity is not exceeded. When she decides to move her pebble, she consumes the first letter(s) from the buffer, thus delaying the construction of her run in comparison to Spoiler's. One can construct pairs of automata such that Duplicator wins the buffered simulation game with an unbounded buffer but not with any bounded buffer [18, Ex. 2.2].

- Simulation games can be seen as special Gale-Stewart games in which input and output words must be equal. As such, one can also consider

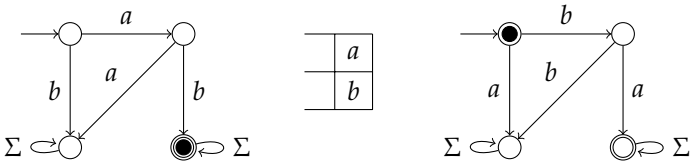
delay games [14] as a form of refined simulation games. Here, one fixes a sequence $(n_i)_{i\geq 0}$ of positive integers and an ω-regular set L of pairs of ω-words as the winning condition. In round i, Spoiler produces n_i letters of his word and Duplicator answers with a single letter of her word. Duplicator wins if the pair of words constructed this way belongs to L.

What distinguishes delay games from buffered simulation is the pre-described relative speed with which Spoiler advances. As a consequence, one obtains: if Duplicator can win then she can win with a fixed delay [14, Thm. 6.4].

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

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

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



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

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

Note that no such commutation in the order of letters in a run can occur when only a single buffer is being used. Single-buffere simulation therefore corresponds to trace inclusion with an empty underlying independence relation which is easily seen to be the usual language inclusion problem between NBA.

Contribution and Organisation of the Paper. This paper contains a complete study of decidability and complexity issues arising with multi-buffer simulation, including the special case of single-buffer simulation. It provides complete proofs of the results that were announced in preliminary work [15, 16, 18].

The paper is organised as follows. In Section 2 we recall the preliminary notions of nondeterministic Büchi automata, trace equivalence, and fair simulation. Section 3 introduces multi-buffer simulation games, both informally and formally as an infinite-duration game.

Section 4 is devoted to upper bounds. We start by considering parity games and simulations of parity games. To prove upper bounds, we then transform multi-buffer simulation games to parity games. Analysing these parity games, we show that multi-buffer simulation can be decided in polynomial time for any fixed number of buffers of fixed bounded capacities. When buffer capacities can be unbounded then multi-buffer simulation does not comprise a game with a finite state-space anymore, hence, decidability cannot be taken for granted. It then hinges on the number of buffers involved in a game. Single-buffer games can be decided in EXPTIME, whereas the general case of several buffers can only be placed in Δ_2^1 in the analytic hierarchy. We also consider a version of buffered simulation that restricts Duplicator's moves, called the *flushing* variant. Such games are simpler to decide: the single-buffer case falls into PSPACE and the multi-buffer case into Π_1^1.

Section 5 shows that the results of Section 4 are essentially tight by presenting matching lower bounds. Note that Δ_2^1 does not have complete problems [21, Theorem 16.1.X]; however we can show that unbounded multi-buffer simulation is hard for $\Sigma_1^1 \cup \Pi_1^1$ (and even more than that), already in the simplest case of one unbounded and one bounded buffer of capacity 0.

Section 6 concludes the paper with remarks on further work.

2. Preliminaries

2.1. Nondeterministic Büchi Automata

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 $\kappa \in \mathbb{N} \cup \{\omega\}$, we write $\Sigma^{\leq \kappa}$ for the set of *finite* words w of length $|w| \leq \kappa$. In particular, $\Sigma^{\leq \omega} = \Sigma^*$ since we restrict to *finite* words. We write $|w|_a$ for a $w \in \Sigma^\infty$ and $a \in \Sigma$ to denote the number of occurrences of the letter a in the word w. For a natural number k, we set $[k] = \{1, 2, \ldots, k\}$.

A *nondeterministic Büchi automaton* or *NBA* is a tuple $\mathcal{A} = (Q, \Sigma, \iota, \delta, F)$ where Q is a finite set of *states*, Σ is an alphabet, $\iota \in Q$ 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*. We demand

the transition functions to be total, i.e. $\delta(q,a) \neq \emptyset$ for every $q \in Q$ and $a \in \Sigma$. However, in order to keep the presentation short, our examples may show NBAs with non-total transition relations. They can easily be extended to total relations by adding one more non-accepting state without effecting the essence of these examples.

For $q \in Q$, we write $\mathcal{A}[q]$ to denote $(Q, \Sigma, q, \delta, F)$, i.e. the NBA that results from $\mathcal{A}$ by taking q as initial state.

A *run* of $\mathcal{A}$ is a finite or infinite sequence $\rho = (q_i, a_i, q_{i+1})_{0 \leq i < \kappa}$ with $q_{i+1} \in \delta(q_i, a_i)$ for all $0 \leq i < \kappa$ (where $\kappa \in \mathbb{N} \cup \{\omega\}$); it is *initial* if $q_0 = \iota$ or $\kappa = 0$. Its *label* is the word $\mathsf{lab}(\rho) = (a_i)_{0 \leq i < \kappa} \in \Sigma^\infty$. If $\kappa \in \mathbb{N} \setminus \{0\}$, then $\mathsf{target}(\rho) = q_\kappa$ is the *target* of the run ρ, if $\kappa = 0$, we set $\mathsf{target}(\rho) = \iota$. The set of finite runs is denoted $\mathsf{FRuns}(\mathcal{A})$, $\mathsf{iFRuns}(\mathcal{A})$ is the set of finite and initial runs.

An infinite run $\rho = (q_i, a_i, q_{i+1})_{i \geq 0}$ is *accepting* if it is initial and $q_i \in F$ for infinitely many $i \geq 0$. We write $\mathsf{ARuns}(\mathcal{A})$ for the set of infinite accepting runs of $\mathcal{A}$ and $L(\mathcal{A}) \subseteq \Sigma^\omega$ for the set of labels of accepting runs.

Let $w \in \Sigma^+$ be some finite and nonempty word.To simplify notation, we write $\delta(p, w)$ for the set of states reachable from $p \in Q$ by some w-labeled run, i.e., $q \in \delta(p, w)$ if there exists some w-labeled run ρ from p with $q = \mathsf{target}(\rho)$. For $w = \varepsilon$, we set $\delta(q, \varepsilon) = \{q\}$. A finite run $\rho = (q_i, a_i, q_{i+1})_{0 \leq i < \kappa}$ *visits some accepting state* if there exists $0 \leq i < \kappa$ with $q_i \in F$ or $\mathsf{target}(\rho) \in F$. Then $\delta_F(p, w)$ is the set of states reachable from $p \in Q$ by some w-labeled run that visits some accepting state, i.e., $r \in \delta_F(p, w)$ if there is some factorisation $w = uv$ and some $q \in F \cap \delta(p, u)$ with $r \in \delta(q, v)$. For $w = \varepsilon$, this yields $\delta_F(q, \varepsilon) = \{q\} \cap F$.

2.2. Trace Languages

We shortly introduce the notions of finite and infinite traces, for a detailed treatment see [5]. Traces generalise words – which can be seen as a sequence of actions performed by a single process – to the setting of several parallel processes. Then the order underlying the performed actions (or occurring letters) is not a total one anymore since some actions may be performed by different processes in parallel and therefore cannot be ordered in time.

Formally, a *trace alphabet* is a triple (Σ, σ, k) where Σ is an alphabet, $k \in \mathbb{N}$ and $\sigma \colon \Sigma \to \mathcal{P}([k]) \setminus \{\emptyset\}$ is a mapping. Intuitively, a letter $a \in \Sigma$ denotes an action that is performed by the set of processes $\sigma(a)$. For $i \in [k]$, let $\Sigma_i = \{a \in \Sigma \mid i \in \sigma(a)\}$ be the alphabet of process i. Furthermore, $\pi_i \colon \Sigma^\infty \to \Sigma_i^\infty$ is the natural projection function that deletes all letters from a word that do not belong to Σ_i.

Two words $u, v \in \Sigma^\infty$ are *σ-equivalent*, denoted $u \sim_\sigma v$, if $\pi_i(u) = \pi_i(v)$ for all $i \in [k]$. The relation $\sim_\sigma$ is called *trace equivalence* and $[u]$ denotes the equivalence class of $u \in \Sigma^*$ wrt. this trace equivalence.

2.3. Fair Simulation

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. The *fair* or *ordinary simulation game* is played between players Spoiler and Duplicator by moving two pebbles across the NBAs' state spaces.

The game starts with Spoiler's pebble placed on p_I and Duplicator's on q_I. In each round with pebbles placed on p_i and q_i,

1. Spoiler first chooses a letter a_i and a state $p_{i+1} \in \delta^{\mathcal{A}}(p_i, a_i)$ and
2. Duplicator responds with a state $q_{i+1} \in \delta^{\mathcal{B}}(q_i, a_i)$.

In an infinite play, Spoiler will have constructed a run ρ of $\mathcal{A}$, and Duplicator will have constructed a run ρ' of $\mathcal{B}$ (note that ρ and ρ' are runs on the same ω-word). Such a play is won by Duplicator if ρ is not an accepting run or ρ' is an accepting run.

We write $\mathcal{A} \sqsubseteq \mathcal{B}$ if Duplicator has a winning strategy for the fair simulation game on $\mathcal{A}$ and $\mathcal{B}$. The following is well-known.

Proposition 1 (e.g. [13]) *Let $\mathcal{A}$ and $\mathcal{B}$ be two NBA. If $\mathcal{A} \sqsubseteq \mathcal{B}$ then $L(\mathcal{A}) \subseteq L(\mathcal{B})$. There are, however, Büchi automata $\mathcal{A}$ and $\mathcal{B}$ such that $L(\mathcal{A}) \subseteq L(\mathcal{B})$ but $\mathcal{A} \not\sqsubseteq \mathcal{B}$.*

2.4. Infinite-Duration Games

An *infinite-duration game* is a tuple $\mathcal{G} = (V, V_0, V_1, E, v_I, \mathcal{W})$ such that

- V is a set of *positions* partitioned into the sets V_p of *positions of Player p* for $p \in \{0, 1\}$,
- $E \subseteq V \times V$ is the set of *moves* such that, for every $v \in V$, there exists $v' \in V$ with $(v, v') \in E$,
- $v_I \in V$ is the *initial position*, and
- the *winning condition* $\mathcal{W}$ is a set of infinite paths starting in v_I.

Intuitively, the game $\mathcal{G}$ is played between Player 0 and Player 1 by constructing an infinite path step-by-step: first, a pebble is placed in the initial position v_I. Whenever the pebble is placed on a position $v \in V_p$, then it is Player p's turn to move it to one of the successors of v (for $p \in \{0, 1\}$). By the assumption on the set of moves, this is always possible. This way, the players produce an infinite path (called an infinite play). If that infinite play belongs to $\mathcal{W}$, then it is won by Player 1, otherwise it is won by Player 0.

Formally, a *play* is a finite or infinite path that starts at v_I, i.e., a sequence $v_0 v_1 v_2 \cdots$ of positions such that $v_0 = v_I$ and $(v_i, v_{i+1}) \in E$ for all i. We write $\mathsf{Plays}(\mathcal{G}) \subseteq V^*$ for the set of finite plays of $\mathcal{G}$.

Whenever a finite play ends in a position from V_1, Player 1 can choose between possibly more than one ways to extend the play. A deterministic strategy for Player 1 instructs her what to do exactly, a (nondeterministic) strategy only gives some minimal requirement on how to extend. Formally, a *strategy (for Player 1)* is a non-empty set of finite plays $\chi \subseteq \mathsf{Plays}(\mathcal{G})$ such that the following hold for all $v \in V$ and all finite plays $\pi\, v \in \chi$:

1. $\pi \in \chi$,
2. if $v \in V_0$, then $\pi\, v\, v' \in \chi$ for all $v' \in vE = \{w \in V \mid (v, w) \in E\}$, and
3. if $v \in V_1$, then there is (at least one) $v' \in vE$ such that $\pi\, v\, v' \in \chi$.

A χ-*play* is an infinite play such that all finite prefixes are in χ. In other words, all the choices made by Player 1 along a χ-play were made according to the strategy χ. The strategy χ is a *winning strategy (for Player 1)* if all χ-plays are won by Player 1, i.e., belong to $\mathcal{W}$. Player 1 *wins the infinite-duration game* if there exists a winning strategy for Player 1. A strategy χ for Player 1 is *positional* if there exists a function $s\colon V_1 \to V$ such that

$$\pi\, v\, v' \in \chi \iff s(v) = v'$$

for all $\pi\, v\, v' \in \mathsf{Plays}(\mathcal{G})$ with $v \in V_1$. In other words, iff

$$\chi = \mathsf{Plays}(\mathcal{G}) \cap V^* \cdot \left(\bigcup_{v \in V_0} \{v\}\,(vE) \cup \bigcup_{v \in V_1} \{v\, s(v)\} \right). \tag{1}$$

Strategies for Player 0 are defined similarly. (One has to exchange the roles of V_0 and V_1 in the definitions above.) Then, as above, Player 0 wins the infinite-duration game $\mathcal{G}$ if he has a winning strategy.

3. Multi-buffer Simulations

3.1. An Intuitive Description

Let (Σ, σ, 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 NBAs. The *multi-buffer simulation game* is played between players Spoiler and Duplicator using two pebbles on the state spaces of the two automata, as well as k FIFO queues in order to buffer the choices made by Spoiler so that Duplicator does not have to react to them immediately. Buffers can be addressed by indices, i.e. the 1st, the 2nd, etc., and each of them has a capacity which is either finite or infinite, also called *bounded* and *unbounded*.

The game starts with the two pebbles being placed on the initial states p_I and q_I respectively and all buffers empty. It then proceeds in rounds as follows.

1. Spoiler moves his pebble along a transition in $\mathcal{A}$ that is labeled with some letter $a \in \Sigma$.
2. This a is being put into all buffers named in $\sigma(a)$.
3. Duplicator now has the choice to either skip her turn if no buffer capacity is exceeded, or to choose a nonempty word $u \in \Sigma^+$ such that the contents of all buffers $i \in [k]$ start with $\pi_i(u)$, shift her pebble along some u-labeled path in $\mathcal{B}$ and, for all i, remove $\pi_i(u)$ from the front of buffer i, ensuring that the buffer capacities are respected again after this move.[1]

Note that a buffer can be "overfilled" by one letter momentarily after Spoiler's choice. Duplicator then has to remove something from this buffer immediately to ensure that the capacity is not exceeded at the end of the round.

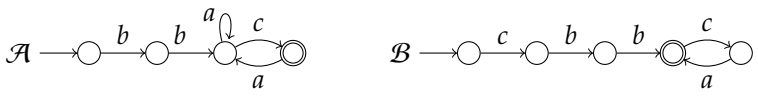
[1] This concerns, most of all, the buffers i such that $\pi_i(u) = \varepsilon$.

Let ρ and ρ' be the paths (or runs) in $\mathcal{A}$ and $\mathcal{B}$, respectively, that the players have moved the pebbles along in a play. Then ρ is necessarily an infinite run while ρ' can be finite (if, from some round on, Duplicator always skips her turn). Duplicator wins this play if

- ρ is not an accepting run
- or
  - ρ' is an accepting infinite run and
  - every letter that Spoiler puts into any of the buffers eventually gets removed from that buffer by Duplicator.

We observe that no player can ever get stuck in such a play. For Spoiler this is a simple consequence of the totality assumption on NBA. For Duplicator, it can be proved by induction on the length of a play that the following invariant holds: there exists a word $w \in \Sigma^+$ such that, for all $i \in [k]$, buffer i contains the word $\pi_i(w)$. Hence, by the totality assumption on the NBA $\mathcal{B}$, she can always empty the buffers completely and therefore, in particular, make some legal move.

Example 2 Consider the following two NBA over the trace alphabet $\Sigma = \{a, b, c\}$ with $\sigma(a) = \{1\}$, $\sigma(b) = \{1, 2\}$, and $\sigma(c) = \{3\}$.



Duplicator wins the multi-buffer game on $\mathcal{A}$ and $\mathcal{B}$ with buffer capacities $(\omega, 2, 0)$. 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 two 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 forever, 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.

A variant of this multi-buffer simulation game is the *multi-buffer flushing* game. In that game, Duplicator is required to either skip her turn or to empty all buffers completely.

3.2. A Formal Definition as Infinite-Duration Game

Let $\mathcal{A}, \mathcal{B}$ be NBA over a trace alphabet (Σ, σ, k) as above and $\kappa\colon [k] \to \mathbb{N} \cup \{\omega\}$ be a function that assigns a capacity to each of k buffers. Formally, the *multi-buffer simulation game* on $\mathcal{A}$ and $\mathcal{B}$ with buffer capacities κ is an infinite-duration game

$$\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B}) := (V, V_0, V_1, E, v_I, \mathcal{W}) .$$

Here, Spoiler is Player 0 and Duplicator is Player 1. The game is defined as follows.

Positions. As always, we have $V := V_0 \cup V_1$ where

$$V_0 := \{0\} \times Q^{\mathcal{A}} \times \Sigma_1^{\leq \kappa(1)} \times \cdots \times \Sigma_k^{\leq \kappa(k)} \times Q^{\mathcal{B}}$$

is the set of tuples $(0, p, w_1, \ldots, w_k, q)$ with $w_i \in \Sigma_i^*$ and $|w_i| \leq \kappa(i)$ for all $i \in [k]$. Likewise,

$$V_1 := \{1\} \times Q^{\mathcal{A}} \times \Sigma_1^{\leq 1+\kappa(1)} \times \cdots \times \Sigma_k^{\leq 1+\kappa(k)} \times Q^{\mathcal{B}}$$

is the set of tuples $(1, p, w_1, \ldots, w_k, q)$ with $w_i \in \Sigma_i^*$ and $|w_i| \leq 1 + \kappa(i)$ for all $i \in [k]$. (Recall that $1 + \omega = \omega$ such that $\Sigma^{\leq 1+\omega} = \Sigma^*$).

The initial node is

$$v_I := (0, p_I, \underbrace{\varepsilon, \ldots, \varepsilon}_{k \text{ times}}, q_I) .$$

Moves. The set of moves, i.e., the set E, is the set of the following pairs from $(V_0 \times V_1) \cup (V_1 \times V_0)$:

- $\Big((0, p, w_1, \ldots, w_k, q), (1, p', w_1', \ldots, w_k', q')\Big)$ provided that there is $a \in \Sigma$ such that
  - $p' \in \delta^{\mathcal{A}}(p, a)$,
  - $w_i' = w_i \, \pi_i(a)$ for all $i \in [k]$ (i.e., $w_i' = w_i a$ for $i \in \sigma(a)$ and $w_i' = w_i$ for $i \in [k] \setminus \sigma(a)$), and
  - $q' = q$.
- $\Big((1, p, w_1, \ldots, w_k, q), (0, p', w_1', \ldots, w_k', q')\Big)$ provided that there is $u \in \Sigma^*$ such that
  - $p' = p$,
  - $\pi_i(u) \, w_i' = w_i$ for all $i \in [k]$, and
  - $q' \in \delta^{\mathcal{B}}(q, u)$.

Note that in the description of moves from V_0 to V_1 (i.e., in moves made by Player 0), the letter a is uniquely given by the two positions. Differently, the word u in the description of moves from V_1 to V_0 is only unique up to trace equivalence since the two positions only determine the projections $\pi_i(u)$ for all

$i \in [k]$. In particular, the two positions determine the number of occurrences of every letter a in the word u.

Also note that Player 1's skipping of a turn is modelled by the ability to choose $u = \varepsilon$ in her moves.

Winning conditions. For this, let $\pi = (v_n)_{n \geq 0}$ be an infinite play with $v_0 = v_I$ and $v_n = (p_n, w_{n,1}, w_{n,2}, \ldots, w_{n,k}, q_n)$.

Since $v_0 \in V_0$ and since moves alternate between V_0 and V_1, we get $v_{2n} \in V_0$ and $v_{2n+1} \in V_1$ for all $n \geq 0$. Now let $n \geq 0$. Since $(v_{2n}, v_{2n+1}) \in E$, there is a unique letter $a_n \in \Sigma$ that justifies this move. In particular $p_{2n+1} \in \delta^{\mathcal{A}}(p_{2n}, a_n)$ and $q_{2n+1} = q_{2n}$. Since $(v_{2n+1}, v_{2(n+1)}) \in E$, there is a word $u_n \in \Sigma^*$ that justifies this move. In particular, $p_{2(n+1)} = p_{2n+1}$ and $q_{2(n+1)} \in \delta^{\mathcal{B}}(q_{2n+1}, u_n)$. As explained above, this word u_n is not uniquely determined. The infinite play π belongs to the winning condition $\mathcal{W}$ if the words u_n can be chosen in such a way that

- $p_n \in F^{\mathcal{A}}$ for finitely many $n \geq 0$
- or
  - $q_{2(n+1)} \in \delta_F^{\mathcal{B}}(q_{2n+1}, u_n)$ and $u_n \neq \varepsilon$ for infinitely many $n \geq 0$ and
  - $|a_0 a_1 a_2 \ldots|_a = |u_0 u_1 u_2 \ldots|_a$ for all letters $a \in \Sigma$.

The *multi-buffer flushing game* on $\mathcal{A}$ and $\mathcal{B}$ with buffer capacities κ is the infinite-duration game

$$\mathcal{F}^{\kappa}(\mathcal{A}, \mathcal{B}) := (V, V_0, V_1, E', v_I, \mathcal{W}')$$

where V, V_0, V_1 and v_I are as above. The set of moves E' is a subset of the set of moves of the multi-buffer simulation game: It contains all moves from V_0 to V_1 (i.e., moves of Player 0 are not restricted), but Player 1 can only do the following moves:

$\Big((1, p, w_1, \ldots, w_k, q), (0, p', w_1', \ldots, w_k', q')\Big) \in E'$ provided that there is $u \in \Sigma^*$ such that

- $p' = p$,
- $\pi_i(u) = w_i$ and $w_i' = \varepsilon$ for all $i \in [k]$,
  or $u = \varepsilon$ and $w_i' = w_i$ for all $i \in [k]$ and
- $q' \in \delta^{\mathcal{B}}(q, u)$.

Since $E' \subseteq E$, any play of the multi-buffer flushing game $\mathcal{F}^{\kappa}(\mathcal{A}, \mathcal{B})$ is also a play of the multi-buffer simulation game $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$. Hence we can define the winning condition $\mathcal{W}'$ of the multi-buffer flushing game *mutatis mutandis*: it is the set of infinite plays of the multi-buffer flushing game $\mathcal{F}^{\kappa}(\mathcal{A}, \mathcal{B})$ that belong to $\mathcal{W}$ as defined above.

Definition 3 Let $\kappa \colon [k] \to \mathbb{N} \cup \{\omega\}$ for some $k \geq 1$.

Then $\sqsubseteq^{\kappa}$ is the set of tuples $(\mathcal{A}, \mathcal{B}, \Sigma, \sigma)$ such that (Σ, σ, k) is a trace alphabet, $\mathcal{A}$ and $\mathcal{B}$ are NBAs over this trace alphabet, and Player 1 has a winning strategy

for the simulation game $\mathcal{G}^\kappa(\mathcal{A}, \mathcal{B})$. For $(\mathcal{A}, \mathcal{B}, \Sigma, \sigma) \in \sqsubseteq^\kappa$, we write $\mathcal{A} \sqsubseteq^\kappa \mathcal{B}$ (and let the trace alphabet be implicit).

Similarly, $\mathcal{A} \sqsubseteq^\kappa_{\mathsf{fl}} \mathcal{B}$ denotes that Player 1 has a winning strategy for the flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$.

Sometimes we will consider the special case of $k = 1$, i.e., the multi-buffer simulation or flushing game on a single buffer. This will also be referred to as the *single-buffer* simulation, respectively flushing game.

4. Upper Bounds

In this section, we will determine upper bounds for the question whether Player 1 can win the multi-buffer simulation, respectively the multi-buffer flushing game. Qualitatively, these upper bounds depend on the number and capacities of the buffers. In order to determine these upper bounds, we first translate the games in question into parity games.

4.1. Parity Games

A *parity game* is a tuple $(V, V_0, V_1, E, v_I, \Omega)$ such that

- $(V, V_0, V_1, E, v_I, \emptyset)$ is an infinite-duration game and
- $\Omega\colon V \to \{1, 2, \ldots, o\}$ is the *priority function* (for some number of priorities $o \geq 1$).

Such a parity game determines an infinite-duration game $(V, V_0, V_1, E, v_I, \mathcal{W})$ where $\mathcal{W}$ is the set of infinite plays $\pi = (v_n)_{n \geq 0}$ such that

$$\limsup_{n \to \infty} \Omega(v_n) \text{ is odd.}$$

We will identify the tuple $(V, V_0, V_1, E, v_I, \Omega)$ with the infinite-duration game $(V, V_0, V_1, E, v_I, \mathcal{W})$ and speak, e.g., of a winning strategy in the parity game.

In every parity game $\mathcal{G}$, one of the players has a strategy that is both, winning and memory-less [7]. In particular, one of the two players wins the game. The question who of the two players is the winner, is refered to as "solving the parity game". The question whether finite parity games can be solved in truly polynomial time is subject to current research and has led to a multitude of algorithms. We only consider very special parity games, hence for our purposes it is enough to appeal to a result which does not necessarily provide the best asymptotic worst-case complexity but is simple to state.

Proposition 4 ([19]) *Parity games with n nodes and o distinct priorities can be solved in time $O(n^{2+\lceil o/2 \rceil})$. Hence, solving finite parity games with a fixed number of priorities is in P.*

We will – in the following sections – translate multi-buffer simulation and flushing games into parity games. In order to solve these parity games, we will simplify them further. Using simulations (that are defined next), we will be able to prove that these simplifications are appropriate (i.e., preserve the winner).

Definition 5 Let $\mathcal{G} = (V, V_0, V_1, E, v_I, \Omega)$ and $\mathcal{G}' = (V', V'_0, V_1, E', v'_I, \Omega')$ be two parity games. A relation $R \subseteq V \times V'$ is a *simulation of $\mathcal{G}$ in $\mathcal{G}'$* if $(v_I, v'_I) \in R$ and, for all $(v, v') \in R$, the following hold:

(1) $v \in V_0$ iff $v' \in V'_0$;
(2) $\Omega(v) = \Omega'(v')$;
(3) if $v \in V_1$ and $w \in vE$, then there exists $w' \in v'E'$ with $(w, w') \in R$;
(4) if $v' \in V'_0$ and $w' \in v'E'$, then there exists $w \in vE$ with $(w, w') \in R$.

A *bisimulation of $\mathcal{G}$ and $\mathcal{G}'$* is a relation $R \subseteq V \times V'$ that is a simulation of $\mathcal{G}$ in $\mathcal{G}'$ such that, conversely, $R^{-1} = \{(v', v) \mid (v, v') \in R\}$ is a simulation of $\mathcal{G}'$ in $\mathcal{G}$.

Note that a relation R is a bisimulation iff, besides (1) and (2) from the above definition, we have

(3^+) if $w \in vE$, then there exists $w' \in v'E'$ with $(w, w') \in R$;
(4^+) if $w' \in v'E'$, then there exists $w \in vE$ with $(w, w') \in R$.

Lemma 6 *Let $\mathcal{G}$ and $\mathcal{G}'$ be two parity games and let R be a simulation of $\mathcal{G}$ in $\mathcal{G}'$. If Player 1 wins $\mathcal{G}$, then she wins $\mathcal{G}'$.*

Proof Since Player 1 wins $\mathcal{G}$, she has a winning strategy χ on $\mathcal{G}$. Let $\leq$ be a well-order on the set V of positions in the game $\mathcal{G}$.

Let $\pi = (v_i)_{0 \leq i \leq k} \in \mathsf{Plays}(\mathcal{G})$ and $\pi' = (v'_i)_{0 \leq i \leq k} \in \mathsf{Plays}(\mathcal{G}')$ be finite plays of the same length. We say that π is the witness for π' if the following holds for all $0 \leq i < k$:

$$v_{i+1} \in V \text{ is the } \leq\text{-minimal position } v \text{ such that } (v_0, v_1, \ldots, v_i, v) \in \chi \text{ and } (v, v'_{i+1}) \in R.$$

Note that not every play $\pi' \in \mathsf{Plays}(\mathcal{G}')$ has a witness, but that any play $\pi' \in \mathsf{Plays}(\mathcal{G}')$ has at most one witness. Note also that (with $i = k - 1$) any witness belongs to the winning strategy χ. Furthermore, if π is the witness of π', then $(v_i, v'_i) \in R$ for all $0 \leq i \leq k$.

Now define $\chi' \subseteq \mathsf{Plays}(\mathcal{G}')$ as the set of plays in $\mathcal{G}'$ that have a witness.

We will show that χ' is a winning strategy for the game $\mathcal{G}'$. To this aim, we first verify that χ' is a strategy:

- Note that $(v_I) \in \mathsf{Plays}(\mathcal{G})$ is the witness for the play $(v'_I) \in \mathsf{Plays}(\mathcal{G}')$. Hence χ' is not empty.
- By construction, χ' is closed under prefixes.

- Let $\pi' = (v'_i)_{0 \le i \le k}$ be a finite play in χ' with $v'_k \in V'_1$. Then, by definition of χ', the play π' has a witness $\pi = (v_i)_{0 \le i \le k} \in \chi$. Since, in particular, $(v_k, v'_k) \in R$, condition (1) from Definition 5 ensures $v_k \in V_1$. Since χ is a strategy, there is $v \in v_k E$ with $\pi\, v \in \chi$. Since $\le$ is a well-order, there is a minimal position v with this property that we call v_{k+1}. Condition (3) implies the existence of $v'_{k+1} \in v'_k E'$ with $(v_{k+1}, v'_{k+1}) \in R$. Hence $\pi'\, v'_{k+1} \in \mathsf{Plays}(\mathcal{G}')$ and $\pi\, v_{k+1}$ is a witness of this play, i.e., $\pi'\, v'_{k+1} \in \chi'$.
- Let $\pi' = (v'_i)_{0 \le i \le k}$ be a finite play in χ' with $v'_k \in V'_0$ and let $v'_{k+1} \in v'_k E'$. By definition of χ', the play π' has a witness $\pi = (v_i)_{0 \le i \le k} \in \chi$. Since, in particular, $(v_k, v'_k) \in R$, condition (1) ensures $v_k \in V_0$. Furthermore, condition (4) implies the existence of $v \in v_k E$ with $(v, v'_{k+1}) \in R$. Since $\le$ is a well-order, there is a minimal position v with this property that we call v_{k+1}. Since χ is a strategy, we get $\pi\, v_{k+1} \in \chi$. Since this play is a witness of $\pi'\, v'_{k+1}$, we obtain $\pi'\, v'_{k+1} \in \chi'$ by definition of χ'.

Let us finally show that χ' is a winning strategy, i.e., that Player 1 wins every χ'-play. So let $\pi' = (v'_i)_{i \ge 0}$ be an (infinite) χ'-play, and, for $k \in \mathbb{N}$, let $\pi'_k = (v'_i)_{0 \le i < k}$ be its prefix of length k. Then $\pi'_k \in \chi'$ since π' is a χ'-play. By definition of χ', there is, for every $k \ge 0$, a witness $\pi_k \in \chi$ for π'_k. Since π_k and π_{k+1} are witnesses of π'_k and $\pi'_{k+1} = \pi'_k\, v'_{k+1}$, resp., there is v_{k+1} such that $\pi_{k+1} = \pi_k\, v_{k+1}$ and $(v_{k+1}, v'_{k+1}) \in R$. Thus, the sequence $\pi = (v_i)_{i \ge 0}$ of positions is a χ-play and thus a winning play. By condition (2), π' is a winning play. ∎

In this paper, bisimulations will be used to simplify parity games. More precisely, we consider a bisimulation R of $\mathcal{G}$ and $\mathcal{G}'$ with $\mathcal{G}' = \mathcal{G}$ that is, at the same time, an equivalence relation on the set of positions of the game $\mathcal{G}$. The following lemma is immediate:

Lemma 7 *Let $\mathcal{G} = (V, V_0, V_1, E, v_I, \Omega)$ be a parity game and R an equivalence relation on V. Then R is a bisimulation of $\mathcal{G}$ and $\mathcal{G}$ itself iff it satisfies (1) and (2) from Definition 5 as well as the following: if $(v, v') \in R$ and $w \in vE$, then there exists $w' \in v'E$ with $(w, w') \in R$.*

For the proof of this lemma, it actually suffices to have a symmetric relation R.

If R is a bisimulation of $\mathcal{G}$ and $\mathcal{G}$ itself and an equivalence relation, then it makes sense to consider the following quotient of $\mathcal{G}$ wrt. R.

Definition 8 Let $\mathcal{G} = (V, V_0, V_1, E, v_I, \Omega)$ be a parity game and let R be a bisimulation of $\mathcal{G}$ and $\mathcal{G}$ that is, at the same time, an equivalence relation. Then we define the quotient game $\mathcal{G}/R = (V/_R, V_0/_R, V_1/_R, E/_R, v_I/_R, \Omega/_R)$ as follows:

- $V_0/_R = \{[v] \mid v \in V_0\}$,
- $V_1/_R = \{[v] \mid v \in V_1\}$,
- $v_I/_R = [v_I]$,

- $\Omega/_R([v]) = \Omega(v)$, and
- $([v_1], [v_2]) \in E/_R$ whenever $(v_1, v_2) \in E$.

Lemma 9 *Let $\mathcal{G} = (V, V_0, V_1, E, v_I, \Omega)$ be a parity game and let R be a bisimulation of $\mathcal{G}$ and $\mathcal{G}$ that is, at the same time, an equivalence relation. Then the relation $S = \{(v, [v]) \mid v \in V\}$ is a bisimulation of the parity game $\mathcal{G}$ and the quotient game $\mathcal{G}/R$. Hence Player 1 wins the game $\mathcal{G}$ iff she wins the game $\mathcal{G}/R$.*

Proof We have to verify the conditions (1) and (2) from Definition 5 as well as (3^+) and (4^+). So let $(v, [v]) \in S$.

(1) $v \in V_0 \iff [v] \in V_0/_R$
(2) $\Omega(v) = \Omega/_R([v])$
(3^+) Suppose $w \in vE$. Then $[w] \in [v]E/_R$ and $(w, [w]) \in S$.
(4^+) Suppose and $y \in [v]E/_R$. Then there is some position $w \in vE$ with $y = [w]$. Hence $y = [w] \in [v]E/_R$ and $(w, y) \in S$.

■

4.2. The Multi-Buffer Simulation Game

4.2.1. Multi-Buffer Simulation Games as Parity Games

In this section, we define for any multi-buffer simulation game an equivalent parity game, i.e., a parity game won by the same player as the multi-buffer simulation game. The idea of this construction is as follows:

1. First, we add priorities 1, 2, and 3 to the positions of the multi-buffer simulation game. The game moves into a position of priority 2 if Spoiler moves his pebble into some accepting state of the NBA $\mathcal{A}$. Similarly, the game moves into some position of priority 3 if Duplicator moves her pebble along some path that contains an accepting state of the NBA $\mathcal{B}$; in addition, it is required that she reads from all nonempty buffers between any two visits of priority-3-positions.
2. In order to ensure this last condition, we add another counter taking values between 0 and k to the game positions. This counter can only be increased from i to $i+1$ (modulo k) if buffer i is either empty or read from. Duplicator can only move into a priority-3-position in case the counter has value 0.
3. Finally, note that the multi-buffer simulation game cannot move into a position where the buffer contents are "inconsistent" (meaning that, e.g., $a \in \Sigma_1 \cap \Sigma_2$ appears more often in the first than in the second buffer). More precisely, the multi-buffer simulation game can only move into a position $(X, p, w_1, \ldots, w_k, q)$ if there is a word $w \in \Sigma^*$ with $w_i = \pi_i(w)$ for all $i \in [k]$. Our parity game contains only those positions, i.e., the buffer contents are encoded by a single word.

More formally, let $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, p_I, \delta^{\mathcal{A}}, F^{\mathcal{A}})$ and $\mathcal{B} = (Q^{\mathcal{B}}, \Sigma, q_I, \delta^{\mathcal{B}}, F^{\mathcal{B}})$ be NBA, let (Σ, σ, k) be a trace alphabet and let $\kappa\colon [k] \to \mathbb{N} \cup \{\omega\}$ be a capacity function for k buffers.

The *parity simulation game* on $\mathcal{A}$ and $\mathcal{B}$ with capacities κ is the parity game

$$\mathcal{G}^{\kappa}_{\text{par}}(\mathcal{A}, \mathcal{B}) = (V^{\text{par}}, V_0^{\text{par}}, V_1^{\text{par}}, E^{\text{par}}, v_I^{\text{par}}, \Omega^{\text{par}})$$

defined as follows.

Positions. The set V_0^{par} of Player 0's positions is the set of tuples

$$(0, p, w, q, c, r) \in \{0\} \times Q^{\mathcal{A}} \times \Sigma^* \times Q^{\mathcal{B}} \times \{0, 1, \ldots, k\} \times \{1, 3\}$$

with $|\pi_i(w)| \leq \kappa(i)$ for all $i \in [k]$. The set V_1^{par} of Player 1's positions is the set of tuples

$$(1, p, w, q, c, r) \in \{1\} \times Q^{\mathcal{A}} \times \Sigma^* \times Q^{\mathcal{B}} \times \{0, 1, \ldots, k\} \times \{1, 2\}$$

with $|\pi_i(w)| \leq 1 + \kappa(i)$ for all $i \in [k]$. As always, the set of positions is $V^{\text{par}} := V_0^{\text{par}} \cup V_1^{\text{par}}$.

The initial position is $v_I^{\text{par}} := (0, p_I, \varepsilon, q_I, 0, 1)$. Note that $(0, p_I, \pi_1(\varepsilon), \ldots, \pi_k(\varepsilon), q_I)$ is the initial position in the multi-buffer simulation game.

Priorities. The priority of position $v = (X, p, w, q, c, r)$ is $\Omega^{\text{par}}(v) = r$. In particular, positions of Player 0 have priority 1 or 3 and positions of Player 1 have priority 1 or 2.

Moves. The set E^{par} of moves is the set of the following pairs from $(V_0^{\text{par}} \times V_1^{\text{par}}) \cup (V_1^{\text{par}} \times V_0^{\text{par}})$:

(P0) $\Big((0, p, w, q, c, r), (1, p', w', q', c', r')\Big)$, provided that there is $a \in \Sigma$ such that

- $p' \in \delta^{\mathcal{A}}(p, a)$,
- $w' = wa$,
- $q' = q$,
- $c' = c$, and
- $r' = \begin{cases} 2 & \text{if } p' \in F^{\mathcal{A}} \\ 1 & \text{otherwise} \end{cases}$

Note that the first three conditions are equivalent to saying that there is a move from $(0, p, \pi_1(w), \ldots, \pi_k(w), q)$ to $(1, p', \pi_1(w'), \ldots, \pi_k(w'), q')$ in the multi-buffer simulation game $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$.

(P1) $\Big((1, p, w, q, c, r), (0, p', w', q', c', r')\Big)$, provided that there is $u \in \Sigma^*$ such that

- $p' = p$,
- $uw' \sim_{\sigma} w$,
- $q' \in \delta^{\mathcal{B}}(q, u)$,

- $c' = \begin{cases} 1 & \text{if } c = 0, u \neq \varepsilon, \text{ and } \exists u' \sim_\sigma u \colon q' \in \delta_F^{\mathcal{B}}(q, u') \\ (c+1) \bmod k & \text{if } c > 0 \text{ and } \pi_c(w) \neq \pi_c(w') \text{ or } \pi_c(w) = \varepsilon \\ c & \text{otherwise,} \end{cases}$
- and $r' = \begin{cases} 3 & \text{if } c = 0 \text{ and } c' = 1 \\ 1 & \text{otherwise.} \end{cases}$

As above, the first three conditions describe a move in the multi-buffer simulation game $\mathcal{G}^\kappa(\mathcal{A}, \mathcal{B})$ from position $(1, p, \pi_1(w), \ldots, \pi_k(w), q)$ to position $(0, p', \pi_1(w'), \ldots, \pi_k(w'), q')$.

In order to prove that the multi-buffer simulation game $\mathcal{G}^\kappa(\mathcal{A}, \mathcal{B})$ and the parity simulation game $\mathcal{G}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$ are equivalent, we first map the positions of the parity simulation game to positions of the multi-buffer simulation game. So let $v = (X, p, w, q, c, r) \in V^{\text{par}}$ be some position in the parity simulation game. Then define $f(v) = (X, p, \pi_1(w), \ldots, \pi_k(w), q)$. Then $f(v)$ is a position in the multi-buffer simulation game, i.e., $f(v) \in V$. Now let $\pi = (v_n)_{0 \leq n \leq N}$ be a finite play in the parity simulation game. We extend the function f from positions to plays by setting $f(\pi) = (f(v_n))_{0 \leq n \leq N}$. The remarks in the definition of the moves in the parity simulation game show that $f(\pi)$ is a play in the multi-buffer simulation game.

Using this function f, we can now translate winning strategies from the parity simulation game into winning strategies in the multi-buffer simulation game.

Lemma 10 *Let χ_{par} be a winning strategy for Player 0 in the parity simulation game $\mathcal{G}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$. Then*

$$\chi = \{f(\pi) \mid \pi \in \chi_{\text{par}}\}$$

is a winning strategy for Player 0 in the multi-buffer simulation game $\mathcal{G}^\kappa(\mathcal{A}, \mathcal{B})$.

Proof We first verify that χ is a strategy for Player 0 in the multi-buffer simulation game. Since χ_{par} is a strategy, the shortest play (v_I^{par}) belongs to χ_{par}. Since $f(v_I^{\text{par}}) = v_I$, the play (v_I) belongs to χ, hence $\chi \neq \emptyset$.

Let $\pi\, v \in \chi$ with $v \in V$. By the definition of χ, there exists a play $\pi^{\text{par}}\, v^{\text{par}} \in \chi_{\text{par}}$ with $\pi\, v = f(\pi^{\text{par}}\, v^{\text{par}})$. Since χ_{par} is a strategy, we get $\pi^{\text{par}} \in \chi_{\text{par}}$ and therefore $\pi = f(\pi^{\text{par}}) \in \chi$.

Let $\pi\, v \in \chi$ with $v \in V_1$ and let $(v, v') \in E$. We have to show $\pi\, v\, v' \in \chi$. As above, there exists a play $\pi^{\text{par}}\, v^{\text{par}} \in \chi_{\text{par}}$ with $\pi\, v = f(\pi^{\text{par}}\, v^{\text{par}})$. Suppose $v^{\text{par}} = (1, p, w, q, c, r)$. Then $v = (1, p, \pi_1(w), \ldots, \pi_k(w), q)$ and $v' = (0, p, \pi_1(uw), \ldots, \pi_k(uw), q')$ for some $u \in \Sigma^*$ and some $q' \in \delta^{\mathcal{B}}(q, u)$. Then there are $c' \in \{0, 1, \ldots, j\}$ and $r' \in \{1, 2, 3\}$ such that the position

$$v'^{\text{par}} = (0, p, uw, q', c', r')$$

satisfies $f(v'^{\text{par}}) = v'$ and $(v^{\text{par}}, v'^{\text{par}}) \in E^{\text{par}}$. Since χ_{par} is a strategy for Player 0, this implies $\pi^{\text{par}}\, v^{\text{par}}\, v'^{\text{par}} \in \chi_{\text{par}}$. Now we get $\pi\, v\, v' = f(\pi^{\text{par}}\, v^{\text{par}}\, v'^{\text{par}}) \in \chi$.

Let $\pi\, v \in \chi$ with $v \in V_0$. We have to find $v' \in vE$ with $\pi\, v\, v' \in \chi$. As above, there is a play $\pi^{\mathrm{par}}\, v^{\mathrm{par}} \in \chi_{\mathrm{par}}$ with $f(\pi^{\mathrm{par}}\, v^{\mathrm{par}}) = \pi\, v$. Since χ_{par} is a strategy for Player 0, there is a play $\pi^{\mathrm{par}}\, v^{\mathrm{par}}\, v'^{\mathrm{par}} \in \chi_{\mathrm{par}}$. Setting $v' = f(v'^{\mathrm{par}})$, we get $\pi\, v\, v' = f(\pi^{\mathrm{par}}\, v^{\mathrm{par}}\, v'^{\mathrm{par}}) \in \chi$.

Thus, indeed, χ is a strategy for Player 0 in the multi-buffer simulation game. It remains to be shown that this strategy is winning for Player 0. So let $(v_n)_{n\geq 0}$ be some χ-play.

Let T denote the set of finite plays $\pi^{\mathrm{par}} = (v_n^{\mathrm{par}})_{0\leq n\leq N} \in \chi_{\mathrm{par}}$ with $f(\pi^{\mathrm{par}}) = (v_n)_{0\leq n\leq N}$, i.e., $f(v_n^{\mathrm{par}}) = v_n$ for all $0 \leq n \leq N$. The edge relation K on this set describes the extension of a finite play by one move, i.e., K is the set of all pairs $(\pi^{\mathrm{par}}, \pi^{\mathrm{par}}\, v^{\mathrm{par}})$ for $\pi^{\mathrm{par}}\, v^{\mathrm{par}} \in T$. Since the function $f\colon V \to V^{\mathrm{par}}$ is finite-to-one, this tree is finitely branching. Since $(v_n)_{n\geq 0}$ is a χ-play, all its finite prefixes belong to χ. Hence, for all $N \geq 0$, there exists $\pi_N^{\mathrm{par}} \in \chi_{\mathrm{par}}$ with $f(\pi_N^{\mathrm{par}}) = (v_n)_{0\leq n\leq N}$. It follows that the tree T is infinite. By Kőnig's lemma, the tree T has an infinite branch which defines an infinite χ_{par}-play $(v_n^{\mathrm{par}})_{n\geq 0}$ with $f(v_n^{\mathrm{par}}) = v_n$ for all $n \geq 0$. Since χ_{par} is a winning strategy, this play is won by Player 0. In other words, we have $\Omega^{\mathrm{par}}(v_n^{\mathrm{par}}) = 2$ for infinitely many $n \geq 0$ and $\Omega^{\mathrm{par}}(v_n \mathrm{par}) = 3$ for finitely many $n \geq 0$.

For $n \geq 0$, let $v_n^{\mathrm{par}} = (X_n, p_n, w_n, q_n, c_n, r_n)$ such that

$$v_n = (X_n, p_n, \pi_1(w_n), \ldots, \pi_k(w_n), q_n)\,.$$

For $n \geq 0$, let $a_n \in \Sigma$ be the letter that justifies the move from v_{2n} to v_{2n+1} and let $u_n \in \Sigma^*$ be the word that justifies the move from v_{2n+1} to $v_{2(n+1)}$.

There are infinitely many $n \geq 0$ with $\Omega^{\mathrm{par}}(v_n^{\mathrm{par}}) = 2$. Since the positions of Player 0 cannot have parity 2, there are infinitely many $n \geq 0$ such that $\Omega^{\mathrm{par}}(v_{2n+1}^{\mathrm{par}}) = 2$, i.e., $p_{2(n+1)} = p_{2n+1} \in F^{\mathcal{B}}$. Hence the infinite run $(p_{2n}, a_n, p_{2(n+1)})_{n\geq 0}$ is accepting.

Furthermore, $\Omega^{\mathrm{par}}(v_n^{\mathrm{par}}) = 3$ holds for only finitely many $n \geq 0$. Hence, there is $c \in \{0, 1, \ldots, k\}$ such that $c_n = c$ for all but finitely many n.

First, suppose $c_n = 0$ holds for all but finitely many n. Then, for almost all $n \geq 0$, we have

$$u_n \neq \varepsilon \implies q_{2(n+1)} \notin \delta_F^{\mathcal{B}}(q_{2n+1}, [u_n])\,.$$

But this ensures that, for only finitely many $n \geq 0$, we have $u_n \neq \varepsilon$ and $q_{2(n+1)} \in \delta_F^{\mathcal{B}}(q_{2n+1}, [u_n])$. Thus, if $c = 0$, then the χ-play $(v_n)_{n\geq 0}$ is won by Player 0.

It remains to consider the case $c > 0$. Then, for almost all $n \geq 0$, we have $c = c_n$ and $\varepsilon \neq \pi_c(w_{2n+1}) = \pi_c(w_{2(n+1)})$. Hence, some letter a written into the buffer number c does never get read. Hence $|a_0 a_1 \ldots|_a \neq |u_0 u_1 \ldots|_a$, i.e., Player 0 wins the χ-play $(v_n)_{n\geq 0}$ also in case $c > 0$. ∎

Now we prove the analogous lemma for strategies of Player 1:

Lemma 11 *Let χ_{par} be a winning strategy for Player 1 in the parity simulation game $\mathcal{G}^{\kappa}_{\mathrm{par}}(\mathcal{A}, \mathcal{B})$. Then*

$$\chi = \{f(\pi) \mid \pi \in \chi_{\mathrm{par}}\}$$

is a winning strategy for Player 1 in the multi-buffer simulation game $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$.

Proof We first verify that χ is a strategy for Player 1 in the multi-buffer simulation game. Since χ_{par} is a strategy, the shortest play (v_I^{par}) belongs to χ_{par}. Since $f(v_I^{\mathrm{par}}) = v_I$, the play (v_I) belongs to χ, hence $\chi \neq \emptyset$.

Let $\pi\, v \in \chi$ with $v \in V_0$ and let $(v, v') \in E$. We have to show $\pi\, v\, v' \in \chi$. By the definition of χ, there exists a play $\pi^{\mathrm{par}}\, v^{\mathrm{par}} \in \chi_{\mathrm{par}}$ with $\pi\, v = f(\pi^{\mathrm{par}}\, v^{\mathrm{par}})$. Suppose $v^{\mathrm{par}} = (0, p, w, q, c, r)$. Then $v = (0, p, \pi_1(w), \ldots, \pi_k(w), q)$ and $v' = (1, p', \pi_1(wa), \ldots, \pi_k(wa), q)$ for some $a \in \Sigma$ and $p' \in \delta^{\mathcal{A}}(p, a)$. Set

$$v'^{\mathrm{par}} = (1, p', wa, q, c, r)$$

with $r = 2$ if $p' \in F^{\mathcal{A}}$ and $r = 1$ otherwise. Then $(v^{\mathrm{par}}, v'^{\mathrm{par}}) \in E^{\mathrm{par}}$. Since χ_{par} is a strategy, this implies $\pi^{\mathrm{par}}\, v^{\mathrm{par}}\, v'^{\mathrm{par}} \in \chi_{\mathrm{par}}$. Now we get $\pi\, v\, v' = f(\pi^{\mathrm{par}}\, v^{\mathrm{par}}\, v'^{\mathrm{par}}) \in \chi$.

Let $\pi\, v \in \chi$ with $v \in V_1$. We have to find $v' \in vE$ with $\pi\, v\, v' \in \chi$. As above, there is a play $\pi^{\mathrm{par}}\, v^{\mathrm{par}} \in \chi_{\mathrm{par}}$ with $f(\pi^{\mathrm{par}}\, v^{\mathrm{par}}) = \pi\, v$. Since χ_{par} is a strategy for Player 1, there is a play $\pi^{\mathrm{par}}\, v^{\mathrm{par}}\, v'^{\mathrm{par}} \in \chi_{\mathrm{par}}$. Setting $v' = f(v'^{\mathrm{par}})$, we get $\pi\, v\, v' = f(\pi^{\mathrm{par}}\, v^{\mathrm{par}}\, v'^{\mathrm{par}}) \in \chi$.

Thus, indeed, χ is a strategy for Player 1 in the multi-buffer simulation game. It remains to be shown that this strategy is winning. So let $(v_n)_{n\geq 0}$ be some χ-play. Then, for all $N \geq 0$, there exists $\pi_N^{\mathrm{par}} \in \chi_{\mathrm{par}}$ with $f(\pi_N^{\mathrm{par}}) = (v_n)_{0\leq n\leq N}$. Since the function $f\colon V \to V^{\mathrm{par}}$ is finite-to-one, we obtain, as in the proof of Lemma 10, from Kőnigs Lemma a χ_{par}-play $(v_n^{\mathrm{par}})_{n\geq 0}$ with $f(v_n^{\mathrm{par}}) = v_n$ for all $n \geq 0$. Since χ_{par} is a winning strategy, this play is won by Player 1. In other words, we have $\Omega^{\mathrm{par}}(v_n^{\mathrm{par}}) = 1$ for almost all $n \geq 0$ or we have $\Omega^{\mathrm{par}}(v_n\mathrm{par}) = 3$ for infinitely many $n \geq 0$.

For $n \geq 0$, let $v_n^{\mathrm{par}} = (X_n, p_n, w_n, q_n, c_n, r_n)$ such that

$$v_n = (X_n, p_n, \pi_1(w_n), \ldots, \pi_k(w_n), q_n)\,.$$

In the first case, i.e., if $\Omega^{\mathrm{par}}(v_n\mathrm{par}) = 1$ for all but finitely many $n \geq 0$, we have $p_n \notin F^{\mathcal{B}}$ for almost all $n \geq 0$. Hence the play $(v_n)_{n\geq 0}$ is won by Player 1.

Now consider the latter case $\Omega^{\mathrm{par}}(v_n^{\mathrm{par}}) = 3$ for infinitely many $n \geq 0$. Note that the parity of positions of Player 1 cannot be 3, hence there are infinitely many $n \geq 0$ with $r_{2(n+1)} = 3$. For any such $n \geq 0$, there is a word $u_n \in \Sigma^+$ justifying the move $(v_{2n+1}^{\mathrm{par}}, v_{2(n+1)}^{\mathrm{par}})$ with $q_{2(n+1)} \in \delta_F^{\mathcal{B}}(q_{2n+1}, u_n)$. Furthermore, the counter c_n moves infinitely often from 0 to 1 such that, any buffer $i \in [k]$, is empty infinitely often or shortened by Player 1 infinitely often. As a consequence, all letters ever put into buffer i is eventually read. Thus, Player 1 wins the play $(v_n)_{n\geq 0}$. ∎

The following result is an immediate consequence of the above two lemmata. It allows to analyse the multi-buffer simulation game in terms of parity games.

Theorem 12 *Let $\mathcal{A}$ and $\mathcal{B}$ be NBA over the trace alphabet (Σ, σ, k) and let $\kappa\colon [k] \to \mathbb{N} \cup \{\omega\}$ be a capacity function. Then the games $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$ and $\mathcal{G}^{\kappa}_{\mathrm{par}}(\mathcal{A}, \mathcal{B})$ are won by the same player and the multi-buffer simulation game is determined.*

Proof By [7], one of the players has a winning strategy in the parity simulation game $\mathcal{G}^{\kappa}_{\text{par}}(\mathcal{A}, \mathcal{B})$. By Lemma 10 and 11, the same player wins the multi-buffer simulation game $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$. Thus, the two games are equivalent. In particular, the multi-buffer simulation game is determined. ∎

4.2.2. Solving the multi-buffer simulation game

In this section, we will give upper bounds for the complexity of determining the winner in the multi-buffer simulation game, depending on the number of channels and the capacity function κ. In light of Theorem 12, this can be achieved by analysing the parity simulation game. The first result in this direction gives an upper bound in the analytical hierarchy for the general case.

Corollary 13 *The relations $\sqsubseteq^{\kappa}$ belong to Δ^1_2, even uniformly in the capacity function $\kappa\colon [k] \to \mathbb{N} \cup \{\omega\}$.*

In other words, the set of tuples $(\mathcal{A}, \mathcal{B}, (\Sigma, \sigma, k), \kappa)$ where $\mathcal{A}$ and $\mathcal{B}$ are NBA over the trace alphabet (Σ, σ, k) and $\kappa\colon [k] \to \mathbb{N} \cup \{\omega\}$ is a capacity function such that Player 1 wins the multi-buffer simulation game $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$ belongs to Δ^1_2.

Recall that $\Delta^1_2 = \Sigma^1_2 \cap \Pi^1_2$ is the intersection of the second existential and the second universal level of the analytical hierarchy.

Proof It suffices to prove this claim for the parity simulation game $\mathcal{G}^{\kappa}_{\text{par}}(\mathcal{A}, \mathcal{B})$.

The existence of a winning strategy for Player 1 in this parity game is a statement of the form

$$\exists \text{ some strategy } \chi \, \forall \text{ sequences } \pi = (v_n)_{n \geq 0}\colon \alpha$$

where α is the following arithmetical statement:

$$\Big(\forall n\colon (v_0, \ldots, v_n) \in \chi\Big) \Longrightarrow \; \exists k \, \forall \ell\colon \Omega^{\text{par}}(v_{k+\ell}) = 1 \vee \exists k \, \forall \ell \, \exists m\colon \Omega^{\text{par}}(v_{k+\ell+m}) = 3\,.$$

Since strategies as well as sequences are infinite sets, this is a typical statement from Σ^1_2. Hence the set in question belongs to Σ^1_2.

By Theorem 12, the tuple $(\mathcal{A}, \mathcal{B}, (\Sigma, \sigma, k), \kappa)$ belongs to the set in question if Player 0 does not have a winning strategy for $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$, which is a negated Σ^1_2-statement and therefore a Π^1_2-statement. Hence the set in question also belongs to Π^1_2. ∎

In Section 5, we will see that this upper bound cannot be improved much in case we have at least two buffers of which at least one is unbounded (in particular, the game is highly undecidable in that case). Upper bounds for the two remaining cases, namely $k = 1$ (i.e., only one buffer) and $\kappa(i) \in \mathbb{N}$ for all $i \in [k]$ (i.e., all buffers are bounded) are given by the following two corollaries of this section. First, we consider the case that all buffers are bounded:

Corollary 14 *The relations $\sqsubseteq^{(c_1, c_2, \ldots, c_k)}$ with $k \geq 1$ and $c_1, c_2, \ldots, c_k \in \mathbb{N}$ are decidable in polynomial time (uniformly, they are decidable in time polynomial in the automata and exponential in $k \cdot \max\{\kappa(i) + 1 \mid 1 \leq i \leq k\}$).*

More precisely, given two NBA $\mathcal{A}$ and $\mathcal{B}$ over the trace alphabet (Σ, σ, k) and a capacity function $\kappa\colon [k] \to \mathbb{N}$, it can be decided in time polynomial in $|\mathcal{A}| + |\mathcal{B}| + |\Sigma|^{k \cdot \max\{\kappa(i)+1 | i \in [k]\}}$ whether Player 1 wins the multi-buffer simulation game $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$.

Proof Let $w \in \Sigma^*$ with $|\pi_i(w)| \leq 1 + \kappa(i)$ for all $i \in \kappa$. Then $|w| \leq \sum_{i \in [k]}(1 + \kappa(i)) =: N$. It follows that, with $s = |\Sigma|$, there are at most $\frac{s^{N+1}-s}{s-1} \leq s^{N+1}$ many such words w. Consequently,

$$|V^{\text{par}}| \leq 2 \cdot |Q^{\mathcal{A}}| \cdot |\Sigma|^{N+1} \cdot |Q^{\mathcal{B}}| \cdot (k+1) \cdot 3\,,$$

i.e., the parity simulation game is polynomial in $|\mathcal{A}| + |\mathcal{B}| + |\Sigma|^{k \cdot \max\{\kappa(i) | i \in [k]\}}$. Since the parity simulation game uses only three priorities, the result follows from Theorem 12 and Proposition 4. ∎

The remaining section is devoted to the case $k = 1$ and $\kappa(1) = \omega$, i.e., to a single unbounded buffer. In this case, we will construct a finite parity game that is equivalent to the parity game $\mathcal{G}^{\kappa}_{\text{par}}(\mathcal{A}, \mathcal{B})$. This is achieved by first defining a "modified simulation game" that is equivalent to $\mathcal{G}^{\kappa}_{\text{par}}(\mathcal{A}, \mathcal{B})$. On this modified simulation game, we then define a bisimulation. The quotient of the modified game wrt. this bisimulation will be the announced finite parity game.

For the rest of this section, we fix the number of buffers by $k = 1$ and the capacity function κ by setting $\kappa(1) = \omega$. Consequently, for any trace alphabet $(\Sigma, \sigma, [1])$, we have $\sigma(a) = \{1\}$ for all $a \in \Sigma$.

The idea of the announced modification of the multi-buffer simulation game is as follows:

1. Recall that the two players build runs in their NBA $\mathcal{A}$ and $\mathcal{B}$, resp., and that the game positions recall the states reached so far. In the modified game, the game position recalls the complete runs built so far.
2. Let w be the word in the current game position in the parity simulation game. If the position belongs to Player 1, then she can use an arbitrary prefix of w to extend her run. In the subsequent rounds (while Player 1 skips her turns), Player 0 will extend the remaining word w_1 by some word w_2 until Player 1 decides to use an arbitrary prefix of the resulting word $w_1 w_2$. In the modified game, Player 1 is forced to use w_1, i.e., all the letters left in the buffer after her last proper move, entirely.

More formally, the *modified simulation game* on $\mathcal{A}$ and $\mathcal{B}$ is the parity game

$$\mathcal{G}_{\text{mod}}(\mathcal{A}, \mathcal{B}) = (V^{\text{mod}}, V_0^{\text{mod}}, V_1^{\text{mod}}, E^{\text{mod}}, v_I^{\text{mod}}, \Omega^{\text{mod}})$$

defined as follows.

Positions. V_0^{mod} is the set of tuples

$$(0, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r) \in \{0\} \times \mathsf{iFRuns}(\mathcal{A}) \times \Sigma^* \times \Sigma^* \times \mathsf{iFRuns}(\mathcal{B}) \times \{0,1\} \times \{1,3\}$$

and V_1^{mod} is the set of tuples

$$(1, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r) \in \{0\} \times \mathsf{iFRuns}(\mathcal{A}) \times \Sigma^* \times \Sigma^* \times \mathsf{iFRuns}(\mathcal{B}) \times \{0,1\} \times \{1,2\}\,.$$

As always, $V^{\mathrm{mod}} = V_0^{\mathrm{mod}} \cup V_1^{\mathrm{mod}}$.

Intuitively, $(X, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r)$ corresponds to the position $(X, \mathsf{target}(\rho^{\mathcal{A}}), w_1 w_2, \mathsf{target}(\rho^{\mathcal{B}}), c, r)$ in the parity simulation game.

The initial position is $v_I^{\mathrm{mod}} := (0, \varepsilon, \varepsilon, \varepsilon, \varepsilon, 0, 1)$.

Priorities. The priority of position $v = (X, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r)$ is $\Omega^{\mathrm{mod}}(v) := r$. In particular, positions of Player 0 have priority 1 or 3 and positions of Player 1 have priority 1 or 2.

Moves. The set of moves, i.e., the set E^{mod}, is the set of the following pairs from $(V_0^{\mathrm{mod}} \times V_1^{\mathrm{mod}}) \cup (V_1^{\mathrm{mod}} \times V_0^{\mathrm{mod}})$:

(P0) $\Big((0, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r), (1, \rho'^{\mathcal{A}}, w_1', w_2', \rho'^{\mathcal{B}}, c', r')\Big)$, provided that there is $a \in \Sigma$ such that

- $\rho'^{\mathcal{A}}$ is the extension of $\rho^{\mathcal{A}}$ by some a-labeled transition,
- $w_1' = w_1$ and $w_2' = w_2 a$,
- $\rho'^{\mathcal{B}} = \rho^{\mathcal{B}}$,
- $c' = c$, and
- $r' = \begin{cases} 2 & \text{if } \mathsf{target}(\rho'^{\mathcal{A}}) \in F^{\mathcal{A}} \\ 1 & \text{otherwise.} \end{cases}$

(P1) $\Big((1, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r), (0, \rho'^{\mathcal{A}}, w_1', w_2', \rho'^{\mathcal{B}}, c', r')\Big)$, provided that

1. $(\rho'^{\mathcal{A}}, w_1', w_2', \rho'^{\mathcal{B}}, c') = (\rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c)$ and $r' = 1$ or
2. - $\rho'^{\mathcal{A}} = \rho^{\mathcal{A}}$,
   - $w_1' = w_2$, $w_2' = \varepsilon$,
   - $\rho'^{\mathcal{B}}$ is an extension of $\rho^{\mathcal{B}}$ by some w_1-labeled run $\rho''^{\mathcal{B}}$,
   - $c' = \begin{cases} 1 & \text{if } \rho''^{\mathcal{B}} \neq \varepsilon \text{ sees some accepting state} \\ 0 & \text{otherwise} \end{cases}$
   - and $r' = \begin{cases} 3 & \text{if } \rho''^{\mathcal{B}} \neq \varepsilon \text{ sees some accepting state} \\ 1 & \text{otherwise.} \end{cases}$

Remark 15 The "counter" c is only used to notationally simplify the following proof.

We want to prove that the parity simulation game $\mathcal{G}^{\kappa}_{\mathrm{par}}(\mathcal{A}, \mathcal{B})$ and the modified simulation game $\mathcal{G}_{\mathrm{mod}}(\mathcal{A}, \mathcal{B})$ are equivalent. Providing a simulation, it can be easily shown that Player 1 wins the parity simulation game whenever she wins the modified simulation game (see proof of Lemma 17). For the other implication, we use some topological argument that we prepare by the following definition and result.

The set $\mathsf{ARuns}(\mathcal{A})$ of accepting runs of the NBA $\mathcal{A}$ carries a natural metric: two runs are "close" if they share a long common prefix. More precisely,

the distance between the distinct infinite accepting runs $\rho = (p_i, a_i, p_{i+1})_{i\geq 0}$ and $\rho' = (p'_i, a'_i, p'_{i+1})_{i\geq 0}$ equals $2^{-\ell}$ where

$$\ell = \min\{i \mid (p_i, a_i, p_{i+1}) \neq (p'_i, a'_i, p'_{i+1})\}\,.$$

A function f from $\mathsf{ARuns}(\mathcal{A})$ to $\mathsf{ARuns}(\mathcal{B})$ is *trace preserving* if the runs ρ and $f(\rho)$ carry the same trace (for all $\rho \in \mathsf{ARuns}(\mathcal{A})$), i.e., $\pi_i(u) = \pi_i(v)$ for all $i \in [k]$ for the words u and v accepted by ρ and $f(\rho)$, respectively. Given these notions, we have

Proposition 16 ([15, Theorem 18]) *Let $\mathcal{A}$ and $\mathcal{B}$ be NBA over the trace alphabet (Σ, σ, k) and $\kappa(i) = \omega$ for all $i \in [k]$. Then Player 1 has a winning strategy in the multi-buffer simulation game $\mathcal{G}^\kappa(\mathcal{A}, \mathcal{B})$ if, and only if, there exists a continuous trace preserving function from $\mathsf{ARuns}(\mathcal{A})$ to $\mathsf{ARuns}(\mathcal{B})$.*

Lemma 17 *Let $\mathcal{A}$ and $\mathcal{B}$ be two NBA over the trace alphabet $(\Sigma, \sigma, 1)$ and let $\kappa(1) = \omega$. Then the the multi-buffer simulation game $\mathcal{G}^\kappa(\mathcal{A}, \mathcal{B})$ and the modified simulation game $\mathcal{G}_{\mathrm{mod}}(\mathcal{A}, \mathcal{B})$ are won by the same player.*

Proof First suppose Player 1 wins the modified simulation game. The set of pairs

$$\Big((X, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r), (X, \mathsf{target}(\rho^{\mathcal{A}}), w_1 w_2, \mathsf{target}(\rho^{\mathcal{B}}), c, r)\Big)$$

forms a simulation of the modified game in the parity simulation game. Hence, by Lemma 6, she also wins the parity simulation game and therefore (by Theorem 12) the multi-buffer simulation game.

Conversely, suppose Player 1 wins the multi-buffer simulation game $\mathcal{G}^\kappa(\mathcal{A}, \mathcal{B})$. Hence, by Proposition 16, there is a continuous and trace preserving function $f\colon \mathsf{ARuns}(\mathcal{A}) \to \mathsf{ARuns}(\mathcal{B})$. Since $k = 1$, we get $\mathsf{lab}(\rho^{\mathcal{A}}) = \mathsf{lab}(f(\rho^{\mathcal{A}}))$ for all $\rho^{\mathcal{A}} \in \mathsf{ARuns}(\mathcal{A})$. We will define a positional strategy χ for Player 1 in the modified simulation game $\mathcal{G}_{\mathrm{mod}}(\mathcal{A}, \mathcal{B})$. This requires us to define a function $s\colon V_1^{\mathrm{mod}} \to V^{\mathrm{mod}}$. So let $(1, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, i) \in V_1^{\mathcal{G}_{\mathrm{mod}}(\mathcal{A},\mathcal{B})}$ be a position of Player 1 in the modified simulation game. We consider the set $R^{\mathcal{A}} \subseteq \mathsf{ARuns}(\mathcal{A})$ of all infinite accepting runs $\rho^{\mathcal{A}}_\omega$ that extend the finite run $\rho^{\mathcal{A}}$ and let $R^{\mathcal{B}} = \{f(\rho^{\mathcal{A}}_\omega) \mid \rho^{\mathcal{A}}_\omega \in R^{\mathcal{A}}\} \subseteq \mathsf{ARuns}(\mathcal{B})$. First suppose $R^{\mathcal{A}} \neq \emptyset$ and there is a run $\rho'^{\mathcal{B}}$ of length $|w_1|$ such that $\rho^{\mathcal{B}}\,\rho'^{\mathcal{B}}$ is a prefix of all runs in $R^{\mathcal{B}}$. Then set

$$s(1, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r) = (0, \rho^{\mathcal{A}}, w_2, \varepsilon, \rho^{\mathcal{B}}\,\rho'^{\mathcal{B}}, c', r') \text{ with} \tag{2}$$

$$(c', r') = \begin{cases} (1,3) & \text{if } \rho'^{\mathcal{B}} \neq \varepsilon \text{ meets some state from } F^{\mathcal{B}} \\ (0,1) & \text{otherwise} \end{cases}$$

If $R^{\mathcal{A}} = \emptyset$ or there is no such run $\rho'^{\mathcal{B}}$, then set

$$s(1, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r) = (0, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, 0, 1)\,.$$

This function $s\colon V_1^{\mathrm{mod}} \to V^{\mathrm{mod}}$ defines a positional strategy χ in the modified game.

Let $\pi = (v_i)_{i\geq 0}$ be a χ-play with $v_i = (X_i, \rho_i^{\mathcal{A}}, w_1^i, w_2^i, \rho_i^{\mathcal{B}}, c_i, r_i)$. Since the players play alternatingly, we get $X_i = 1 \iff i$ is odd. Note that the run $\rho_i^{\mathcal{A}}$ is a prefix of the run $\rho_{i+1}^{\mathcal{A}}$ and a proper prefix of $\rho_{i+2}^{\mathcal{A}}$. Hence there is a unique infinite and initial run $\rho^{\mathcal{A}}$ such that all finite runs $\rho_i^{\mathcal{A}}$ are prefixes of $\rho^{\mathcal{A}}$. If this run is not accepting, then there are only finitely many $i \geq 0$ with $r^i = 2$. Hence, in that case, Player 1 wins the play. Now suppose that $\rho^{\mathcal{A}}$ is accepting. Then set $\rho^{\mathcal{B}} = f(\rho^{\mathcal{A}})$. By the definition of the function s and the positional strategy χ, any of the finite runs $\rho_i^{\mathcal{B}}$ is a prefix of $\rho^{\mathcal{B}}$. Suppose there is $k \geq 0$ such that $|\rho_i^{\mathcal{B}}| \leq k$ for all $i \geq 0$. Then, for all $i \geq 0$, there are infinite accepting runs $\rho_1^i, \rho_2^i \in \mathsf{ARuns}(\mathcal{A})$ that extend $\rho_i^{\mathcal{A}}$ (implying $d(\rho_1^i, \rho_2^i) \leq 2^{-|\rho_i^{\mathcal{A}}|} = 2^{-\lfloor \frac{i}{2} \rfloor}$) such that $f(\rho_1^i)$ and $f(\rho_2^i)$ have different prefixes of length $k+1$ (i.e., $d(f(\rho_1^i), f(\rho_2^i)) \geq 2^{-k-1}$). But this contradicts the continuity of f. Consequently, $\rho^{\mathcal{B}}$ can be written as the infinite run

$$\rho_0'^{\mathcal{B}}\, \rho_1'^{\mathcal{B}}\, \rho_2'^{\mathcal{B}} \cdots$$

where the finite nonempty runs $\rho_i'^{\mathcal{B}}$ are the consecutive extensions originating from (2). Since $\rho^{\mathcal{B}} = f(\rho^{\mathcal{A}})$ is accepting, infinitely many of these runs meet some accepting state. Hence, $\Omega^{\mathrm{mod}}(v_i) = 3$ for infinitely many $i \geq 0$, i.e., Player 1 wins the play π. Thus, we showed that χ is a winning strategy for Player 1. ∎

Let $(X, \rho^{\mathcal{A}}, w_1, w_2, \rho^{\mathcal{B}}, c, r)$ be some position in the modified game. Then, all that is really needed to know are the target states of the two runs $\rho^{\mathcal{A}}$ and $\rho^{\mathcal{B}}$ as well as the "behavior" of w_1 and w_2 in the NBA $\mathcal{B}$, i.e., the sets $\delta^{\mathcal{B}}(\mathsf{target}(\rho^{\mathcal{B}}), w_1)$, $\delta_F^{\mathcal{B}}(\mathsf{target}(\rho^{\mathcal{B}}), w_1)$ as well as $\delta^{\mathcal{B}}(p, w_2)$ and $\delta_F^{\mathcal{B}}(p, w_2)$ for all $p \in Q^{\mathcal{B}}$.

We capture this "behavior" by the following function: For a word $w \in \Sigma^*$, we define the function $f_w \colon Q^{\mathcal{B}} \times Q^{\mathcal{B}} \to \{0,1\}^2$ setting $f_w(p,q) = (i,j)$ with

$$i = \begin{cases} 1 & \text{if there is a nonempty } w\text{-labeled run from } p \text{ to } q \\ & \text{that sees an accepting state} \\ 0 & \text{otherwise} \end{cases}$$

$$j = \begin{cases} 1 & \text{if there is some } w\text{-labeled run from } p \text{ to } q \\ & \text{that does not see an accepting state or is empty} \\ 0 & \text{otherwise} \end{cases}$$

Note that the number of functions f_u is at most $4^{|Q^{\mathcal{B}}|^2}$, i.e., exponential in the number of states of $\mathcal{B}$.

From these functions, we derive a binary relation on the positions of the modified game $\mathcal{G}_{\mathrm{mod}}(\mathcal{A}, \mathcal{B})$: Let R be the set of pairs of positions

$$\Big((X_1, \rho_{1\mathcal{A}}, w_{11}, w_{12}, \rho_{1\mathcal{B}}, c_1, r_1), (X_2, \rho_{2\mathcal{A}}, w_{21}, w_{22}, \rho_{2\mathcal{B}}, c_2, r_2),)\Big) \in V^{\mathrm{mod}} \times V^{\mathrm{mod}}$$

satisfying

- $\mathsf{target}(\rho_{1\mathcal{A}}) = \mathsf{target}(\rho_{2\mathcal{A}})$,

- $f_{w_{11}} = f_{w_{21}}$ and $f_{w_{12}} = f_{w_{22}}$,
- $\mathsf{target}(\rho_{1\mathcal{B}}) = \mathsf{target}(\rho_{2\mathcal{B}})$, and
- $(X_1, c_1, r_1) = (X_2, c_2, r_2)$.

Lemma 18 *The relation R is a bisimulation of the modified game $\mathcal{G}_{\mathrm{mod}}(\mathcal{A}, \mathcal{B})$ and itself and, at the same time, an equivalence relation.*

Proof Clearly, R is an equivalence relation. Now let $v_i = (X_i, \rho_{i\mathcal{A}}, w_{i1}, w_{i2}, \rho_{i\mathcal{B}}, c_i, r_i)$ for $i \in \{1, 2\}$ be two positions in the modified game with $(v_1, v_2) \in R$. By the definition of R, we obtain $(X_1, c_1, r_1) = (X_2, c_2, r_2)$.

Hence $v_1 \in V_0 \iff X_1 = 0 \iff X_2 = 0 \iff v_2 \in V_0$ and $\Omega^{\mathrm{mod}}(v_1) = r_1 = r_2 = \Omega^{\mathrm{mod}}(v_2)$.

Next let $v_1 \in V_1^{\mathrm{mod}}$ and $v_1' \in v_2 E^{\mathrm{mod}}$. If $v_1' = (0, \rho_{1\mathcal{A}}, w_{11}, w_{12}, \rho_{1\mathcal{B}}, c_1, r_1)$, set $v_2' = (0, \rho_{2\mathcal{A}}, w_{21}, w_{22}, \rho_{2\mathcal{B}}, c_2, r_2)$. Then $v_2' \in v_1 E^{\mathrm{mod}}$ and $(v_1, v_2) \in R$ implies $(v_1', v_2') \in R$.

Alternatively, $v_1' = (0, \rho_{1\mathcal{A}}, w_{12}, \varepsilon, \rho_{1\mathcal{B}}\, \rho_{1\mathcal{B}}', c_1', r_1')$ where $\rho_{1\mathcal{B}}'$ is some w_{11}-labeled run starting in $\mathsf{target}(\rho_{1\mathcal{B}})$ and

$$(c_1', r_1') = \begin{cases} (1,3) & \text{if this run is nonempty and sees some accepting state} \\ (0,1) & \text{otherwise.} \end{cases}$$

First, suppose that the run $\rho_{1\mathcal{B}}'$ is nonempty and sees some accepting state. Then $f_{w_{11}}(\mathsf{target}(\rho_{1\mathcal{B}}), \mathsf{target}(\rho_{1\mathcal{B}}')) \in \{(1,0), (1,1)\}$. Since $f_{w_{11}} = f_{w_{21}}$, there exists some nonempty w_{21}-labeled run $\rho_{2\mathcal{B}}'$ from $\mathsf{target}(\rho_{1\mathcal{B}}) = \mathsf{target}(\rho_{2\mathcal{B}})$ to $\mathsf{target}(\rho_{1\mathcal{B}}')$ that sees some accepting state. Set

$$v_2' = (0, \rho_{2\mathcal{A}}, w_{22}, \varepsilon, \rho_{2\mathcal{B}}\, \rho_{2\mathcal{B}}', 1, 3)\,.$$

Then $v_2' \in v_2 E^{\mathrm{mod}}$ and $(v_1', v_2') \in R$.

Next, suppose that the run $\rho_{1\mathcal{B}}'$ is empty or does not see an accepting state. Then $f_{w_{11}}(\mathsf{target}(\rho_{1\mathcal{B}}), \mathsf{target}(\rho_{1\mathcal{B}}\rho_{1\mathcal{B}}')) \in \{(0,1), (1,1)\}$. Since $f_{w_{11}} = f_{w_{21}}$, we can extend $\rho_{2\mathcal{B}}$ by some w_{21}-labeled run $\rho_{2\mathcal{B}}'$ that is empty or does not see an accepting state such that $\mathsf{target}(\rho_{1\mathcal{B}}\rho_{1\mathcal{B}}') = \mathsf{target}(\rho_{2\mathcal{B}}\rho_{2\mathcal{B}}')$. We set, similarly to above,

$$v_2' = (0, \rho_{2\mathcal{A}}, w_{22}, \varepsilon, \rho_{2\mathcal{B}}\, \rho_{2\mathcal{B}}', 0, 1)\,.$$

Then $v_2' \in v_2 E^{\mathrm{mod}}$ and $(v_1', v_2') \in R$.

Next let $v_2 \in V_0^{\mathrm{mod}}$ and $v_2' \in v_2 E^{\mathrm{mod}}$. Then there exist $a \in \Sigma$, an a-labeled run $\rho_{\mathcal{A}}'$ from $\mathsf{target}(\rho_{2\mathcal{A}})$ to some state q, and $r' \in \{1, 2\}$ such that $v_2' = (1, \rho_{2\mathcal{A}}\rho_{\mathcal{A}}', w_{21}, w_{22}a, \rho_{2\mathcal{B}}, c_2, r_2')$ where $r' = 2$ if $q \in F^{\mathcal{A}}$ and $r' = 1$ otherwise. Then set

$$v_1' = (1, \rho_{1\mathcal{A}}\rho_{\mathcal{A}}', w_{11}, w_{12}a, \rho_{1\mathcal{B}}, c_1, r')\,.$$

such that $v_1' \in v_1 E^{\mathrm{mod}}$. By the definition of the function $f_{w_{12}} = f_{w_{22}}$, we also obtain $f_{w_{12}a} = f_{w_{22}a}$ and therefore $(v_1', v_2') \in R$. ∎

We consider the quotient game $Q(\mathcal{A}, \mathcal{B}) = \mathcal{G}_{\text{mod}}(\mathcal{A}, \mathcal{B})/_R$: Its number of positions is bounded by

$$|Q_{\mathcal{A}}| \cdot 4^{|Q_{\mathcal{B}}^2|} \cdot |Q_{\mathcal{B}}| \cdot 2 \cdot 3$$

and therefore polynomial in the size of the automaton $\mathcal{A}$ and exponential in the sizes of the automaton $\mathcal{B}$. Furthermore, it uses 3 priorities. Hence, by Proposition 4, the winner can be decided in time polynomial in the size of the quotient game and therefore exponential in the size of the automata.

In summary, we get the following result.

Theorem 19 *$\sqsubseteq^{(\omega)}$ is decidable in exponential time. In other words, let $\kappa\colon [1] \to \mathbb{N} \cup \{\omega\}$ with $\kappa(1) = \omega$. For a trace alphabet $(\Sigma, \sigma, 1)$ and two NBA $\mathcal{A}$ and $\mathcal{B}$, it can be decided in exponential time whether Player 1 wins the single-buffer simulation game $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$.*

Proof By Theorem 12 and Lemma 17, Player 1 wins the multi-buffer simulation game $\mathcal{G}^{\kappa}(\mathcal{A}, \mathcal{B})$ iff she wins the modified game $\mathcal{G}_{\text{mod}}(\mathcal{A}, \mathcal{B})$. By Lemma 18, the relation R is a bisimulation and an equivalence relation on the modified simulation game. Hence, by Lemma 9, Player 1 wins the modified simulation game iff she wins the quotient game $\mathcal{G}_{\text{mod}}(\mathcal{A}, \mathcal{B})/_R$. Since this quotient game uses a fixed number of priorities, its winner can be computed in size polynomial in $\mathcal{G}_{\text{mod}}(\mathcal{A}, \mathcal{B})/_R$ and therefore exponential in $\mathcal{A}$ and $\mathcal{B}$. ∎

This finishes our consideration of upper bounds for solving the multi-buffer simulation game: the problem is in general in Δ_2^1, it can be solved in exponential time if we have a single buffer that is unbounded and in polynomial time if all buffers are bounded.

4.3. The Multi-Buffer Flushing Game

We now turn to the consideration of the multi-buffer flushing game. Recall that, differently from the multi-buffer simulation game, Duplicator now has only the choice of leaving the buffers untouched or emptying them completely.

We will proceed similarly to the above considerations: first, we define a parity game that is equivalent to the multi-buffer flushing game. This parity game will then be studied for four special cases: all buffers are unbounded, there is a single buffer which is unbounded, there is at most one unbounded buffer (and possibly some bounded ones), and all buffers are bounded. In all but the last of these cases, we obtain upper bounds that are (sometimes marginally, sometimes drastically) better than for the multi-buffer simulation game.

4.3.1. Multi-Buffer Flushing Games as Parity Games

In this section, we define for any multi-buffer flushing game an equivalent parity game, i.e., a parity game won by Player 1 iff the multi-buffer game is won by Player 1. The idea is very similar to the definition of the parity simulation game $\mathcal{G}^{\kappa}_{\text{par}}(\mathcal{A}, \mathcal{B})$: we add priorities 1, 2 and 3 to the positions and replace the contents of the k buffers by a single word in order to restrict the buffer contents

to "consistent" ones. The counter c is not needed here since the rules of the flushing game ensure that all buffers are emptied whenever Duplicator does not skip her move.

Let $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, p_I, \delta^{\mathcal{A}}, F^{\mathcal{A}})$ and $\mathcal{B} = (Q^{\mathcal{B}}, \Sigma, q_I, \delta^{\mathcal{B}}, F^{\mathcal{B}})$ be NBA, let (Σ, σ, k) be a trace alphabet and let $\kappa\colon [k] \to \mathbb{N} \cup \{\omega\}$ be a capacity function for k buffers.

The *parity flushing game* on $\mathcal{A}$ and $\mathcal{B}$ with capacities κ is the parity game

$$\mathcal{F}^{\kappa}_{\mathrm{par}}(\mathcal{A}, \mathcal{B}) = (V^{\mathrm{par}}, V_0^{\mathrm{par}}, V_1^{\mathrm{par}}, E^{\mathrm{par}}, v_I^{\mathrm{par}}, \Omega^{\mathrm{par}})$$

defined as follows.

Positions. The set V_0^{par} of Player 0's positions is the set of tuples

$$(0, p, w, q, r) \in \{0\} \times Q^{\mathcal{A}} \times \Sigma^* \times Q^{\mathcal{B}} \times \{1, 3\}$$

with $|\pi_i(w)| \leq \kappa(i)$ for all $i \in [k]$ such that $r = 3$ implies $w = \varepsilon$. The set V_1^{par} of Player 1's positions is the set of tuples

$$(1, p, w, q, r) \in \{1\} \times Q^{\mathcal{A}} \times \Sigma^* \times Q^{\mathcal{B}} \times \{1, 2\}$$

with $|\pi_i(w)| \leq 1 + \kappa(i)$ for all $i \in [k]$.

As always, the set of positions is $V^{\mathrm{par}} = V_0^{\mathrm{par}} \cup V_1^{\mathrm{par}}$.

The initial position is $v_I^{\mathrm{par}} := (0, p_I, \varepsilon, q_I, 3)$. Recall that $(0, p_I, \pi_1(\varepsilon), \ldots, \pi_k(\varepsilon), q_I)$ is the initial position in the multi-buffer flushing game.

Priorities. The priority of position $v = (X, p, w, q, r)$ is $\Omega^{\mathrm{par}}(v) = r$. In particular, positions of Player 0 have priorities 1 or 3 and positions of Player 1 have priorities 1 or 2.

Moves. The set E^{par} of moves is the set of the following pairs from $(V_0^{\mathrm{par}} \times V_1^{\mathrm{par}}) \cup (V_1^{\mathrm{par}} \times V_0^{\mathrm{par}})$:

(P0) $\Big((0, p, w, q, r), (1, p', w', q', r')\Big)$, provided that there is $a \in \Sigma$ such that

- $p' \in \delta^{\mathcal{A}}(p, a)$,
- $w' = wa$,
- $q' = q$,
- $r' = \begin{cases} 2 & \text{if } p' \in F^{\mathcal{A}} \\ 1 & \text{otherwise} \end{cases}$

Note that the first three conditions are equivalent to saying that there is a move from $(0, p, \pi_1(w), \ldots, \pi_k(w), q)$ to $(1, p', \pi_1(w'), \ldots, \pi_k(w'), q')$ in the multi-buffer flushing game $\mathcal{F}^{\kappa}(\mathcal{A}, \mathcal{B})$.

(P1) $\Big((1, p, w, q, r), (0, p', w', q', r')\Big)$, provided that there is $u \in \Sigma^*$ with $u = \varepsilon$ or $u \sim_\sigma w$ such that

- $p' = p$,

- $uw' \sim_\sigma w$ (i.e., $w' = \varepsilon$ if $u \sim_\sigma w$ and $w' \sim_\sigma w$ otherwise)
- $q' \in \delta^{\mathcal{B}}(q, u)$,
- and $r' \in \{1, 3\}$ such that $r' = 3$ implies $u \neq \varepsilon$ and $\exists u' \sim_\sigma u \colon q' \in \delta_F^{\mathcal{B}}(q, u')$.

As above, the first three conditions describe a move in the multi-buffer flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$ from position $(1, p, \pi_1(w), \ldots, \pi_k(w), q)$ to position $(0, p', \pi_1(w'), \ldots, \pi_k(w'), q')$.

The proof of the following theorem follows the same lines as that of Theorem 12 and is therefore omitted.

Theorem 20 *Let $\mathcal{A}$ and $\mathcal{B}$ be NBA over the trace alphabet (Σ, σ, k) and let $\kappa \colon [k] \to \mathbb{N} \cup \{\omega\}$ be a capacity function. Then Player 1 wins the multi-buffer flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$ if, and only if, she wins the parity flushing game $\mathcal{F}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$.*

From this result, we get the following corollary in the same way as Theorem 12 implies Corollary 14.

Corollary 21 *Given two NBA $\mathcal{A}$ and $\mathcal{B}$ over the trace alphabet (Σ, σ, k) and a capacity function $\kappa \colon [k] \to \mathbb{N}$, it can be decided in time polynomial in $|\mathcal{A}| + |\mathcal{B}| + |\Sigma|^{k \cdot \max\{\kappa(i) \mid i \in [k]\}}$ whether Player 1 wins the multi-buffer flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$.*

4.3.2. Unbounded Buffers, only

Suppose $\kappa(i) = \omega$ for all $i \in [k]$, i.e., all buffers are unbounded. We define a restricted version of the above parity flushing game by only allowing some of the moves of Player 1, namely those that do not flush the buffer and those that lead to some position of parity 3. The idea is that Player 1 postpones her moves until she can move into a parity-3-position. More precisely, the sets of positions, the parities, and the moves of Player 0 are as in the parity flushing game, but the set of moves of Player 1 is defined as follows:

(P1) $\Big((1, p, w, q, r), (0, p', w', q', r')\Big)$ is a move provided that there is $u \in \Sigma^*$ justifying $\Big((1, p, w, q, r), (0, p', w', q', r')\Big) \in E^{\text{par}}$ such that, in addition,

- $u \neq \varepsilon$ implies $r' = 3$.

We refer to this *restricted flushing game* by $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B}) = (V^{\text{res}}, V_0^{\text{res}}, V_1^{\text{res}}, v_I^{\text{res}}, E^{\text{res}}, \Omega^{\text{res}})$.

Lemma 22 *Let $\mathcal{A}$ and $\mathcal{B}$ be two NBA over the trace alphabet (Σ, σ, k) and let $\kappa(i) = \omega$ for all $i \in [k]$. Then there exists a simulation of the parity flushing game $\mathcal{F}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$ in the restricted flushing game $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B})$.*

Proof Let $v_{\text{par}} = (X_{\text{par}}, p_{\text{par}}, w_{\text{par}}, q_{\text{par}}, r_{\text{par}}) \in V^{\text{par}}$ be a position in the parity flushing game and let $v_{\text{res}} = (X_{\text{res}}, p_{\text{res}}, w_{\text{res}}, q_{\text{res}}, r_{\text{res}}) \in V^{\text{res}}$ be a position in the restricted flushing game. Then we set $(v_{\text{par}}, v_{\text{res}}) \in R$ iff there exists a finite word $x \in \Sigma^*$ such that

(i) $X_{\text{par}} = X_{\text{res}}$,
(ii) $p_{\text{par}} = p_{\text{res}}$,
(iii) $x\, w_{\text{par}} \sim_\sigma w_{\text{res}}$,
(iv) $q_{\text{par}} \in \delta^{\mathcal{B}}(q_{\text{res}}, x)$, and
(v) $r_{\text{par}} = r_{\text{res}}$.

We verify that R is indeed a simulation. So let v_{par} and v_{res} be as above with $(v_{\text{par}}, v_{\text{res}}) \in R$. We have

$$v_{\text{par}} \in V_0^{\text{par}} \iff X_{\text{par}} = 0 \overset{\text{(i)}}{\iff} X_{\text{res}} = 0 \iff v_{\text{res}} \in V_0^{\text{res}}$$

and

$$\Omega^{\text{par}}(v_{\text{par}}) = r_{\text{par}} \overset{\text{(v)}}{=} r_{\text{res}} = \Omega^{\text{res}}(v_{\text{res}})$$

It remains to be seen that the moves can be matched as well. This is split into two cases: (I) for Player 1's moves, and (II) further below for Player 0's moves.

(I) Suppose $v_{\text{par}} \in V_1^{\text{par}}$, i.e., $X_{\text{par}} = 1$, and let $v'_{\text{par}} = (X'_{\text{par}}, p'_{\text{par}}, w'_{\text{par}}, q'_{\text{par}}, r'_{\text{par}}) \in V^{\text{par}}$ be a position from $v_{\text{par}} E^{\text{par}}$, i.e., Player 1 can move in the parity flushing game from v_{par} to v'_{par}. Since $(v_{\text{par}}, v'_{\text{par}}) \in E^{\text{par}}$ and since the players alternate in the parity flushing game, we obtain $X'_{\text{par}} = 0$.

Since Player 1 can move from v_{par} to v'_{par} in the parity flushing game $\mathcal{F}^{\kappa}_{\text{par}}(\mathcal{A}, \mathcal{B})$, there exists $u_{\text{par}} \in \Sigma^*$ with $u_{\text{par}} = \varepsilon$ or $u_{\text{par}} \sim_\sigma w_{\text{par}}$ such that

- $p'_{\text{par}} = p_{\text{par}}$,
- $u_{\text{par}} w'_{\text{par}} \sim_\sigma w_{\text{par}}$,
- $q'_{\text{par}} \in \delta^{\mathcal{B}}(q_{\text{par}}, u_{\text{par}})$,
- and $r'_{\text{par}} \in \{1, 3\}$ with $r'_{\text{par}} = 3$ only in case $u_{\text{par}} \neq \varepsilon$ and $q'_{\text{par}} \in \delta_F^{\mathcal{B}}(q_{\text{par}}, [u_{\text{par}}])$.

Depending on the value of r'_{par}, we construct a position v'_{res} in the restricted flushing game $\mathcal{F}^{\kappa}_{\text{res}}(\mathcal{A}, \mathcal{B})$ such that $(v_{\text{res}}, v'_{\text{res}}) \in E^{\text{res}}$ and $(v'_{\text{par}}, v'_{\text{res}}) \in R$.

First suppose $r'_{\text{par}} = 3$ such that $u_{\text{par}} \neq \varepsilon$ and $q'_{\text{par}} \in \delta_F^{\mathcal{B}}(q_{\text{par}}, u'_{\text{par}})$ for some word $u'_{\text{par}} \sim_\sigma u_{\text{par}}$. Recall that $u_{\text{par}} \neq \varepsilon$ implies $w'_{\text{par}} = \varepsilon$. Then set $u_{\text{res}} = x u'_{\text{par}}$ and $v'_{\text{res}} = (X'_{\text{res}}, p'_{\text{res}}, w'_{\text{res}}, q'_{\text{res}}, r'_{\text{res}})$ with

- $X'_{\text{res}} = 0$,
- $p'_{\text{res}} = p_{\text{res}}$,
- $w'_{\text{res}} = \varepsilon$,
- $q'_{\text{res}} = q'_{\text{par}}$, and
- $r'_{\text{res}} = 3$.

We first verify that Player 1 can move from v_{res} to v'_{res} in the restricted flushing game $\mathcal{F}^{\kappa}_{\text{res}}(\mathcal{A}, \mathcal{B})$. Note that

$$u_{\text{res}} = xu'_{\text{par}} \sim_\sigma xu_{\text{par}} = xu_{\text{par}}w'_{\text{par}} \sim_\sigma xw_{\text{par}} \sim_\sigma w_{\text{res}}\,.$$

Furthermore, we obtain

- $p'_{\text{res}} = p_{\text{res}}$,
- $u_{\text{res}}w'_{\text{res}} = u_{\text{res}} \sim_\sigma w_{\text{res}}$,
- $q'_{\text{res}} = q'_{\text{par}} \in \delta^{\mathcal{B}}_F(q_{\text{par}}, u'_{\text{par}}) \subseteq \delta^{\mathcal{B}}_F(q_{\text{res}}, xu'_{\text{par}}) = \delta^{\mathcal{B}}_F(q_{\text{res}}, u_{\text{res}})$ and therefore in particular $q'_{\text{res}} \in \delta^{\mathcal{B}}(q_{\text{res}}, u_{\text{res}})$,
- From $u_{\text{par}} \neq \varepsilon$, we obtain $u'_{\text{par}} \neq \varepsilon$, and therefore $u_{\text{res}} \neq \varepsilon$. Furthermore, (by the previous item) $q'_{\text{res}} \in \delta^{\mathcal{B}}_F(q_{\text{res}}, u_{\text{res}})$. Since $r'_{\text{res}} = 3$, this finishes our verification of $(v_{\text{res}}, v'_{\text{res}}) \in E^{\text{res}}$.

Next, we verify $(v'_{\text{par}}, v'_{\text{res}}) \in R$. To this aim, let $x' = \varepsilon$. Then we get

- $X'_{\text{par}} = 0 = X'_{\text{res}}$,
- $p'_{\text{par}} = p_{\text{par}} = p_{\text{res}} = p'_{\text{res}}$,
- $x'w'_{\text{par}} = \varepsilon = w'_{\text{res}}$,
- $q'_{\text{par}} \in \delta^{\mathcal{B}}(q'_{\text{par}}, \varepsilon) = \delta^{\mathcal{B}}(q'_{\text{res}}, x')$, and
- $r'_{\text{par}} = 3 = r'_{\text{res}}$.

This finishes the case $r'_{\text{par}} = 3$. So suppose now that $r'_{\text{par}} = 1$. Then set $u_{\text{res}} = \varepsilon$ and $v'_{\text{res}} = (X'_{\text{res}}, p'_{\text{res}}, w'_{\text{res}}, q'_{\text{res}}, r'_{\text{res}})$ with

- $X'_{\text{res}} = 0$,
- $p'_{\text{res}} = p_{\text{res}}$,
- $w'_{\text{res}} = w_{\text{res}}$,[2]
- $q'_{\text{res}} = q_{\text{res}}$, and
- $r'_{\text{res}} = 1$.

Then we immediately get $(v_{\text{res}}, v'_{\text{res}}) \in E^{\text{res}}$ and it remains to verify $(v'_{\text{par}}, v'_{\text{res}}) \in R$. To this aim, set $x' = xu_{\text{par}}$. Then we get

- $X'_{\text{par}} = 0 = X'_{\text{res}}$,

[2]Since $v_{\text{res}} \in V^{\text{res}}_1$, we have $|\pi_i(w_{\text{res}})| \leq 1 + \kappa(i)$ for all $i \in [k]$. To ensure $v'_{\text{res}} \in V^{\text{res}}_0$, we need $|\pi_i(w'_{\text{res}})| \leq \kappa(i)$. Since $\kappa(i) = \omega$, this is indeed the case. It is not clear whether the lemma can be shown without the assumption $\kappa(i) = \omega$ for all $i \in [k]$.

- $p'_{\text{par}} = p_{\text{par}} = p_{\text{res}} = p'_{\text{res}}$,
- $w'_{\text{res}} = w_{\text{res}} \sim_\sigma xw_{\text{par}} \sim_\sigma xu_{\text{par}}w'_{\text{par}} = x'w'_{\text{par}}$,
- $q'_{\text{par}} \in \delta^{\mathcal{B}}(q_{\text{par}}, u_{\text{par}}) \subseteq \delta^{\mathcal{B}}(q_{\text{res}}, xu_{\text{par}}) = \delta^{\mathcal{B}}(q'_{\text{res}}, x')$, and
- $r'_{\text{par}} = 1 = r'_{\text{res}}$

which also settles the case $r'_{\text{par}} = 1$.

(II) Now suppose $v_{\text{res}} \in V_0^{\text{res}}$, i.e., $X_{\text{res}} = 0$, and let $v'_{\text{res}} = (X'_{\text{res}}, p'_{\text{res}}, w'_{\text{res}}, q'_{\text{res}}, r'_{\text{res}}) \in V^{\text{res}}$ be a position from $v_{\text{res}}E^{\text{res}}$, i.e., Player 0 can move in the restricted flushing game from v_{res} to v'_{res}. It follows that there is $a \in \Sigma$ such that

- $p'_{\text{res}} \in \delta^{\mathcal{A}}(p_{\text{res}}, a)$,
- $w'_{\text{res}} = w_{\text{res}}a$,
- $q'_{\text{res}} = q_{\text{res}}$, and
- $r'_{\text{res}} = \begin{cases} 2 & \text{if } p'_{\text{res}} \in F^{\mathcal{A}} \\ 1 & \text{otherwise.} \end{cases}$

Then set $v'_{\text{par}} = (X'_{\text{par}}, p'_{\text{par}}, w'_{\text{par}}, q'_{\text{par}}, r'_{\text{par}})$ with

- $X'_{\text{par}} = 1$,
- $p'_{\text{par}} = p'_{\text{res}}$,
- $w'_{\text{par}} = w_{\text{par}}a$,
- $q'_{\text{par}} = q_{\text{par}}$, and
- $r'_{\text{par}} = \begin{cases} 2 & \text{if } p'_{\text{par}} \in F^{\mathcal{A}} \\ 1 & \text{otherwise.} \end{cases}$

It is immediate that $(v_{\text{par}}, v'_{\text{par}}) \in E^{\text{par}}$, i.e., Player 0 can move from v_{par} to v'_{par} in the parity flushing game. We show $(v'_{\text{par}}, v'_{\text{res}}) \in R$:

- $X'_{\text{par}} = 1 = X'_{\text{res}}$,
- $p'_{\text{par}} = p'_{\text{res}}$,
- $xw'_{\text{par}} = xw_{\text{par}}a \sim_\sigma w_{\text{res}}a$,
- $q'_{\text{par}} = q_{\text{par}} \in \delta^{\mathcal{B}}(q_{\text{res}}, x) = \delta^{\mathcal{B}}(q'_{\text{res}}, x)$, and
- $r'_{\text{par}} = 2$ iff $p'_{\text{par}} \in F^{\mathcal{A}}$ iff $p'_{\text{res}} \in F^{\mathcal{A}}$ iff $r'_{\text{res}} = 2$.

Thus, indeed, R is a simulation of the parity flushing game $\mathcal{F}^{\kappa}_{\text{par}}(\mathcal{A}, \mathcal{B})$ in the restricted flushing game $\mathcal{F}^{\kappa}_{\text{res}}(\mathcal{A}, \mathcal{B})$. ∎

Lemma 23 *Let $\mathcal{A}$ and $\mathcal{B}$ be two NBA over the trace alphabet (Σ, σ, k) and let $\kappa(i) = \omega$ for all $i \in [k]$. Then Player 1 wins the multi-buffer flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$ if, and only if, she wins the restricted flushing game $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B})$.*

Proof First suppose Player 1 wins the restricted flushing game $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B})$. The set R of pairs (v, v) for $v \in V^{\text{res}} = V^{\text{par}}$ is a simulation of $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B})$ in the parity flushing game $\mathcal{F}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$. Hence, by Lemma 6, Player 1 wins the parity flushing game $\mathcal{F}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$ and therefore (by Theorem 20), the multi-buffer flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$.

Conversely, suppose Player 1 wins the multi-buffer flushing game $\mathcal{G}^\kappa(\mathcal{A}, \mathcal{B})$ and therefore (by Theorem 20) the parity flushing game $\mathcal{F}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$. The previous lemma provides a simulation of $\mathcal{F}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$ in $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B})$, hence Player 1 also wins the latter game by Lemma 6. ∎

Let $S_0 = \{0\} \times Q^{\mathcal{A}} \times \{\varepsilon\} \times Q^{\mathcal{B}} \times \{3\}$ be the set of all parity-3-positions in the restricted flushing game $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B})$. We define, inductively, a decreasing sequence of sets $S_0 \supseteq S_1 \supseteq S_2 \cdots$ by defining S_{n+1} to be the set of priority-3-positions $v = (0, p, \varepsilon, q, 3) \in S_n$ such that the following holds:

> For all infinite accepting runs $(p_i, a_i, p_{i+1})_{i \geq 0} \in \mathsf{ARuns}(\mathcal{A}[p])$, there exist $i \geq 0$, $w_i \sim_\sigma a_0 a_1 \dots a_i$, and $q' \in \delta^{\mathcal{B}}_F(q, w_i)$ with $(0, p_{i+1}, \varepsilon, q', 3) \in S_n$.

Since this defines a decreasing sequence of subsets of S_0 and since $|S_0| = |Q^{\mathcal{A}} \times Q^{\mathcal{B}}|$ is finite, we get $S_{|Q_{\mathcal{A}} \times Q_{\mathcal{B}}|} = \bigcap_{n \geq 0} S_n$.

Lemma 24 *$\bigcap_{n \geq 0} S_n$ is the set of parity-3-positions in the winning region for player 1.*

Proof First we provide a strategy for Player 1 that is winning from every position in $\bigcap_{n \geq 0} S_n$. This positional strategy χ tries to move into $\bigcap_{n \geq 0} S_n$ whenever possible: if the position is $(1, p, w, q, r)$ and there are $w' \sim_\sigma w$ and $q' \in \delta^{\mathcal{B}}_F(q, w')$ such that $(0, p, \varepsilon, q', 3) \in \bigcap_{n \geq 0} S_n$, then Player 1 moves to such a position. Otherwise, she moves to $(0, p, w, q, 1)$.

The claim is that χ is a winning strategy for Player 1 starting from an arbitrary position in $\bigcap_{n \geq 0} S_n$. Assume by contradiction that there is a χ-play starting in $\bigcap_{n \geq 0} S_n$ and lost by Player 1. Since that play meets a priority-3-position (and therefore a position from $\bigcap_{n \geq 0} S_n$) only finitely often, there is such a play $\pi = (v_i)_{i \geq 0}$ satisfying $v_0 \in \bigcap_{n \geq 0} S_n$ and $v_i \notin \bigcap_{n \geq 0} S_n$ for all $i > 0$. Let $v_i = (X_i, p_i, w_i, q_i, r_i)$ for $i \geq 0$. By the definition of the restricted flushing game $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B})$, there are letters $a_i \in \Sigma$ with $p_{2(i+1)} = p_{2i+1} \in \delta^{\mathcal{A}}(p_{2i}, a_i)$ and $w_{2i+1} = w_{2i} a_i$ for all $i \geq 0$. Since Player 1 looses the play π, there are infinitely many $i \geq 0$ with $\Omega(v_i) = 2$. Since $p_{2(i+1)} = p_{2i+1}$, there are consequently infinitely many $i \geq 0$ with $p_{2i} \in F^{\mathcal{A}}$. Hence $(p_{2i}, a_i, p_{2(i+1)})_{i \geq 0}$ is an accepting run of $\mathcal{A}[p_0]$. Let $N = |Q^{\mathcal{A}} \times Q^{\mathcal{B}}|$ such that $\bigcap_{n \geq 0} S_n = S_N$. Since $v_0 \in S_N = S_{N+1}$, there exist $i \geq 0$, $w \sim_\sigma a_0 a_1 \dots a_i$, and $q' \in \delta^{\mathcal{B}}_F(q_0, w)$ with $(0, p_{2(i+1)}, \varepsilon, q', 3) \in S_N$. Let i be minimal with this property. Since π is a χ-play, we get $v_{2i+1} = (1, p_{2i+1}, a_0 a_1 \dots a_i, q_0, r_{2i+1})$ and $v_{2(i+1)} \in S_N = \bigcap_{n \geq 0} S_n$. But this contradicts the choice of the infinite play π. Hence, indeed, Player 1 has a winning strategy from all positions in $\bigcap_{n \geq 0} S_n$.

To prove the converse implication, we provide, by induction on n, a winning strategy for Player 0 for every $v = (0, p, \varepsilon, q, 3) \in S_n \setminus S_{n+1}$. So first let $n = 0$, i.e., $v \in S_0 \setminus S_1$. By definition of S_1, there is some accepting run $\rho = (p_i, a_i, p_{i+1})_{i \geq 0} \in \mathsf{ARuns}(\mathcal{A}[p])$ such that $(0, p_i, \varepsilon, q, 3) \notin S_0$ for all $i \geq 0$, all $w \sim_\sigma a_0 a_1 \dots a_i$, and all $q \in \delta_F^{\mathcal{B}}(p, w)$. The strategy χ_v for Player 0 now is to follow this accepting run ρ such that every χ_v-play $\pi = (v_j)_{j \geq 0}$ with $v_0 = v$ satisfies $v_j = (j \bmod 2, p_{\lfloor \frac{j}{2} \rfloor}, w_j, q_j, r_j)$ for all $j \geq 0$. We want to show that χ_v is winning for Player 0. Towards a contradiction, suppose the χ_v-play $\pi = (v_j)_{j \geq 0}$ is won by Player 1. Since $p_i \in F_{\mathcal{A}}$ for infinitely many $i \geq 0$, we get $r_{2i+1} = 2$ for infinitely many $i \geq 0$. Since the play π is won by Player 1, there must be infinitely many $i \geq 0$ with $r_{2i} = 3$. Let $i \geq 0$ be minimal with $r_{2i} = 3$. It follows that $i > 0$ and $w_{2i-1} = a_0 a_1 \dots a_i$. Since $r_{2i} = 3$, there are $w \sim_\sigma w_{2i-1}$ and $q_{2i} \in \delta_F^{\mathcal{B}}(q_0, w)$ with $v_{2i} = (0, p_i, \varepsilon, q_{2i}, 3) \in S_0$. But this contradicts the choice of the run ρ. Hence, indeed, for every $v \in S_1 \setminus S_0$, Player 0 has a winning strategy (namely, χ_v).

Now let $v \in S_n \setminus S_{n+1}$. Then, again, there is some accepting run $\rho = (p_i, a_i, p_{i+1})_{i \geq 0} \in \mathsf{ARuns}(\mathcal{A}[p])$ such that $(0, p_i, \varepsilon, q', 3)) \notin S_n$ for all $i \geq 0$, all $w \sim_\sigma a_0 a_1 \dots a_i$, and all $q' \in \delta^*_{\mathcal{B},F}(q, w)$. Player 0's strategy χ_v is now to first play that run ρ. If Player 1 never moves into a priority-3-position, then the play is lost by Player 1 since it meets a priority-2-position infinitely often and a priority-3-position only once. Now suppose Player 1 moves into a priority-3-position w. Then, by the assumption on the run ρ, we get $w \in S_0 \setminus S_n$. Hence there is $m \in [n-1]$ with $w \in S_m \setminus S_{m+1}$. From that moment on, Player 0 plays his winning strategy χ_w that was already constructed by the induction hypothesis.

Thus, Player 0 wins from every position in $S_0 \setminus \bigcap_{n \geq 0} S_n$. In other words, Player 1's winning priority-3-positions are all contained in $\bigcap_{n \geq 0} S_n$. ∎

Lemma 25 $\bigcap_{n \geq 0} S_n \in \Pi^1_1$.

Proof With $N = |Q^{\mathcal{A}} \times Q^{\mathcal{B}}|$, we get $S_N = S_{N+1}$ and therefore $\bigcap_{n \geq 0} S_n = S_N$. Hence it suffices to be shown that S_n belongs to Π^1_1 for all $n \in \mathbb{N}$. Note that S_0 is finite and decidable. Hence, this set belongs to Π^1_1. The definition of S_{n+1} has the form

$$\forall f \colon \mathbb{N} \to Q^{\mathcal{A}}, g \colon \mathbb{N} \to \Sigma \, \exists i \colon \varphi(f, g) \to \psi(f(i+1), g(0)g(1) \cdots g(i))$$

with the following subexpressions φ and ψ:

$$\begin{aligned} \varphi(f, g) = \forall j \colon & f(j+1) \in \delta^{\mathcal{A}}(f(j), g(j)) \\ & \wedge f(0) = p \\ & \wedge \forall j \, \exists k \colon f(j+k) \in F^{\mathcal{A}} \end{aligned}$$

and

$$\psi(p', u) = \bigvee_{\substack{w \sim_\sigma u \\ q' \in \delta_F^{\mathcal{B}}(q, w)}} (0, p', \varepsilon, q', 3) \in S_n \, .$$

Since the disjunction in ψ is finite and computable from u (the automaton $\mathcal{B}$ and the trace alphabet σ), the expression ψ belongs to Π^1_1. Clearly, φ is an arithmetical expression (since its quantification extends over natural numbers). In summary, the definition of S_{n+1} has the form

$$\forall F\, \exists i\, \forall G\colon \xi$$

for some arithmetical expression ξ. By [21, Theorem 16.1.III], it can be reformulated into an equivalent expression of the form

$$\forall F\, \forall G\, \exists i\colon \xi'$$

where, again, ξ' is arithmetical. But this expression belongs to Π^1_1. Thus, we obtain a Π^1_1-definition for S_{n+1} and therefore for $S_N = \bigcap_{n\geq 0} S_n$. ∎

Theorem 26 *The relations $\sqsubseteq^{(\omega,\omega,\ldots,\omega)}$ belong to Π^1_1, even uniformly in the number of buffers k. In other words, the set of all tuples $(\mathcal{A}, \mathcal{B}, \Sigma, \sigma, k)$ where*

- *(Σ, σ, k) is a trace alphabet and $\mathcal{A}$ and $\mathcal{B}$ are NBA over Σ and*
- *Player 1 wins the multi-buffer flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$ with $\kappa(i) = \omega$ for all $i \in [k]$*

belongs to Π^1_1.

Proof Let $\mathcal{A}$ and $\mathcal{B}$ be two NBA over a trace alphabet (Σ, σ, k) and let $\kappa\colon [k] \to \{\omega\}$ be a capacity function. Then Player 1 wins the multi-buffer flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$ iff she wins the restricted flushing game $\mathcal{F}^\kappa_{\text{res}}(\mathcal{A}, \mathcal{B})$ (Lemma 23) iff the initial position v_I^{res} of this game belongs to $\bigcup_{n\geq 0} S_n$ (Lemma 24 since the parity of v_I^{res} is 3). But this latter is a statement from Π^1_1 by Lemma 25. ∎

4.3.3. A single and unbounded buffer

We next consider the case of a single buffer that is unbounded, i.e., $k = 1$ and $\kappa(1) = \omega$. In that case, we can improve the result of Lemma 25 considerably:

Lemma 27 *$\bigcap_{n\geq 0} S_n$ is computable in space polynomial in $|Q^{\mathcal{A}} \times Q^{\mathcal{B}}|$.*

Proof As in the proof of Lemma 25, it suffices to compute S_n in polynomial space for all $n \in \mathbb{N}$. This claim is trivial for $n = 0$ since $S_0 = \{0\}\times Q^{\mathcal{A}}\times\{\varepsilon\}\times Q^{\mathcal{B}}\times\{3\}$. Since in case $k = 1$, the relation $\sim_\sigma$ is the identity on Σ^*, we have $v = (0, p, \varepsilon, q, 3) \in S_{n+1}$ if, for all infinite accepting runs $(p_i, a_i, p_{i+1})_{i\geq 0} \in \mathsf{ARuns}(\mathcal{A}[p])$, there exist $i \geq 0$ and $q' \in \delta^{\mathcal{B}}_F(q, a_0 a_1 \ldots a_i)$ with $(0, p_{i+1}, \varepsilon, q', 3) \in S_n$.

To verify whether this holds, we consider the following NBA C. It is meant to accept the empty language if, and only if, $v = (0, p, \varepsilon, q, 3) \notin S_{n+1}$.

- States are the tuples $(r, M, N) \in Q^{\mathcal{A}} \times \mathcal{P}(Q^{\mathcal{B}}) \times \mathcal{P}(Q^{\mathcal{B}})$ such that, for all $s \in N$, we have $(0, r, \varepsilon, s, 3) \notin S_n$. In addition, we have the initial state ι.
- $\delta((r, M, N), a)$ is the set of all states $(r', \delta^{\mathcal{B}}(M, a), \delta^{\mathcal{B}}(N, a) \cup (M \cap F^{\mathcal{B}}))$ of C that satisfy $r' \in \delta^{\mathcal{A}}(r, a)$.

- $\delta(\iota, a)$ is the set of all states $(r, \delta^{\mathcal{B}}(p,a), \delta^{\mathcal{B}}(q,a) \cap F^{\mathcal{B}})$ of C that satisfy $r \in \delta^{\mathcal{A}}(p,a)$.
- A state (r, M, N) is accepting iff $r \in F^{\mathcal{A}}$.

Let $w \in \Sigma^*$ be some finite word and let (r, M, N) be a state from C that is reachable from the initial state ι. Then $r \in \delta^{\mathcal{A}}(p, w)$, $M = \delta^{\mathcal{B}}(q, w)$, and $N = \delta_F^{\mathcal{B}}(q, w)$. In addition, all states (r', M', N') on the path from ι to (r, M, N) satisfy $(0, r', \varepsilon, s, 3) \notin S_n$ for all $s \in N'$.

Hence the Büchi-automaton C accepts $\alpha = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ iff there is an accepting run $(r_i, a_i, r_{i+1})_{i \geq 0} \in \mathsf{ARuns}(\mathcal{A}[p])$ such that, for all $i \geq 0$, the set $\delta_F^{\mathcal{B}}(q, a_0 a_1 \dots a_i)$ does not contain any state s with $(0, r_i, \varepsilon, s, 3) \in S_n$. But this is the case iff $v \notin S_{n+1}$.

Since the size of the Büchi-automaton C is exponential in $|Q^{\mathcal{A}} \times Q^{\mathcal{B}}|$, the emptiness of its language can be decided in space polynomial in $|Q^{\mathcal{A}} \times Q^{\mathcal{B}}|$. ∎

Replacing in the proof of Theorem 26 the reference to Lemma 25 by a reference to Lemma 27, we obtain the following result.

Theorem 28 *The relation $\sqsubseteq_{\mathsf{fl}}^{(\omega)}$ is decidable in polynomial space. In other words, given two NBA $\mathcal{A}$ and $\mathcal{B}$ over the trace alphabet $(\Sigma, \sigma, 1)$ and the capacity function $\kappa\colon [1] \to \{\omega\}$, it can be decided in space polynomial in $|Q^{\mathcal{A}} \times Q^{\mathcal{B}}|$ whether Player 1 wins the multi-buffer flushing game $\mathcal{F}^\kappa(\mathcal{A}, \mathcal{B})$.*

4.3.4. One unbounded and some bounded buffers

We now consider the case of capacity functions κ with $\kappa(i) \in \mathbb{N}$ for all $i \in [k-1]$ and $\kappa(k) = \omega$. In this situation, we show that the parity flushing game can be solved algorithmically (consequently, the multi-buffer flushing game is solvable). To this aim, we abstract the word w in a position (X, p, w, q, r) of the parity flushing game. All that is needed to know about this word is how it behaves in the NBA $\mathcal{B}$, i.e., we need to know the sets

$$\delta^{\mathcal{B}}(q, [w]) := \bigcup_{w' \sim_\sigma w} \delta^{\mathcal{B}}(q, w') \text{ and } \delta_F^{\mathcal{B}}(q, [w]) := \bigcup_{w' \sim_\sigma w} \delta_F^{\mathcal{B}}(q, w').$$

Therefore, our first aim is to show that this information can be maintained while Player 0 makes his moves. More precisely, we need some finite piece of information on the word w that allows

1. the sets $\delta^{\mathcal{B}}(q, [w])$ and $\delta_F^{\mathcal{B}}(q, [w])$ to be determined, and
2. the corresponding information on wa for $a \in \Sigma$ to be computed.

The finite piece of information alluded to above is defined as follows.

Definition 29 Let $q \in Q^{\mathcal{B}}$ and $w \in \Sigma^*$ with $|\pi_i(w)| \leq \kappa(i)$ for all $i \in [k-1]$. The set $\mathcal{I}(q, w)$ consists of all tuples

$$t = (q', (u_i', v_i')_{i \in [k-1]}, b')$$

with $q' \in Q^{\mathcal{B}}$, $u_i', v_i' \in \Sigma_i^{\leq \kappa(i)}$ for all $i \in [k-1]$ and $b' \in \{0, 1\}$ such that there exists $z \in \Sigma^*$ satisfying

(i) $u'_i = \pi_i(w)$ and $v'_i = \pi_i(z)$ for all $i \in [k-1]$,
(ii) $\pi_k(z) = \pi_k(w)$,
(iii) $q' \in \delta^{\mathcal{B}}(q, z)$, and
(iv) $b' = 1 \iff q' \in \delta_F^{\mathcal{B}}(q, z)$.

Let $z \sim_\sigma w$ and $q' \in \delta^{\mathcal{B}}(q, z)$. Since $\pi_i(w) = \pi_i(z)$ for all $i \in [k]$, there exists a tuple $t = (q', (u'_i, v'_i)_{i\in[k-1]}, b')$ in $\mathcal{I}(q, w)$ with $u'_i = v'_i = \pi_i(w)$ for all $i \in [k-1]$. In addition, there is such a tuple with $b' = 1$ iff $q' \in \delta_F^{\mathcal{B}}(q, [w])$. Thus, the sets $\delta^{\mathcal{B}}(q, [w])$ and $\delta_F^{\mathcal{B}}(q, [w])$ can be determined from those tuples $t \in \mathcal{I}(q, w)$ that satisfy $u'_i = v'_i$ for all $i \in [k-1]$. But from these tuples alone, we cannot determine the corresponding tuples from $\mathcal{I}(q, wa)$. Therefore, in the above definition, the set $\mathcal{I}(q, w)$ contains tuples with $u'_i \neq v'_i$.

We now prove that $\mathcal{I}(q, wa)$ can be computed from $\mathcal{I}(q, w)$ and the letter $a \in \Sigma$.

Lemma 30 *Let $q^0 \in Q^{\mathcal{B}}$, $w \in \Sigma^*$ and $a \in \Sigma$ with $|\pi_i(wa)| \leq \kappa(i)$ for all $i \in [k-1]$. Then $\mathcal{I}(q^0, wa)$ is the set of tuples*

$$t^2 = (q^2, (u_i^2, v_i^2)_{i\in[k-1]}, b^2)$$

with $q^2 \in Q^{\mathcal{B}}$, $u_i^2, v_i^2 \in \Sigma_i^{\leq\kappa(i)}$ for all $i \in [k-1]$ and $b^2 \in \{0, 1\}$ such that there exist a tuple

$$t^1 = (q^1, (u_i^1, v_i^1)_{i\in[k-1]}, b^1) \in \mathcal{I}(q^0, w)$$

and a word $x \in \Sigma^*$ with

(a) $u_i^2 = u_i^1 \pi_i(a)$ *and* $v_i^2 = v_i^1 \pi_i(x)$ *for all* $i \in [k-1]$,
(b) $\pi_k(x) = \pi_k(w)$,
(c) $q^2 \in \delta^{\mathcal{B}}(q^1, x)$, *and*
(d) $b^2 = 1 \iff b^1 = 1$ *or* $q^2 \in \delta_F^{\mathcal{B}}(q^1, x)$.

Proof First suppose $t^1 \in \mathcal{I}(q^0, w)$ and $x \in \Sigma^*$ satisfying (a)-(d). From $t^1 \in \mathcal{I}(q^0, w)$, we obtain a word $z^1 \in \Sigma^*$ such that the tuple (w, t^1, z^1) satisfies (i)-(iv).

Set $z^2 = z^1\, x$ We have to verify conditions (i)-(iv) for (wa, t^2, z^2):

(i) $u_i^2 = u_i^1\, \pi_i(a) = \pi_i(w)\, \pi_i(a) = \pi_i(wa)$ and $v_i^2 = v_i^1\, \pi_i(x) = \pi_i(z^1)\, \pi_i(x) = \pi_i(z^2)$ for all $i \in [k-1]$.
(ii) $\pi_k(z^2) = \pi_k(z^1 x) = \pi_k(w)\, \pi_k(a) = \pi_k(wa)$.
(iii) $q^2 \in \delta^{\mathcal{B}}(q^1, x) \subseteq \delta^{\mathcal{B}}(q^0, z^1 x) = \delta^{\mathcal{B}}(q^0, z^2)$.
(iv) $b^2 = 1$ iff $b^1 = 1$ or $q^2 \in \delta_F^{\mathcal{B}}(q^1, x)$ iff $q^1 \in \delta_F^{\mathcal{B}}(q^0, z^1)$ or $q^2 \in \delta_F^{\mathcal{B}}(q^1, x)$ iff $q^2 \in \delta_F^{\mathcal{B}}(q^0, z^1 x) = \delta_F^{\mathcal{B}}(q^0, z^2)$.

Thus, indeed, $t^2 \in \mathcal{I}(q^0, wa)$.

Conversely, suppose $t^2 \in \mathcal{I}(q^0, wa)$. Then there exists $z^2 \in \Sigma^*$ such that (wa, t^2, z^2) satisfies (i)-(iv).

We define a tuple t^1 and a word z^1 such that (w, t^1, z^1) satisfies (i)-(iv) and such that (a)-(d) hold.

Since $\pi_k(w)$ is a prefix of $\pi_k(wa) = \pi_k(z^2)$, there exists a factorization $z^2 = z^1\, x$ with $\pi_k(w) = \pi_k(z^1)$. This already ensures (ii) for (w, t^1, z^1). Next let $u_i^1 = \pi_i(w)$ and $v_i^1 = \pi_i(z^1)$ for all $i \in [k]$. This ensures (i) for (w, t^1, z^1). Since w is a prefix of wa and z_1 is a prefix of z_2, we also get $|u_i^1| \leq |u_i^2| \leq \kappa(i)$ as well as $|v_i^1| \leq |v_i^2| \leq \kappa(i)$. Since $q^2 \in \delta^{\mathcal{B}}(q^0, z^2)$, there is some z^2-labeled path ρ from q^0 to q^2. If $q^2 \in \delta_F^{\mathcal{B}}(q^0, z^2)$, then chose ρ such that it contains some accepting state. Since $z^2 = z^1\, x$, there is a state q^1 such that ρ splits into some z^1-labeled path ρ_1 from q^0 to q^1 and some x-labeled path ρ_2 from q^1 to q^2. If ρ sees some accepting state, then one of the paths ρ_1 and ρ_2 sees some accepting state. In particular, we have $q^1 \in \delta^{\mathcal{B}}(q^0, z^1)$ (ensuring (iii) for (w, t^1, z^1)) and $q^2 \in \delta^{\mathcal{B}}(q^1, x)$ (ensuring (c)). Finally, set $b^1 = 1$ iff $q^1 \in \delta_F^{\mathcal{B}}(q^0, w)$ (ensuring (iv) for (w, t^1, z^1)).

Thus, we defined $t^1 = (q^1, (u_i^1, v_i^1)_{i \in [k-1]}, b^1) \in \mathcal{I}(q^0, w)$ and it remains to verify (a)-(d) (note that (c) has already been checked).

(a) $u_i^2 = \pi_i(wa) = u_i^1 \pi_i(w)$ and $v_i^2 = \pi_i(z^2) = \pi_i(z^1\, x) = v_1^1\, \pi_i(x)$ for all $i \in [k-1]$.

(b) $\pi_k(w)\, \pi_k(x) = \pi_k(z^1 x) = \pi_k(z^2) = \pi_k(wa) = \pi_k(w)\, \pi_k(a)$ implies $\pi_k(x) = \pi_k(a)$.

(d) $b^2 = 1$ iff $q^2 \in \delta_F^{\mathcal{B}}(q^0, z^2)$ iff $q^2 \in \delta_F^{\mathcal{B}}(q^1, x)$ or $q^1 \in \delta_F^{\mathcal{B}}(q^0, z^1)$ (by the choice of the path ρ) iff $b^1 = 1$ or $q^2 \in \delta_F^{\mathcal{B}}(q^1, x)$.

∎

The information $\mathcal{I}(q, w)$ can be used to define a bisimulation of the parity flushing game that is also an equivalence relation. Since the quotient of the parity flushing game wrt. this simulation is finite, the quotient game and therefore the parity flushing game is solvable.

Theorem 31 *The relations $\sqsubseteq_{\mathrm{fl}}^{(c_1, c_2, \ldots, c_{k-1}, \omega)}$ (for $k \geq 1$ and $c_1, \ldots, c_{k-1} \in \mathbb{N}$) are decidable in doubly exponential time, even uniformly in k and the tuple $(c_1, \ldots, c_{k-1}) \in \mathbb{N}^{k-1}$.*

In other words, given two NBA $\mathcal{A}$ and $\mathcal{B}$ over the trace alphabet (Σ, σ, k) and a capacity function $\kappa \colon [k] \to \mathbb{N} \cup \{\omega\}$ with $\kappa(i) \in \mathbb{N}$ for all $i \in [k-1]$ and $\kappa(k) = \omega$, it can be decided in doubly exponential time whether Player 1 wins the multi-buffer flushing game $\mathcal{F}^{\kappa}(\mathcal{A}, \mathcal{B})$.

Proof By Theorem 20, it suffices to solve the parity flushing game $\mathcal{F}_{\mathrm{par}}^{\kappa}(\mathcal{A}, \mathcal{B})$.

Let R be the set of pairs

$$\Big((X_1, p_1, w_1, q_1, r_1), (X_2, p_2, w_2, q_2, r_2)\Big)$$

of positions of the parity flushing game $\mathcal{F}_{\mathrm{par}}^{\kappa}(\mathcal{A}, \mathcal{B})$ with

- $X_1 = X_2$, $p_1 = p_2$, $q_1 = q_2$, $r_1 = r_2$, and
- $\mathcal{I}(q_1, w_1) = \mathcal{I}(q_2, w_2)$.

Then the relation R is clearly an equivalence relation and it is not difficult to verify that it is also a bisimulation of $\mathcal{F}_{\mathrm{par}}^{\kappa}(\mathcal{A}, \mathcal{B})$ and $\mathcal{F}_{\mathrm{par}}^{\kappa}(\mathcal{A}, \mathcal{B})$: Let $(v_1, v_2) \in R$. Then $v_i = (X, p, w_i, q, r)$ for $i \in \{1, 2\}$ and we get the following:

1. $v_1 \in V_0^{\text{par}} \iff X = 0 \iff v_2 \in V_0^{\text{par}}$
2. $\Omega^{\text{par}}(v_1) = r = \Omega^{\text{par}}(v_2)$
3. Suppose $v_1 \in V_1^{\text{par}}$ and $(v_1, v_1') \in E^{\text{par}}$. Then there is some word $u_1 \in \{\varepsilon\} \cup [w_1]$ that justifies the move from v_1 to v_1'.
   If $u_1 = \varepsilon$, then $v_1' = (0, p, w_1, q, 1)$. In this case, we set $v_2' = (0, p, w_2, q, 1)$ which satisfies $v_2' \in v_2 E^{\text{par}}$ and $(v_1', v_2') \in R$.
   Now let $u_1 \neq \varepsilon$ and therefore $u_1 \sim_\sigma w_1$. Then $v' = (0, p, \varepsilon, q', r')$ with $q' \in \delta^{\mathcal{B}}(q, u_1)$ and $r' = 1$ or $q' \in \delta_F^{\mathcal{B}}(q, [u_1]) = \delta_F^{\mathcal{B}}(q, [w_1])$. Consider the tuple $t = (q', (u_i', v_i')_{i \in [k-1]}, b)$ from $\mathcal{I}(q, w_1)$ that results from this word $z = u_1$. Note that $u_i' = v_i'$ since $z = u_1 \sim_\sigma w_1$. Since $\mathcal{I}(q, w_1) = \mathcal{I}(q, w_2)$, this tuple also belongs to the latter set. Hence there exists a word $u_2 \in \Sigma^*$ that induces this tuple in $\mathcal{I}(q, w_2)$. From $u_i' = v_i'$ for $i \in [k-1]$, we obtain $u_2 \sim_\sigma w_2$. Set $v_2' = (0, p, \varepsilon, q', r')$. Then $v_2' \in v_2 E^{\text{par}}$ and $(v_1', v_2') \in R$.
4. Similarly, we can argue if $v_1 \in V_0^{\text{par}}$ and $(v_1, v_1') \in E^{\text{par}}$.

Consequently, by Lemmas 9 and 6, Player 1 wins the parity flushing game $\mathcal{F}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})$ if and only if she wins the quotient game $\mathcal{G} = \mathcal{F}^\kappa_{\text{par}}(\mathcal{A}, \mathcal{B})/_R$.

The positions of this quotient game $\mathcal{G}$ can be given as tuples (X, p, W, q, r) for some $X \in \{0, 1\}$, $p \in Q^{\mathcal{A}}$, $q \in Q^{\mathcal{B}}$, $r \in \{0, 1, 2\}$ and W a subset of

$$Q^{\mathcal{B}} \times \prod_{i \in [k-1]} \left(\Sigma_i^{\leq \kappa(i)} \times \Sigma_i^{\leq \kappa(i)} \right) \times \{0, 1\}.$$

Hence the number of positions of the quotient game $\mathcal{G}$ is bounded by

$$2 \cdot |Q^{\mathcal{A}}| \cdot 2^{|Q^{\mathcal{B}}| \cdot \prod_{i \in [k-1]} |\Sigma_i|^{2(\kappa(i)+1)} \cdot 2} \cdot |Q^{\mathcal{B}}| \cdot 3 \leq 6 \cdot |Q^{\mathcal{A}}| \cdot |Q^{\mathcal{B}}| \cdot 4^{|Q^{\mathcal{B}}| \cdot |\Sigma|^{2(K+k)}}$$

with $K = \sum_{i \in [k-1]} \kappa(i)$. In other words, the game $\mathcal{G}$ is polynomial in the size of the NBA $\mathcal{A}$, exponential in the size of the NBA $\mathcal{B}$ and the size of the alphabets, and doubly exponential in the size and number of buffers. Hence the claim follows from Proposition 4. ∎

5. Lower Bounds

The aim of this section is to show that the upper bounds of the previous section are tight by providing matching lower bounds. The general upper bound of Δ_2^1 from Cor. 13 for solving unbounded multi-buffer games is an exception, since this class does not contain complete problems [21, Theorem 16.1.X]. Instead we will provide a lower bound that gets very close to it. We start with the computationally easier cases of single-buffer games, though.

5.1. A PSPACE lower bound for unbounded single-buffer flushing games

A typical PSPACE-complete problem is the *n-corridor tiling problem* [23]. Its input is a tiling system $\mathcal{T} = (T, H, V, t_I)$, where T is a finite set of tiles, $H, V \subseteq T \times T$ are the horizontal and vertical matching relations, $t_I \in T$ is the initial tile, and a natural number n encoded unarily. The problem is to decide whether there is a function $\tau : \mathbb{N} \times \{0, \ldots, n-1\} \to T$ satisfying

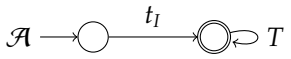
1. $t(0,0) = t_I$,
2. for all $i \in \mathbb{N}$, all $j < n-1$ we have $(\tau(i,j), \tau(i,j+1)) \in H$,
3. for all $i \in \mathbb{N}$, all $j < n$ we have $(\tau(i,j), \tau(i+1,j)) \in V$.

Such a function is also called a *valid tiling*.

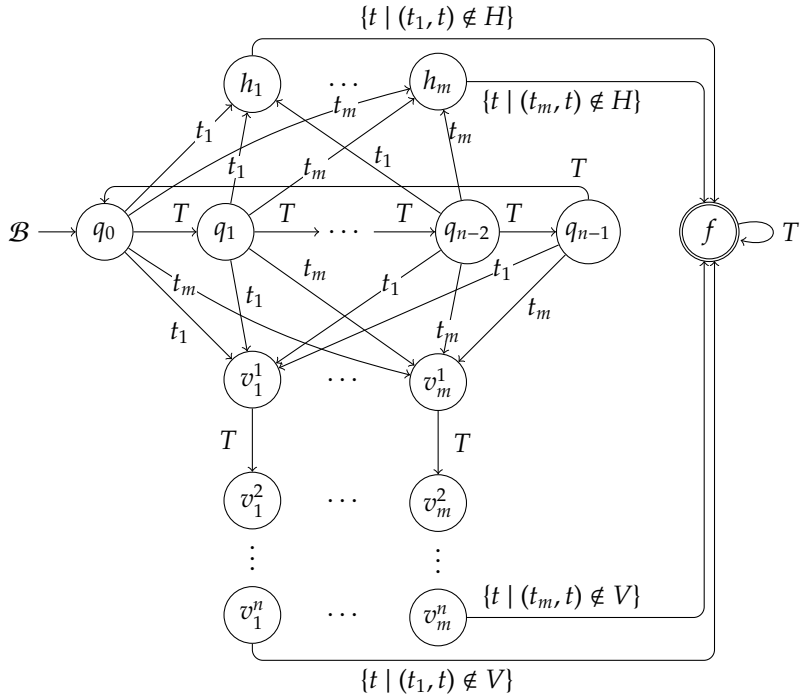
Theorem 32 *Deciding $\sqsubseteq_{\mathsf{fl}}^{(\omega)}$ is PSPACE-hard.*

Proof By a reduction from the n-corridor tiling problem. Given a tiling system $\mathcal{T} = (T, H, V, t_I)$ and a unarily encoded n, we construct two NBA $\mathcal{A}, \mathcal{B}$ over the (trace) alphabet T such that Player 0 wins $\mathcal{F}^{(\omega)}(\mathcal{A}, \mathcal{B})$ iff there is a valid $\mathcal{T}$-tiling for the n-corridor. Note that PSPACE is closed under complements. Hence, we also get PSPACE-hardness of the question whether Player 1 wins this game.

W.l.o.g. assume that $T = \{t_1, \ldots, t_m\}$ and $t_I = t_1$. Intuitively, $\mathcal{A}$ allows Player 0 to construct a tiling of the n-corridor by simply choosing the tiles $\tau(0,0), \tau(0,1), \ldots, \tau(0,n-1), \tau(1,0), \ldots, \tau(1,n-1), \tau(2,0), \ldots$



The intuition behind Player 1's NBA $\mathcal{B}$ is that it allows her to spot any mistakes which Player 0 might have made. Note that $\mathcal{A}$ has an accepting run for any sequence of tiles starting with t_I, not just those that respect the horizontal and vertical matching relation on the n-corridor. $\mathcal{B}$ allows Player 1 to wait until Player 0 produces a tile that is either not matching vertically to the one in the row below, i.e. the n-last in the sequence that Player 0 produced; or does not match the immediate predecessor in this sequence, with the exception of every n-th tile at the beginning of each row. $\mathcal{B}$ makes use of nondeterminism to let Player 1 guess which tile Player 0 cannot produce a match for.



The size of $\mathcal{A}$ is 2, and that of $\mathcal{B}$ is $(n+1)\cdot(|T|+1)$, and both can clearly be constructed in polynomial time. It remains to be seen that Player 1 wins $\mathcal{F}^\omega(\mathcal{A},\mathcal{B})$ iff there is no valid $\mathcal{T}$-tiling of the n-corridor.

"$\Rightarrow$" Suppose there is a valid tiling τ. A strategy for Player 0 consists of playing the sequence $\tau(0,0),\tau(0,1),\ldots,\tau(0,n-1),\tau(1,0),\ldots$ as above. The only run on $\mathcal{B}$ that is trace-equivalent to this is cycling through the sequence $(q_0,\ldots,q_{n-1})$ of states of $\mathcal{B}$, i.e. it does not form an accepting run. Hence, this is a winning strategy for Player 0.

"$\Leftarrow$" Suppose conversely that there is no valid tiling for the n-corridor. Take an arbitrary word $w = a_0a_1\ldots \in t_I T^\omega$ which Player 0 could play. This induces a tiling τ_w in a natural way: $\tau_w(i,j) = a_{i\cdot n+j}$. By assumption, no such τ_w is a valid tiling. Since all of them start with t_I, it must contain a horizontal or a vertical mismatch, i.e. there is an $h\in\mathbb{N}$ such that

1. $(a_{h-1},a_h)\notin H$ and $h\not\equiv 0 \pmod n$, or
2. $(a_{h-n},a_h)\notin V$.

This gives Player 1 the following strategy χ: she skips her turns until Player 0 has played a_h. At this point, the buffer content is $a_0\ldots a_h$. Let (i,j) be uniquely chosen such that $h = i\cdot n+j$ and $j<n$. Player 1 then flushes the entire buffer producing one of the following runs.

- If the defect is a horizontal one then we must have $j>0$, and there are x,y such that $a_{h-1}=t_x$, $a_h=t_y$ and $(t_x,t_y)\notin H$. Player 1 chooses the run $(q_0,\ldots,q_{n-1})^i,q_0,\ldots,q_{j-1},h_x,f$ to flush the buffer. From then on she can

consume every letter played by Player 0 whilst staying in state f and thus producing an accepting run.

- If the defect is a vertical one then we have $i > 0$ and there are x, y such that $a_{h-n} = t_x$, $a_h = t_y$ and $(t_x, t_y) \notin V$. Likewise, Player 1 can now flush the buffer along the sequence $(q_0, \ldots, q_{n-1})^{i-1}, q_0, \ldots, q_j, v_x^1, \ldots, v_x^n, f$ and then react to each of Player 0's moves immediately, producing an accepting run.

Note that χ does not depend on the word w played by Player 0. It is a winning strategy in the flushing game because it allows Player 1 to produce an accepting run regardless of which word is chosen by Player 0, and whenever it requires her to move she can do so by flushing the buffer. ∎

We note that the NBA in the construction above is very small with only two states. It is a standard exercise to amend $\mathcal{B}$ such that Player 1 can check whether Spoiler played t_I first, thus reducing $\mathcal{A}$ to a single state only. Moreover, there is a fixed tiling system for which the n-corridor tiling problem is PSPACE-hard (similar to the availability of universal Turing machines). Using standard coding tricks, it is possible to reduce the alphabet to a binary one, resulting in the following.

Corollary 33 *There is an NBA $\mathcal{A}$ such that deciding whether Player 1 wins the flushing game $\mathcal{F}^{(\omega)}(\mathcal{A}, \mathcal{B})$ for a given NBA $\mathcal{B}$ is PSPACE-hard.*

5.2. An EXPTIME lower bound for unbounded single-buffer simulation games

Next we turn to the non-flushing variant, i.e. the single-buffer simulation game played with an unbounded buffer. We show that this is – under some standard complexity-theoretic assumptions – harder than the flushing variant, namely EXPTIME-hard. A candidate for a suitable reduction is the tiling *game* played on the corridor of width n [3]. It is played by two players called Starter and Completer in order to produce a valid tiling of the n-corridor, given some tiling system $\mathcal{T} = (T, H, V, t_I)$. The game proceeds in rows, beginning with the lowest. Starter chooses the tile in the first column with the exception of the first row; here the tile has to be t_I. Then Completer chooses tiles for the remaining columns in this row. A player wins if their opponent is unable to place a tile that matches its horizontal and vertical predecessors. (Note that Starter never has to obey the horizontal matching relation but only the vertical one.) Additionally, Completer wins any infinite play.

Intuitively, the two players in the simulation game model the two players in the tiling game: as in the reduction above, Player 0 produces the tilings for the rows and Player 1 checks that the tiling is correct. Additionally, it should be Player 0's opponent, i.e. Player 1, who chooses the first tile of each row. This is problematic in principle because in the simulation game on an unbounded buffer, there is no obvious mechanism which can be used to force Player 1 to make a move. Moreover, Player 0 has to move in each round; so a more

sophisticated trick is needed to model the interaction of the players in the tiling game.

We will construct Player 1's NBA as a product of four automata. The first three monitor the content of the buffer and check that it corresponds to a sensible encoding of a tiling of the n-corridor. In the fourth component, Player 1 keeps a record of the first tile of the currently constructed row and, by moving to a certain state, implicitly makes a choice about the first tile of the next row.

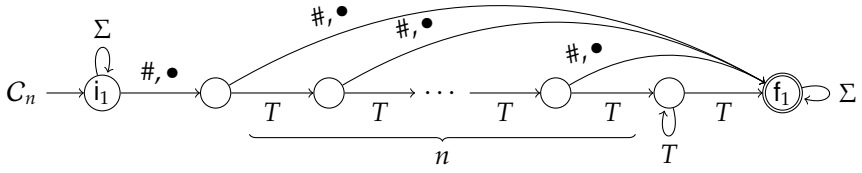
To formalise this, let $\mathcal{T} = (T, H, V, t_I)$ be a tiling system and $n \in \mathbb{N}$. W.l.o.g. we assume $T = \{t_1, \dots, t_m\}$ and $t_I = t_1$. We use the alphabet $\Sigma = T \cup \{\#, \bullet\}$. The two additional symbols are used to mark the beginning of a row in a row-wise encoding of a tiling of the n-corridor.

Let τ be a tiling of the n-corridor. We say that a word $w \in \Sigma^\omega$ encodes τ, if it is of the form $(\# T^n(\bullet T^n)^*)^\omega$ such that the following holds. Let

$$w = \# v_{0,0} \bullet \dots \bullet v_{0,n_0} \# v_{1,0} \bullet \dots \bullet v_{1,n_1} \# v_{2,0} \bullet \dots \bullet v_{2,n_2} \# \dots$$

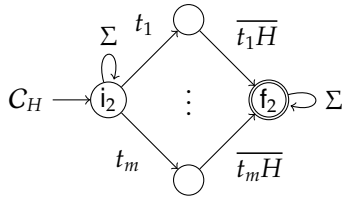
such that $v_{i,j} \in T^n$ for all i, j. We then require that $v_{i,j} = \tau(i,0) \dots \tau(i, n-1)$ for each $i \in \mathbb{N}$ and $j \leq n_i$. Note that this implies that $v_{i,j} = v_{i,j'}$ for all i, j, j', i.e. all the T-words before the i-th occurrence of # represent the i-th row of the tiling τ.

The first automaton C_n checks that Player 0 produces sequences of the form $((\# + \bullet)T^n)^\omega$ in the sense that whenever he does *not*, then Player 1 can reach a particular state. Note that this is just one component of Player 1's NBA $\mathcal{B}$.



Lemma 34 *There is a run in C_n from i_1 to f_1 under a word w iff $w = u(\# + \bullet)v\,v'$ for some u, v, v' such that v is not of the form $T^n(\# + \bullet)$.*

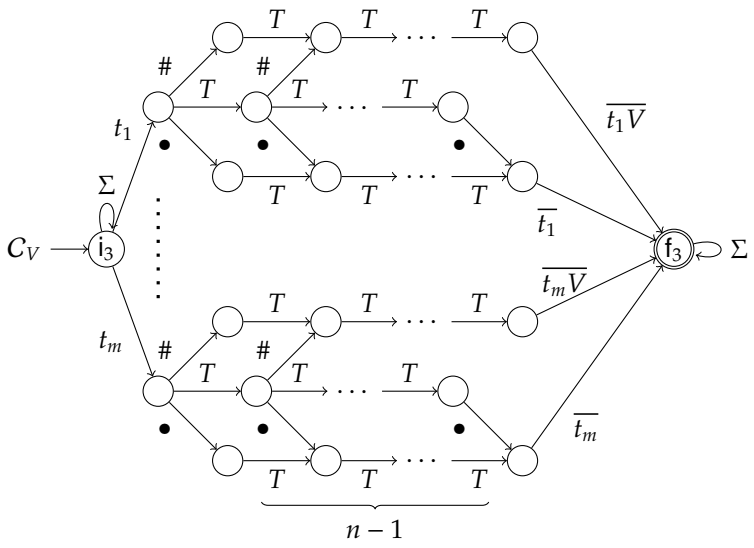
The next component is used by Player 1 to check that each adjacent two tiles match horizontally. We write $\overline{tH}$ for $\{t' \in T \mid (t, t') \notin H\}$ and likewise for $\overline{tV}$.



Lemma 35 *There is a run in C_H from i_2 to f_2 under a word w iff $w = u\,t\,t'\,, v$ for some u, t, t', v such that $t, t' \in T$ and $(t, t') \notin H$.*

The third component is used to check the vertical matching relation. It examines two letters representing tiles that are exactly $n+1$ positions apart and

assumes that in between there is exactly one occurrence of either # or •. If it is • then the two tiles must be equal. If it is # then they must be adjacent with repsect to V.



Note the asymmetry w.r.t. the handling of • and # at the end of each of the triples of horizontal rows. This ensures that Player 1 can only detect mismatches of the form $t\,T^i\,\#\,t'$ with $(t,t') \notin V$ and $i < n-1$ because only then do t and t' represent tiles that are not first in their respective row. Remember that Player 1 needs to choose the first tiles in each row, hence she should not be able to win by deliberately creating a mismatch and then detecting it herself.

Lemma 36 *There is a run in C_V from i_3 to f_3 under a word w iff $w = u\,t\,u'\,x\,v'\,t'\,v$ for some $t,t' \in T$, $u,v \in \Sigma^*$, $u',v' \in T^*$, $x \in \{\#,\bullet\}$ such that*

- $|u'\,v'| = n-1$,
- $t \neq t'$ *if* $x = \bullet$,
- $(t,t') \notin V$ *and* $|u'| < n-1$ *if* $x = \#$.

The last component C is used to make Player 1 choose the first tile in each row. It is shown in Fig. 1. C has two states $\mathsf{c}_i, \mathsf{c}'_i$ for each tile t_i. Intuitively, by being in state c_i, Player 1 has made a choice of t_i as the first tile in the row currently under construction. In these states she can read any symbol but # which takes her to c'_i. Here, she needs any letter other than t_i in order to reach f_4. With t_i she can get to any $\mathsf{c}_{i'}$ for $i' \in [m]$ such that $(t_i, t_{i'}) \in V$.

Lemma 37 *There is a run in C from any c_i to f_4 under a word w containing a single occurrence of #, iff w is of the form $u\,\#\,t_j v$ and $j \neq i$.*

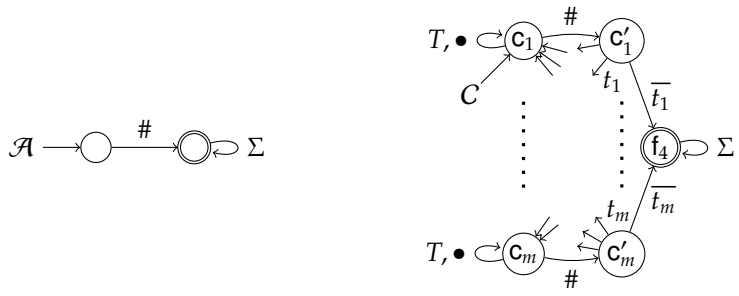


Figure 1: Automata used in the proof of Thm. 38.

Theorem 38 *Deciding $\sqsubseteq^{(\omega)}$ is EXPTIME-hard.*

Proof By a reduction from the n-corridor tiling game problem. Let $\mathcal{T}$ and n be given as above. We construct two NBA $\mathcal{A}$ and $\mathcal{B}$ of size polynomial in $|T|$ and (the value of) n such that Player 1 wins $\mathcal{G}^{(\omega)}(\mathcal{A}, \mathcal{B})$ iff Starter wins the $\mathcal{T}$-tiling game on the n-corridor. Player 0's NBA $\mathcal{A}$ is shown in Fig. 1 on the left. .

Player 1's NBA $\mathcal{B}$ is obtained as the synchronous product of C_n, C_H, C_V and C. Its initial state is $(\mathsf{i}_1, \mathsf{i}_2, \mathsf{i}_3, \mathsf{c}_1)$. Transitions are of the form $(q_1, q_2, q_3, q_4) \xrightarrow{a} (q'_1, q'_2, q'_3, q'_4)$ whenever $q_i \xrightarrow{a} q'_i$ for all $i = 1, \ldots, 4$. The acceptance condition is disjunctive, deviating from the usual notion of a synchronous product of finite automata: a state (q_1, q_2, q_3, q_4) is accepting iff $q_i = \mathsf{f}_i$ for some $i = 1, \ldots, 4$. Clearly, the size of $\mathcal{A}$ is $O(1)$, and the size of $\mathcal{B}$ is polynomial in $|\mathcal{T}|$ and n. Correctness of this construction remains to be shown.

"⇒" By contraposition. Suppose that Completer has a winning strategy χ for the n-corridor $\mathcal{T}$-tiling game. We write $\chi(i, j)$ with $i \in \mathbb{N}, 0 < j < n$ for the tile placed at position (i, j) when all previous rows and all previous columns in row i have been filled already. Note that Completer's choices with χ will naturally depend on Starter's choices of the first tiles in the current and previous rows.

We describe a strategy for Player 0 in $\mathcal{G}^{(\omega)}(\mathcal{A}, \mathcal{B})$. It divides the game into phases. The i-th phase in the simulation game corresponds to the construction of the i-th row in the tiling game. At the beginning of each phase, Player 1 has her pebble on a state $(\mathsf{i}_1, \mathsf{i}_2, \mathsf{i}_3, \mathsf{c}_h)$ for some $h \in [m]$. Let $w_i = t_h\, \chi(i, 1) \ldots \chi(i, n-1)$. Player 0 plays the letters of the word $\# w_i (\bullet w_i)^\omega$ for as long as Player 1 remains in state c_h (of the 4th component in $\mathcal{B}$). Note that Player 1 can only move out of c_h by moving to c'_h with #, and this is the first letter in the buffer at the beginning. Moreover, since w_i starts with t_h, Player 1 cannot reach f_4. Hence, in order not to lose by never moving she eventually has to make the move $\mathsf{c}_h \xrightarrow{\#} \mathsf{c}'_h$ which can only be followed by a move $\mathsf{c}'_h \xrightarrow{t_h} \mathsf{c}_{h'}$ such that $(t_h, t_{h'}) \in V$. Now she is on a non-accepting state and would therefore also lose unless she eventually moves to some $\mathsf{c}_{h'}$. Then Player 0 possibly continues until he has played a word of the form $\# w_i (\bullet w_i)^*$. This completes the i-th phase, and Player 0 moves to the $(i+1)$-th phase.

It should be clear that any run constructed by Player 1 in this way is accepting. Thus, it remains to be seen that Player 0 cannot construct an accepting run against this strategy. In order to do so, she would have four possibilities, namely reaching any of the states $\mathsf{f}_1, \ldots, \mathsf{f}_4$ in the respective component. She cannot reach f_1 by Lemma 34 because there are always exactly n letters from T between each times that Player 0 play # or $\bullet$. She also cannot reach f_2 or f_3 by Lemmas 35 and 36 since χ is supposed to be a winning strategy in the tiling game. Hence, Player 0 does not produce horizontal mismatches anywhere, or vertical mismatches in either of the columns $1, \ldots, n-1$.

Finally, she cannot reach f_4 because of the invariant about phases stated above and Lemma 37: when Player 0 is in state c_i of component C, then the letter following the next # in the buffer is t_i.

"$\Leftarrow$" In a similar way. Suppose that Starter has a winning strategy χ in the tiling game. We write $\chi(i)$ to denote the tile that is placed at position $(i, 0)$ by Starter according to strategy χ, provided that the rows $0, \ldots, i-1$ have been constructed already. This induces a strategy for Player 1 in the simulation game. Starting in position $(\mathsf{i}_1, \mathsf{i}_2, \mathsf{i}_3, \mathsf{c}_1)$ she maintains the following invariant: she consumes any $x \in T \cup \{\bullet\}$ until the buffer content is of the form $\# T^n (\bullet\, T^n)^+$. If the T-parts between each occurrence of # or $\bullet$ are not exactly of length n then she moves to state f_1 using Lemma 34. If there is a horizontal mismatch in any of these words she moves to f_2 using Lemma 35; and if the T-parts are not identical then she moves to f_3 using Lemma 36. Otherwise, she interprets any such T-part as the tiles in the previous row $i-1$, and then moves to state c_h if $\chi(i) = t_h$, via c'_i.

Since Completer would win any infinite play in the tiling game, he must eventually get stuck, i.e. produce a horizontal mismatch (which Player 1 can deal with as described above) or a vertical mismatch. This can also be detected by Player 1 once Player 0 has placed the tiles of the second row: then she can always play such that the buffer content contains a word in T^n before the first occurrence of #, representing the current row, followed by the next row. Using Lemma 36 she can move to q_3 in case there was a vertical mismatch. ∎

5.3. Undecidability for flushing games

In the remainder of this section, we provide undecidability results. We start with the relatively simple case of two unbounded buffers for the flushing game. We give a reduction from the following problem, known as the ω-regular Post's Correspondence Problem (ω-PCP(REG)).

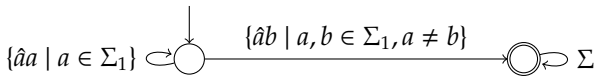
Input:	a list of pairs of non-empty words $\big((u_1, v_1), \ldots, (u_n, v_n)\big)$ over some alphabet Δ, and an NBA C over the alphabet $[n]$
Output:	is there an ω-word $w = i_0 i_1 i_2 \ldots \in L(C)$ such that $u_{i_0} u_{i_1} u_{i_2} \ldots = v_{i_0} v_{i_1} v_{i_2} \ldots$?

Theorem 39 *Let $\kappa = (\omega, \omega, \kappa')$ be a vector of capacities with at least two unbounded buffers. Then $\sqsubseteq^{\kappa}_{\mathsf{fl}}$ is Π^1_1-hard.*

Proof It suffices to prove the claim for $\kappa = (\omega, \omega)$ since the availability of more buffers (that are possibly unused by the trace alphabet) cannot make the problem simpler.

The proof provides a reduction from the complement of ω-PCP(REG) which is known to be Σ_1^1-hard [10]. Let $P = \Big((u_1, v_1), \ldots, (u_n, v_n)\Big)$ be a list of pairs of words over some finite alphabet Δ and an NBA C over $[n]$ be given. We construct two NBA $\mathcal{A}$ and $\mathcal{B}$ over a 2-partitioned alphabet $\Sigma := \Sigma_1 \cup \Sigma_2$ with $\Sigma_1 := \Delta$ and $\Sigma_2 := \{\hat{a} \mid a \in \Sigma_1\}$ as follows. For a word $w \in \Sigma_1^*$ we write $\widehat{w}$ for its translation into Σ_2 via the homomorphic extension of $\hat{\cdot}$.

$\mathcal{A}$ is obtained from C by replacing every transition of the form $p \xrightarrow{i} p'$ by a sequence of transitions (adding states correspondingly) for the word $u_i\widehat{v_i}$. $\mathcal{B}$ is the following NBA.



We claim that Player 0 wins $\mathcal{F}^{(\omega,\omega)}(\mathcal{A}, \mathcal{B})$ iff (P, C) is a positive instance of ω-PCP(REG). For the direction "$\Leftarrow$" suppose there is a $w = i_0 i_1 \ldots \in [n]^\omega$ such that (1) $w \in L(C)$ and (2) $u_{i_0} u_{i_1} \ldots = v_{i_0} v_{i_1} \ldots$. Let ρ be an accepting run of C on w. This induces a strategy for Player 0 in $\mathcal{F}^{(\omega,\omega)}(\mathcal{A}, \mathcal{B})$: starting in the initial state and with $j = 0$, if the j-th letter of w is i then he moves to the $(j+1)$-th state in ρ, thus putting u_i into buffer 1 and $\widehat{v_i}$ into buffer 2. Because of assumption (1), the run he produces in $\mathcal{A}$ is accepting. Because of (2), the limit of the contents of the two buffers is the same, which means that Player 1 cannot reach her accepting state. Hence, this is a winning strategy for Player 0.

For the direction "$\Rightarrow$" suppose that (P, C) is not a positive instance to ω-PCP(REG). Hence, for every word $w = i_0 i_1 \ldots$ we have (1) $w \notin L(C)$ or (2) there is a minimal $j \in \mathbb{N}$ such that the j-th letters of $u_{i_0} u_{i_1} \ldots$ and $v_{i_0} v_{i_1} \ldots$ differ. Consider any accepting run ρ by Player 0 through $\mathcal{A}$. It determines a unique word $w_\rho \in [n]^\omega$ with $w \in L(C)$. Hence, assumption (2) must be the case. Then Player 0 can wait until both buffers contain at least j many elements. Remember that P only contains non-empty words. Hence, such a moment must be reached eventually. Then Player 1 can flush both buffers whilst moving to her accepting state. From then on, she can react immediately to any letter put into either buffer, forming an accepting run. Hence, this is a winning strategy for Player 1. ∎

5.4. Undecidability for simulation games

An immediate consequence of the proof of Thm. 39 is Π_1^1-hardness of $\sqsubseteq^{(\omega,\omega)}$. Note that Player 0's winning strategy in the proof above also prevails against any of Player 1's strategies in which she is allowed not to flush the buffers entirely at any point. However, for simulation games we can show an even higher degree of undecidability (including Σ_1^1-hardness as well) for an even more restricted form of buffer capacities, namely $(\omega, 0)$.

For any class $\mathcal{S}$ of languages let $\mathbb{B}(\mathcal{S})$ denote the closure of $\mathcal{S}$ under the Boolean operations union, intersection and complement. We write $\mathbb{B}^+(\mathcal{S})$ for the positive Boolean closure, i.e. closure under union and intersection only. Due to the deMorgan laws, $\mathbb{B}(\mathcal{S}) = \mathbb{B}^+(\mathcal{S} \cup \{\overline{L} \mid L \in \mathcal{S}\})$.

We are particularly interested in the class $\mathbb{B}(\Sigma^1_1)$ or, equivalently, $\mathbb{B}^+(\Sigma^1_1 \cup \Pi^1_1)$.

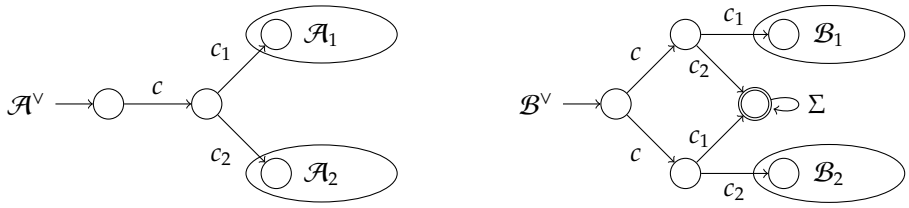
Lemma 40 *From two pairs of NBA $(\mathcal{A}_1, \mathcal{B}_1)$ and $(\mathcal{A}_2, \mathcal{B}_2)$ over some trace alphabet (Σ, σ, k) and some vector $\kappa = (\kappa', 0)$ of capacities, one can construct pairs of NBA $(\mathcal{A}^\vee, \mathcal{B}^\vee)$ and $(\mathcal{A}^\wedge, \mathcal{B}^\wedge)$ such that*

a) Player 1 wins $\mathcal{G}^\kappa(\mathcal{A}^\vee, \mathcal{B}^\vee)$ iff she wins one of the games $\mathcal{G}^\kappa(\mathcal{A}_1, \mathcal{B}_1)$ and $\mathcal{G}^\kappa(\mathcal{A}_2, \mathcal{B}_2)$,

b) Player 1 wins $\mathcal{G}^\kappa(\mathcal{A}^\wedge, \mathcal{B}^\wedge)$ iff she wins both games, $\mathcal{G}^\kappa(\mathcal{A}_1, \mathcal{B}_1)$ and $\mathcal{G}^\kappa(\mathcal{A}_2, \mathcal{B}_2)$.

Proof The simple constructions hinge on the fact that the positive Boolean combinations $\vee$ and $\wedge$ can be mimicked by choices made by the players, namely Player 0 in case of conjunctions and Player 1 for disjunctions. Let c, c_1, c_2 be three new symbols that are assigned to the last buffer (that has capacity 0).

(a) Consider the following NBA



Note that by reacting appropriately to Player 0's first choice of c, Player 1 makes an effective choice of whether to continue the play on $\mathcal{A}_1$ and $\mathcal{B}_1$, or on $\mathcal{A}_2$ and $\mathcal{B}_2$.

(b) The conjunction can be modelled in an even simpler way because logically it should be Player 0 who makes the choice which of the two subgames to play in. This is achieved using a single new state with c_1- and c_2-transitions into the initial states of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, and the same on the other side with $\mathcal{B}_1$ and $\mathcal{B}_2$. ∎

Lemma 40 allows us to transfer hardness results for simulation games to positive Boolean closures, provided that an appropriate buffer is available.

Corollary 41 *Let κ be a capacity vector s.t. $\kappa = (\kappa', 0)$ for some κ'. If deciding $\sqsubseteq^\kappa$ is hard for a class $\mathcal{S}$ of languages, then it is also hard for $\mathbb{B}^+(\mathcal{S})$.*

Hence, all that remains to see is that deciding $\sqsubseteq^{(\omega,0)}$ is hard for both Σ^1_1 and Π^1_1. The latter is relatively easy.

Theorem 42 *Deciding $\sqsubseteq^{(\omega,0)}$ is Π^1_1-hard.*

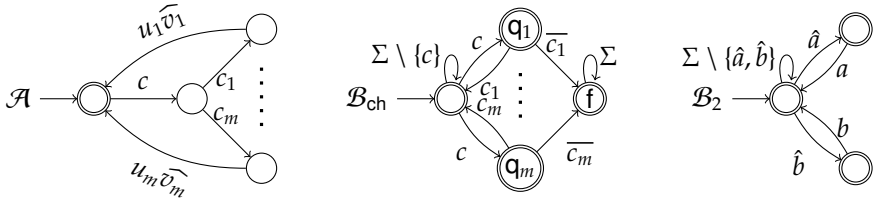


Figure 2: NBA used to show that $\sqsubseteq^{(\omega,0)}$ is also Σ^1_1-hard.

Proof By adjusting the proof of Thm. 39 which uses capacities (ω, ω) appropriately. First we note that the standard Σ^1_1-hardness proof of the ω-PCP(REG) is obtained by a reduction from the problem of deciding whether a given Turing machine has an infinite computation that satisfies a Büchi condition [10, 12]. The reduction produces a list of word pairs $\Big((u_i, v_i) \mid i = 1, \ldots, n\Big)$ such that for all $i = 1, \ldots, n$ we have $|u_i| \geq |v_i|$.

Now consider the flushing game constructed from such instances in the proof of Thm. 39. During the course of the game, buffer 2 can never contain a word that is longer than the current word in buffer 1. This is because the rules require Player 1 to either skip her turn or flush both buffers. Hence, she cannot remove something from buffer 1 unless she also does so for buffer 2 and in both cases, it has to be the entire content. Moreover, since $|u_i| \geq |v_i|$ for all i, and Player 0 can only add buffer content in the form $u_i\widehat{v_i}$ we maintain the invariant that at any position in the game where the buffer contents are (w_1, w_2), we have $|w_1| \geq |w_2|$.

In order to prove the claim of the current theorem, which regards the *simulation* rather than *flushing* game, we note that Player 1 is now allowed to remove content from any buffer without flushing it, in particular buffer 2. Since she can never reach a situation in which buffer 2 contains a letter but buffer 1 does not, and her automaton $\mathcal{B}$ as shown in the proof of Thm. 39 allows her to always make steps by taking one letter from each buffer, she can in fact play in such a way that any letter put into buffer 2 is immediately removed. Hence, if she is not required to play according to the flushing game rules then she can already win using buffer capacities $(\omega, 0)$ only. ∎

Theorem 43 *Deciding $\sqsubseteq^{(\omega,0)}$ is Σ^1_1-hard.*

Proof Again, we use the ω-PCP(REG). However, here we construct two NBA $\mathcal{A}, \mathcal{B}$ such that the game $\mathcal{G}^{(\omega,0)}(\mathcal{A}, \mathcal{B})$ is won by Player 1 iff the given ω-PCP(REG) instance has a solution. Let (L, C) be such an instance with $L = \Big((u_1, v_1), \ldots, (u_m, v_m)\Big)$. W.l.o.g. we can assume the PCP-alphabet to be binary, say, $\{a, b\}$. Moreover, as used in the proof of Thm. 42 already, we can restrict our attention to solutions that are of the form $w = i_0 i_1 i_2 \ldots$ such that

$w \in L(C)$, $u_{i_0}u_{i_1}\ldots = v_{i_0}v_{i_1}\ldots$ and, additionally, for every $n \in \mathbb{N}$ we have $|u_{i_0}\ldots u_{i_n}| \geq |v_{i_0}\ldots v_{i_n}|$.

The two NBA are defined over the alphabet $\Sigma := \Sigma_1 \cup \Sigma_2$ with $\Sigma_1 := \{a, b\}$ and $\Sigma_2 := \{\hat{a}, \hat{b}, c\} \cup \{c_i \mid i = 1, \ldots, m\}$. Intuitively, buffer 1 is used to hold the u-part of a potential solution produced by Player 1, the letters $\hat{a}$ and $\hat{b}$ are used to encode the v-part of the proposed solution, and $c, c_1, \ldots, c_m$ are used to inform Player 0 about Player 1's choices of the next index in such a potential solution.

The NBA $\mathcal{A}$ for Player 0 is shown in Fig. 2. Intuitively, it lets Player 0 ask Player 1 for a choice of an $i \in [m]$ by playing c. Since this letter goes into the capacity-0-buffer, Player 1 has to respond to it immediately. Suppose it takes her to some state which has outgoing transitions labeled with c_j for all but one $j \in [m]$ to a state that can accept anything. Then this implicitly models a choice of hers of this $j \in [m]$ which is communicated to Player 0 who, after playing c_j next, does not lose immediately. After doing so, $\mathcal{A}$ requires Player 0 to put u_j into the first buffer and v_j into the second. We use the shorthand notation $\widehat{v}$ for the word that results from v by applying the function $\hat{\cdot}$ to every letter.

Again, it is easier to present Player 1's NBA as a product of three automata $\mathcal{B}_{\mathsf{ch}}, \mathcal{B}_2$ and $\mathcal{B}_C$. The first two components are shown in Fig. 2 as well. The automaton $\mathcal{B}_{\mathsf{ch}}$ is used to communicate Player 1's choices of the next index i_h in successively constructing a solution $i_0 i_1 \ldots$. The automaton $\mathcal{B}_2$ is used to consume letters from the ω-buffer as soon as there is a letter pending to be consumed from buffer 2, i.e. the one with capacity 0.

The third component $\mathcal{B}_C$ is simply obtained from C by relabelling every transition $q \xrightarrow{i} q'$ as $q \xrightarrow{c_i} q'$, and adding self-loops $q \xrightarrow{x} q$ for every $x \in \{a, b, \hat{a}, \hat{b}, c\}$ to any state q in it. Then the projection of any run in $\mathcal{B}_C$ onto the letters $c_1, \ldots, c_m$ is a run of C.

Player 1's NBA $\mathcal{B}$ is obtained as the direct product of $\mathcal{B}_{\mathsf{ch}}, \mathcal{B}_2$ and $\mathcal{B}_C$. Here, the acceptance condition is a conjunct, i.e. a state (q, q', q'') is accepting iff all three of them are accepting in their respective components. Clearly, $\mathcal{A}$ and $\mathcal{B}$ are constructible in polynomial time in the size of (L, C). It remains to be seen that this construction is correct.

"$\Leftarrow$" Suppose $i_0 i_1 \ldots$ is a solution to the ω-PCP(REG) instance (L, C). We describe a strategy for Player 1. Again, we divide the game into phases, each phase starting when Player 0 is in the initial state of $\mathcal{A}$, and Player 1 is in a state composed of the initial states of $\mathcal{B}_{\mathsf{ch}}$ and $\mathcal{B}_2$, together with any state from $\mathcal{B}_C$. Then Player 0 opens the j-th phase by playing c. Player 1 then reacts by moving to q_{i_j}. If Player 0's next move is not c_{i_j} then Player 1 can easily win the remaining play by moving to f. So, Player 0 is forced to play c_{i_j} next, followed by $u_{i_j}\widehat{v_{i_j}}$. The first part u_{i_j} is put into buffer 1, and the second part $\widehat{v_{i_j}}$ needs to be consumed by Player 1 immediately. By assumption, this matches the current beginning of buffer 1, so she does not get stuck in component $\mathcal{B}_2$ which forces her to remove this matching part from buffer 1. The play then continues in the $(j+1)$-th phase.

By assumption, $i_0 i_1 \ldots \in L(C)$. Hence, the unique play enforced by this strategy constructs an accepting run because it visits final states in $\mathcal{B}_C$, and

therefore in $\mathcal{B}$, infinitely often.

"$\Rightarrow$" We can define a simple universal strategy for Player 0: whenever Player 1 is in state q_i in her component $\mathcal{B}_{\mathsf{ch}}$ then play c_i next. It should be clear that he will not get stuck with this strategy, and Player 1 cannot reach state f with it. Now suppose that the underlying ω-PCP(REG) (L, C) has no solution. Then any word $i_0 i_1 \ldots$ is not accepted by C or contains a mismatch, i.e. $u_{i_0} u_{i_1} \ldots \neq v_{i_0} v_{i_1} \ldots$. Any Player 1 strategy gives rise to such a word by considering the sequence of states $\mathsf{q}_{i_0}, \mathsf{q}_{i_1}, \ldots$ visited by this strategy against Player 0's universal strategy. If $i_0 i_1 \ldots \notin L(C)$ then the run corresponding to this word in $\mathcal{B}$ will eventually not see any accepting states in $\mathcal{B}_C$ and therefore in $\mathcal{B}$ anymore. Moreover, if the sequences $u_{i_0} u_{i_1} \ldots$ and $v_{i_0} v_{i_1} \ldots$ contain a mismatch, then Player 1 will eventually get stuck in component $\mathcal{B}_2$. In both cases she is not able to construct an accepting run, thus, this universal strategy for Player 0 is winning if the ω-PCP(REG) has no solution. ∎

Putting these constructions together we obain the following.

Corollary 44 *Let κ be a vector of capacities with at least one unbounded and at least one bounded capacity. Deciding $\sqsubseteq^{\kappa}$ is $\mathbb{B}(\Sigma^1_1)$-hard.*

Proof From Thm. 42 and 43, and Cor. 41, we obtain that $\sqsubseteq^{(\omega,0)}$ is hard for $\mathbb{B}(\Sigma^1_1)$. By [16, Thm. 7], $\sqsubseteq^{(\omega,n)}$ can be reduced to $\sqsubseteq^{(\omega,0)}$ for any $n \geq 0$, hence also $\sqsubseteq^{(\omega,n)}$ is hard for $\mathbb{B}(\Sigma^1_1)$. Finally, having more buffers (that are possibly unused by the trace alphabet (Σ, σ, k)) cannot make the problem simpler. ∎

6. Conclusion and Further Work

We have studied the decidability and computational complexity of multi-buffer simulation games – a variant of fair simulation, characterised by a game between Spoiler and Duplicator in which Spoiler's moves get stored in FIFO buffers of bounded or unbounded capacities. This gives Duplicator the possibility to delay her responses and therefore better foresight in constructing a run corresponding to the one constructed by Spoiler. This better foresight leads to better approximations to language-theoretic inclusion problems which are important in the verification of reactive and concurrent systems.

Considerations on the expressive power of such games, namely how well they approximate such inclusion problems etc. have not been studied here, simply not to make this article even longer. Nevertheless, a thorough, complete and overviewing study of the expressive power of multi-buffer simulation games, including their relationship to other simulation games, is planned future work. Some results on this can be found scattered in the preliminary work that the decidability-and-complexity study in this article is also based upon [15, 16, 18].

It is worth mentioning that the range of complexities of solving multi-buffer simulation games – including levels of undecidability, and when parametrised

capacities \ game type	multi-buffer simulation	multi-buffer flushing
$(\omega, \ldots, \omega, n, \ldots, n)$	$\in \Delta_2^1$, $\mathbb{B}(\Sigma_1^1)$-hard [Cor. 13, Cor. 44]	?
$(\omega, \ldots, \omega)$		Π_1^1-complete [Thm. 26,Thm. 39]
$(\omega, n, \ldots, n)$		$\in$ 2EXPTIME [Thm. 31]
(ω)	EXPTIME-complete [Thm. 19, Thm. 38]	PSPACE-complete [Thm. 28, Thm. 32]
$(n, \ldots, n)$	P-complete	

Figure 3: Summary of the decidability and complexity results on multi-buffer games. Here, n stands for an arbitrary finite capacity; when two references are given then the first one contains the upper, the second the lower bound. Lower bounds hold when $(n, \ldots, n) = (0)$ already.

by buffer capacities – spans a wide range from P (when all buffers are bounded) to somewhere between $\mathbb{B}(\Sigma_1^1)$ and Δ_2^1 in the analytic hierarchy (when at least one unbounded and one more buffer are available), and capturing PSPACE and EXPTIME in special cases. This can be seen as an indication of the richness of this framework.

Our results show in particular that the flushing variant is a genuinely simpler subcase of multi-buffer simulations. This becomes particularly clear in the case of a single unbounded and several bounded buffers (where multi-buffer simulation is highly undecidable of maximal complexity and the flushing variant is solvable in doubly exponential time).

Besides the aforementioned conclusive study on expressiveness that remains to be done in the future, there is a little bit of work left to be done on complexity issues, which we could not answer to full satisfaction here. These concern the exact complexity of the flushing variant in the case when some buffers are bounded and some are unbounded, see Fig. 3.

References

[1] P. A. Abdulla, A. Bouajjani, L. Holík, L. Kaati, and T. Vojnar, *Computing simulations over tree automata*, TACAS'08, 2008, pp. 93–108.

[2] J. R. Büchi, *On a decision method in restricted second order arithmetic*, Proc. Congress on Logic, Method., and Philosophy of Science, 1962, pp. 1–12.

[3] B. S. Chlebus, *Domino-tiling games*, Journal of Computer and System Sciences **32** (1986), 374–392.

[4] L. Clemente and R. Mayr, *Advanced automata minimization*, POPL'13, 2013, pp. 63–74.

[5] V. Diekert and G. Rozenberg, *The book of traces*, World Scientific Publ. Co., 1995.

[6] D. L. Dill, A. J. Hu, and H. Wong-Toi, *Checking for language inclusion using simulation relations*, CAV'91, 1992, pp. 255–265.

[7] E. A. Emerson and C. S. Jutla, *Tree automata, μ-calculus and determinacy*, FOCS'91, 1991, pp. 368–377.

[8] K. Etessami, *A hierarchy of polynomial-time computable simulations for automata*, CONCUR'02, 2002, pp. 131–144.

[9] K. Etessami, T. Wilke, and R. A. Schuller, *Fair simulation relations, parity games, and state space reduction for Büchi automata*, ICALP'01, 2001, pp. 694–707.

[10] O. Finkel, *Three applications to rational relations of the high undecidability of the infinite Post Correspondence Problem in a regular ω-language*, Int. J. Found. Comput. Sci. **23** (2012), no. 7, 1481–1498.

[11] C. Fritz and T. Wilke, *Simulation relations for alternating Büchi automata*, Theor. Comput. Sci. **338** (2005), no. 1-3, 275–314.

[12] D. Harel, *Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness*, J. ACM **33** (1986), no. 1, 224–248.

[13] T. A. Henzinger, O. Kupferman, and S. K. Rajamani, *Fair simulation*, Inf. Comput. **173** (2002), no. 1, 64–81.

[14] M. Holtmann, L. Kaiser, and W. Thomas, *Degrees of lookahead in regular infinite games*, Log. Meth. in Comp. Sci. **8** (2012), no. 3.

[15] M. Hutagalung, N. Hundeshagen, D. Kuske, M. Lange, and É. Lozes, *Multi-buffer simulations for trace language inclusion*, GandALF'16, 2016, pp. 213–227.

[16] ———, *Two-buffer simulation games*, CASSTING'16, 2016, pp. 27–38.

[17] M. Hutagalung, M. Lange, and É. Lozes, *Revealing vs. concealing: More simulation games for Büchi inclusion*, LATA'13, 2013, pp. 347–358.

[18] M. Hutagalung, M. Lange, and E. Lozes, *Buffered simulation games for Büchi automata*, AFL'14, 2014, pp. 286–300.

[19] M. Jurdziński, *Small progress measures for solving parity games*, STACS'00, 2000, pp. 290–301.

[20] R. Milner, *An algebraic definition of simulation between programs*, IJCAI'71, 1971, pp. 481–489.

[21] H. Rogers, *Theory of recursive functions and effective computability (reprint from 1967)*, MIT Press, 1987.

[22] J. Sakarovitch, *The "last" decision problem for rational trace languages*, LATIN'92, 1992, pp. 460–473.

[23] P. van Emde Boas, *The convenience of tilings*, Complexity, Logic, and Recursion Theory, 1997, pp. 331–363.