On the Complexity of Multi-Pushdown Games

Roland Meyer
TU Braunschweig, Germany
roland.meyer@tu-bs.de

Sören van der Wall
TU Braunschweig, Germany
s.van-der-wall@tu-bs.de

Abstract

We study the influence of parameters like the number of contexts, phases, and stacks on the complexity of solving parity games over concurrent recursive programs. Our first result shows that k-context games are b-EXPTIME-complete, where $b = \max\{k-2, 1\}$. This means up to three contexts do not increase the complexity over an analysis for the sequential case. Our second result shows that for ordered k-stack as well as k-phase games the complexity jumps to k-EXPTIME-complete.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases concurrency, complexity, games, infinite state, multi-pushdown

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.52

Funding This work was partially supported by DFG grant 417532197 *Effective Denotational Semantics for Synthesis* (*EDS@SYN*).

1 Introduction

Software verification and synthesis are difficult, even more so when concurrency comes into play. Algorithmically, both tasks often amount to solving games [33] over an operational model that captures implementation and specification details [62, 40]. What makes these games hard to solve is the size of the underlying graph, which easily ends up having an infinite set of positions. One reason is that software often computes over infinite data domains. Another reason is that the control flow tends to be structured into recursive procedures or even functional code. Despite this difficulty, efficent algorithms and tools for solving games over infinite graphs have been proposed. Data aspects are discharged to logical reasoning engines [9, 21]. Recursive functions are summarized to their call-return relationship [58, 54, 61, 10], an idea that generalizes to functional programs [3, 46, 45, 55, 34, 41, 36]. Alternatively, the set of reachable call stacks is tracked symbolically and saturated until a fixed point is reached [20, 14, 32, 23], which is again applicable to functional programs [15, 37, 26, 17, 25]. There are efficient implementations of saturation [18, 19]. Yet, tools that participate in the Software Verification Competition [11], like CPAchecker [1, 12, 13] and the ULTIMATE framework [2, 38, 39], favor summarization.

What remains a challenge, not only for game solvers but already for verification engines, is concurrency. When combined with recursion, even the simplest analysis problems become undecidable [31, 53]. One way out is under-approximation, analyzing only a (critical) subset of the semantics. In *context-bounded* computations [52] the thread holding the processor (the context) may switch only a bounded number of times. *Phase-bounded* computations [47] generalize the idea. During a phase all threads may push their stack but only one thread can pop. In *ordered* computations [16], the threads are ordered and a pop transition may only be performed by the smallest thread whose stack is non-empty. The complexity is similar to the phase-bounded case [6, 5]. Technically, the above results are obtained for multi-pushdown systems [16], a programming model with multiple stacks accessed by a sequential control flow representing the interleaving of the threads.

Table 1 Overview of the state-of-the-art and new results with technical highlights marked.

	Previous results		New results	
Upper bounds	k fixed	k input	k fixed	k input
k-stack ordered	—	—	k-EXP	non-elem.
k-context	k-EXP [57]	non-elem. [57]	$\max\{k-2,1\}$-EXP	non-elem.
k-phase	k-EXP [57]	non-elem. [57]	k-EXP	non-elem.
Lower bounds				
k-stack ordered	—	—	k-EXP	non-elem.
k-context	—	—	$\max\{k-2,1\}$-EXP	non-elem.
k-phase	—	non-elem. [7, 8]	k-EXP	non-elem.

The aforementioned works are limited to (linear-time) verification. There are considerably less results towards under-approximate synthesis (and branching-time model checking). Seth was the first to study parity games over multi-pushdown systems, multi-pushdown games (MPDG) for short [57]. He considered phase boundedness and gave a summarization-based decision procedure. It can be lifted to a subclass of concurrent higher-order programs [56]. Using saturation, Hague [35] was able to capture the full class of concurrent higher-order programs. The algorithm works for orderedness, bounded phases, and the bounded scopes explained below. The winning condition is reachability. Also using saturation, Atig et al. [8] showed how to reduce the number of phases in an MPDG, leading to a recursive decision procedure. It yields a k-EXPTIME upper bound for k phases. If the phases are part of the input, the upper bound is non-elementary and the authors present a matching lower bound. An also non-elementary lower bound was shown for the related problem of branching-time model checking under a given context bound [7].

Contribution. We determine the precise influence of the number of contexts, phases, and stacks on the complexity of solving parity MPDG (Table 1). The practically most relevant and at the same time technically most interesting case is k-context parity MPDG for which we show $\max\{k-2,1\}$-EXPTIME-completeness.[1] The upper bound reflects the fact that three contexts can be translated into a single stack pushdown. Interestingly, each further context increases the complexity by one exponent, as in the case of the seemingly more expressive phase-bounded MPDG. There, the complexity settles at k-EXPTIME-complete for k phases. The same complexity holds for ordered k-stack parity MPDG.

Motivated by the success of summarization algorithms [1, 13, 2, 38], we decided to derive our upper bounds by summarization. The algorithms reduce the given MPDG to a finite game by abstracting plays through completed function calls (between matching pushes and pops) to their effect on the control states. This approach has been pioneered by Walukiewicz for pushdown games [61] and generalized to phase-bounded MPDG by Seth [57]. Our algorithms are generalizations and optimizations of Seth's work. Unlike [57], we do not assume a constant number of stacks under context bounds.

Unfortunately, we also discovered a flaw in [57] that makes the existing finite game construction unsound. We explain and fix the problem, and thus obtain the first (correct) summarization algorithm for MPDG.

We complement the findings by matching lower bounds. They work by reductions from space-bounded alternating Turing machines. To demonstrate the expressiveness of MPDG without getting lost in the case distinctions often involved with Turing machine reductions,

[1] Class k-EXPTIME is the union of all $\mathsf{DTIME}(\exp_k(\mathrm{poly}(n)))$ with $\exp_0(n) = n$, $\exp_{k+1}(n) = 2^{\exp_k(n)}$.

we propose the formalism of first-order relations. These are relations among words formulated in a fragment of first-order logic. Our main result shows that k-phase, $(k+2)$-context, and k-stack ordered MPDG can decide first-order relations over words of length $(k-1)$-fold exponential. Reachability is sufficient as the winning condition. The reductions are a considerable step beyond the non-elementary and parameterwise rough lower bounds in [7, 8]. We build on ideas in [7], the result for phase-bounded MPDG in [8] cannot be adapted to context boundedness.

Related Work. There are further restrictions on concurrent recursive programs. Round-bounded computations [48] schedule the threads in round-robin fashion for a given number of rounds. Scope-bounded computations [49] require a matching pop to occur within a bounded number of contexts from the corresponding push. Very recently, hole boundedness [4] has been proposed as a generalization of bounded scope. Common to these notions is that they limit the ability of the scheduler in contrast to the studied restrictions.

A framework that has led to algorithmic meta-theorems of the above form is bounded tree-width [28, 50] and its developments like split-width [29, 22]. The idea is to capture a programming model that acts on infinite storage by a finite-state device operating over enriched computations. So far, this approach has not been lifted to games.

Remotely related is higher-order model checking [51]. The complexity is similar, namely k-EXPTIME for schemes of order k. Moreover, besides Ong's game semantics approach [51], there are saturation [17] and summarization [46] algorithms. The technical challenges, however, are different. In HOMC, the task is to represent and manipulate recursively defined functions of higher order. In MPDG, the task is to capture the interferences among threads.

2 Multi-Pushdown Games

Multi-Pushdown Systems. A *multi-pushdown system (MPDS)* is a finite-control program that operates on finitely many stacks of unbounded height [16]. Formally, it is a tuple $P = (Q, \Gamma, \delta, n)$, where Q is a finite set of control states, n is the number of stacks, Γ is a finite stack alphabet, and $\delta = \delta_{int} \cup \delta_{push} \cup \delta_{pop}$ is a set of internal, push, and pop transitions with

$$\delta_{int} \subseteq Q \times [1..n] \times Q \qquad \delta_{push} \subseteq Q \times [1..n] \times \Gamma \times Q \qquad \delta_{pop} \subseteq Q \times \Gamma \times [1..n] \times Q\,.$$

Each transition acts on a stack $r \in [1..n]$. We refer to all internal transitions for stack r with $\delta_{int,r}$, and similarly for $\delta_{push,r}$ and $\delta_{pop,r}$. Let $\delta_r = \delta_{int,r} \cup \delta_{push,r} \cup \delta_{pop,r}$. The size of P is given by $|P| = |Q| + |\Gamma| + |\delta|$.

The behavior of MPDS is defined in terms of configurations and labeled transitions between them. A configuration of P is a pair $(q, \mathcal{P})$ consisting of a control state $q \in Q$ and a vector of stack contents $\mathcal{P} \in (\Gamma^*)^n$. We use $C = Q \times (\Gamma^*)^n$ for the set of all configurations. The labeled transition relation $\to \subseteq C \times \delta \times C$ implements the transitions given by P on its configurations. We have $(q, \mathcal{P}) \xrightarrow{\tau} (q', \mathcal{P}')$ if one of the following holds:

$$\begin{aligned}
\tau = (q, r, q') \in \delta_{int} \quad &\text{and} \quad \mathcal{P}' = \mathcal{P} \\
\tau = (q, r, s, q') \in \delta_{push} \quad &\text{and} \quad \mathcal{P}' = \mathcal{P}_{[r \mapsto s.\mathcal{P}[r]]} \\
\tau = (q, s, r, q') \in \delta_{pop} \quad &\text{and} \quad \mathcal{P}'_{[r \mapsto s.\mathcal{P}'[r]]} = \mathcal{P}.
\end{aligned}$$

If transition label τ is not important, we may omit it. Vector $\mathcal{P}_{[j \mapsto y]}$ is defined to coincide with $\mathcal{P}$ except for the content of stack j which is replaced by y, $\mathcal{P}_{[j \mapsto y]}[j] = y$ and $\mathcal{P}_{[j \mapsto y]}[z] = \mathcal{P}[z]$ for all $z \neq j$. We use $\mathcal{P}[1..r] = y$ to indicate that stacks 1 to r hold content y.

Ordered Computations, Contexts, and Phases. A computation of P is a finite or infinite sequence of configurations $(q_0, \mathcal{P}_0) \xrightarrow{\tau_0} (q_1, \mathcal{P}_1) \xrightarrow{\tau_1} \dots$ that respects the transition relation. It is *ordered* [16] if for every pop transition from stack r the stacks $1, \dots, r-1$ are empty, $\mathcal{P}_p[1..r-1] = \varepsilon$ for all $\tau_p \in \delta_{pop,r}$. The computation is a *context on stack* r if all transitions act on that stack, for all τ_p we have $\tau_p \in \delta_r$. It is a *phase on stack* r if every pop transition acts on that stack, for all $\tau_p \in \delta_{pop}$ we have $\tau_p \in \delta_{pop,r}$. A computation is said to have k *contexts* [52] if it decomposes into k contexts but does not decompose into $k-1$ contexts, and similar for k *phases* [47].

Graph Games. A *graph game* is a two-player zero-sum game played by moving a pebble along the edges of a potentially infinite graph [33]. Formally, it is a tuple (V, E, own, win), where (V, E) is a directed graph, $own : V \to \{\text{Eve}, \text{Ana}\}$ is an ownership function, and $win : V^\omega \to \{\text{Eve}, \text{Ana}\}$ is a winning condition. We call V the positions and E the moves of the game. We call a graph game finite, if the set of positions is finite.

A *play* is a maximal path π in the graph (V, E) underlying the game. *Eve wins the play* if either the play is infinite and $win(\pi) = \text{Eve}$, or the play is finite and ends in a position from V_{Ana} (with no move left for Ana). Otherwise, *Ana wins the play.* Whenever a play reaches a position, the owner of the position has to decide about the next move. A *strategy for Eve* is a function $\sigma : V^* V_{\text{Eve}} \to V$ such that $v\, E\, \sigma(\pi v)$ holds for all $\pi v \in V^* V_{\text{Eve}}$. The strategy is *positional* if it only depends on the current position. In this case, the strategy can be given as $\sigma : V_{\text{Eve}} \to V$. A play $\pi = \pi_0 \pi_1 \dots$ is *compliant with* strategy σ for Eve if for all $\pi_p \in V_{\text{Eve}}$ we have $\pi_{p+1} = \sigma(\pi_0 \dots \pi_p)$. A strategy for Eve is *winning from position* v if Eve wins all compliant plays that start in v. If there is a strategy that is winning from v, we call v a *winning position.* The definitions for Ana are similar.

A *reachability winning condition* win_W is defined by a set $W \subseteq V$ of so-called winning positions. We define $win_W(\pi) = \text{Eve}$ if a winning position $\pi_i \in W$ is visited, and $win_W(\pi) = \text{Ana}$ otherwise. A *reachability game* is a tuple (V, E, own, W). A *parity winning condition* win_Ω is defined by a mapping from positions to *priorities*, $\Omega : V \to [0..max]$. The winner of a play is determined by the highest priority that occurs infinitely often during the play, i.e. $win_\Omega(\pi) = \text{Eve}$ if the highest infinitely often occurring priority is even, and $win_\Omega(\pi) = \text{Ana}$ if it is odd. A *parity game* is a tuple (V, E, own, Ω).

Multi-Pushdown Games. An *n-stack multi-pushdown game (MPDG)* [57] is a triple of the form $G = (P, own, win)$ consisting of a multi-pushdown system $P = (Q, \Gamma, \delta, n)$, an ownership function $own : Q \to \{\text{Eve}, \text{Ana}\}$, and a winning condition win. The MPDG induces the graph game $(C, \to, own, win)$, and we say that Eve wins the MPDG if she wins the induced graph game. The set of positions C and the set of moves $\to$ are the configurations and transition relation of P as defined above. The ownership function carries over from control states to configurations, $own(q, \mathcal{P}) = own(q)$ for all $(q, \mathcal{P}) \in C$. To be precise, in a *reachability* MPDG we are given a set of control states Q_{reach} to represent the winning set $C_{reach} = Q_{reach} \times (\Gamma^*)^n$. In a *parity* MPDG, the winning condition is given by a priority assignment $\Omega : Q \to [0..max]$ to the states. Again, we lift it to configurations by $\Omega(q, \mathcal{P}) = \Omega(q)$. The size of a reachability MPDG G is $|G| = |P|$, the size of a parity MPDG is $|G| = |P| + max$.

A *k-context multi-pushdown game* (G, k) is a restriction of the MPDG G so that plays have at most k contexts [7]. The formal definition tracks the number of contexts within the positions. Moves that would introduce context $k+1$ do not exist. The definition of *k-phase multi-pushdown games* is similar [57, 8]. An *ordered n-stack multi-pushdown game* only admits moves that lead to an ordered play: a pop transition on stack r exists only in positions where stacks 1 to $r-1$ are empty.

In our development, it will be convenient to assume that the MPDG of interest does not deadlock. Every MPDG G can be turned into a deadlock-free MPDG G' that has the same winner, the same highest priority, and is larger by only a linear factor. This continues to hold with a polynomial factor under the aforementioned restrictions.

3 Upper Bound for Ordered MPDG

We give an algorithm for computing the winning positions in an ordered n-stack MPDG that works in n-EXPTIME. The algorithm is a slight optimization of Seth's summarization construction for phase-bounded MPDG [57]. We discovered a bug in this construction that we explain and show how to correct. Details can be found in Appendix B.

▶ **Theorem 1.** *Given an ordered n-stack* MPDG G *with parity winning condition, we can compute Eve's winning positions of the form (q, ε^n) in time $exp_n(\text{poly}(|G|^2))$.*

The algorithm constructs from the given MPDG G a finite parity game F and solves the latter. The finite game preserves the winner for the positions of interest, the set of priorities, and is not too costly to compute. For the complexity, note that n is not part of the input but fixed. Further, parity games are solved in time exponential only in the number of priorities [42, 43].

▶ **Lemma 2.** *Let G be an ordered n-stack parity MPDG. In time $exp_n(\text{poly}(|G|^2))$ we can compute a finite parity game F so that Eve wins G from position (q, ε^n) if and only if she wins F from a corresponding position. The priorities in G and F coincide.*

3.1 Summarization for Ordered MPDG

We explain the summarization construction from [57], highlight our optimization for ordered MPDG, and finally make the construction formal. The game G has infinitely many position due to arbitrarily growing stacks. The finite parity game F removes the stacks and instead tracks the current top of stack symbol for each stack. This forbids pop transitions in F. When one player decides to make a pop transition F ends and determines a winner.

In order to model G's behavior after a pop, F implements a summarization mechanism. When a symbol s is pushed onto a stack r, Eve proposes a set of summaries. This set can be understood as fixing a strategy for Eve in G that she will follow for as long as s remains on stack r. Fixing a strategy results in the set of all plays that are compliant to it and lead from the push of s to a situation where s is popped again. Each summary captures such a situation and thereby abstracts a play from this set. The finite game F can thus skip any of the abstracted plays up to the captured situation after the pop of s.

There is no guarantee that Eve will be honest in the sense that the proposed set of summaries indeed abstracts all plays that pop s and are compliant to some fixed strategy. To account for this, Ana is allowed to react to the proposal. First, she may trust Eve by choosing a summary from the proposed set. In this case, F executes the skip of the abstracted play and replaces (parts of) the current position with the position after the pop as captured by the summary. This can be understood as also fixing a strategy for Ana in G that, together with Eve's strategy, leads to the abstracted play.

Second, she may doubt Eve's proposal by executing the push transition instead of skipping. This replaces the top of stack symbol for stack r. Executing the push also stores the set of summaries proposed by Eve in the position of F. It is remembered for as long as s remains the topmost symbol of stack r. To be precise, the position will hold a separate set of

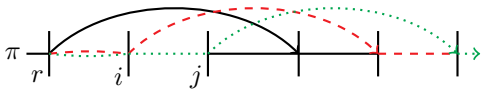

Figure 1 A fixed play π in G. Each arc matches a push to its pop. The highlighted paths show different plays in F.

summaries for the topmost symbol of each stack. When a stack is popped, the remembered set of summaries for that stack is checked for containing a summary that captures the current situation of the play. Eve wins if and only if some summary in the set applies.

Finally, there may be another reason for Ana to execute the push transition. In this case, she trusts Eve's proposed set, but her own strategy will make sure that s is never removed from the stack.

Ordered Summaries. To abstract a play π in G from a push to a matching pop on a stack r, a summary for an ordered MPDG, or ordered summary for short, takes the shape

$$(q, m, \mathcal{T}, \mathcal{M}, \mathcal{S})\,.$$

The entries $q \in Q$ and $\mathcal{T} \in \Gamma^{n-r}$ describe the configuration resulting from the final pop transition, with q the control state and $\mathcal{T}$ the topmost symbol for each stack. The orderedness restriction forces the stacks $1, \dots, r-1$ to be empty. The entry $m \in [0..max]$ is the maximal priority encountered during the abstracted play, from after the push up to before the pop. Each entry $\mathcal{M}[j]$ of $\mathcal{M} \in [0..max]^{n-r}$ is the maximal priority encountered since after the push of the topmost symbol $\mathcal{T}[j]$ of stack j.

The summary recursively holds a vector $\mathcal{S} \in (2^{\mathbb{OS}})^{n-r}$ of sets of summaries for the other non-empty stacks. Here, $\mathbb{OS}$ is the set of all summaries as defined below. Assume the abstracted play pushes the symbol $\mathcal{T}[j]$ onto a stack $j \neq r$ and does not contain a matching pop. The set of summaries $\mathcal{S}[j]$ is Eve's proposed set for how the top of stack symbol $\mathcal{T}[j]$ can be popped after the abstracted play. When the play skips to the position captured by this summary, $\mathcal{S}[j]$ becomes the set of summaries stored for $\mathcal{T}[j]$.

Pathing. When Ana doubts a set of summaries for the push on stack r she might have a strategy that pops stack r with a combination of control state, highest priority and top of stack symbols, that are not present in the set. Alternatively, they coincide but after the pop on stack r, her strategy pops stack j in a situation not captured by $\mathcal{S}[j]$. Observe that if Ana wants to doubt a set $\mathcal{S}[j]$, she needs to find a play in F, which runs into a pop on stack j without running into a pop on another stack. Alternatively, if she wants to skip to a situation captured by a summary within $\mathcal{S}[j]$, she needs to steer the play to run into the push of $\mathcal{T}[j]$, so she gets the option of skipping.

To understand this, assume some summary abstracts a play π in G from the push to a pop on stack r, which first contains a push on stack i and then a push on stack j (Figure 1). If the set of summaries $\mathcal{S}[j]$ does not capture the situation how π pops $\mathcal{T}[j]$, Ana can run into its pop as illustrated by the dashed path in the figure, i.e. Ana executes the push on stack r and then skips upon the push on stack i. The play reaches the pop on stack j and is checked against $\mathcal{S}[j]$. If it is captured, Ana wants to continuetheplaybeyondthepopofstack$j$ byskipping to a summary in $\mathcal{S}[j]$. To achieve this, she executes both, the push on stack r and i. Then she can skip upon reaching the push on stack j as illustrated by the dotted path.

▶ **Definition 3.** *We proceed by induction on the stack from n down to one. The set of* ordered summaries for stack r *is*

$$\mathbb{OS}_r \;=\; Q \times [0..max] \times \Gamma^{n-r} \times [0..max]^{n-r} \times \prod_{r<j\leq n} 2^{\mathbb{OS}_j}.$$

In the base case $r = n$, the last three components are defined to be absent. The set of all ordered summaries is $\mathbb{OS} = \bigcup_{1\leq r\leq n} \mathbb{OS}_r$.

Our optimization targets the orderedness restriction. During an abstracted play between a push and a pop on stack r, if we find an unmatched push on another stack j, then we can conclude that $j > r$. This means a summary for stack r only needs to contain summaries for stacks of larger order. The largest set is $\mathbb{OS}_1$ which has size $exp_{n-1}(\mathcal{O}(|G|^2))$, Appendix A.1. When we construct the game F, it will be convenient to assume that all vectors have length n. We fill the missing entries for stacks one to r with ε for $\mathcal{T}$, 0 for $\mathcal{M}$, and $\emptyset$ for $\mathcal{S}$.

3.2 The Finite Parity Game

Consider the ordered n-stack MPDG with parity winning condition $G = (P, own, \Omega)$, where $P = (Q, \Gamma, \delta, n)$, $own : Q \to \{\text{Eve}, \text{Ana}\}$, and $\Omega : Q \to [0..max]$. We define the finite parity game F explained above, following Seth [57] but correcting a mistake. Rather than giving the positions of F right away, we explain the behavior of the game and introduce them together with their moves. Game F regularly visits check positions $(\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})$ with $q \in Q$, $\mathcal{T} \in \Gamma^n$, $\mathcal{M} \in [0..max]^n$, and $\mathcal{S} \in (2^{\mathbb{OS}})^n$. Note that there is a set of summaries for each stack. The owner and the priority are the ones for q.

Internal Transitions. Internal transitions $(q, r, p) \in \delta_{int}$ of game G are mirrored in F. They only update the priorities:

$$(\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S}) \;\to\; (\text{Check}, p, \mathcal{T}, upd(\mathcal{M}, p), \mathcal{S})\,. \tag{1}$$

The new priority vector is defined by $upd(\mathcal{M}, q)[j] = \max\{\mathcal{M}[j], \Omega(q)\}$, for each stack j. Note that we use an implicit universal quantification over the parameters that are not specified further, meaning the transition exists for all $\mathcal{T}$, $\mathcal{M}$, and $\mathcal{S}$.

Push Transitions. Push transitions $(q, r, s, p) \in \delta_{push}$ in G lead to a series of transitions in F originating from $(\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})$:

$$(\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S}) \;\to\; (\text{Push}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, p, s) \tag{2}$$
$$(\text{Push}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, p, s) \;\to\; (\text{Claim}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, p, s, S) \tag{3}$$
$$(\text{Claim}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, p, s, S) \;\to\; (\text{Check}, p, \mathcal{T}_{[r\mapsto s]}, upd(\mathcal{M}, p)_{[r\mapsto\Omega(p)]}, \mathcal{S}_{[r\mapsto S]}) \tag{4}$$
$$(\text{Claim}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, p, s, S) \;\to\; (\text{Jump}_r, q', m', \mathcal{T}'', \mathcal{M}', \mathcal{S}'', \mathcal{M}[r]) \tag{5}$$
$$(\text{Jump}_r, q', m', \mathcal{T}'', \mathcal{M}', \mathcal{S}'', \mathcal{M}[r]) \;\to\; (\text{Check}, q', \mathcal{T}'', \mathcal{M}'', \mathcal{S}'')\,. \tag{6}$$

The transitions introduce intermediary push, claim, and jump positions with the following ownership and priority assignments:

$$own(\text{Push}_r, -) = \text{Eve} \qquad \Omega(\text{Push}_r, -) = 0$$
$$own(\text{Claim}_r, -) = \text{Ana} \qquad \Omega(\text{Claim}_r, -) = 0$$
$$own(\text{Jump}_r, -) = \text{Eve} \qquad \Omega(\text{Jump}_r, q', m', \mathcal{T}'', \mathcal{M}', \mathcal{S}'', \mathcal{M}[r]) = m'.$$

Move (2) remembers control state p and symbol s and gives Eve the next move. With Move (3), Eve proposes a set of summaries $S \subseteq \mathbb{OS}_r$ for the symbol to be pushed. By implicit universal quantification, there is a transition for each such set. Move (4) performs the push: the priority vector takes into account the priority of p for all stacks. For stack r, $\Omega(p)$ is the highest (and only) priority seen since the push of s. Move (5) corresponds to a skip and exists for every summary $(q', m', \mathcal{T}', \mathcal{M}', \mathcal{S}') \in S$. For stack r, we preserve the top of stack symbol $\mathcal{T}[r]$ and the set of summaries $\mathcal{S}[r]$. For the other stacks, we use the information given by the summary. Thus, $\mathcal{T}'' = \mathcal{T}'_{[r \mapsto \mathcal{T}[r]]}$ and $\mathcal{S}'' = \mathcal{S}'_{[r \mapsto \mathcal{S}[r]]}$. The role of the jump position is to make visible the priority m' of the summary. In Move (6), we update the priority vector to $\mathcal{M}''$. For stack r, note that $\mathcal{T}[r]$ remains the topmost symbol after the skip. Hence, the priority assignment has to take into account $\mathcal{M}[r]$, the highest priority seen before the skip. Thus, $\mathcal{M}'' = upd(\mathcal{M}'_{[r \mapsto \max\{\mathcal{M}[r], m'\}]}, q')$.

3.3 Pop Transitions and a Correction to a Mistake

As defined in [57] Ana may win F in cases where she does not win G. We correct the definition and explain the difference to the original formulation. The ordered MPDG G can only perform a pop $(q, s, r, p) \in \delta_{pop}$ of symbol s from stack r if the stacks 1 to $r-1$ are empty. Given the side condition, the finite game F has simulating moves only in positions $(\text{Check}, q, \mathcal{T}_{[r \mapsto s]}, \mathcal{M}, \mathcal{S})$ where $\mathcal{T}[1..r-1] = \varepsilon$, $\mathcal{M}[1..r-1] = 0$, and $\mathcal{S}[1..r-1] = \emptyset$. The simulating moves immediately decide about the winner of the game and take the following shape. The positions EveWin and AnaWin are winning for Eve resp. Ana. Both are owned by Eve, have self-loops, and EveWin has priority 0 while AnaWin has priority 1.

$$(\text{Check}, q, \mathcal{T}_{[r \mapsto s]}, \mathcal{M}, \mathcal{S}) \;\to\; \text{EveWin} \tag{7}$$

$$(\text{Check}, q, \mathcal{T}_{[r \mapsto s]}, \mathcal{M}, \mathcal{S}) \;\to\; \text{AnaWin}\,. \tag{8}$$

Recall that the goal of a pop transition is to check whether Ana caught Eve lying on the proposal of summaries for the popped symbol. If the current position is captured by a summary, Eve was honest and Ana could have found a path to skip after the current pop. Otherwise, Eve was lying, Ana was right in questioning the proposal and wins.

To check whether position $(\text{Check}, q, \mathcal{T}_{[r \mapsto s]}, \mathcal{M}, \mathcal{S})$ is captured, we compare it to each summary stored for stack r. The finite game has Move (7) if and only if there is a summary $x = (p, m, \mathcal{T}', \mathcal{M}', \mathcal{S}') \in \mathcal{S}[r]$ with $m = \mathcal{M}[r]$, $\mathcal{T}' = \mathcal{T}_{[r \mapsto \varepsilon]}$, $\mathcal{M}' = \mathcal{M}_{[r \mapsto 0]}$, and

$$\mathcal{S}'[j] \;\subseteq\; \mathcal{S}[j] \text{ for all } j \neq r\,. \tag{9}$$

If summary x exists, it indeed captures the current position in the game: The state after the pop transition is p, the priority m is equal the maximal priority seen during the play with s on stack r, i.e. $m = \mathcal{M}[r]$. The top of stack symbols $\mathcal{T}'$ coincide with the current ones $\mathcal{T}$, also the maximal priorities encountered since the moment these symbols have been pushed coincide, $\mathcal{M}[j] = \mathcal{M}'[j]$.

The problem in [57] refers to the relationship between $\mathcal{S}'$ and $\mathcal{S}$. The incorrect definition required an equality and therefore missed Moves (7) winning for Eve. The correction is to require Inclusion (9). To understand the problem with equality, consider the play π in an ordered MPDG G depicted in Figure 2. The play has two pushes with corresponding pops, one on stack r and drawn above the play, the other on stack j and drawn below the play. The push on stack j is simulated in the finite game F in two different ways.

Upon the push of stack r, Eve chooses a strategy up until the pop of stack r, enumerates all compliant plays (up to the pop), and summarizes them in the proposed set S_1. The play π is among the compliant plays and yields summary $x \in S_1$. The part of π abstracted by x

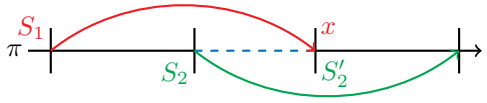

Figure 2 Matching pushes and pops, the above on stack r, the below on stack j.

contains a push on stack $j \neq r$. Eve extends her strategy and enumerates all plays from the push to the pop of stack j *that coincide with* π *on the already fixed dashed part, from the push of stack* j *to the pop of stack* r. The resulting set of summaries S_2' is contained in x.

Ana may decide against skipping and execute the push on stack r. The play may follow π and reach the push on stack j. Eve is again asked to summarize all plays up to the pop on stack j, and proposes a set S_2. Even though the proposed sets are for the same push in the same play, the result may be $S_2 \neq S_2'$. When forming S_2', the play was already fixed on the dashed part, up to the pop of stack r. When forming S_2, this does not hold. Hence, there may be plays starting with the push of stack j that pop stack r in a situation not captured by x and later pop stack j (e.g. with a different highest priority seen). However, Eve will have to at least propose a summary for each play that coincides with π on the dashed part and later pops stack j. Thus, the formed set of summaries S_2 is a superset of the set S_2'. Accordingly, if Ana also executes the push on stack j, and the play runs into the pop of stack r, checking whether summary x captures the situation at the pop of stack r requires that $S_2' \subseteq S_2$ and not $S_2' = S_2$.

4 Upper Bound for Context-Bounded MPDG

We give an algorithm to solve context-bounded MPDG that takes $\max\{1, k-2\}$-EXPTIME when considering k contexts. From a practical point of view, the interesting observation is that communication across two context switches does not increase the complexity over the problem of solving (sequential) pushdown games, which are EXPTIME-complete [61]. This compares well to the fact that a 3-context MPDS can be encoded as a single stack PDS. Interestingly, beyond the third context the complexity rises at the same pace as for phase-bounded MPDG, namely by one exponent per context/phase (Section 5).

▶ **Theorem 4.** *Given a* k*-context* MPDG G *with parity winning condition, we can compute Eve's winning positions of the form* (q, ε^n) *in time* $exp_{\max\{1,k-2\}}(\text{poly}(|G|))$.

Note that we do not assume the number of stacks to be fixed. The observation is that in a play with k contexts we can only make use of k stacks and it can be converted into an MPDG with only k stacks at only polynomial overhead (cf. Appendix C).

Our algorithm starts by reducing the number of stacks, if necessary. Afterwards, we construct a finite parity game identical to the one from the previous section except for a different set of summaries. The correctness statement is therefore a variant of Lemma 2, with a similar proof that can be found in [60].

▶ **Lemma 5.** *Let* G *be a* k*-context* k*-stack parity MPDG. In time* $exp_{\max\{1,k-2\}}(\text{poly}(|G|))$ *we can compute a finite parity game* F *so that Eve wins* G *from position* (q, ε^n) *if and only if she wins* F *from a corresponding position. The priorities in* G *and* F *coincide.*

The set of summaries we use to construct F is an optimization of Seth's summaries for phase-bounded MPDG. We repeat Seth's definition in our notation.

▶ **Definition 6** ([57]). *We fix a stack r and define the set of* phase summaries $\mathbb{PS}_{r,c}$ *by induction on the phase c from k down to 1:*

$$\mathbb{PS}_{r,c} \;=\; Q \times \{c\} \times [1..max] \times \Gamma^{n-1} \times [1..max]^{n-1} \times \prod_{j \neq r} 2^{\mathbb{PS}_{j,>c}}$$

where $\mathbb{PS}_{j,>c} = \bigcup_{i=c+1}^{k} \mathbb{PS}_{j,i}$.

A phase summary $(q, c, m, \mathcal{T}, \mathcal{M}, \mathcal{S})$ in $\mathbb{PS}_{r,c}$ still contains the information required to skip a play from a push on stack r to the matching pop. The matching pop is defined to occur in phase c. The meaning of $q, m, \mathcal{T}, \mathcal{M}$ is unchanged from Definition 3. The sets of summaries for stacks $j \neq r$ refer to phases later than c. If a symbol is popped in phase c from stack r then stack j can only be popped in a later phase.

When considering context-bounded rather than phase-bounded MPDG, the key insight is that the summaries for the first and the second context can be simplified. A summary for these contexts describes a situation where the push and the matching pop happen within the same context. As a consequence, the other stacks will not change between the push and the pop. This means a summary for context one and two does not need to contain entries for other stacks. This yields the following optimization of Definition 6.

▶ **Definition 7.** *Consider stack r. We define the set of* context summaries *by $\mathbb{CS}_{r,c} = \mathbb{PS}_{r,c}$ for c from 3 to k. For $c = 1, 2$, the stack has no influence on the definition:*

$$\mathbb{CS}_c \;=\; Q \times \{c\} \times [1..max] \;.$$

The largest set of summaries is $\mathbb{CS}_{r,3}$, which has size $exp_{k-3}(\mathcal{O}(|G|^{2k}))$ or $exp_{k-3}(\text{poly}(|G|))$ since k is fixed, Appendix A.2. Note that the optimization in Definition 7 is not sound for phase-bounded MPDG. There, symbols can be pushed on all stacks in phases one and two. The difference also manifests itself in the lower bound.

5 Lower Bounds

We show lower bounds on the complexity of solving context-bounded, phase-bounded, and ordered MPDG. They match the upper bounds established in the previous sections. The lower bounds already hold for reachability as the winning condition and thus carry over to parity. Interestingly, for phase and context-bounded MPDG we only need two stacks.

▶ **Theorem 8.** *Solving $(k+2)$-context respectively k-phase 2-stack reachability MPDG is k-EXPTIME-hard. Solving ordered n-stack reachability MPDG is n-EXPTIME-hard.*

The proofs are by reduction from the membership problem for space-bounded alternating Turing machines [27]. We want to focus on the main ideas. A detailed presentation can be found in Appendix D and [60]. Let M be an alternating Turing machine that is guaranteed to terminate (decider) and operate with space bound $exp_{k-1}(\text{poly}(|w|))$. Given an input word w, we show how to construct a 2-stack reachability MPDG G_w satisfying the following.

▶ **Lemma 9.** *Eve wins G_w if and only if $w \in L(M)$. No play in G_w exceeds $(k+2)$ contexts and k phases. The construction of G_w works in time polynomial in $|w|$.*

Note that the same MPDG G_w proves the lower bound for the context-bounded and for the phase-bounded case. Appendix D.4 explains how to adapt the construction to ordered MPDG. In that setting, we need n stacks. Together, this proves Theorem 8.

We give the construction of G_w in three steps. First, we explain the overall idea of how G_w simulates M. Next, we present the key techniques used in the construction, first-order relations defined by a fragment of first-order logic, and Stockmeyer's nested indexing [59, 24, 36]. Finally, we give details of the construction.

5.1 Reduction

We recall the semantics of alternating Turing machines. Configurations (q, c) of M on input w consist of a state q and tape content c. States are defined to be existentially or universally branching. We generalize this terminology to configurations and speak of existential and universal configurations, respectively. The tape content is a word over the tape alphabet together with a marker denoting the head of the alternating Turing machine. We do not use an additional work tape. A computation of M on w yields a tree. The nodes are configurations, the edges are transitions. Existential configurations have one successor configuration, if a transition is possible. Universal configurations have a successor for each possible transition. A configuration is final if it is universal and no transitions are possible. The tree is accepting if every branch reaches a final configuration. There may be different computation trees and M accepts w if one of them is accepting.

Configurations (q, c) of the alternating Turing machine will be modeled by positions in the game G_w. Alternation will be reflected by the ownership function: Eve will own the positions modeling existential configurations, and hence decide about the transition to take from there. Transitions between configurations will be mimicked by moves. A play of G_w reflects a branch in some computation tree of M on input w. A winning strategy for Eve will yield a computation tree of M on w that is accepting. A winning strategy for Ana will find a branch that violates acceptance in any computation tree.

When modeling configuration (q, c), state q will be the control state of G_w. To understand how tape content is stored, consider a computation branch of M that leads to (q, c). It is a sequence of configurations $(q_0, c_0) \dots (q_m, c_m)(q, c)$. The game stores the tape contents on the first stack, in the form $c\#c_m\# \dots \#c_0$. The owner of q chooses a transition δ to take from (q, c), that results in a configuration (q', c'). The game pushes the tape content c' onto the first stack and sets the new control state to q'.

The difficulty is that tape content c' is of size $exp_{k-1}(len)$ with $len = \text{poly}(|w|)$ while the size of G_w has to remain polynomial. This means the tape content cannot be pushed in a faithful way by only using the control states of the MPDG. Instead, we let Eve propose a sequence of symbols γ from an appropriate alphabet. Afterwards, we give Ana the opportunity to check the sequence for correctness. Correctness is expressed by a number of relations between γ and its predecessor c on the first stack.

For each relation, we show how to construct a verification mechanism, an MPDG that is entered when Ana chooses to check correctness. Once entered, the verification mechanism cannot be left again. It is constructed in such a way that Eve has a winning strategy from the entry point if and only if the topmost sequences γ and c on the first stack are in the required relation. The use of verification mechanisms forces a winning strategy for Eve in the overall game to push a sequence γ that satisfies all relations.

Consider the verification mechanism required to implement the relation of a Turing machine transition δ, say from configuration (q, c) to (q', c'). The verification mechanism has to check that $\gamma = c'$, the proposed word is the tape content of the successor configuration. If not, then a single position will witness the mismatch. It is either a position that changed without having the head, or the change did not respect δ. The verification mechanism thus needs to compare the same position in γ and c. Still, the number of positions is not polynomial and verification mechanisms cannot store the position in the control state.

The idea is to annotate the letters in c and in γ with their positions. This reduces counting to the problem of comparing annotations. However, even if the positions are encoded binary the annotation is still $exp_{k-2}(len)$ long. The problem can be addressed in the same way, we again annotate the letters of the encoding with their positions. This recursive process is known as *Stockmeyer's nested indexing* [59, 24, 36], written here as function *enc*. As a consequence, the first stack will actually hold the sequence $enc(c)\#enc(c_m)\#\ldots\#enc(c_0)$.

The relations between γ and $enc(c)$ that Ana will have to check by means of verification mechanisms are: is γ a correct encoding at all, $\gamma = enc(u)$ for some u, and is the encoded tape content u the successor c' of content c according to a Turing machine transition δ. We already discussed how to check the latter relation. For the former, the essence is to check the binary increment relation. We rely on further auxiliary relations.

Our key observation is that all relations required for the reduction are defined by a finite alternation of quantifiers over the set of positions. We introduce *first-order relations*, a formalism sufficiently expressive to capture each of these relations. Then we show how to construct a verification mechanism for any first-order relation. This is the main technical contribution of the section.

5.2 First-Order Relations

A first-order relation is a relation $u \sim_\varphi v$ between words u and v that is defined by a closed formula φ from a fragment of first-order logic. For the definition of the fragment, let Σ be a finite alphabet ranged over by s. Let $\mathcal{V}$ be a countable set of so-called position variables ranged over by y. A term t is either an alphabet symbol or the symbol at position y in the first or in the second word. A formula φ quantifies over positions, compares positions, and compares symbols given by terms t_1 and t_2:

$$t \;::=\; s \mid symb_1(y) \mid symb_2(y) \qquad \varphi \;::=\; y_1 \leq y_2 \mid t_1 = t_2 \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \exists y.\varphi\,.$$

The remaining logical connectives, the universal quantifier, and common predicates are defined as abbreviations. A formula is closed if every position variable is bound by a quantifier.

Formulas φ are evaluated over pairs of words $u = u_0 \ldots u_{n-1}$ and $v = v_0 \ldots v_{n-1}$ over Σ of the same length together with a valuation of the free position variables $val : \mathcal{V} \to [0..n-1]$. The semantics of terms is

$$[\![s]\!]^{val}_{u,v} \;=\; s \qquad [\![symb_1(y)]\!]^{val}_{u,v} \;=\; u_{val(y)} \qquad [\![symb_2(y)]\!]^{val}_{u,v} \;=\; v_{val(y)}\,.$$

The semantics of first-order formulas is as expected [30]. For closed formulas, it is independent of the valuation. A closed formula φ defines a so-called *first-order relation* among words that contains all models of the formula, $\sim_\varphi = \{(u,v) \mid u, v \models \varphi\}$. To give an example, the ordering $<$ on natural numbers in most-significant-bit-first encoding is defined by

$$\exists y_1.\forall y_2.\; [(y_2 < y_1) \;\rightarrow\; symb_1(y_2) = symb_2(y_2)] \;\wedge\; symb_1(y_1) = 0 \;\wedge\; symb_2(y_1) = 1\,.$$

A first-order formula is in prenex normal form if is has the shape $Q_1 y_1 \ldots Q_m y_m.\varphi$, where $Q_i \in \{\exists, \forall\}$ and φ does not contain quantifiers. Every first-order formula can be transformed into an equivalent formula in prenex normal form [30].

5.3 Stockmeyer's Nested Indexing

Our reduction relies on Stockmeyer's nested indexing [59, 24, 36]. It takes a word of length $exp_d(n)$ and appends to each letter an index. The index is the letter's position given in most-significant-bit-first encoding. Each index has length $exp_{d-1}(n)$. This encourages to index it as well, and do so on until the indices have polynomial length.

For each nesting depth $d > 0$ of indices we introduce the alphabet $\Sigma_d = \{0_d, 1_d\}$. Let function $msbf_d$ assign to a number its *most-significant-bit-first encoding* over Σ_d. We define the *d-nested indexing* $ind_d(u)$ of words $u = u_0 \dots u_m$ by induction. The base case is $ind_0(u) = u$, the word itself and the inductive case is

$$ind_{d+1}(u) = u_0 x_0 \dots u_{m-1} x_{m-1} , \quad \text{where } x_i = ind_d(msbf_{d+1}(i)) .$$

To give an example, $ind_2(\text{abra}) = \text{a}0_2 0_1 0_2 1_1 \text{b}0_2 0_1 1_2 1_1 \text{r}1_2 0_1 0_2 1_1 \text{a}1_2 0_1 1_2 1_1$.

In our reduction, all words indexed by ind_d are of length $exp_d(len)$ for some $len \in \mathbb{N}$. Then all indices in each layer d have the same length and their $msbf_d$ encoding ranges from $0^{exp_{d-1}(len)}$ to $1^{exp_{d-1}(len)}$. Since the tape contents in the reduction are of length $(k-1)$-fold exponential in the input word of the Turing machine, we define this to be our encoding, $enc(c) = ind_{k-1}(c)$. As a result, the lowest layer indices are of polynomial length.

5.4 Verification Mechanisms

We proceed by induction on the depth d of the nested indexing. We show how to construct for every closed first-order formula φ a verification mechanism, a 2-stack MPDG G^d_φ that *decides* $\sim_\varphi$ *over words of length* $exp_d(len)$ in the following sense. We have $u \sim_\varphi v$ if and only if Eve has a winning strategy from the initial position with $u, v \in \Sigma^{exp_d(len)}$ on top of the first stack. The initial position takes the shape

$$(\text{Check}^d_\varphi, ind_d(u)\gamma_1 ind_d(v)\gamma_2, \gamma_3) ,$$

where $\gamma_1, \gamma_2, \gamma_3$ are arbitrary stack contents up to some delimiting symbols. For the reduction, we want to verify the relation of a Turing machine transition δ between encoded configurations of length $(k-1)$-fold exponential in the input. We invoke the next lemma with $d = k-1$ and $len = \text{poly}(|w|)$ and w the input to M. The verification mechanism thus takes at most $k+1$ contexts and k, once entered. The first phase for pushing the configurations has no fixed stack, so it merges with the first phase of the verification mechanism for a total of $k+2$ contexts and k phases.

▶ **Lemma 10.** *Let φ be a first-order formula over Σ and $d \in \mathbb{N}$. In time $\text{poly}(d + |\Sigma| + len)$ we can construct a 2-stack reachability MPDG G^d_φ that decides φ over words of length $exp_d(len)$. Any play takes at most $d+2$ contexts and $d+1$ phases.*

We explain the main ideas behind the construction. Details can be found in Appendix D and [60]. Let $\varphi = Qy_1 \dots . Qy_m . \psi$ be a closed first-order formula in prenex normal form. The game reflects the choice of a valuation val for $y_1, \dots, y_m$, discharging the quantifier alternation to the players. For each y_j, the responsible player chooses a position $val(y_j)$ whose binary representation has length $exp_{d-1}(len)$. Then she pushes the encoding $ind_{d-1}(val(y_j))$ on the second stack (cf. Appendix D.3). When all variables have been processed, the second stack holds a sequence $y_m ind_{d-1}(msbf_d(val(y_m))) \dots y_1 ind_{d-1}(msbf_d(val(y_1)))$ representing val, where $y_1, \dots, y_m$ serve as delimiting symbols.

After val has been chosen, we are interested in the value $[\![\psi]\!]^{val}_{u,v}$. Eve wins if and only if it is true. The difficult case is to evaluate the atomic formulas in ψ. The idea is to let Eve propose auxiliary information about u, v, and $y_1, \dots, y_m$, which is sufficient to evaluate the atomic formulas. The size of this information is independent from d, len, so it can be stored in the control state. Together, this means the game does not need to access the stack during evaluation. As before, Ana may verify the proposed information.

Atomic formulas can take the shape $[\![symb_{1/2}(y)]\!]^{val}_{u,v}$. Eve proposes symbol assignments $symb_u$, $symb_v : \{y_1, \ldots, y_m\} \to \Sigma$, mapping each variable y stored on stack two to the symbol of word u, v at the position $val(y)$. If Ana chooses to verify $symb_u(y)$ (resp. $symb_v(y)$) for some y, Eve removes symbols from stack one until she claims to have found position $val(y)$. Ana then decides to either compare the symbol found on stack one to $symb_u(y)$ ($symb_v(y)$) or verify the equality of the valuation $ind_{d-1}(val(y))$ on stack two and the annotated index on stack one (Appendix D.1).

Alternatively, an atomic formula can be $[\![y \leq y']\!]^{val}_{u,v}$. For these, Eve proposes a variable order, a sequence of variables interleaved with relations of the form

$$y_{l_1}\, \theta_1\, y_{l_2}\, \theta_2\, \ldots\, \theta_{m-1}\, y_{l_m}\,, \qquad \text{with } \theta_j \in \{=, <\}\,.$$

Verifying an entry $y\,\theta\,y'$ amounts to verifying one of the first-order relations $=$, $<$, and $>$ on the corresponding valuations stored on stack two. This is an application of induction, since the valuations are of smaller nested depth, i.e. $ind_{d-1}(val(y))$ and $ind_{d-1}(val(y'))$.

Note that the number of possible $symb_u$, $symb_v$, and variable orders is independent of the indexing depth d and the input length len, but exponential in the size of φ.

Contexts and Phases. The verification mechanism begins by pushing the valuation onto stack two. This process either succeeds with the final valuation on stack two, which costs one context or phase, but does not fix the stack for the phase, or fails and takes at most $d+1$ contexts and d phases (Appendices D.3, D.2).

Then, Eve proposes information. Ana may doubt that $symb_1(y)$ or $symb_2(y)$ equals $u_{val(y)}$ resp. $v_{val(y)}$. Popping the first stack introduces a second context and sets the stack for the first phase. The comparison routine adds $(d-1)+2$ contexts and phases (Appendix D.1). Since the context and phase for popping merges with the first context or phase of this routine, the resulting play has up to $d+2$ contexts and $d+1$ phases.

Alternatively, Ana may doubt an entry $y\,\theta\,y'$ of the variable order. This includes removing irrelevant variable values from stack two, which continues the first context and sets the stack for the first phase. Then $G^{d-1}_{\varphi_\theta}$ adds up to $(d-1)+2$ contexts and $(d-1)+1$ phases by the induction hypothesis. This yields a bound of $d+2$ contexts and d phases.

If Ana believes the proposal, the game ends without a further context or phase. The maximum across all plays is thus $d+2$ contexts and $d+1$ phases.

References

1 CPAchecker. URL: `https://cpachecker.sosy-lab.org/index.php`.

2 ULTIMATE Automizer and ULTIMATE Taipan. URL: `https://monteverdi.informatik.uni-freiburg.de/tomcat/Website/`.

3 K. Aehlig. A finite semantics of simply-typed lambda terms for infinite runs of automata. *LMCS*, 3(3), 2007.

4 S. Akshay, P. Gastin, S. Krishna, and S. Roychowdhury. Revisiting underapproximate reachability for multipushdown systems. In *TACAS*, volume 12078 of *LNCS*, pages 387–404. Springer, 2020. `doi:10.1007/978-3-030-45190-5_21`.

5 M. F. Atig. From multi to single stack automata. In *CONCUR*, volume 6269 of *LNCS*, pages 117–131. Springer, 2010.

6 M. F. Atig. Global model checking of ordered multi-pushdown systems. In *FSTTCS*, volume 8 of *LIPIcs*, pages 216–227. Dagstuhl, 2010.

7 M. F. Atig, A. Bouajjani, K. N. Kumar, and P. Saivasan. Model checking branching-time properties of multi-pushdown systems is hard. *CoRR*, abs/1205.6928, 2012. URL: `http://arxiv.org/abs/1205.6928`.

8 M. F. Atig, A. Bouajjani, K. N. Kumar, and P. Saivasan. Parity games on bounded phase multi-pushdown systems. In *NETYS*, volume 10299 of *LNCS*, pages 272–287. Springer, 2017.

9 T. A. Beyene, S. Chaudhuri, C. Popeea, and A. Rybalchenko. A constraint-based approach to solving games on infinite graphs. In *POPL*, pages 221–234. ACM, 2014.

10 T. A. Beyene, S. Chaudhuri, C. Popeea, and A. Rybalchenko. Recursive games for compositional program synthesis. In *VSTTE*, volume 9593 of *LNCS*, pages 19–39. Springer, 2015.

11 D. Beyer. Advances in automatic software verification: SV-COMP 2020. In *TACAS*, volume 12079 of *LNCS*, pages 347–367. Springer, 2020.

12 D. Beyer, M. Dangl, and P. Wendler. A unifying view on SMT-based software verification. *JAR*, 60(3):299–335, 2018.

13 D. Beyer and M. E. Keremoglu. CPAchecker: A tool for configurable software verification. In *Proc. CAV*, volume 6806 of *LNCS*, pages 184–190. Springer, 2011.

14 A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR*, volume 1243 of *LNCS*, pages 135–150. Springer, 1997.

15 A. Bouajjani and A. Meyer. Symbolic reachability analysis of higher-order context-free processes. In *FSTTCS*, volume 3328 of *LNCS*, pages 135–147. Springer, 2004.

16 L. Breveglieri, A. Cherubini, C. Citrini, and S. Crespi-Reghizzi. Multi-push-down languages and grammars. *Int. J. Found. Comput. Sci.*, 7(3):253–292, 1996.

17 C. H. Broadbent, A. Carayol, M. Hague, and O. Serre. A saturation method for collapsible pushdown systems. In *ICALP*, volume 7392 of *LNCS*, pages 165–176. Springer, 2012.

18 C. H. Broadbent, A. Carayol, M. Hague, and O. Serre. C-SHORe: a collapsible approach to higher-order verification. In *ICFP*, pages 13–24. ACM, 2013.

19 C. H. Broadbent and N. Kobayashi. Saturation-based model checking of higher-order recursion schemes. In *CSL*, volume 23 of *LIPIcs*, pages 129–148. Dagstuhl, 2013.

20 J. R. Büchi. Regular canonical systems. *Archiv für mathematische Logik und Grundlagenforschung*, 6(3):91–111, 1964.

21 T. Cathcart Burn, C.-H. L. Ong, and S. J. Ramsay. Higher-order constrained Horn clauses for verification. *Proc. ACM Program. Lang.*, 2(POPL):11:1–11:28, 2018.

22 Aiswarya C. *Verification of communicating recursive programs via split-width*. PhD thesis, ENS Cachan, 2014.

23 T. Cachat. Symbolic strategy synthesis for games on pushdown graphs. In *ICALP*, volume 2380 of *LNCS*, pages 704–715. Springer, 2002.

24 T. Cachat and I. Walukiewicz. The complexity of games on higher order pushdown automata. *CoRR*, abs/0705.0262, 2007. URL: `http://arxiv.org/abs/0705.0262`.

25 A. Carayol and M. Hague. Saturation algorithms for model-checking pushdown systems. In *AFL*, volume 151 of *EPTCS*, pages 1–24, 2014.

26 A. Carayol, M. Hague, A. Meyer, C.-H. L. Ong, and O. Serre. Winning regions of higher-order pushdown games. In *LICS*, pages 193–204. IEEE, 2008.

27 A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *JACM*, 28(1):114–133, 1981.

28 B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic*. CUP, 2012.

29 A. Cyriac, P. Gastin, and K. N. Kumar. MSO decidability of multi-pushdown systems via split-width. In *CONCUR*, volume 7454 of *LNCS*, pages 547–561. Springer, 2012. `doi:10.1007/978-3-642-32940-1_38`.

30 H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

31 J. Esparza. On the decidability of model checking for several μ-calculi and Petri nets. In *CAAP*, volume 787 of *LNCS*, pages 115–129. Springer, 1994.

32 A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. In *Infinity*, volume 9 of *ENTCS*, pages 27–37. Elsevier, 1997.

33 E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*. Springer, 2002.

34 C. Grellois and P.-A. Melliès. Finitary semantics of linear logic and higher-order model-checking. In *MFCS*, volume 9234 of *LNCS*, pages 256–268. Springer, 2015.

35 M. Hague. Saturation of concurrent collapsible pushdown systems. In *FSTTCS*, volume 24 of *LIPIcs*, pages 313–325. Dagstuhl, 2013.

36 M. Hague, R. Meyer, and S. Muskalla. Domains for Higher-Order Games. In *MFCS*, volume 83 of *LIPIcs*, pages 59:1–59:15. Dagstuhl, 2017.

37 M. Hague and C.-H. L. Ong. Symbolic backwards-reachability analysis for higher-order pushdown systems. *LMCS*, 4(4), 2008.

38 M. Heizmann, Y.-W. Chen, D. Dietsch, M. Greitschus, A. Nutz, B. Musa, C. Schätzle, C. Schilling, F. Schüssele, and Andreas Podelski. Ultimate Automizer with an on-demand construction of Floyd-Hoare automata. In *TACAS*, volume 10206 of *LNCS*, pages 394–398. Springer, 2017.

39 M. Heizmann, J. Hoenicke, and A. Podelski. Nested interpolants. In *POPL*, pages 471–482. ACM, 2010.

40 T. A. Henzinger. Games in system design and verification. In *TARK*, pages 1–4. National University of Singapore, 2005.

41 M. Hofmann and J. Ledent. A cartesian-closed category for higher-order model checking. In *LICS*, pages 1–12. IEEE, 2017.

42 M. Jurdzinski. Small progress measures for solving parity games. In *STACS*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000. doi:10.1007/3-540-46541-3.

43 M. Jurdzinski and R. Lazic. Succinct progress measures for solving parity games. In *LICS*, pages 1–9. IEEE, 2017.

44 M. Jurdzinski, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. C.*, 38(4):1519–1532, 2008.

45 N. Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *POPL*, pages 416–428. ACM, 2009.

46 N. Kobayashi and C.-H. L. Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *LICS*, pages 179–188. IEEE, 2009.

47 S. La Torre, P. Madhusudan, and G. Parlato. A robust class of context-sensitive languages. In *LICS*, pages 161–170. IEEE, 2007.

48 S. La Torre, P. Madhusudan, and G. Parlato. Model-checking parameterized concurrent programs using linear interfaces. In *CAV*, volume 6174 of *LNCS*, pages 629–644. Springer, 2010.

49 S. La Torre and M. Napoli. Reachability of multistack pushdown systems with scope-bounded matching relations. In *CONCUR*, volume 6901 of *LNCS*, pages 203–218. Springer, 2011.

50 P. Madhusudan and G. Parlato. The tree width of auxiliary storage. In *POPL*, pages 283–294. ACM, 2011.

51 C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS*, pages 81–90. IEEE, 2006.

52 S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS*, volume 3440 of *LNCS*, pages 93–107. Springer, 2005.

53 G. Ramalingam. Context-sensitive synchronization-sensitive analysis is undecidable. *ACM ToPLaS*, 22(2):416–430, 2000.

54 T. Reps, S. Horwitz, and M. Sagiv. Precise interprocedural dataflow analysis via graph reachability. In *POPL*, pages 49–61. ACM, 1995.

55 S. Salvati and I. Walukiewicz. A model for behavioural properties of higher-order programs. In *CSL*, volume 41 of *LIPIcs*, pages 229–243. Dagstuhl, 2015.

56 A. Seth. Games on higher order multi-stack pushdown systems. In *RP*, volume 5797 of *LNCS*, pages 203–216. Springer, 2009.

57 A. Seth. Games on multi-stack pushdown systems. In *LFCS*, volume 5407 of *LNCS*, pages 395–408. Springer, 2009.

58 M. Sharir and A. Pnueli. Two approaches to interprocedural data flow analysis. TR 2, NYU, 1978.

59 L. J. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, MIT, 1974.

60 S. van der Wall. Bounded analysis of concurrent and recursive programs. Master's thesis, TU Braunschweig, 2019. URL: `http://www.tcs.cs.tu-bs.de/documents/thesis-van-der-wall-2019.pdf`.

61 I. Walukiewicz. Pushdown processes: games and model-checking. *IC*, 164(2):234–263, 2001.

62 I. Walukiewicz. A landscape with games in the background. In *LICS*, pages 356–366. IEEE, 2004.

A Details on Section 3

We argue that Theorem 1 follows from Lemma 2. To check whether Eve wins G, by the first statement in the lemma it is sufficient to check whether she wins F. We thus construct the finite game F, which takes $(k-2)$-fold exponential time. We apply a modern parity game solving algorithm to determine the winner in F, like the recent subexponential algorithms [43, 44]. The algorithm takes time exponential only in the priorities of F, which by the second statement in the lemma are the priorities $[0..max]$ in G. We thus obtain an overall time complexity

$$exp_{k-2}(\text{poly}(|G|)) + exp_{k-2}(\text{poly}(|G|))^{max} \leq exp_{k-2}((1+max)\,\text{poly}(|G|)),$$

which is still $exp_{k-2}(\text{poly}(|G|))$.

A.1 Size of the sets of ordered summaries

We estimate the size of the optimized sets of summaries by induction on the stack. In the base case:

$$|\mathbb{OS}_n| \;=\; |Q| \cdot max \;=\; exp_{n-n}(\mathcal{O}(|G|^2))\,.$$

For the induction step, assume $|\mathbb{OS}_{r+1}| = exp_{n-(r+1)}(\mathcal{O}(|G|^2))$. For stack r we obtain

$$\begin{aligned} |\mathbb{OS}_r| \;&=\; |Q| \cdot \prod_{j=r+1}^{n} |\Gamma| \cdot max \cdot 2^{\sum_{i=r+1}^{n} |\mathbb{OS}_i|} \;\leq\; |Q| \cdot |\Gamma|^{n-r} \cdot max^{n-r} \cdot \prod_{j=r+1}^{n} 2^{n|\mathbb{OS}_{r+1}|} \\ &\leq\; |G|^{2(n-r)+1} \cdot 2^{(n-r)n|\mathbb{OS}_{r+1}|} \;\leq\; 2^{(2n+1)|G|+n^2|\mathbb{OS}_{r+1}|} \end{aligned}$$

The equality is by definition. The first inequality relies on $|\mathbb{OS}_r| \geq \ldots \geq |\mathbb{OS}_n|$. Since n is a constant,

$$2^{(2n+1)|G|+n^2|\mathbb{OS}_{r+1}|} \;=\; 2^{(2n+1)|G|+n^2 exp_{n-(r+1)}(\mathcal{O}(|G|^2))} = exp_{n-r}(\mathcal{O}(|G|^2))\,.$$

A.2 Size of the sets of context summaries

We estimate the size of the optimized sets of summaries by induction on the context. In the base case:

$$|\mathbb{OS}_{r,1}| \;=\; |\mathbb{OS}_{r,2}| \;\leq\; |\mathbb{OS}_{r,k}| \;\leq\; |Q| \cdot max \cdot |\Gamma|^{k-1} \cdot max^{k-1} \;=\; exp_{k-k}(\mathcal{O}(|G|^{2k}))\,.$$

The function is indeed polynomial as the number of stacks k is fixed. For the induction step, assume $|\mathbb{OS}_{r,c+1}| = exp_{k-(c+1)}(\mathcal{O}(|G|^{2k}))$ with $4 \leq c+1 \leq k$. For context c we obtain

$$\begin{aligned} |\mathbb{OS}_{r,c}| \;&=\; |Q| \cdot max \cdot \prod_{j=1}^{r-1} |\Gamma| \cdot max \cdot 2^{\sum_{l=c+1}^{k} |\mathbb{OS}_{j,l}|} \\ &\quad \cdot \prod_{j=r+1}^{k} |\Gamma| \cdot max \cdot 2^{\sum_{l=c+1}^{k} |\mathbb{OS}_{j,l}|} \\ &\leq |G|^{2k} \cdot \prod_{j=1}^{k} 2^{k|\mathbb{OS}_{j,c+1}|} \;\leq\; 2^{2k|G|+k^2|\mathbb{OS}_{r,c+1}|}\,. \end{aligned}$$

The equality is by definition. The first inequality relies on $|\mathbb{OS}_{r,c}| \geq \ldots \geq |\mathbb{OS}_{r,k}|$. Also, $|\mathbb{OS}_{j,c}| = |\mathbb{OS}_{j',c}|$ for stacks $j \neq j'$ by Symmetry.

$$2^{2k|G|+k^2|\mathbb{OS}_{r,c+1}|} \;=\; 2^{2k|G|+k^2\, exp_{k-(c+1)}(\mathcal{O}(|G|^{2k}))} = exp_{k-c}(\mathcal{O}(|G|^{2k})))\,.$$

B Equivalence of the MPDG G and the finite game F

In the following, we give a construction intuition for how winning strategies for Eve can be converted between G and F.

B.1 Transforming a winning strategy from F to G

We proceed in the following steps: First, we introduce a strategy transducer T to transport σ from $F = (V_F, E_F, own, \Omega)$ to $G = (P, own, \Omega)$. Then, we define the strategy ν it implements and show an invariant between plays compliant with that strategy and runs of T (Lemma 14). Next, we show that if T can perform a run from a stair $(q, \mathcal{R})$ to some configuration $(q', \mathcal{R}')$, then there is a play compliant to σ in F from $Tr(q, \mathcal{R})$ to $Tr(q', \mathcal{R}')$ (Lemma 15 and Lemma 16). Lastly, putting the previous results together, we get that a run compliant to ν that is losing for Eve leads to a play in F that is compliant to σ losing for Eve, which contradicts it being a winning strategy. Thus, ν is also a winning strategy.

Intuitively, the transducer T is a multi-pushdown system with the same number of stacks as P. At any point in the game, the stack heights of T are identical to the stack heights of P. However, the transitions are amplified to update all top of stack contents with each transition.

The strategy automaton remembers information of the finite state game in the following sense. If the finite state game would be in a position $(\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})$, then the strategy automaton is in state q and the top of stack symbol of each stack r is a tuple $(\mathcal{T}[r], \mathcal{M}[r], \mathcal{S}[r])$. The automaton mimics a play of G. Whenever Eve has the next move, she can use it to follow her strategy σ for F.

▶ **Definition 11.** *Given a strategy σ for Eve in F, the* strategy automaton *T is a tuple $T = (Q, \Gamma \times [0..max] \times 2^{\mathbb{OS}}, \mapsto, n)$ where Q is the state space, $\Gamma \times [0..max] \times 2^{\mathbb{OS}}$ is the stack alphabet, and $\mapsto$ is a transition relation, which we will define directly on the set of configurations C_T.*

A configuration of $T \in C_T$ is $(q, \mathcal{R})$, where $\mathcal{R} : [1..n] \to (\Gamma \times [0..max])^* \times 2^{\mathbb{OS}}$ are the stack contents. For each stack j, the stack contents $\mathcal{R}[j] = {}^{j}\mathcal{R}_{|\mathcal{R}[j]|}\, {}^{j}\mathcal{R}_{|\mathcal{R}[j]|-1} \ldots {}^{j}\mathcal{R}_1$ is a seqence of tuples, and ${}^{j}\mathcal{R}_{|\mathcal{R}[j]|}$ is the top of stack symbol. We denote the tuple contents by ${}^{j}\mathcal{R}_i = ({}^{j}\gamma_i, {}^{j}m_i, {}^{j}S_i)$.

For the sake of notation, we introduce a context sensitive top of stack pointer $\uparrow$. It obtains the index value of the top of stack symbol:

$$\mathcal{R}[j] = {}^{j}\mathcal{R}_{|\mathcal{R}[j]|}\,{}^{j}\mathcal{R}_{|\mathcal{R}[j]|-1}\dots{}^{j}\mathcal{R}_1 = {}^{j}\mathcal{R}_{\uparrow}\,{}^{j}\mathcal{R}_{\uparrow-1}\dots{}^{j}\mathcal{R}_1.$$

This notation also carries over to the individual tuple contents. Thus,

$${}^{j}\gamma_{\uparrow} = {}^{j}\gamma_{|\mathcal{R}[j]|} \qquad {}^{j}S_{\uparrow} = {}^{j}S_{|\mathcal{R}[j]|} \qquad {}^{j}m_{\uparrow} = {}^{j}m_{|\mathcal{R}[j]|}.$$

▶ **Definition 12.** *We define a transformation function $Tr : C_T \to V_F$ mapping configurations of T to positions in F by $Tr(q,\mathcal{R}) = (\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})$, where $\mathcal{T}[j] = {}^{j}\gamma_{\uparrow}$, $\mathcal{M}[j] = {}^{j}m_{\uparrow}$ and $\mathcal{S}[j] = {}^{j}S_{\uparrow}$.*

To define the transition relation $\mapsto \subseteq C_T \times \delta \times C_T$, let $Tr(q,\mathcal{R}) = (\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})$ and $Tr(q',\mathcal{R}') = (\text{Check}, q', \mathcal{T}', \mathcal{M}', \mathcal{S}')$. For every transition rule τ of P, T has a transition $(q,\mathcal{R}) \overset{\tau}{\mapsto} (q',\mathcal{R}')$, if either q is owned by Eve and σ tells her for position $Tr(q,\mathcal{R})$ to use the move simulating τ, or $own(q) = \text{Ana}$.

Further, for all stacks $j \neq r$, $\mathcal{T}'[j] = {}^{j}\gamma'_{\uparrow} = {}^{j}\gamma_{\uparrow} = \mathcal{T}'[j]$ and T updates the stackcontents $\mathcal{R}$ to $\mathcal{R}'$ dependent on the transition:

Case 1 (τ is an internal transition (q,r,q'))**:** $\mathcal{R} = \mathcal{R}'$ except for each stack j, ${}^{j}m'_{\uparrow} = \max\{\Omega(q'), {}^{j}m_{\uparrow}\}$.

Case 2 (τ is a push transition (q,r,s,q'))**:** $\mathcal{R} = \mathcal{R}'$, except for each stack $j \neq r$, ${}^{j}m'_{\uparrow} = \max\{\Omega(q'), {}^{j}m_{\uparrow}\}$. And for stack r, $\mathcal{R}'[r] = (s, \Omega(q'), \mathcal{S})\mathcal{R}[r]$, where $\mathcal{S}$ is determined by σ:

$$\sigma(\text{Push}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s) = (\text{Claim}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s, S).$$

Case 3 (τ is a pop transition $(q,s,r,q') \in \delta_{pop}$ and $\mathcal{R}[j] = \varepsilon$ for all $j < r$)**:**
Case 3.1 (there is $(q', \mathcal{M}[r], \mathcal{T}_{[r\mapsto\varepsilon]}, \mathcal{M}_{[r\mapsto 0]}, \mathcal{S}) \in {}^{r}S_{\uparrow}$ s.t. for each $j > r$, $\mathcal{S}[j] \subseteq {}^{j}S_{\uparrow}$)*:* $\mathcal{R} = \mathcal{R}'$, except for each $j > r$, ${}^{j}m'_{\uparrow} = \max\{\Omega(q'), {}^{j}m_{\uparrow}\}$ and ${}^{j}S'_{\uparrow} = \mathcal{S}[j]$. And for stack r, $\mathcal{R}'[r] = ({}^{r}\gamma_{\uparrow-1}, \max\{{}^{r}m_{\uparrow}, {}^{r}m_{\uparrow-1}, \Omega(q')\}, {}^{r}S_{\uparrow-1}){}^{r}\mathcal{R}_{\uparrow-2}\,{}^{r}\mathcal{R}_{\uparrow-3}\dots{}^{r}\mathcal{R}_1$.
Case 3.2 (there is **no** $(q', \mathcal{M}[r], \mathcal{T}_{[r\mapsto\varepsilon]}, \mathcal{M}_{[r\mapsto 0]}, \mathcal{S}) \in {}^{r}S_{\uparrow}$ s.t. for each $j > r$, $\mathcal{S}[j] \subseteq {}^{j}S_{\uparrow}$)*:* $\mathcal{R} = \mathcal{R}'$, except for each $j > r$, ${}^{j}m'_{\uparrow} = \max\{\Omega(q'), {}^{j}m_{\uparrow}\}$ and for stack r, $\mathcal{R}'[r] = ({}^{r}\gamma_{\uparrow-1}, \max\{{}^{r}m_{\uparrow}, {}^{r}m_{\uparrow-1}, \Omega(q')\}, {}^{r}S_{\uparrow-1}){}^{r}\mathcal{R}_{\uparrow-2}\,{}^{r}\mathcal{R}_{\uparrow-3}\dots{}^{r}\mathcal{R}_1$.

Note that case 3.2 is almost a copy of case 3.1. It simply does not find a matching summary. We will later use case distinction on whether a transition is due to case 3.1 or 3.2. Also note that when a configuration $(q,\mathcal{R})$ belongs to Eve, T can only perfom the transition which σ wants to simulate from $Tr(q,\mathcal{R})$.

The following lemma tells us, that during a run of T, the summary for a stack symbol can only shrink during the run.

▶ **Lemma 13.** *Let $\eta = (q,\mathcal{R})$ and $\eta' = (q',\mathcal{R}')$ with $\eta \overset{\tau}{\mapsto} \eta'$ in T. For each stack $j \in [1..n]$, let $sh_j = \min\{|\mathcal{R}[j]|, |\mathcal{R}[j]'|\}$.*
For each stack $j \in [1..n]$, ${}^{j}S'_{sh_j} \subseteq {}^{j}S_{sh_j}$ and for each $u \in [1..sh_j - 1]$, ${}^{j}\mathcal{R}_u = {}^{j}\mathcal{R}'_u$.

Proof. By construction. Only transition case 3.1 changes ${}^{j}S_{sh_j}$. When transition case 3.1 happens for a stack r, it changes on stacks $j > r$ the set ${}^{j}S_{\uparrow}$ to $\mathcal{S}[j]$. But $\mathcal{S}[j] \subseteq {}^{j}S_{\uparrow}$ by the conditions for transition 3. ◀

A run of the strategy automaton is a sequence $(q_{\text{init}}, (\varepsilon, \varnothing, \Omega(q_{init})^n) = \eta_0 \overset{\tau_0}{\longmapsto} \eta_1 \overset{\tau_1}{\longmapsto} \dots$. We can use the strategy automaton to define a strategy ν on G for starting positions of the form $(q_{\text{init}}, \varepsilon^n)$. We define ν inductively on the play prefix. The proof of the next lemma does both, it defines the strategy and states an invariant between plays in G conform to it and runs of T.

▶ **Lemma 14.** *Let $\pi = \pi_0 \xrightarrow{\tau_0} \dots \xrightarrow{\tau_{l-1}} \pi_l$ be a play prefix compliant to the strategy ν and $\eta = \eta_0 \overset{\tau_0}{\longmapsto} \dots \overset{\tau_{l-1}}{\longmapsto} \eta_l$ the corresponding run of T. At any position $p \in \mathbb{N}$, if $\eta_p = (q, \mathcal{R})$, then*

1. $\pi_p = (q, ({}^1\gamma_\uparrow\, {}^1\gamma_{\uparrow-1} \dots {}^1\gamma_1, {}^2\gamma_\uparrow\, {}^2\gamma_{\uparrow-1} \dots {}^2\gamma_1, \dots, {}^n\gamma_\uparrow\, {}^n\gamma_{\uparrow-1} \dots {}^n\gamma_1))$.
2. *If $\operatorname{lup}_j^\pi(p) \neq \bot$ is defined, then $\max_{u \in [\operatorname{lup}_j^\pi(p)..p]}\{\Omega(q^u)\} = {}^j m_\uparrow$. And if $\operatorname{lup}_j^\pi(p) = \bot$ is undefined, then $\max_{u \in [0..p]}\{\Omega(q^u)\} = {}^j m_\uparrow$.*

Proof.

▶ **Base Case (π_0, η_0).** $\pi_0 = (q_{\text{init}}, \varepsilon^n)$, $\eta_0 = ((q_{\text{init}}, 0, 0), (\varepsilon, \varnothing, \Omega(q_{init}))^n)$. Both invariants hold immediately.

▶ **Inductive Case (π_i to π_{i+1}, η_i to η_{i+1}).** Let $\pi = \pi_0 \overset{\tau_0}{\longmapsto} \dots \overset{\tau_{i-1}}{\longmapsto} \pi_i = (q, \mathcal{P}) \overset{\tau_i}{\longmapsto} (q, \mathcal{P}')$ be a play prefix in G. By induction, the strategy automaton has a run prefix $\eta_0 \overset{\tau_1}{\longmapsto} \dots \overset{\tau_{i-1}}{\longmapsto} \eta_i = (q, \mathcal{R})$, that fulfills the lemma.

In case of $own(\eta_i) = \text{Eve}$, by construction, T has only an enabled transition for a single τ at η_i. Namely the one, which σ would choose to simulate from $\psi(q, \mathcal{P})$. We choose $\nu(\pi_0 \overset{\tau_0}{\longmapsto} \dots \overset{\tau_{i-1}}{\longmapsto} \pi_i)$ to use that transition.

In the other case, for every transition τ_i enabled in π_i, a corresponding is enabled in T by construction.

If π is compliant with ν, we continue η by the corresponding enabled transition $\eta_i = (q, \mathcal{R}) \overset{\tau_i}{\longmapsto} (q', \mathcal{R}') = \eta_{i+1}$.

Remains to show, that the lemma holds for position $i+1$ as well.

For all stacks $j \neq r$, the third condition is fulfilled by construction and induction: By induction, the maximal parity seen since position $\operatorname{lup}_j^\pi(i)$ up to π_i is ${}^j m_\uparrow$. For all stacks $j \neq r$, the stack height does not change. There has been seen a new parity $\Omega(q')$. The construction sets ${}^j m'_\uparrow$ appropriately to $\max\{\Omega(q'), {}^j m_\uparrow\}$.

Case 1 ($\tau_i = (q, r, q') \in \delta_{int}$)**:** The first condition is fulfilled trivially, since the symbols in the stack contents did not change. For the second condition, in this case, the same arguments apply as for the other stacks.

Case 2 ($\tau_i = (q, r, s, q') \in \delta_{push}$)**:** For the first condition, the symbols in the stack contents don't change for all stacks $j \neq r$. For stack r however, $\mathcal{R}'[r] = (s, \Omega(q'), \mathcal{S})\mathcal{R}[r]$, such that s is the new additional symbol, which is the pushed symbol by τ_i. This meets the lemma's requirenment.

For the third condition, $\Omega(q')$ is the only parity seen since the push of s.

Case 3 ($\tau_i = (q, s, r, q') \in \delta_{pop}$)**:** Be aware that T contains multiple possible transitions for τ_i. These only differ in the sets of summaries ${}^j S_\uparrow$ for each stack $j > r$. The lemma does not state any conditions on the summary sets, so there is no need for case distinction.

For the first condition, the symbols in the stack contents don't change for stacks $j \neq r$. For stack r however, $(s, \Omega(q'), \mathcal{S})\mathcal{R}[r]' = \mathcal{R}[r]$, removing the top most symbol from the stack, which is the symbol removed by τ_i. This meets the lemma's requirenment.

For the second condition, the maximal parity seen since the push of ${}^{r}\gamma'_{\uparrow}$ is determined by the maximal parity seen since the last unmatched push before pushing s, the just popped symbol. This is the position $\mathrm{lup}_r^{\pi}(i+1) = \mathrm{lup}_r^{\pi}(\mathrm{lup}_r^{\pi}\, i)$.

The correct priority is chosen by ${}^{r}m'_{\uparrow} = \max\{{}^{r}m_{\uparrow}, {}^{r}m_{\uparrow-1}, \Omega(q')\}$. ◀

The next lemma ensures that the strategy automaton T and the mapping Tr of configurations from T to positions in F behave well with respect to the strategy σ itself. When the strategy automaton is able to make a move $\eta \overset{\tau}{\mapsto} \eta'$, then there should be transitions $Tr(\eta) \mapsto \cdots \mapsto Tr(\eta')$ compliant with σ in F.

▶ **Lemma 15.** *Let σ be a strategy for Eve in F and T the strategy automaton. Let $\eta = \eta_0 \overset{\tau_0}{\mapsto} \eta_1 \overset{\tau_1}{\mapsto} \dots$ be a computation of T starting in a stair η_0.*

For each position $i \in \mathbb{N}$, let

$$\eta_i = (q^i, \mathcal{R}^i) \qquad\qquad Tr(\eta_i) = (\mathit{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i)$$

where for each stack j, $\mathcal{R}^i[j] = ({}^{j}\gamma^i_{\uparrow}, {}^{j}m^i_{\uparrow}, {}^{j}S^i_{\uparrow})({}^{j}\gamma^i_{\uparrow-1}, {}^{j}m^i_{\uparrow-1}, {}^{j}S^i_{\uparrow-1}) \dots ({}^{j}\gamma^i_1, {}^{j}m^i_1, {}^{j}S^i_1)$.

Let $i \in \mathbb{N}$. Let $\eta_i \overset{\tau_i}{\mapsto} \eta_{i+1}$ be a transition of T that is not by transition case 3.2.

Then the following transitions exist in F and are compliant with σ.

If $\tau_i = (q^i, r, q^{i+1}) \in \delta_{int}$:

$$Tr(\eta_i) = (\mathit{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i) \mapsto (\mathit{Check}, q^{i+1}, \mathcal{T}^{i+1}, \mathcal{M}^{i+1}, \mathcal{S}^{i+1}) = Tr(\eta_{i+1})$$

$\tau_i = (q, r, s, q') \in \delta_{push}$:

$$\begin{aligned} Tr(\eta_i) &= (\mathit{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i) \\ &\mapsto (\mathit{Push}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s) \\ &\mapsto (\mathit{Claim}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s, {}^{r}S^{i+1}_{\uparrow}) \\ &\mapsto (\mathit{Check}, q^{i+1}, \mathcal{T}^i_{[r\mapsto s]}, \mathcal{M}^{i+1}, \mathcal{S}^i_{[r\mapsto {}^{r}S^{i+1}_{\uparrow}]}) = Tr(\eta_{i+1}) \end{aligned}$$

If $\tau_i = (q, s, r, q') \in \delta_{pop}$: *Since η_0 is a stair, position i is in a push-pop-pair (t, i).*

$$\begin{aligned} Tr(\eta_t) &= (\mathit{Check}, q^t, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t) \\ &\mapsto (\mathit{Push}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s) \\ &\mapsto (\mathit{Claim}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s, {}^{r}S^{t+1}_{\uparrow}) \\ &\mapsto (\mathit{Jump}_r, q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^{i+1}, \mathcal{M}^i_{[r\mapsto 0]}, \mathcal{S}^{i+1}, \mathcal{M}^t[r]) \\ &\mapsto (\mathit{Check}, q^{i+1}, \mathcal{T}^{i+1}, \mathcal{M}^{i+1}, \mathcal{S}^{i+1},) = Tr(\eta_{i+1}) \end{aligned}$$

Proof. For any of the following, the correctness of q^{i+1} is immediate and therefore skipped. Also be aware that both, F and T, respect the orderedness restriction.

Case 1 ($\tau_i = (q^i, r, q^{i+1}) \in \delta_{int}$)**:** By construction of F, τ_i causes the existence of the transition

$$\begin{aligned} Tr(\eta_i) = (\mathrm{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i) &\mapsto (\mathrm{Check}, q^{i+1}, \mathcal{T}^i, \mathcal{M}', \mathcal{S}^i) \\ &\overset{!}{=} (\mathrm{Check}, q^{i+1}, \mathcal{T}^{i+1}, \mathcal{M}^{i+1}, \mathcal{S}^{i+1}) = Tr(\eta_{i+1}) \end{aligned}$$

Remains to show the equality of the last two check states. By definition of T, $\mathcal{R}^i = \mathcal{R}^{i+1}$. Together with Tr we get that $\mathcal{T}^i = \mathcal{T}^{i+1}$ and $\mathcal{S}^i = \mathcal{S}^{i+1}$. By construction of F and T, for any stack j,

$$\mathcal{M}[j]^{i+1} = {}^{j}m^{i+1}_{\uparrow} = \max\{{}^{j}m^i_{\uparrow}, \Omega(q^{i+1})\} = \max\{\mathcal{M}[j]^i, \Omega(q^{i+1})\} = \mathcal{M}[j]'.$$

Also, by construction of T, the transition $\eta_i \overset{\tau_i}{\mapsto} \eta_{i+1}$ only exists, if $own(q^i) = \text{Ana}$ or $\sigma(\text{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i) = (\text{Check}, q^{i+1}, \mathcal{T}^i, \mathcal{M}', \mathcal{S}^i)$. The transition is compliant with σ.

Case 2 ($\tau_i = (q^i, r, s, q^{i+1}) \in \delta_{push}$)**:** By construction of F, τ_i causes the existence of the transitions

$$\begin{aligned} Tr(\eta_i) &= (\text{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i) \\ &\mapsto (\text{Push}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s) \\ &\mapsto (\text{Claim}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s, {}^rS_\uparrow^{i+1}) \\ &\mapsto (\text{Check}, q^{i+1}, \mathcal{T}^i_{[r\mapsto s]}, \mathcal{M}', \mathcal{S}^i_{[r\mapsto {}^rS_\uparrow^{i+1}]}) \\ &\overset{!}{=} (\text{Check}, q^{i+1}, \mathcal{T}^{i+1}, \mathcal{M}^{i+1}, \mathcal{S}^{i+1}) = Tr(\eta_{i+1}) \end{aligned}$$

Remains to show the equality of the last two check states. By definition of T, $\mathcal{R}^i = \mathcal{R}^{i+1}$, except for each $j \neq r$, ${}^jm_\uparrow^{i+1} = \max\{\Omega(q^{i+1}), {}^jm_\uparrow^i\}$, and for stack r, $\mathcal{R}^{i+1}[r] = (s, \Omega(q^{i+1}), {}^rS_\uparrow^{i+1})\mathcal{R}[r]$, where

$$\sigma(\text{Push}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s) = (\text{Claim}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s, {}^rS_\uparrow^{i+1}).$$

This immediately yields $\mathcal{T}^i_{[r\mapsto s]} = \mathcal{T}^{i+1}$ and $\mathcal{S}^i_{[r\mapsto {}^rS_\uparrow^{i+1}]} = \mathcal{S}^{i+1}$.

Also, $\mathcal{M}'[r] = \Omega(q^{i+1}) = {}^rm_\uparrow^{i+1}$ and for each stack $j \neq r$:

$$\mathcal{M}'[j] = \max\{\Omega(q^{i+1}), \mathcal{M}[j]^i\} = \max\{\Omega(q^{i+1}), {}^jm_\uparrow^i\} = {}^jm_\uparrow^{i+1}.$$

Since the transition $\eta_i \overset{\tau_i}{\mapsto} \eta_{i+1}$ exists in T, by its construction,

$$\sigma(\text{Push}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s) = (\text{Claim}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s, {}^rS_\uparrow^{i+1}).$$

Also, either $own(q^i) = \text{Ana}$ or

$$\sigma(\text{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i) = (\text{Push}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s).$$

Thus, the transitions exist in F and are compliant with σ.

Case 3 ($\tau_i = (q, r, s, q') \in \delta_{pop}$ and (t, i) is a push-pop-pair)**:** Since (t, i) is a push-pop-pair, there is a push transition $\tau_t = (q^t, r, s, q^{t+1})$.

By construction of F, τ_t causes the existence of the transitions

$$\begin{aligned} Tr(\eta_t) &= (\text{Check}, q^t, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t) \\ &\mapsto (\text{Push}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s) \\ &\mapsto (\text{Claim}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s, {}^rS_\uparrow^{t+1}) \end{aligned}$$

which are compliant with σ, as discussed in the previous case. Since (t, i) is a push-pop-pair, for any position $t < p < i$, $|\mathcal{R}^{t+1}[r]| = |\mathcal{R}^i[r]| \leq |\mathcal{R}^p[r]|$. Then, by repetitive use of Lemma 13, ${}^rS_\uparrow^i \subseteq {}^rS_\uparrow^{t+1}$.

Next, we show that

$$(\text{Claim}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s, {}^rS_\uparrow^{t+1}) \mapsto (\text{Jump}_r, q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^{i+1}, \mathcal{M}^i_{[r\mapsto 0]}, \mathcal{S}^{i+1}, \mathcal{M}^t[r])$$

is a valid transition in F. Since the transition $\eta_i \overset{\tau_i}{\mapsto} \eta_{i+1}$ is not by transition case 3.2, it must be by transition case 3.1 of T. Thus, T finds a summary $(q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^i_{[r\mapsto\varepsilon]}, \mathcal{M}^i_{[r\mapsto 0]}, \mathcal{S}) \in$

$\mathcal{S}^i[r]$ such that for each stack $j \neq r$, $\mathcal{S}[j] \subseteq \mathcal{S}^i[j]$. By construction of F, there is the transition

$$\begin{aligned}&(\text{Claim}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s, {}^rS_\uparrow^{t+1})\\ &\mapsto (\text{Jump}_r, q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^i_{[r\mapsto\mathcal{T}^t[r]]}, \mathcal{M}^i_{[r\mapsto 0]}, \mathcal{S}_{[r\mapsto\mathcal{S}^t[r]]}, \mathcal{M}^t[r])\\ &\qquad\qquad \overset{!}{=} (\text{Jump}_r, q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^{i+1}, \mathcal{M}^i_{[r\mapsto 0]}, \mathcal{S}^{i+1}, \mathcal{M}^t[r])\end{aligned}$$

To find the last equation, we need to identify $\mathcal{T}^i_{[r\mapsto\mathcal{T}^t[r]]} = \mathcal{T}^{i+1}$ and $\mathcal{S}_{[r\mapsto\mathcal{S}^t[r]]} = \mathcal{S}^{i+1}$. Remember that $\tau_i = (q^i, s, r, q^{i+1})$ is a popping transition with

$$\eta_i = (q^i, \mathcal{R}^i) \overset{\tau_i}{\longmapsto} (q^{i+1}, \mathcal{R}^{i+1}) = \eta_{i+1},$$

that used transition case 3.1 of the strategy automaton conditions for a transition with the prediction $\mathcal{S} \in {}^rS^i_\uparrow = \mathcal{S}^i[r]$. Since (t, i) is a push-pop-pair, for all positions $t+1 \leq p \leq i$,

$$|\mathcal{R}^t[r]| = |\mathcal{R}^{t+1}[r]| - 1 = |\mathcal{R}^i[r]| - 1 = |\mathcal{R}^{i+1}[r]| < |\mathcal{R}^p[r]|.$$

As T does not change the symbol of its tuples and by repetitive use of Lemma 13,

$$\mathcal{T}^{i+1}[r] = {}^r\gamma^{i+1}_\uparrow = {}^r\gamma^t_\uparrow = \mathcal{T}^t[r] \quad \text{and} \quad \mathcal{S}^t[r] = {}^rS^t_\uparrow = {}^rS^{t+1}_{\uparrow-1} = {}^rS^i_{\uparrow-1} = {}^rS^{i+1}_\uparrow = \mathcal{S}^{i+1}[r].$$

Since T used $\mathcal{S}$ for its case 3.1 transition, by its construction: For all stacks $j \neq r$, $\mathcal{S}[j]^{i+1} = {}^jS^{i+1}_\uparrow = \mathcal{S}[j]$.

The last transition is

$$\begin{aligned}&(\text{Jump}_r, q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^{i+1}, \mathcal{M}^i_{[r\mapsto 0]}, \mathcal{S}^{i+1}, \mathcal{M}^t[r]) \mapsto (\text{Check}, q^{i+1}, \mathcal{T}^{i+1}, \mathcal{M}', \mathcal{S}^{i+1})\\ &\qquad\qquad \overset{!}{=} (\text{Check}, q^{i+1}, \mathcal{T}^{i+1}, \mathcal{M}^{i+1}, \mathcal{S}^{i+1}) = Tr(\eta_{i+1})\end{aligned}$$

It remains to show $\mathcal{M}' = \mathcal{M}^{i+1}$. For any stack $j > r$, by construction of F and T,

$$\mathcal{M}^{i+1}[j] = {}^jm^{i+1}_\uparrow = \max\{{}^jm^i_\uparrow, \Omega(q^{i+1})\} = \max\{\mathcal{M}^i[j], \Omega(q^{i+1})\} = \mathcal{M}'[j].$$

For stack r, again since (t, i) is a push-pop-pair, for all positions $t+1 \leq p \leq i$ holds:

$$|\mathcal{R}^t[r]| = |\mathcal{R}^{t+1}[r]| - 1 = |\mathcal{R}^i[r]| - 1 = |\mathcal{R}^{i+1}[r]| < |\mathcal{R}^p[r]|.$$

Repetitive use of Lemma 13 leads to ${}^rm^t_\uparrow = {}^rm^{t+1}_{\uparrow-1} = {}^rm^i_{\uparrow-1}$. Finally,

$$\begin{aligned}\mathcal{M}^{i+1}[r] &= \max\{{}^rm^i_\uparrow, {}^rm^i_{\uparrow-1}, \Omega(q^{i+1})\}\\ &= \max\{{}^rm^i_\uparrow, {}^rm^t_\uparrow, \Omega(q^{i+1})\} = \max\{\mathcal{M}^i[r], \mathcal{M}^t[r], \Omega(q^{i+1})\} = \mathcal{M}'[r].\end{aligned}$$

The positions

$$(\text{Claim}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s, {}^rS^{t+1}_\uparrow), \quad (\text{Jump}_r, q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^{i+1}, \mathcal{M}^i_{[r\mapsto 0]}, \mathcal{S}^{i+1}, \mathcal{M}^t[r])$$

are both not owned by Eve. The transitions are compliant with σ. ◀

Let σ be a winning strategy for Eve in F from $(\text{Check}, q_{init}, \varepsilon^n, 0^n, \varnothing^n)$. We use T to derive a strategy ν for Eve in G as described above (Lemma 14). Let there be a play π compliant with ν together with its strategy automaton run η. Towards contradiction, assume π is losing for Eve. We construct a play $\rho = \rho_0 \dots$ in F compliant with σ from $\rho_0 = (\text{Check}, q_{init}, \varepsilon^n, 0^n, \varnothing^n)$ that is losing for Eve.

For each position $i \in \mathbb{N}$, let

$$\eta_i = (q^i, \overline{\mathcal{R}}^i), \qquad Tr(\eta_i) = (\text{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i).$$

where for each stack j, $\mathcal{R}[j]^i = ({}^j\gamma^i_\uparrow, {}^jm^i_\uparrow, {}^jS^i_\uparrow)({}^j\gamma^i_{\uparrow-1}, {}^jm^i_{\uparrow-1}, {}^jS^i_{\uparrow-1}) \dots ({}^j\gamma^i_1, {}^jm^i_1, {}^jS^i_1)$.

In the following, assume that for all $i \in \mathbb{N}$ there is no transition $\eta_i \overset{\tau_i}{\mapsto} \eta_{i+1}$ following transition case 3.2 of T for having a transition. We handle that case later.

▶ **Lemma 16.** *Let $p \in \mathbb{N}$ be a stair of η. There is $\psi : \mathbb{N} \to \mathbb{N}$, such that for each $i \in \mathbb{N}$ with $p \leq i$, there is a play $\rho^i = \rho^i_1 \mapsto \dots \mapsto \rho^i_{\psi(i)}$ of length $\psi(i)$ compliant with σ in F from $\rho^i_1 = Tr(\eta_p)$ to $\rho^i_{\psi(i)} = Tr(\eta_i)$ such that*

$$\max_{u \in [p..i]} \{\Omega(q^u)\} = \max_{u \in [1..\psi(i)]} \{\Omega(\rho_u)\}.$$

Proof. We show this by induction.

▶ **Base Case ($i = p$).** Set $\psi(p) = 1$. Immediatly, $Tr(\eta_p) = Tr(\eta_i)$ and $\Omega(q^p) = \Omega(Tr(\eta_p))$.

▶ **Inductive Case ($i \mapsto i+1$).** Assume, that for each $t \in [p..i]$, there is a play ρ^t compliant with σ from $Tr(\eta_p) = \rho^t_1$ to $Tr(\eta_t) = \rho^t_{\psi(t)}$.

We create a play in F from $Tr(\eta_p)$ to $Tr(\eta_{i+1})$ compliant with σ.

Case 1 ($\tau_i \in \delta_{int}$)**:** Set $\psi(i+1) = \psi(i) + 1$. The desired play is a continuation of ρ^i. By Lemma 15, the following transition is compliant with σ.

$$Tr(\eta_i) = (\text{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i) \mapsto (\text{Check}, q^{i+1}, \mathcal{T}, \mathcal{M}^{i+1}, \mathcal{S}) = Tr(\eta_{i+1})$$

By induction,

$$\begin{aligned}\max_{u \in [p..i+1]} \{\Omega(q^u)\} &= \max\{\max_{u \in [p..i]} \{\Omega(q^u)\}, \Omega(q^{i+1})\} \\ &= \max\{\max_{u \in [1..\psi(i)]} \{\Omega(\rho^i_u)\}, \Omega(\rho_{\psi(t+1)})\} = \max_{u \in [1..\psi(i+1)]} \{\Omega(\rho^{i+1}_u)\}\end{aligned}$$

Case 2 ($\tau_i = (q^i, r, s, q^{i+1}) \in \delta_{push}$)**:** The desired play is a continuation of ρ^i, which ends in $Tr(\eta_i)$. By Lemma 15, the following transitions are compliant with σ.

$$\begin{aligned}Tr(\eta_i) &= (\text{Check}, q^i, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i) \\ &\mapsto (\text{Push}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s) \\ &\mapsto (\text{Claim}_r, \mathcal{T}^i, \mathcal{M}^i, \mathcal{S}^i, q^{i+1}, s, {}^rS^{i+1}_\uparrow) \\ &\mapsto (\text{Check}, q^{i+1}, \mathcal{T}^{i+1}, \mathcal{M}^{i+1}, \mathcal{S}^{i+1}) = Tr(\eta_{i+1})\end{aligned}$$

By induction,

$$\begin{aligned}\max_{u \in [p..i+1]} \{\Omega(q^u)\} &= \max\left\{\max_{t \in [p..i]} \{\Omega(q^t)\}, \Omega(q^{i+1})\right\} \\ &= \max\left\{\max_{u \in [1..\psi(i)]} \{\Omega(\rho^p_u)\}, 0, 0, \Omega(q^{i+1})\right\} \\ &= \max\left\{\max_{u \in [1..\psi(i)]} \{\Omega(\rho^p_u)\}, \Omega(\text{Push}_r, -), \Omega(\text{Claim}_r, -),\right. \\ &\qquad \left.\Omega(\text{Check}, q^{i+1}, -)\right\} \\ &= \max_{u \in [1..\psi(i+1)]} \{\Omega(\rho^p_u)\}\end{aligned}$$

Case 3 ($\tau_i = (q^i, s, r, q^{i+1}) \in \delta_{pop}$)**:** Since p is a stair, position i is in a push-pop-pair (t, i) such that $p \leq t < i$. Set $\psi(i+1) = \psi(t) + 4$. The desired play is a continuation of ρ^t. Because $\eta_i \overset{\tau_i}{\longmapsto} \eta_{i+1}$ is not by transition case 3.2 of T's transition conditions, by Lemma 15, the following transitions are compliant with σ.

$$\begin{aligned} Tr(\eta_t) &= (\text{Check}, q^t, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t) \\ &\mapsto (\text{Push}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s) \\ &\mapsto (\text{Claim}_r, \mathcal{T}^t, \mathcal{M}^t, \mathcal{S}^t, q^{t+1}, s, {}^r S^{t+1}_{\uparrow}) \\ &\mapsto (\text{Jump}_r, q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^{i+1}, \mathcal{M}^i_{[r \mapsto 0]}, \mathcal{S}^{i+1}, \mathcal{M}^t[r]) \\ &\mapsto (\text{Check}, q^{i+1}, \mathcal{T}^{i+1}, \mathcal{M}^{i+1}, \mathcal{S}^{i+1}) = Tr(\eta_{i+1}) \end{aligned}$$

Since (t, i) is push-pop-pair, $\text{lup}^{\eta}_r(i) = t$. By Lemma 14,

$$\mathcal{M}^i[r] = \max_{u \in [\text{lup}^{\eta}_r(i)..i]} \{\Omega(q^u)\} = \max_{u \in [t..i]} \{\Omega(q^u)\}.$$

By induction,

$$\begin{aligned} \max_{u \in [p..i+1]} \{\Omega(q^u)\} &= \max \left\{ \max_{u \in [p..t]} \{\Omega(q^u)\}, \max_{u \in [t..i]} \{\Omega(q^u)\}, \Omega(q^{i+1}) \right\} \\ &= \max \left\{ \max_{u \in [1..\psi(t)]} \{\Omega(\rho^t_u)\}, \mathcal{M}[r]^i, \Omega(\rho^t_{\psi(i+1)}) \right\} \\ &= \max_{u \in [1..\psi(i+1)]} \{\Omega(\rho^{i+1}_u)\} \end{aligned}$$

◀

Since $(\mathbb{N}^n, \leq_n)$ is a well-quasi ordering, η contains an infinite set of stairs $\mathcal{ST}_\eta = \{p_1, p_2, \dots\} \subseteq \mathbb{N}$ with $p_1 < p_2 < \dots$. Towards contradiction, we can now construct a play $\rho = \rho_0 \mapsto \rho_1 \mapsto \dots$ in F that is winning for Ana and is compliant with σ. We need a function $\phi : \mathcal{ST} \to \mathbb{N}$, such that for any $p \in \mathcal{ST}$, $Tr(\eta_p) = \rho_{\phi(p)}$. Furthermore, we want for each $p_i, p_{i+1} \in \mathcal{ST}$ that

$$\max_{u \in [p_i..p_{i+1}]} \{\Omega(\pi_u)\} = \max_{u \in [\phi(p_i)..\phi(p_{i+1})]} \{\Omega(\rho_u)\},$$

which leads to

$$\begin{aligned} \max_{u \in \mathbb{N}} \inf\{\Omega(\pi_u)\} = \max_{i \in \mathbb{N}} \inf\{\Omega(\pi_{p_i}), \max_{p_i < u < p_{l+1}} \Omega(\pi_u)\} = \\ \max_{i \in \mathbb{N}} \inf\{\Omega(\rho_{\phi(p_i)}), \max_{\phi(p_i) < u < \phi(p_{i+1})} \Omega(\rho_u)\} = \max_{u \in \mathbb{N}} \inf\{\Omega(\rho_u)\}. \end{aligned}$$

And thus ρ is a play compliant with σ, that is won by Ana, contradicting σ being a winning strategy for Eve.

▶ **Base Case** (p_1)**.** Initial position of the play is $\rho_0 = (\text{Check}, q_{init}, \varepsilon^n, 0^n, \varnothing^n) = Tr(\eta_0)$, which is a stair. Thus, $p_1 = 0$.

▶ **Inductive Case** ($p_i \to p_{i+1}$)**.** Assume, we constructed ρ and the function ϕ, such that

$$\rho_0 = Tr(\eta_0) \mapsto \dots \mapsto \rho_{\phi(p_1)} = Tr(\eta_1) \mapsto \dots \mapsto \rho_{\phi(p_2)} = Tr(\eta_2) \mapsto \dots \mapsto \rho_{\phi(p_i)} = Tr(\eta_{p_i})$$

and for all $i \in [1..i-1]$,

$$\max_{u \in [p_i..p_{i+1}]} \{\Omega(\pi_u)\} = \max_{u \in [\phi(p_i)..\phi(p_{i+1})]} \{\Omega(\rho_u)\}.$$

Since η_{p_i} is a stair and $Tr(\eta_{p_i}) = \rho_{\phi(p_i)}$, by Lemma 16, we can find a position for $\phi(p_{i+1})$ and continue ρ by some transitions $Tr(\eta_{p_i}) = \rho_{\phi(p_i)} \mapsto \cdots \mapsto \rho_{\phi(p_{i+1})} = Tr(\eta_{p_{i+1}})$, such that

$$\max_{u \in [p_i..p_{i+1}]} \{\Omega(\pi_u)\} = \max_{u \in [\phi(p_i)..\phi(p_{i+1})]} \{\Omega(\rho_u)\}.$$

Now we handle the case where one of the transitions in η is due to transition case 3.2 of the strategy automaton. Let there is a minimal position $i \in \mathbb{N}$, such that $\eta_i \overset{\tau_i}{\longmapsto} \eta_{i+1}$ is due to transition case 3.2 of T's transition conditions. Be aware, that the induction in Lemma 16 still works up to position i. Thus, there is a play ρ compliant with σ from $(\text{Check}, q_{init}, \varepsilon^n, 0^n, \varnothing^n)$, which is a stair, to $\psi(\eta_i)$. Since T had a transition for τ_i, either $own(\psi(\eta_i)) = \text{Ana}$ or σ chose the transition introduced to F caused by τ_i. In either case, the following transition is compliant with σ:

$$\psi(\eta_i) = (\text{Check}, q^i, \mathcal{T}^i, \mathcal{S}^i, \mathcal{M}^i) \mapsto \text{AnaWin}.$$

Because τ_i used transition case 3.2, we know that there is no $(q^{i+1}, \mathcal{M}^i[r], \mathcal{T}^i_{[r \mapsto \varepsilon]}, \mathcal{M}^i_{[r \mapsto 0]}, \mathcal{S}) \in \mathcal{S}[r]$ such that for each stack $j > r$, $\mathcal{S}[j] \subseteq {}^jS_{\uparrow}$.

Thus the above transition is indeed a continuation of ρ, compliant with σ, that is won by Ana, contradicting σ being a winning strategy for Eve.

B.2 Transforming a winning strategy from G to F

We handle a lot of play prefixes in this section. Let us introduce the notation $\pi_{..i} = \pi_0 \pi_1 \dots \pi_i$ for play prefixes of π.

Let ν be a winning strategy for Eve in G. We construct a strategy σ for Eve in F. For this, we need to maintain a play prefix of G. During a play ρ in F, we build up and continue this prefix and use it to determine the moves to be taken by σ in ρ.

▶ **Definition 17.** *Given a strategy ν for Eve in G, a play prefix $\pi_{..l} = \pi_0 \overset{\tau_0}{\longmapsto} \dots \overset{\tau_{l-1}}{\longmapsto} \pi_l$ compliant with ν and an unmatched pushing position $p \in [0..l-1]$ with $\tau_p = (q, r, s, q') \in \delta_{push}$, we define the summary set $S^{\pi_{..l}, \nu, p} \subseteq \mathbb{OS}_r$ recursively:*
For every play $\pi_{..l'}$ that is a continuation of $\pi_{..l}$, i.e. $l < l'$, and compliant with ν, if p is matched in $\pi_{..l'}$, i.e. (p, t) is a push-pop-pair in $\pi_{..l}$, we add a summary as follows to $S^{\pi_{..l}, \nu, p}$:

Let q'' be the state at position $t+1$, and $\mathcal{T}$ be the top of stack symbols at π_{t+1}. For each stack $j \in [1..n]$, let $t_j = \text{lup}^{\pi_{..t}}_j(t)$. Let $\mathcal{M}$ be such that

$$\mathcal{M}[j] = \begin{cases} \max_{u \in [t_j+1..t]} \{\Omega(\pi_u)\} & t_j \neq \bot \\ \max_{u \in [0..t]} \{\Omega(\pi_u)\} & t_j = \bot \end{cases}$$

We add $(q'', \mathcal{M}[r], \mathcal{T}_{[r \mapsto \varepsilon]}, \mathcal{M}_{[r \mapsto 0]}, \mathcal{S})$ to $S^{\pi_{..l}, \nu, p}$, where $\mathcal{S}[j]$ for each stack $j > r$,

$$S[j] = S^{\pi_{..t+1}, \nu, t_j}.$$

Be aware, that this construction is finite and the result is an actual prediction: The sets $S^{\pi_{..t+1}, \nu, t_j}$ contain sets of summaries for only stacks greater, thus the recursion terminates for stack n. Furthermore, this construction is finite as $\mathbb{OS}_j$ is finite.

For a play prefix $\pi_{..l}$ and its continuation $\pi_{..l'}$, i.e. $l < l'$, it is immediate that $S^{\pi_{..l'}, \nu, p} \subseteq S^{\pi_{..l}, \nu, p}$. This is because the set of play continuations for $\pi_{..l'}$ is a subset of the play continuations for $\pi_{..l}$.

For a play ρ in F compliant with σ, we maintain the play prefix of G. In order to keep the construction short, we define the strategy and an invariant (Lemma 18) between the

two plays at the same time. For this, define the set $\text{Checks}^\rho \subseteq \mathbb{N}$ of all indecies where ρ is in a Check-state. We can order these positions by their occurence in ρ so we get $\text{Checks}^\rho = \{p_1, p_2, \dots\}$ with $p_1 < p_2 < \dots$. We define a function $\psi : \text{Checks}^\rho \to \mathbb{N}$ that maps play indecies of Check-states in ρ to positions in π.

▶ **Lemma 18.** *Let ρ be a play compliant with σ and π the corresponding play created.*

- *π is compliant with ν*
- *For any position $p \in \text{Checks}^\rho$, if $\rho_p = (\mathit{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})$, then $\pi_{\psi(p)}$ is in state q and the top of stack symbols are $\mathcal{T}$. Let $t_j = \text{lup}_j^{\pi_{..\psi(p)}}(\psi(p))$. For every stack j with $t_j \neq \bot$, $S^{\pi_{..\psi(p)},\nu,t_j} \subseteq \mathcal{S}[j]$.*
- *for $p_i, p_{i+1} \in \text{Checks}^\rho$,*

$$\max_{u \in [p_i..p_{i+1}]} \{\Omega(\rho_u)\} = \max_{u \in [\psi(p_i)..\psi(p_{i+1})]} \{\Omega(\pi_u)\}$$

Proof and Construction. by induction.

▶ **Base Case** $(i = 1)$**.** This is only the initial position. $\pi_0 = (q_{init}, \varepsilon^n)$, $\rho_0 = (\text{Check}, q_{init}, \varepsilon^n, \Omega(q_{init})^n, \varnothing^n)$. $\psi(p_1) = \psi(0) = 0$.

▶ **Inductive Case** $(i \to i+1)$**.** We first show how σ continues ρ and $\pi_{..\psi(p_i)}$ before focussing on the invariant stated in Lemma 18.

If $\mathit{own}(\rho_{p_i}) = \text{Eve}$, we need to construct σ for $\rho_{p_i} = (\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})$. In that case, let $\pi_{\psi(p_i)} \overset{\tau}{\mapsto} \nu(\pi_{\psi(p_i)})$ be the transition used by ν in G. If $\mathit{own}(\rho_{p_i}) = \text{Ana}$, Ana takes some transition in F that was introduced by some transition τ enabled in $\pi_{\psi(p_i)}$.

Let $\pi_{\psi(p_i)} = (q, \mathcal{P})$, where by indcution, the top of stack symbols form $\mathcal{T}$.

Case 1 $(\tau = (q, r, q') \in \delta_{int})$**:** ρ continues with the transition introduced in F:

$$\rho_{p_i} = (\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S}) \overset{\tau}{\mapsto} (\text{Check}, q', \mathcal{T}, \mathcal{M}', \mathcal{S}) = \rho_{p_{i+1}}$$

Further, we set $\psi(p_{i+1}) = \psi(p_i) + 1$ and continue π by

$$\pi_{\psi(p_i)} = (q, \mathcal{P}) \overset{\tau}{\mapsto} (q', \mathcal{P}) = \pi_{\psi(p_{i+1})}.$$

to arrive at $\pi_{..\psi(p_{i+1})}$.

To the invariant: This tranisition is compliant with ν. The state conditions are fulfilled by construction, as well as the top of stack condition. The prediction sets did not change, thus $S^{\pi_{..\psi(p_{i+1})},\nu,t_j} \subseteq S^{\pi_{..\psi(p_i)},\nu,t_j} \subseteq \mathcal{S}[j]$. And for the parity condition,

$$\max_{u \in [p_i..p_{i+1}]} \{\Omega(\rho_u)\} = \max\{\Omega(\rho_{p_i}), \Omega(\rho_{p_{i+1}})\} = \max\{\Omega(\pi_{\psi(p_i)}), \Omega(\pi_{\psi(p_{i+1})})\} = \max_{u \in [\psi(p_i)..\psi(p_{i+1})]} \{\Omega(\pi_u)\}.$$

Case 2 $(\tau = (q, r, s, q') \in \delta_{push})$**:** ρ continues with the transition introduced in F:

$$\rho_{p_i} = (\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S}) \mapsto (\text{Push}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s)$$

First, we continue $\pi_{..\psi(p_i)}$ by $\pi_{..\psi(p_i)+1} = \pi_{..\psi(p_i)} \overset{\tau}{\mapsto} \pi_{\psi(p_i)+1}$ which is compliant with ν.

Then, Eve has to make a claim in F. To define their strategy, we use the game prediction from above.

$$\sigma(\text{Push}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s) = (\text{Claim}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s, S^{\pi_{..\psi(p_i)+1},\nu,\psi(p_i)})$$

Case 2.1 (Ana continues to $(\text{Check}, q', r, \mathcal{T}_{[r\mapsto s]}, \mathcal{M}', \mathcal{S}_{[r\mapsto S^{\pi_{..\psi(p_i)+1},\nu,\psi(p_i)}]})$)*:* The transitions in F up to p_{i+1} are

$$\begin{aligned}\rho_{p_i} &= (\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})\\ &\mapsto (\text{Push}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s)\\ &\mapsto (\text{Claim}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s, S^{\pi_{..\psi(p_i)+1},\nu,\psi(p_i)})\\ &\mapsto (\text{Check}, q', r, \mathcal{T}_{[r\mapsto s]}, \mathcal{M}', \mathcal{S}_{[r\mapsto S^{\pi_{..\psi(p_i)+1},\nu,\psi(p_i)}]}) = \rho_{p_{i+1}}\end{aligned}$$

Thus, $p_{i+1} = p_i + 3$. Set $\psi(p_{i+1}) = \psi(p_i) + 1$. Then, $\pi_{..\psi(p_{i+1})} = \pi_{..\psi(p_i)+1}$.

To the invariant, state conditions are fulfilled by construction, as well as the top of stack condition. The prediction sets for all stacks $j \neq r$ did not change and $t_j = \text{lup}_j^{\pi_{..\psi(p_i)}}(\psi(p_i)) = \text{lup}_j^{\pi_{..\psi(p_{i+1})}}(\psi(p_{i+1}))$, thus $S^{\pi_{..\psi(p_{i+1})},\nu,t_j} \subseteq S^{\pi_{..\psi(p_i)},\nu,t_j} \subseteq S[j] = \mathcal{S}_{[r\mapsto S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}]}[j]$.

For stack r, we have $\text{lup}_j^{\pi_{..\psi(p_{i+1})}}(\psi(p_{i+1})) = \psi(p_i)$. Adequately, $S^{\pi_{..\psi(p_{i+1})},\nu,\psi(p_i)} = S_{[r\mapsto S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}]}[r]$.

For the parity condition, we have

$$\max_{u\in[p_i..p_{i+1}]}\{\Omega(\rho_u)\} = \max\{\rho_{p_i}, (\text{Push}_r, -), (\text{Claim}_r, -), \rho_{p_{i+1}}\} =$$
$$\max\{\rho_{p_i}, \rho_{p_{i+1}}\} = \max\{\pi_{\psi(p_i)}, \pi_{\psi(p_{i+1})}\} = \max_{u\in[\psi(p_i)..\psi(p_{i+1})]}\{\Omega(\pi_u)\}.$$

Case 2.2 (Ana continues to $(\text{Jump}_r, q'', m, \mathcal{T}'_{[r\mapsto\mathcal{T}[r]]}, \mathcal{M}', \mathcal{S}'_{[r\mapsto S[r]]}, \mathcal{M}[r])$)*:* The transitions in F up to p_{i+1} are

$$\begin{aligned}\rho_{p_i} &= (\text{Check}, q, \mathcal{T}, \mathcal{M}, \mathcal{S})\\ &\mapsto (\text{Push}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s)\\ &\mapsto (\text{Claim}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s, S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)})\\ &\mapsto (\text{Jump}_r, q'', m, \mathcal{T}'_{[r\mapsto\mathcal{T}[r]]}, \mathcal{M}', \mathcal{S}'_{[r\mapsto S[r]]}, \mathcal{M}[r])\\ &\mapsto (\text{Check}, q'', \mathcal{T}'_{[r\mapsto\mathcal{T}[r]]}, \mathcal{M}'', \mathcal{S}'_{[r\mapsto S[r]]}) = \rho_{p_{i+1}}\end{aligned}$$

We set $p_{i+1} = p_i + 4$. Further, it must be that $(q'', m, \mathcal{T}', \mathcal{M}', \mathcal{S}') \in S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$ in order for

$$(\text{Claim}_r, \mathcal{T}, \mathcal{M}, \mathcal{S}, q', s, S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}) \mapsto (\text{Jump}_r, q'', m, \mathcal{T}'_{[r\mapsto\mathcal{T}[r]]}, \mathcal{M}', \mathcal{S}'_{[r\mapsto S[r]]}, \mathcal{M}[r])$$

to exist. By construction of $S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, there is a play continuation $\pi_{..l}$ of $\pi_{..\psi(p_i)}$ compliant with ν, such that $\psi(p_i)$ is in a push-pop-pair $(\psi(p_i), t)$ with $\psi(p_i) < t < l$.

Finally, we continue $\pi_{..\psi(p_i)}$ to $\pi_{..t+1}$ and set $\psi(p_{i+1}) = t + 1$.

To the invariant: By construction of $S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, this is compliant with ν.

By construction of $S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, π_{t+1} is in state q'' with the top of stack symbols being $\mathcal{T}'_{[r\mapsto\gamma[r]]}$.

Furthermore, for each stack $j > r$, since $(q'', m, \mathcal{T}', \mathcal{M}', \mathcal{S}') \in S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, with $t_j = \text{lup}_j^{\pi_{..t}}(t)$,

$$\mathcal{S}'_{[r\mapsto\mathcal{S}[r]]}[j] = \mathcal{S}'[j] = \begin{cases} S^{\pi_{..t},\nu,t_j} & t_j \neq \bot \\ \varnothing & t_j = \bot \end{cases}.$$

For stack r, we know that since $(\psi(p_i), t)$ is a push-pop-pair, that

$$t_r = \text{lup}_r^{\pi_{..\psi(p_i)}}(\psi(p_i)) = \text{lup}_r^{\pi_{..t+1}}(t+1) = \text{lup}_r^{\pi_{..\psi(p_{i+1})}}(\psi(p_{i+1})).$$

Due to $\pi_{..\psi(p_{i+1})}$ being a continuation of $\pi_{..\psi(p_i)}$, we arrive at

$$S^{\pi_{..\psi(p_{i+1})},\nu,t_r} \subseteq S^{\pi_{..\psi(p_i)},\nu,t_r} \subseteq \mathcal{S}[r] = \mathcal{S}'_{[r\mapsto\mathcal{S}[r]]}[r].$$

For the parity condition, be aware, that by construction of $S^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$,

$$m = \max_{u\in[\psi(p_i)+1..t]}\{\Omega(\pi_u)\}.$$

Together, we arrive at:

$$\begin{aligned}\max_{u\in[p_i..p_{i+1}]}\{\Omega(\rho_u)\} &= \max\{\rho_{p_i},\rho_{p_{i+1}},(\mathrm{Push}_r,-),(\mathrm{Claim}_r,-),\\ &\qquad (\mathrm{Jump}_r,q'',m,\mathcal{T}'_{[r\mapsto\mathcal{T}[r]]},\mathcal{M}',\mathcal{S}'_{[r\mapsto S[r]]},\mathcal{M}[r])\}\\ &= \max\{\rho_{p_i},\rho_{p_{i+1}},m\} = \max_{u\in[\psi(p_i)..\psi(p_{i+1})]}\{\Omega(\pi_u)\}.\end{aligned}$$

Case 3 ($\tau = (q,s,r,q') \in \delta_{pop}$)**:** ρ continues with the transition introduced for τ. This is either

$$\rho_{p_i} = (\mathrm{Check},q,\mathcal{T},\mathcal{M},\mathcal{S}) \mapsto \mathrm{EveWin} \quad \text{or}$$
$$\rho_{p_i} = (\mathrm{Check},q,\mathcal{T},\mathcal{M},\mathcal{S}) \mapsto \mathrm{AnaWin}.$$

We show, that the second case is impossible. Since τ is enabled in $\pi_{\psi(p_i)}$, we can continue $\pi_{..\psi(p_i)}$ by $\pi_{\psi(p_i)} \overset{\tau}{\mapsto} \pi_{\psi(p_i)+1}$. Due to the enabledness of a pop-transition, there is $t_r = \mathrm{lup}_r^{\pi_{..\psi(p_i)}}(\psi(p_i))$ and by definition of $S^{\pi_{..\psi(p_i)},\nu,t_r}$, there is a summary $(q',\mathcal{M}[r],\mathcal{T}_{[r\mapsto\varepsilon]},\mathcal{M}_{[r\mapsto 0]},\mathcal{S}') \in S^{\pi_{..\psi(p_i)},\nu,t_r} \subseteq \mathcal{S}[r]$, such that for all stacks $j > r$:

$$\mathcal{S}'[j] = S^{\pi_{..\psi(p_i)+1},\nu,t_j} \subseteq \mathcal{S}[j].$$

Thus, by construction of F, the second transition does not exist.

Now we can show, that ρ is winning for Eve:

Case 1 (ρ contains EveWin)**:** This play is winning for Eve.

Case 2 (ρ contains AnaWin)**:** We have just shown, that this can not happen.

Case 3 (ρ contains infinitly many Check states)**:** In this case, Checks^ρ is infinite and

$$\begin{aligned}\max_{u\in\mathbb{N}}\{\Omega(\rho_u)\} &= \max_{i\in\mathbb{N}}\left\{\max_{u\in[p_i..p_{i+1}]}\{\Omega(\rho_u)\}\right\}\\ &= \max_{i\in\mathbb{N}}\left\{\max_{u\in[\psi(p_i)..\psi(p_{i+1})]}\{\Omega(\pi_u)\}\right\} = \max_{u\in\mathbb{N}}\{\Omega(\pi_u)\},\end{aligned}$$

which is winning for Eve, since π is compliant with ν, which is a winning strategy. ◀

C Stack Elimination for Context-Bounded MPDG

For k-context bounded MPDG can only visit up to k stacks in a play, we can eliminate stacks to obtain a k-context k-stack MPDG.

▶ **Lemma 19.** *For every k-context-bounded n-stack MPDG $G = (P, own, \Omega)$ with MPDS $P = (Q,\Gamma,\delta,n)$, there is k-context-bounded k-stack MPDG $G' = (P', own', \Omega')$ with $P' = (Q',\Gamma,\delta',k)$ such that Eve wins G if and only if she wins G'. The set of priorities coincide and G' is constructible in time $\mathcal{O}(|G|\cdot n^{k+1})$.*

First, present the construction of P': The new state space is $Q' = [1..n]^k \times [0..k] \times [1..n] \times [1..k] \times Q$. A configuration of P' is thus $((f, k, d, e, q), \mathcal{R})$, where the task of the different parameters is as follows. f is an injective mapping from the available stacks $[1..k]$ to the used stacks of P. The inverse function is f^{-1}. k tracks the current context. d tracks the stack of the current context. e tracks the number of different stacks used so far. $\mathcal{R} : [1..k] \to \Gamma^*$ are the stack contents.

For each transition $\tau \in \delta_{d'}$ with $(q, \mathcal{P}) \xrightarrow{\tau} (q', \mathcal{P}')$, P' has transitions

$$((f, k, d, e, q), \mathcal{R}) \xrightarrow{\tau} ((f', k', d', e', q), \mathcal{R}_{[(f')^{-1}(d') \mapsto \mathcal{P}'[d']]}),$$

where either

- $d' = d$ and $f' = f$, $k' = k$, $e' = e$ or
- $d \neq d'$ and $k + 1 \leq k$ and $k' = k + 1$ and $f^{-1}(d') \neq \bot$ and $e' = e$ and $f' = f$ or
- $d \neq d'$ and $k + 1 \leq k$ and $k' = k + 1$ and $f^{-1}(d') = \bot$ and $e' = e + 1$ and $f' = f_{[e' \mapsto d']}$.

Let $\pi = \pi_0 \to \cdots \to \pi_l$ be a play prefix of G. We define the function g_π, which takes a position p of the run π and transforms it to stack contents $\mathcal{R}^p$ for P'. It takes the stack contents of $\pi_p = (q^p, \mathcal{P}^p)$ and reduces them to the stacks to which a transition belonged in $\pi_0 \to \ldots \to \pi_p$, then reorders them, so that they are in the order in which the stacks were visited with their first respective context. Further, let f^p be the corresponding stack assigning function, e^p the number of stacks visited so far, k^p the context, and d^p the stack of that context at position p. Thus, for all already visited stacks d, $\mathcal{P}^p[d] = \mathcal{R}^p[f(d)]$ Define the function $h(\pi) = ((f^0, k^0, d^0, e^0, q^0), g(0)) \to \ldots \to ((f^l, k^l, d^l, e^l, q^l), g(l))$

Vice versa, let $\pi' = \pi'_0 \to \cdots \to \pi'_l$ be a play prefix of G'. We create the function $h'(\pi') = (q^0, \mathcal{P}^0) \to \ldots \to (q^l, \mathcal{P}^l)$, where for every position p and stack j,

$$\mathcal{P}^p[j] = \begin{cases} \varepsilon & (f^p)^{-1}(j) = \bot \\ \mathcal{R}^p[(f^p)^{-1}(j)] & \text{otherwise}. \end{cases}$$

▶ **Lemma 20.** *h and h' form a bijection on the play prefixes starting with empty stack contents, i.e. $h'(h(\pi)) = \pi$ and $h(h'(\pi')) = \pi'$ for all play prefixes π and π' starting with empty stacks.*

Proof. By induction on the length l of the plays.

Base Case. At π_0, no stacks were visited. Thus, f is undefined, no context has been visited and there is no active stack, and no stacks have been used. Rerversely, at π'_0, f is undefined for every stack. Thus, $h'(h(\pi)) = \pi = \pi_0$ and $h(h'(\pi')) = \pi' = \pi'_0$.

Ind. Case. Let $\pi = \pi_0 \to \ldots \to \pi_l \xrightarrow{\tau} \pi_{l+}$ with $h'(h(\pi_0 \ldots \pi_l)) = \pi_0 \ldots \pi_l$ with $h(\pi_0 \ldots \pi_l) = \pi'_0 \to \ldots \to \pi'_l$. Then, $\pi'_l = ((f^l, k^l, d^l, e^l, q^l), \mathcal{R}^l)$, where $\mathcal{R}^l = g_\pi(l)$.

We have $h(\pi) = h(\pi_0 \ldots \pi_l) \to \pi'_{l+1}$, where $\pi'_{l+1} = ((f^{l+1}, k^{l+1}, d^{l+1}, e^{l+1}, q^{l+1}), \mathcal{R}^{l+1})$ and $\mathcal{R}^{l+1} = g_\pi(l)$.

Case 1. $\tau \in \delta_{d^p}$. Then, there is the transition to $((f', k', d', e', q), \mathcal{R}_{[(f')^{-1}(d') \mapsto \mathcal{P}'[d']]})$, where $d' = d^p$ and $f' = f^p$, $k' = k^p$, $e' = e^p$. Since the stack did not change, these coincide with d^{p+1}, f^{p+1}, k^{p+1}, and e^{p+1}. The stack contents did also change for the stack representing stack d^p by $\mathcal{R}_{[(f^p)^{-1}(d^p) \mapsto \mathcal{P}'[d^p]]}$. Thus, $h(\pi)$ is a play prefix in G'.

Case 2. $\tau \in \delta_{d^{p+1}}$ and $d^{p+1} \neq d^p$ and $(f^p)^{-1}(d^{p+1}) = \bot$. To be k-bounded, k^p must be less than k. Then, there is the transition to $((f', k', d', e', q), \mathcal{R}_{[(f')^{-1}(d') \mapsto \mathcal{P}'[d']]})$, where $e' = e^p + 1$ and $f' = f^p_{[e' \mapsto d']}$. Since the stack was not seen before (by induction, it was not found in f^p)), $f' = f^{p+1}$, further, $k^{p+1} = k' = k^{p+1}$, the next context is introduced and the

active stack is $d^{p+1} = d'$. The stack contents did also change for the stack representing stack d^p by $\mathcal{R}_{[(f^{p+1})^{-1}(d^{p+1}) \mapsto \mathcal{P}'[d^{p+1}]]}$. Thus, $h(\pi)$ is a play prefix in G'.

Case 3. $\tau \in \delta_{d^{p+1}}$ and $d^{p+1} \neq d^p$ and $(f^p)^{-1}(d^{p+1}) \neq \bot$. To be k-bounded, k^p must be less than k. Then, there is the transition to $((f', k', d', e', q), \mathcal{R}_{[(f')^{-1}(d') \mapsto \mathcal{P}'[d']]})$, where $e' = e^p$ and $f' = f$. Since the stack was seen before (by induction, it was found in f^p)), $f' = f^{p+1}$, further, $k^{p+1} = k' = k^{p+1}$, the next context is introduced and the active stack is $d^{p+1} = d'$. The stack contents did also change for the stack representing stack d^p by $\mathcal{R}_{[(f^{p+1})^{-1}(d^{p+1}) \mapsto \mathcal{P}'[d^{p+1}]]}$. Thus, $h(\pi)$ is a play prefix in G'.

Further, by induction $h'(h(\pi))) = \pi_0 \dots \pi_l \to (q^{l+1}, \mathcal{P})$, where for every stack j,

$$\mathcal{P}[j] = \begin{cases} \mathcal{R}^{l+1}[(f^p)^{-1}(j)] = \mathcal{P}^{l+1}[j] & (f^p)^{-1}(j) \neq \bot \\ \varepsilon = \mathcal{P}^{l+1}[j] & (f^p)^{-1}(j) = \bot \end{cases}$$

Thus, $h'(h(\pi)) = \pi$.

Showing $h(h'(\pi')) = \pi'$ is analogue. ◀

Proof of Lemma 19. Together with the ownership assignment of $own'((f, k, d, e, q), \mathcal{R}) = own(q)$ and priority assignment $\Omega'((f, k, d, e, q), \mathcal{R}) = \Omega(q)$, the bijection of play prefixes immediatly presents a portation of strategies for both players.

Starting positions with non-empty stacks need to be encoded into the MPDS P first. This can be done by encoding them into the state space in the following sense: When a context would first be initiated on a stack, it first pushes their stack content (The player doing this is unimportant). When the first transition on that stack would be a pop, the player chosing the pop transition will lose, if after the pushes, the symbol to be popped is not on top. ◀

D Construction of MPDG for the Lower Bound

Formally, we introduce 3 gadgets to prove correctness. Each represents a verification mechanism.

- G^d_{comp}, which checks, whether on top of two stacks is the same encoded word,
- G_{ind_d}, which checks, whether the top of a stack is a valid encoding, and
- G^d_φ, which checks the two topmost encoded words for the relation $\sim_\varphi$.

We construct them by induction. Notably, the latter two are constructed by simultaneous induction: G^d_φ needs $G_{ind_{d-1}}$ and G^{d-1}_φ internally. The latter mechanism is described in section 5.4.

D.1 Construction of G^d_{comp}

This gadget expects the top of both stacks to be d-nested indexings w_1, w_2 of words u, v of length $exp_d(len)$. Eve has a winning strategy, if they index the same word.

To be precise, it is also sufficient if they are not on top, but marked by a delimiter Symbol. In our construction we need the latter case for the verification mechanism G^d_φ, where Ana wants to verify the position (of a variable after doubting the symbol is correct) and the valuation is still complete on the second stack. Remembering the correct variable in the control state is no problem as there are a constant amount of variables. For presentational reasons, this differs a little from our detailed version [60].

▶ **Lemma 21.** *There is a 2-stack MPDG such that Eve has a winning strategy from positions* $(CheckEq_d, w_1\sigma_1\gamma_1\bot, w_2\sigma_2\gamma_2\bot)$ *if and only if* $u = v$. *It is contructible in time* $\text{poly}(d + |\Sigma| + n)$. *The maximal number of contexts or phases of any play from such a position is at most* $d + 2$.

Proof idea. At the top of the stacks are the d-nested indexings w_1 and w_2, where $w_1 = ind_d(u)$ and $w_2 = ind_d(v)$ for some $u, v \in \Sigma^{max_d}$. They have form

$$w_1 = u_0 x_0 \dots u_{max_d} x_{max_d}, \quad w_2 = v_0 x_0 \dots v_{max_d} x_{max_d}.$$

Intuitively, Ana removes a sequence of $(\Sigma\Sigma^*_{\leq d})^*$ from stack one, until she claims to have found a position p, where $u_p \neq v_p$. She leaves x_p on top of stack one and store the symbol u_p in the control state. Then, Eve has to pop a sequence of $(\Sigma\Sigma^*_{\leq d})^*$ from stack two. They are supposed to find the corresponding position in v. Removing the sequence leaves $v_{p'} x_{p'}$ on top of stack two for some position p'. After storing $v_{p'}$ in the control state, Ana may now choose to

1. Believe Eve's choice to be $p' = p$. Then, if $u_p = v_{p'}$, Eve wins and vice versa.
2. Doubt Eve's choice and claim $p \neq p$. Since w_1, w_2 are d-nested indexings, $x_p = ind_{d-1}(msbf(p))$ and $x_{p'} = ind_{d-1}(msbf(p'))$. Checking $p \neq p$ can thus be done by $(d-1)$-Equality(Σ_d).

Number of Contexts and Phases. The first context or phase starts by Ana removing symbols from stack one. The second context or phase is then started by Eve popping from stack two to find the corresponding position. Then, we can use a copy of G^{d-1}_{comp}, where the stacks are swapped, so that it first pops from stack two. This way, the first context or phase of that game merges with the second context or phase. By induction from Lemma 21, this results in a bound of $1 + (d-1) + 2 = d + 2$ contexts or phases. ◀

D.2 Construction of G_{ind_d}

This gadget checks the top of the first stack for whether it is a valid d-nested indexing.

▶ **Lemma 22.** *There is a 2-stack MPDG G_{ind_d} such that Eve has a winning strategy from an initial position (Check $ind_d, w\sigma\gamma_1\bot, \gamma_2\bot$) if and only if $w = ind_d(u)$, where u is any word. It is contructible in time* $\mathrm{poly}(d + |\Sigma| + n)$. *Any play has at most $d+2$ contexts and $d+1$ phases.*

From the initial position, with $w \in (\Sigma \cup \Sigma_{\leq d})^*$, the goal is to check whether w is a valid d-nested indexing. This holds if and only if the following three conditions are met. (1) The word has the shape $w = u_0 x_0 \dots u_m x_m \in (\Sigma\Sigma^*_{\leq d})^+$. (2) Each x_p is a valid $(d-1)$-nested indexing. (3) We have

$$x_0 = ind_{d-1}(msbf_d(0)) = ind_{d-1}(0_d^{exp_{d-1}(len)}),$$
$$x_m = ind_{d-1}(msbf_d(max_d)) = ind_{d-1}(1_d^{exp_{d-1}(len)}),$$

and for all positions $1 \leq p < m$ with indexing $x_p = ind_{d-1}(msbf_d(i))$ and indexing $x_{p+1} = ind_{d-1}(msbf_d(i'))$ we have $i' = i + 1$.

We let Ana choose which condition is violated. In the first case, Eve has to prove that w is of the form $(\Sigma\Sigma^*_{\leq d})^+$. This can be done by a popping loop.

In the second case, Ana identifies a position p by removing a sequence from $\Sigma(\Sigma^*_{\leq d}\Sigma)^*$ and leaving x_p on top of the stack. We use $G_{ind_{d-1}}$ from the induction hypothesis to check whether x_p is a $(d-1)$-nested indexing.

In the last case, there are first-order formulas φ_0 and φ_1 for the constant conditions and a formula φ_{+1} for the successor relation under most-significant-bit-first encodings. With the induction hypothesis for Lemma 10, we construct the corresponding games $G^{k-1}_{\varphi_0/\varphi_1/\varphi_{+1}}$. For checking relation φ_1, before invoking the game, Eve has the task of removing symbols until x_m is on top of the stack. For checking relation φ_{+1}, Ana first pops symbols to find a position where $x_p = ind_{d-1}(msbf_d(i))$ and $x_{p+1} = ind_{d-1}(msbf_d(i'))$, but $i + 1 \neq i'$.

Number of Contexts and Phases. In either case the play starts by popping, leading to a first context or phase on stack one. Actually, in the first case the play already ends after having popped stack one.

In the second case, the play continues to invoke the game $G_{ind_{d-1}}$. This adds $(d-1)+2$ contexts or $(d-1)+1$ phases respectively, by the induction hypothesis for Lemma 22. However, the first context or phase in $G_{ind_{d-1}}$ also acts on stack one. So it merges with the previous context or phase for popping from stack one, leading to $d+1$ contexts or d phases.

In the last case, the play enters the game G_{φ}^{d-1} for φ_0, φ_1, or φ_{+1}, leading to $(d-1)+2$ contexts or $d-1$ phases respectively, by induction from Lemma 10. With the initial context or phase, we arrive at $d+2$ contexts and d phases

Together, this is at most $d+2$ contexts and d phases. This covers the required $d+1$ phases in the Lemma. The base case requires the additional phase.

D.3 Details on how the players push a valuation

Intuitively, we want to reuse the same principles for pushing successing configuration to push a correctly indexed valuation for variable y. Eve pushes any sequence and afterwards, Ana can verify that this is indeed a $(d-1)$-nested indexing by the use of $G_{ind_{d-1}}$. However, when Ana has to choose the valuation, we can not check that Eve pushed the correct position (that is a $exp_{d-1}(n)$ long sequence from $\{0_d, 1_d\}$ arbitrarily chosen by Ana). We also cannot swap the roles: Whenever Ana gets the chance of pushing arbitrary long sequences, she can just push symbols infinitely and win the safety winning condition. Thus, we need to let Eve determine when to stop pushing symbols. We do so by letting Eve push sequences in between Ana's choices for single digits of the position. Also, Eve may choose to end the pushing of the sequence at any time. Afterwards, Ana may choose to check, whether the result is a $(d-1)$-nested indexing.

D.4 Adaptions for ordered multi-pushdown systems

It is possible to adapt the lowerbound construction, so that it provides the same strategies for ordered pushdowns. The key idea is to use d stacks to simulate the d phases in the lowerbound construction. To this end, we need instances for the gadgets created in the previous sections not only for two stacks, but for combinations of stacks j, r with $j < r$. Further, it should be noted that the gadget G^d_{comp} can not be used as is, since it pops symbols from both stacks alternatingly, which cannot be done with an ordered pushdown. Instead, it will need some intermediate steps, which will copy the contents to be compared to the another stack (higher in order). Further adaptions are of minor importance and will be mentioned later for completeness.

To this end, we will adapt the gadgets and recieve for each stack $j < r$,

- $G^d_{comp}(j, r)$, comparing the top of stacks j and r,
- $G^d_{copy}(j, r)$, copying the top of stack indexing from stack j to r,
- $G_{ind_d}(j)$, checking the top of stack j for a d-indexing and
- $G^d_{\varphi}(j)$, checking the $\sim_\varphi$ relation on the marked indexings on stack j.

In this, $G^d_{copy}(j, r)$ will internally use $G^d_{comp}(j, r)$, which, in turn, uses $G^{d-1}_{copy}(j, r)$. Again, we create the gadget in simultaneous induction together with $G_{ind_d}(j, r)$.

The adaptions for $G_{ind_d}(j)$ and $G^d_{\varphi}(j)$ are rather small: $G_{ind_d}(j)$ only needs the stack of the transitions to be changed to j. $G^d_{\varphi}(j)$ also needs the stacks to be changed; stack one becomes j and stack two becomes $j+1$. Further, when doubting the suggested variable order, stack j needs to be emptied, before $G^{d-1}_{\varphi}(j+1)$ can be called.

After these adaptions, the construction of the multi-pushdown game simulating an alternating Turing machine is the same as in Section 5. The only difference is, that the ordered pushdown-system created posseses d stacks.

The following lemma states the same as Lemma 21, but for $G^d_{comp}(j, r)$. Let $u, u' \in \Sigma^{l_d+1}$, $w_1 = ind_d(u)$, $w_2 = ind_d(u')$ and $j < r \leq n - d$. Note that the latter requires this gadget to have at least $d + 2$ stacks.

▶ **Lemma 23.** *There is an n-stack ordered MPDG such that Eve has a winning strategy from positions* $(\text{start}, [w_1\gamma_1]_j, [w_2\gamma_1]_r)$ *if and only if $u = v$. It is contructible in time* $\text{poly}(d + |\Sigma| + n)$.

▶ **Base Case** ($d = 0$)**.** The gadget is almost the same as G^0_{comp}, where transition rules for stack one (2) are swapped for transition rules for stack j (r). The stacks j to $r - 1$ are emptied before the transitions popping from r are executed. Correctness is follows as for G^0_{comp}.

▶ **Inductive Case** ($d > 0$)**.** Instantiate $G^{d-1}_{copy}(j, r + 1)$ on $(\text{copyPos}, s)$ with out state $(\text{copyDone}, s)$ for each $s \in \Sigma$.

Instantiate $G^{d-1}_{comp}(r, r + 1)$ on state disbelievePos.

Let s, s' range over Σ.

$(start, (\Sigma\Sigma^*_{\leq d})^* s, j, (\text{copyPos}, s))$

$((\text{copyDone}, s), j, (\text{reproducePos}, s))$

$((\text{reproducePos}, s), (\Sigma\Sigma^*_{\leq d})^* s', r, (\text{claimPos}, s, s'))$

$((\text{claimPos}, s, s'), r, (\text{believe}, s, s'))$

$((\text{claimPos}, s, s'), r, \text{disbelievePos})$

$((\text{believe}, s, s), r, \text{EveWin})$

$((\text{believe}, s, s'), r, \text{AnaWin})$ $\qquad s \neq s'$

Note that the induction hypothesis (Lemma 24) for the copy gadget holds: $r + 1 \leq n - d + 1$. Given by induction (Lemma 24) that each player possesses a strategy from $((\text{copyPos}, s), [xs\gamma]_j, [x's'\gamma]_r, [\varepsilon]_{r+1})$ to $((\text{copyDone}, s), [\varepsilon]_j, [x's'\gamma]_r, [xs\gamma]_{r+1})$, the proof is analogue to the proof for Lemma 21.

And for copying.

▶ **Lemma 24.** *Let $u \in \Sigma^{l_d+1}$, $w_1 = ind_d(u)$ and $j \leq n - d$. Each player possesses a strategy from* $(\text{start}, [w_1\sigma_1\gamma_1]_j, [\varepsilon]_r)$ *to* $(\text{Out}, [\varepsilon]_j, [w_1]_r)$.
The number of states of this gadget (not counting the states of additionally instantiated gadgets) is polynomially in the size of Σ and l_0.

The construction is pretty straightforward: Eve guesses the stackcontent for stack r and Ana may doubt or believe it.

Instantiate $G_{ind_d}(r)$ on position disbelieveValidity.

Instantiate $G^d_{comp}(j, r)$ on position disbelieveEquality.

$(start, (\Sigma\Sigma^*_{\leq d})^*, r, \text{pushed})$

$(\text{pushed}, r, \text{disbelieveValidity})$

$(\text{pushed}, r, \text{disbelieveEquality})$

$(\text{pushed}, j, (\Sigma \cup \Sigma_{\leq d})^*, Out)$

Be aware, that the definition of MPDS does not allow for testing a stack for ε. One can, however, implement such a transition rule for games given the allowed transition rules.

Now to show that each player possesses a strategy from position $(start, [w_1\sigma_1\gamma_1]_j, [\varepsilon]_r)$ to $(start, [\varepsilon]_j, [w_1]_r)$.

Be aware, that $r \leq n - d$ holds. The following assumes that Lemma 23 and 22 already hold for the current induction step, which has been shown already.

The strategy for Eve pushes w_1 on stack r in the first transition. If Ana chooses to go to disbelieveValidity or disbelieveEquality, Eve wins (Lemma 22 and Lemma 23).

The strategy for Ana analyzes the pushed sequence w from Eve. If it is not a d-indexing, Ana wins the play using the move to disbelieveValidity (Lemma 22). If it is a valid d-indexing, but $w \neq w_1$, i.e. $w = ind_d(u')$ with $u' \neq u$, Ana wins the play using the move to disbelieveEquality (Lemma 23).