Bounded Analysis of Concurrent and Recursive Programs

Sören van der Wall
Institute of Theoretical Computer Science
Technische Universität Braunschweig
Supervised by Prof. Dr. Roland Meyer

October 30, 2019

Abstract Solving games on models of computation is a typical approach in program synthesis. Concurrent, recursive programs can be modelled by multi-pushdown systems. Previous work has shown an upper bound of k-EXPTIME for k bounded phase multi-pushdown parity games [13]. We present a k-EXPTIME lower bound on the complexity of reachability games on $k+2$ bounded context switching and k bounded phase multi-pushdown systems and a k-EXPTIME upper bound on parity games over $k+2$ bounded context switching multi-pushdown systems.

Contents

1 Introduction and related work

Programs are omnipresent in every sector of today's world, ranging from entertainment systems, communication and science to defense, health care and transportation. Dependent on project size and application, software is developed in a vast variety of fashions, often including globally distributed teams of developers. Up until today, programs are handwritten and, by human nature, they regularly contain bugs. The critical application fields dramatically increase the need for programs that are known to be correct.

Even though formal proof systems with the capability of showing program correctness exist, they are rarely used in practice [8–10]. This is mainly due to additional complexity during the development process, which is already consumed by a rapidly growing demand for software solutions.

Today's most common approach to bug reduction in a program is testing. Testing can be very dynamic and finds bugs in any component of a development process, but it is "hopelessly inadequate for showing their absence" [7]. This is particularly true for concurrent programs where bugs are hard to reproduce. By their nature, the interaction of threads in a concurrent program increases the number of possible computations drastically.

This motivates a shift towards automated procedures ensuring program correctness. In theoretical computer science, showing correctness of a program belongs to the field of verification. The perspectives for formal verification of program properties seemed to be very limited after the results of Turing's halting problem and Rice's theorem [12, 15]. They conclude that even simple properties like termination of programs are already undecidable. In principle, for any looping programs that have two counters and the ability to test them for being zero, most properties are undecidable. If the counters are bound however, the state space of the program is finite and those properties become decidable. However, complexity theory states most verification problems as computationally hard.

One part of formal verification is Model checking. Model checking introduces an automatic way to use logical specifications for bug finding, making it applicable for practical use. The basic concept of model checking is simple. The paradigm consists of the following parts:

- Modelling. The program is abstracted to a transition system which is an adequate model for finite state systems like programs. The specific type of model depends on the type of program.
- Specification. A language to formalize conditions on finite state systems is usually utilized in the form of logics. Common examples are temporal logics

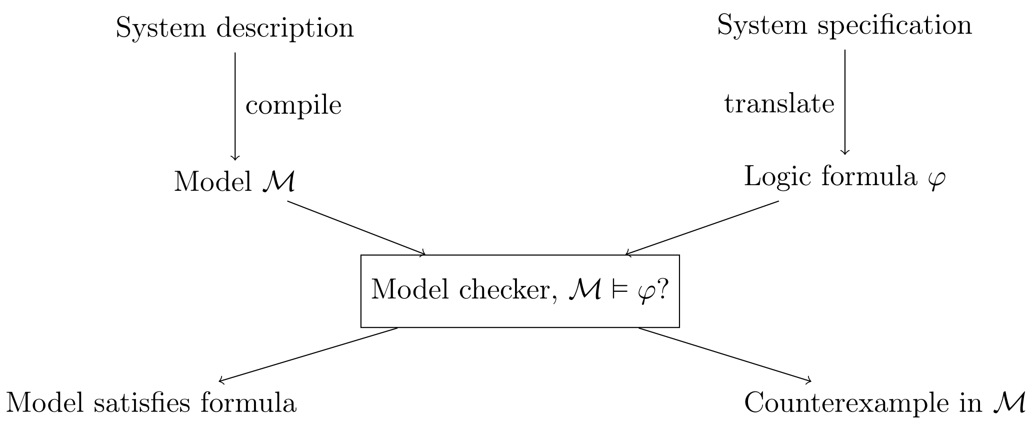


Figure 1.1: Model checking

like LTL or BTL, but the logic is usually tied to the expressiveness of the modelled system.

- Algorithms. These form the backbone of model checking as they are procedures for solving whether the model satisfies the specification.

The typical methodology for model checking involves each of the parts, its general concept is displayed in Figure 1.1. A system description (program) is compiled into the model $\mathcal{M}$ of the system, some kind of transition system, and the system specification is translated into a formula φ of the logic. Both are serve as input to the model checker, where a decision procedure concludes whether the model satisfies the formula ($\mathcal{M} \models \varphi$) or not ($\mathcal{M} \not\models \varphi$). If $\mathcal{M} \not\models \varphi$, the model checker usually also yields a counterexample, witnessing the violation of the formula in the model.

Work in the branch of model checking focuses on two major parts. Finding new algorithms for existing models and specifications and showing their computational hardness or creating new models and specifications in order to model new kinds of programs or hardware.

The introduction of model checking offers some advantages over other methods known. Firstly, the algorithmic nature of model checking. As such, model checking can be automated and meets the goal to be integrated into the development process of programs. Secondly, model checking can be used on very different systems, from abstract modeling languages to hardware implementations. While each needs its own model and specification language, the methodology stays the same across all applications of model checking and knowledge can be spread across the fields. At last, it can be used for checking concurrent programs, for which testing is not a sufficient option due to the interaction of parallel processes. As an automated tool designed to explore large state spaces, model checking is applicable for showing correctness of concurrent programs.

Moving beyond the verification part of theoretical computer science, newer research has explored the problem of synthesis. While verification focuses on showing the correctness of a program, synthesis is about constructing a system that acts compliant with a specification. The motivation for synthesis is relief for the programmer from implementation details, increasing their focus on conceptual design. At the same time, synthesis entails a formal correctness proof for each generated program.

A typical setting for synthesis are "reactive systems". Their specification consists of input and output parameters and they are generated such that the correct outputs for any inputs are produced. The synthesis problem motivates an extension to the model checking methodology by the use of games.

A typical model checking algorithm for a model $\mathcal{M}$ and a formula φ creates an automaton $A_{\neg\varphi}$ for the negated formula and builds the product model $\mathcal{M} \times A_{\neg\varphi}$ and runs an language emptiness test on it. If there is no accepting path in it, $\mathcal{M}$ satisfies φ. If there is an accepting path, it is a witness for a faulty behaviour. One could interpret these actions as a question "is there any input sequence such that $\mathcal{M}$ enters a faulty state?". Naturally, the product automaton is a non-deterministic automaton where the environment of the program chooses the input parameters.

In the synthesis environment, this question needs to be adapted to "is there a model $\mathcal{M}$ that reacts to every input with the correct output, such that no input sequence brings $\mathcal{M}$ to a faulty state?". Implicitly, this question brings a second kind of non-determinism into the synthesis problem. The first type of non-determinism is the environment, deciding the inputs as in typical model checking, the second one is the synthesizer, deciding the outputs with the ability to react to the input. Formally, synthesis can be seen as a two player perfect information game and the task of finding a model $\mathcal{M}$ corresponds to finding a winning strategy in such a such game.

The concept of synthesis is displayed in Figure 1.2. A specification is translated into a formula φ, which is further translated into an automaton A. The automaton is then modified into a game G. Intuitivly, we split each transition of A into two halfs: the input transitions, which belong to the environment, and the output transitions, which belong to the synthesizer. If there is a winning strategy for the synthesizer, it outputs a model $\mathcal{M}$ that satisfies φ. Otherwise, there is no model for the specification. Synthesis based on this paradigm shares the positive properties of model checking. It can be automated and integrated into the development process. However, the computational complexity of solving games on a model is usually a lot harder than the verification problem.

In this work, we focus on a model for concurrent, recursive programs: Multi-pushdown systems. Pushdowns can be used to model call stacks of recursive programs, while a finite state space abstracts the set of (bounded) global variables. In a concurrent world of recursive programs, each having their own call stack, the adequate model also needs multiple pushdowns. As discussed earlier, only two pushdowns and a common finite state space are sufficient for most properties to become undecidable. Various under-approximations have been suggested in order to arrive at a decidable model, including bounded context switching and bounded phase com-

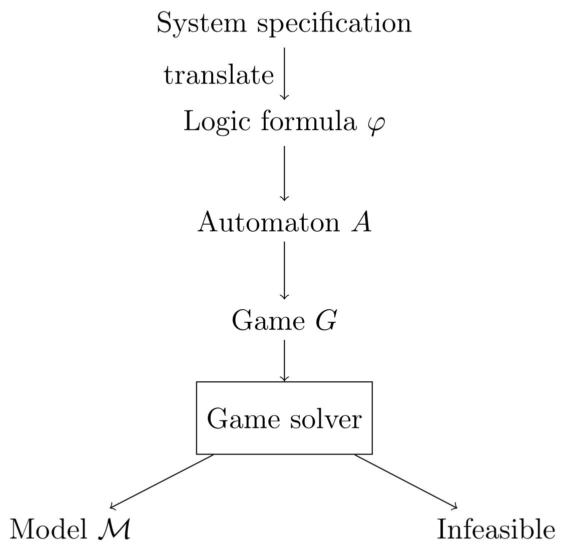


Figure 1.2: Program synthesis

putations [11, 14]. A context is part of a computation where only one pushdown is active, i.e. the system uses transitions acting on that stack only for the duration of the context. Similarly, a phase is part of a computation where only one pushdown uses pop transitions, i.e. the system uses any transitions execpt pop transitions. These may only occur for the active pushdown during the phase. A multi-pushdown system with bounded context switching or bounded phase allows only computations that do not exceed a fixed number of contexts or phases.

The model checking problem for multi-pushdown systems with bounded context switching (phase) has been solved in EXPTIME (2-EXPTIME) [3], where the number k of contexts (phases) is part of the input. In this work, we focus on reachability and parity games on multi-pushdown systems with bounded context switching and phase, settling the complexity for game based synthesis and other proceedures looking to solve games on this model.

1996, Walukiewicz introduced the complexity of EXPTIME for single pushdown games by creating a finite state game, exponential in the size of the pushdown system [4, 16]. Key to the construction is to only remember the top of stack symbol of the pushdown and maintain a prediction of possible popping scenarios. Whenever a symbol would be pushed, one of the players proposes such a prediction $\mathcal{P} \subseteq Q$ which is a subset of the states. The other player may either skip to one of the proposed states or remember the set in the state. Whenever a symbol would be popped from the pushdown, the winner of the play is determined depending on whether $\mathcal{P}$ contained the popping scenario. The exponential size of the finite state game is immediate, because the states space of the game consits of the whole powerset of states. However, the construction could only determine the winner of positions with an empty stack. 2003, Cachat provided a uniform solution for pushdown games including arbitrary starting configurations [5].

Seth adapted Walukiewicz's approach in 2009 to find an algorithm solving bounded phase multi-pushdown games [13]. The construction contains a slight error, leading to missing winning strategies for the predicting player even though the original bounded phase multi-pushdown game was winning for them. We construct a very similar game to solve this problem for the bounded context switching case, resulting in a difference of two exponents in complexity to the bounded phase problem. In that work, Seth also suggests a non-elementary lower bound, encouraging further work in this direction.

Very recently, in 2017, there has been work in order to find a uniform solution for bounded phase multi-pushdown games as well [2]. The authors also mention an idea on how to show the non-elementary lower bound. In another publication [1], they find a non-elementary lower bound on model checking branching time logic for bounded context switching multi-pushdown systems by reducing from a Turing machine.

In this work, we settle a tight k-EXPTIME lower bound for $k + 2$ context bound multi-pushdown games and k phase bound multi-pushdown games. This confirms the non-elementary lower bound for both problems and tightens them for each fix bound on the contexts or phases. We use key elements of the construction in [1] of bounded context switching and bounded phase multi-pushdown games such as the idea of storing a computation of the Turing machine on one stack, encoding the configurations of the Turing machine into a layered indexing system. We also use a kind of "Guess & Check" approach heavily for the correctness of the construction.

We also provide an upper bound of k-EXPTIME for $k + 2$ context bounded multi-pushdown games. Together with Seth's upper bound on the bounded phase case, we can conclude k-EXPTIME completness for $k + 2$ context bounded and k phase bounded multi-pushdown games.

2 Preliminaries

First settle notation and definition of basic structures involved in multi-pushdown games. We define multi-pushdown systems, games, plays, winning conditions and games on multi-pushdown systems.

2.1 Multi-pushdown systems

A multi-pushdown system consists of a single state space, a finite number of stacks and a transition relation. Like a regular pushdown system, each transition pushes or pops a symbol on one of the stacks. Each stack is affected only by the transitions acting on them. A transition acting on one stack does not change the contents of any other stack.

Definition 1. A *multi-pushdown system* (*mpds*) P is a tuple (Q, Γ, δ, n) where Q is a finite set of states, n is the number of stacks, Γ is a finite alphabet and $\delta = \delta_{int} \cup \delta_{push} \cup \delta_{pop}$ is a set of transition rules with

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

We denote the set of transition rules acting on stack r by

$$\delta_r = (\delta_{int} \cap (Q \times \{r\} \times Q)) \cup (\delta_{push} \cap (Q \times \{r\} \times \Gamma \times Q)) \cup (\delta_{pop} \cap (Q \times \Gamma \times \{r\} \times Q)).$$

A configuration of P is of the form $(q, \mathcal{S})$ where $q \in Q$ is the configuration's state and $\mathcal{S} : [1..n] \to \Gamma^*\bot$ are its stack contents. Each stack r is being assigned its content $\mathcal{S}_r \in \Gamma^*\bot$. The bottom of each stack consists of a non-removable symbol $\bot$. We sometimes denote configurations of mpds with n stacks by $(q, \mathcal{S}_1, \ldots, \mathcal{S}_n)$. The top of the stack is on the left side in our notation.

The set of all configurations is $C = Q \times (\Gamma^*\bot)^n$. We define the transition relation of the associated transition system.

Definition 2. The *transition relation* $\mapsto \subseteq C \times \delta \times C$ implements the transition rules given by P on its configurations. For any two configurations, $((q, \mathcal{S}), \tau, (q', \mathcal{S}')) \in \mapsto$ if

- $\tau = (q, r, q') \in \delta_{int}$ and $\mathcal{S}_r = \mathcal{S}'_r$ or
- $\tau = (q, r, s, q') \in \delta_{push}$ and $s\mathcal{S}_r = \mathcal{S}'_r$ or
- $\tau = (q, s, r, q') \in \delta_{pop}$ and $\mathcal{S}_r = s\mathcal{S}'_r$

and for all stacks $j \neq r$, $\mathcal{S}_j = \mathcal{S}'_j$.

We also denote this by $(q, \mathcal{S}) \xmapsto{\tau} (q', \mathcal{S}')$. If the transition τ is of no importance, we may omit it. A computation of P is a sequence $\pi = \pi_0 \dots \pi_l \in C^*$ of configurations with $\pi_0 \mapsto \pi_1 \mapsto \dots \mapsto \pi_l$.

We restrict our computational model by introducing a bound to phases or contexts.

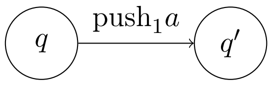
Definition 3. A *context* on stack r is a non-empty computation $\alpha = \alpha_0 \xmapsto{\tau_0} \alpha_1 \xmapsto{\tau_1} \dots \xmapsto{\tau_{l-1}} \alpha_l$ such that each transition rule acts on stack r. I.e. for each position $0 \leq p < l$, $\tau_p \in \delta_r$.

A *phase* on stack r is a non-empty computation $\alpha = \alpha_0 \xmapsto{\tau_0} \alpha_1 \xmapsto{\tau_1} \dots \xmapsto{\tau_{l-1}} \alpha_l$ such that each pop transition rule acts on stack r. Formally, there is a transition rule $\tau_t \in \delta_r \cap \delta_{pop}$ and for each position $0 \leq p < l$, if $\tau_p \in \delta_{pop}$ then $\tau_p \in \delta_r \cap \delta_{pop}$.

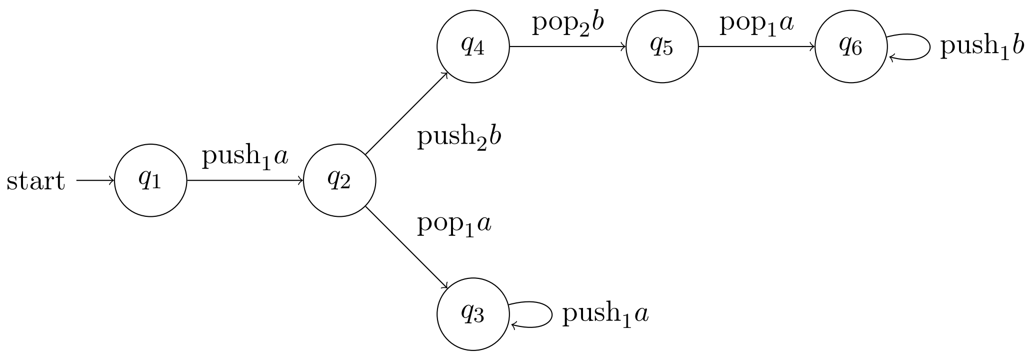
A computation $\pi = \pi_0 \xmapsto{\tau_0} \pi_1 \xmapsto{\tau_1} \dots \xmapsto{\tau_{l-1}} \pi_l$ is a k *context (phase) computation*, if it consists of k contexts (phases):

$$\alpha_1 = \pi_0 \xmapsto{\tau_0} \dots \xmapsto{\tau_{p_1-1}} \pi_{p_1} \quad \alpha_2 = \pi_{p_1} \xmapsto{\tau_{p_1}} \dots \xmapsto{\tau_{p_2-1}} \pi_{p_2} \quad \dots$$
$$\alpha_k = \pi_{p_{k-1}} \xmapsto{\tau_{p_{k-1}}} \dots \xmapsto{\tau_{l-1}} \pi_l.$$

Example 4. Let $P = (\{q_1, \dots, q_6\}, \{a, b\}, \delta, 2)$ where δ is given by the following figure. A transition



states the transition rule $(q, 1, a, q')$ and vice versa for popping transitions.



An example configuration of P is $(q_2, \mathcal{S})$ where $\mathcal{S}_1 = a\bot$ and $\mathcal{S}_2 = \bot$, also denoted by $(q_2, a\bot, \bot)$.

An example computation is

$$(q_1, \bot, \bot) \xmapsto{(q_1,1,a,q_2)} (q_2, a\bot, \bot) \xmapsto{(q_2,2,b,q_4)} (q_4, a\bot, b\bot) \xmapsto{(q_4,b,2,q_5)}$$
$$(q_5, a\bot, \bot) \xmapsto{(q_5,1,a,q_6)} (q_6, \bot, \bot) \xmapsto{(q_6,1,b,q_6)} (q_6, b\bot, \bot) \xmapsto{(q_6,1,b,q_6)} (q_6, bb\bot, \bot) \dots$$

which is a 3 context computation. It switches context at the transition $(q_2, 2, b, q_4)$ and $(q_5, 1, a, q_6)$ and then stays in context 3 forever.

When limiting contexts to 2 for example, the computation would stop in $(q_5, a\bot, \bot)$ and no transitions could be used anymore.

2.2 Games

A game consists of a set of positions, a set of moves, two players, called Eve and Ana, and a winning condition. Each player owns a partition of the positions. They can play the game from some starting position and whenever the play is in a position owned by Eve or Ana respectivly, they choose a move from the set of moves to continue the play to the next position. When they complete a play, the winning condition decides the winner of that play.

Definition 5. A *game* is a tuple $(V, E, \mathbb{W})$ together with an *ownership* function $\mathit{own} : V \to \{\text{Eve}, \text{Ana}\}$, where V is a non-empty set of positions, $E \subseteq V^2$ is the set of moves and $\mathbb{W} : V^\omega \to \{\text{Eve}, \text{Ana}\}$ is the winning condition.

Let $\Box \in \{\text{Eve}, \text{Ana}\}$ and $V_\Box = \{v \in V \mid \mathit{own}(v) = \Box\}$.

A *play* is a sequence $\pi = (\pi_p)_{p\in\mathbb{N}}$, such that for all p, $\pi_p\pi_{p+1} \in E$.

A play π is *winning* for $\Box$ if $\mathbb{W}(\pi) = \Box$.

A *strategy* for $\Box$ is a function $\sigma : V^*V_\Box \to V$ such that $v\sigma(\pi v) \in E$.

A strategy is *positional*, if $\sigma : V_\Box \to V$.

A play π is *compliant with* strategy σ of $\Box$ if for all $\pi_p \in V_\Box$: $\pi_{p+1} = \sigma(\pi_p)$.

A strategy σ is called *winning* for $\Box$ from $v \in V$ if all plays compliant with σ starting in v are won by $\Box$.

There are a variety of different winning conditions. We focus on two winning conditions relevant in model checking: reachability and parity winning condition.

Definition 6. The *reachability* winning condition $\mathbb{W}_{reach}$ consists of a set of winning positions $V_{reach} \subseteq V$ and the winner of a play is determined by visiting a winning position:

$$\mathbb{W}_{reach}(\pi) = \begin{cases} \text{Eve} & \text{if there is } \pi_i \text{ with } \pi_i \in V_{reach} \\ \text{Ana} & \text{otherwise} \end{cases}.$$

A *reachability game* is the tuple (V, E, V_{reach}).

The *parity* winning condition $\mathbb{W}_m$ consists of a parity assignment $\Omega : V \to [1..\mathit{max}]$ and the winner of a play is determined by the greatest infinitly occuring parity during the play:

$$\mathbb{W}_{parity}(\pi) = \begin{cases} \text{Eve} & \text{if the greatest infinitly occuring parity in } (\Omega(\pi_i))_{i\in\mathbb{N}} \text{ is even} \\ \text{Ana} & \text{otherwise} \end{cases}.$$

A *parity game* is the tuple (V, E, Ω).

Even though plays are infinite by definition, we will allow games to contain positions v that do not offer a move, i.e. $E(v) = \varnothing$. A play ending in such a position v is losing for $own(v)$ and thus winning for their opponent. There is a game equivalent to G, where we introduce the positions Evewin and Anawin with the moves $(\square\text{win}, \square\text{win})$ and $(v, \square\text{win})$ if $E(v) = \varnothing$ and $own(v) = \blacksquare \neq \square$. We also complete the winning condition by

$$\mathbb{W}_{new}(\pi) = \begin{cases} \square & \text{if } \square\text{win occurs in } \pi \\ \mathbb{W}(\pi) & \text{otherwise} \end{cases} .$$

We use the term "$\square$ possesses a strategy from v to V'", where $V' \subseteq V$, as abbreviation for: Player $\square$ possesses a strategy σ, such that each play π compliant with σ starting in v visits a position $\pi_j \in V'$ or is winning for player $\square = \mathbb{W}(\pi)$.

2.2.1 Game simulation

Later, we want to carry strategies from one reachability game to another. We introduce game simulation for reachability games in order to find those strategies without having to explicitly name them. Intuitivly, a game simulating another offers to reproduce moves from the original game. However, these do not have to be a single move. Instead we require a strategy for the owner of each state, that will bring them to the desired successing position.

Definition 7. A reachability game (V^A, E^A, V^A_{reach}) is *simulated* by (V^B, E^B, V^B_{reach}) if there is a function $f : V^A \to V^B$ such that for each position $v \in V^A$, with $own(v) = \square$ and $\blacksquare$ their opponent,

- $own(v) = own(f(v))$
- $v \in V^A_{reach}$ if and only if $f(v) \in V^B_{reach}$
- If $v' \in E^A(v)$, then $\square$ possesses a strategy $\sigma_{v,v'}$ from $f(v)$ to $f(v')$. Player $\blacksquare$ possesses a strategy σ_v from $f(v)$ to $f(E^A(v))$
- Let $\pi = \pi_0\pi_1\ldots$ be any play from $f(v)$ compliant with $\sigma_{v,v'}$ that contains a first position p with $\pi_p = f(v')$. Then, for all $0 < u < p$, $\pi_u \notin V^B_{reach}$
- Let $\pi = \pi_0\pi_1\ldots$ be any play from $f(v)$ compliant with σ_v that contains a first position p with $\pi_p \in f(E^A(v))$. Then, for all $0 < u < p$, $\pi_u \notin V^B_{reach}$.

Lemma 8. *Let $A = (V^A, E^A, V^A_{reach})$ be simulated by $B = (V^B, E^B, V^B_{reach})$ via f. A position $v \in V^A$ is winning for Eve if and only if $f(v)$ is winning for Eve.*

Proof. Let $\square$ possess a winning strategy σ^A from a position $v \in V^A$. We construct a winning strategy σ^B for $\square$ from $f(v)$. We construct this non-positional strategy by remembering a play prefix $\pi_0\pi_1\ldots\pi_i$ of A that is compliant with σ^A. We show, how σ^B acts to continue a play prefix $\pi'_0\pi'_1\ldots\pi'_{\psi(i)}$ in B, where $\psi : \mathbb{N} \to \mathbb{N}$ maps positions in π to positions in π' such that

- for each $u \in [0..i]$, $f(\pi_u) = \pi'_{\psi(u)}$
- for each $t \in [0..i-1]$ and all $\psi(t) < u < \psi(t+1)$, $\pi'_u \notin V^B_{reach}$.

Strategy σ^B either wins the play π' from $\pi'_{\psi(i)}$ or continues to $\pi'_{\psi(i+1)}$ together with some π_{i+1}, such that $\pi_0 \dots \pi_i \pi_{i+1}$ is compliant with σ^A. We construct σ^B and show the correctness via induction:

Base Case ($i = 0$)**.** $\pi_i = \pi_0 = v$, $\psi(0) = 0$, $\pi'_{\psi(i)} = \pi'_0 = f(v)$.

Inductive Case. Assume $\pi'_0 \dots \pi'_{\psi(i)}$ is compliant with σ^B and $\pi_0 \dots \pi_i$ is compliant with σ^A and ψ is constructed according to the above invariant.

Case 1 (there is a position p, such that $\pi'_p \in V^B_{reach}$)*:* By induction, the play prefixes are according to the invariant. Thus, for each position $0 \leq t < i$, all $\psi(t) < u < \psi(t+1)$ fulfill $\pi'_u \notin V^B_{reach}$. The only position possible for p is $p = \psi(t)$ for some $0 \leq t \leq i$. By the invariant, $f(\pi_t) = \pi'_{\psi(t)} = \pi'_p$, and by definition of simulation $\pi_t \in V^A_{reach}$. Any play with the prefix $\pi_0 \dots \pi_i$ is winning for Eve. Since σ^A was winning for $\square$ and by induction, $\pi_0 \dots \pi_i$ is compliant with σ^A, Eve $= \square$. Indeed, the winner of any play with prefix $\pi'_0 \dots \pi'_{\psi(i)}$ is winning for Eve.

For the rest of the inductive case, assume that for all positions $0 \leq p \leq \psi(i)$, $\pi'_p \notin V^B_{reach}$. In particular, the winning condition is independent of the play up to $\pi_{\psi(i)}$.

Case 2 ($own(v) = \square$)*:* Let $v' = f(\sigma^A(v))$, Player $\square$ continues the play by using strategy $\sigma_{v,v'}$, until
Case 2.1 (there occurs a first position p in π', such that $\pi'_p = f(v')$): Set $\psi(i+1) = p$. Continue $\pi_0 \dots \pi_i$ by $\pi_{i+1} = v'$. By definition of simulation and $\sigma_{v,v'}$, for all $\psi(i) < u < \psi(i+1)$, $\pi'_u \notin V^B_{reach}$.
Case 2.2 ($f(v')$ is not visited by π'): By definition, $\sigma_{v,v'}$ is a strategy from $f(v)$ to $f(v')$. Thus, any play compliant with that strategy that does not visit $f(v')$ is winning for $\square$.

Case 3 ($own(v) = \blacksquare$)*:* Player $\square$ continues the play by using strategy σ_v, until
Case 3.1 (there occurs a first position p in π', such that $\pi'_p = f(v')$, where $v' \in E^A(v)$): Set $\psi(i+1) = p$. Continue $\pi_0 \dots \pi_i$ to $\pi_{i+1} = v'$. By definition of simulation and σ_v, for all $\psi(i) < u < \psi(i+1)$, $\pi'_u \notin V^B_{reach}$.
Case 3.2 ($f(E^A(v))$ is not visited by π'): By definition, σ_v is a strategy from $f(v)$ to $f(E^A(v))$. Thus, any play compliant with that strategy that does not visit $f(E^A(v))$ is winning for $\square$.

Towards contradiction assume a play π' compliant with σ^B is losing for $\square$. Then, neither Case 1, Case 2.2 nor Case 3.2 happend during the play. Thus, π' is an infinite play compliant with σ^B and there is an infinite play π compliant with σ^A together with a function ψ according to the above conditions.

Case 1 ($\Box$ = Eve)*:* Since σ^A is winning for Eve, there is a position $\pi_p \in V^A_{reach}$ and thus $\pi'_{\psi(p)} = f(\pi_p) \in f(V^A_{reach}) \subseteq V^B_{reach}$, i.e. π' is winning for Eve.

Case 2 ($\Box$ = Ana)*:* Since σ^A is winning for Ana, π does not visit V^A_{reach} and thus for all $i \in \mathbb{N}$, $\pi'_{\psi(i)} \notin V^B_{reach}$. Furthermore, for all $\psi(i) < u < \psi(i+1)$, $\pi'_u \notin V^B_{reach}$. Then, π' does not visit V^B_{reach} and the play is winning for Ana.

$\Box$

2.3 Multi-pushdown games

A multi-pushdown game is a game on the configurations of a multi-pushdown system. The set of positions is the set of configurations of the multi-pushdown system and the set of moves is the transition relation. Ownership is determined by partition of the state space of the multi-pushdown system.

Definition 9. Let $P = (Q, \Gamma, \delta, n)$ be a multi-pushdown system.
A *multi-pushdown game* (*mpdg*) $(P, \mathbb{W})$ is the game $(C, \mapsto, \mathbb{W})$ with ownership function $own : Q \to \{\text{Eve}, \text{Ana}\}$ that carries over to positions of the game via $own(q, \mathcal{S}) = own(q)$.

For the winning condition, we consider state reachability games, configuration reachability games and parity games.

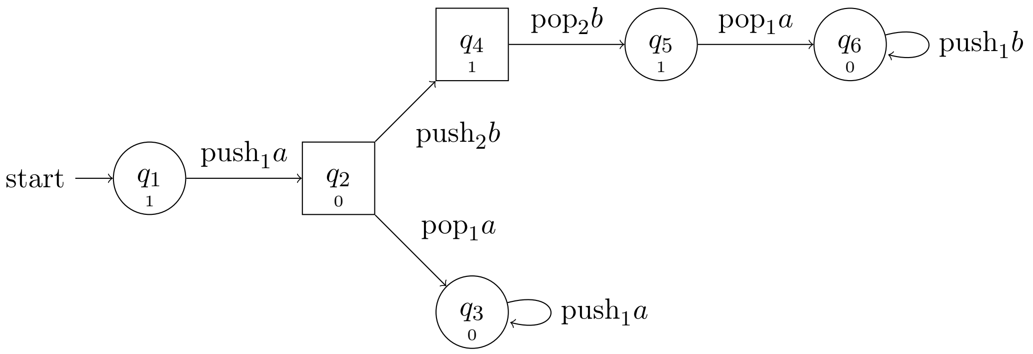
Definition 10. The *configuration reachability* mpdg (P, C_{reach}) with winning region $C_{reach} \subseteq C$ is the reachability game $(C, \mapsto, C_{reach})$.
The *state reachability* mpdg (P, Q_{reach}) with winning states $Q_{reach} \subseteq Q$ is the configuration reachability mpdg $(P, Q_{reach} \times (\Gamma^*)^n)$.
The *parity* mpdg (P, Ω) with parity assignment $\Omega : Q \to [1..max]$ is the parity game $(C, \mapsto, \mathbb{W}_{parity})$ where the parity assignment carries over to positions via $\Omega(q, \mathcal{S}) = \Omega(q)$.

As a stronger computational model than multi-pushdown systems, multi-pushdown games are also Turing complete. We apply the previous restrictions for multi-pushdown systems.

Definition 11. A k-context (phase) bounded multi-pushdown game $(P, \mathbb{W}, k)$ is a variation of the mpdg $(P, \mathbb{W})$ where no play may exceed the k'th context (phase). Moves that would introduce a $k+1$'th context (phase) do not exist.

We omit a more formal definition for the sake of notation. It should be noted that this way some plays may not be continued at all. There are different ways to approach this. We choose that the player owning the position, from which the play can not be continued, looses the game. The lower bound construction is unaffected by this, as the mpdg constructed there will stay within k contexts (phases) for any computation and will always offer a possible transition. For the upper bound, we assume that there will always be a possible transition from each position without introducing a $k+1$'th context. An equivalent mpdg with this condition can be constructed analoge to the construction in section 2.2 about games in general.

Example 12. We use the multi-pushdown system from example 4 for an example multi-pushdown game. We add the ownership function $own : \{q_1, \dots, q_6\} \to \{\text{Eve}, \text{Ana}\}$ by shaping the states in the mpds graph. A round state belongs to Eve and a rectangle one to Ana. We also set a parity function $\Omega : \{q_1, \dots, q_6\} \to \{0, 1\}$ by writing the parities at the bottom of each state.



Since the parities $\Omega(q_3) = \Omega(q_6) = 0$ are winning for Eve, each play is winning for Eve in this multi-pushdown game.
If we limit the amount of contexts to 2 however, Ana gains a winning strategy, by taking using the transition rule $(q_2, 2, b, q_4)$. The play continues to the configuration $(q_5, a\bot, \bot)$. As in example 4, the number of contexts can not be further increased and Eve, who owns q_5, has no transitions they can take which causes them to loose the play.

For better construction of mpdg, we introduce push and pop transitions of regular expressions instead of single symbols. Each such transition is a short notion for a set of transitions, that allow the player owning the state to push or pop any word matching the regular expression.

Definition 13. Let $\mathcal{R}$ be a regular expression and $(Q_A, \delta_A, I_A, F_A)$ be the NFA accepting $\mathcal{R}$.
We write the transition rules

$$\begin{aligned} &(q, r, \mathcal{R}, q') && \text{for pushing a word belonging to } \mathcal{R} \text{ onto stack } r \\ &(q, \mathcal{R}, r, q') && \text{for popping a word belonging to } \mathcal{R} \text{ from stack } r \end{aligned}$$

as a replacement for a finite number of transition rules.
A pushing transition rule $(q, r, \mathcal{R}, q')$ is replaced by finitly many transition rules:

$$\begin{tikzcd} q \arrow[r, "\mathrm{int}_r"] & q_{\mathrm{init}} \end{tikzcd}$$ for each $q_{\mathrm{in}} \in I_{\mathrm{A}}$

$$\begin{tikzcd} q_{\mathrm{A}} \arrow[r, "\mathrm{push}_r s"] & q'_{\mathrm{A}} \end{tikzcd}$$ for each $(q_{\mathrm{A}}, s, q'_{\mathrm{A}}) \in \delta_{\mathrm{A}}$

$$\begin{tikzcd} q_{\mathrm{out}} \arrow[r, "\mathrm{int}_r"] & q' \end{tikzcd}$$ for each $q_{\mathrm{out}} \in F_{\mathrm{A}}$

We also set the ownership of all $q_{\mathrm{A}} \in Q_{\mathrm{A}}$ to $own(q_{\mathrm{A}}) = own(q)$.
Analogously for a popping transition rule, except that we pop instead of pushing:

$$\begin{tikzcd} q_{\mathrm{A}} \arrow[r, "\mathrm{pop}_r s"] & q'_{\mathrm{A}} \end{tikzcd}$$ for each $(q_{\mathrm{A}}, s, q'_{\mathrm{A}}) \in \delta_{\mathrm{A}}$.

However, this construction has a flaw. In a reachability mpdg, a pushing transition of the above form may allow Ana to loop in an infinite cycle within the NFA and not ever complete the transition. This would result in a winning play for them. Popping transitions do not suffer this problem as the stack height is finite. We circumvent this problem by only using regular expressions that do not contain kleene iteration or complement when giving Ana a regular expression pushing transition, as there are NFA without cycles to accept them.

Example 14. The regular expression $\{a, b\}^*$ has the NFA

$$\begin{tikzcd} \text{start} \arrow[r] & q_{\mathrm{A}} \arrow[loop right, "{a, b}"] \end{tikzcd}$$

A pushing transition $(q, r, \{a, b\}^*, q')$ is replaced by the transitions

$$\begin{tikzcd} q \arrow[r, "\mathrm{int}_r"] & q_{\mathrm{A}} \arrow[r, "\mathrm{int}_r"] \arrow[loop below, "{\mathrm{push}_r a, b}"] & q' \end{tikzcd}$$

In a reachability game, Ana can loop forever in q_{A} which might form a winning strategy for them, as no other state would ever be visited.

3 Upper bound

In this section, we present a finite game that is equivalent to a context bounded multi-pushdown game. The size of the finite game depends on the number of contexts allowed.

Theorem 15. *$k+2$-context bounded multi-pushdown parity games can be solved in k-EXPTIME.*

For the rest of the section, let $G_P = (P, \Omega, k)$ be a k-context bound multi-pushdown game with $P = (Q, \Gamma, \delta, n)$, $\Omega : Q \to [0..max]$ and $own : Q \to \{\text{Eve}, \text{Ana}\}$. The game G_P consists of infinitly many positions: The set of configurations of P is infinite, because the stack contents can grow arbitrarily large. The goal is to reduce this amount of positions to a finite number. Intuitivly, we forget the exact stack contents and only remember the top of stack symbol for each stack. This idea is similar to Walukiewicz's construction [16]. In fact it is a variation of Anil Seth's construction for bounded phase multi-pushdown games, which contains a slight constructional error. We will see an example for which Eve fails to obtain a winning strategy in Seth's finite game, even though they possess a winning strategy in the original mpdg.

Remembering only the top of stack symbol for each stack makes it impossible to continue a play that used a pop transition, since the next top of stack symbol was forgotten. To circumvent this problem, we use a Guess & Check approach that ends any play when it discovers its first pop transition. Whenever a symbol is pushed on one of the stacks, Eve suggests a set of popping scenarios for this symbol. The idea is that Eve can fix their strategy and they can predict all plays compliant with this strategy. They can collect all situations where eventually the pushed symbol is popped and create the desired set of popping scenarios. We call this set a prediction set. Given a prediction set, Ana has two choices. They can choose one of the popping scenarios given in the prediciton set and move to its corresponding position, skipping the play in between. Or they actually perform the pushing transition. In the second case, we remember the prediction set in the positions of the finite game. This is necessary to determine a winner, when a popping transition is used. Whenever a popping transition is used, the game looks up whether the remembered prediction set contains the current situation as a popping scenario and determines the winner of the play.

3.1 Push-pop-pairs, stairs and notation

We first settle some notation and define push-pop-pairs, last unmatched pushes and stairs. Remember that a play of G_P is a computation $(q^0, \mathcal{S}^0) \xmapsto{\tau_0} (q^1, \mathcal{S}^1) \xmapsto{\tau_1} \ldots$

where each q is a state and each $\mathcal{S}$ is a function assigning each stack its contents.

In the rest of the section occur many functions of the form $\overline{T} : [1..n] \to A$ assigning each stack some information.

Definition 16. Let $\overline{T} : [1..n] \to A$ for some set A. For each $r \in [1..n]$ and $x \in A$, we use the notation $\overline{T}_j$ for $\overline{T}(j)$ and $\overline{T}[x/r]$ which replaces the information for stack r by x:

$$\overline{T}[x/r]_j = \begin{cases} x & j = r \\ \overline{T}_j & \text{otherwise.} \end{cases}$$

A push-pop-pair of a play for an mpdg are two positions. The first position is a pushing transition and the second is the popping transition that pops precisly the symbol pushed by the push transition.

Definition 17. A push-pop-pair of a play $(q^0, \mathcal{S}^0) \overset{\tau_0}{\longmapsto} (q^1, \mathcal{S}^1) \overset{\tau_1}{\longmapsto} \dots$ of G_P are two indices $(p, p') \in \mathbb{N}^2$ such that there is a stack r with

- $p < p'$
- $\tau_p \in \delta_r \cap \delta_{push}$
- $\tau_{p'} \in \delta_r \cap \delta_{pop}$
- $\min\{|\mathcal{S}_r^{p+1}|, \dots, |\mathcal{S}_r^{p'}|\} > |\mathcal{S}_r^p| = |\mathcal{S}_r^{p'+1}|$

Note that each $p \in \mathbb{N}$ is in at most one push-pop-pair.

Definition 18. Let $\pi = \pi_0 \overset{\tau_0}{\longmapsto} \pi_1 \dots$ be a play of G_P. We call a pushing position $p \in \mathbb{N}$ with $\tau_p \in \delta_{push}$ matched, if there is $p' \in \mathbb{N}$ such that (p, p') is a push-pop-pair. Otherwise, the push is unmatched.

For each stack j, we define the function $\mathrm{lup}_j^\pi : \mathbb{N} \to \mathbb{N}$ which finds the last unmatched push of stack j in π.

$$\mathrm{lup}_j^\pi(p) = \max_{\substack{0 \le u < p \\ \tau_u \in \delta_{push}}} \{u \mid u \text{ unmatched in } \pi_0 \dots \pi_p\}$$

Be aware, that if no such pushing position exists, $\mathrm{lup}_j^\pi(p) = \bot$ is undefined.

Definition 19. Let π be a play and $p \in \mathbb{N}$. Let $\pi_p = (q^p, \mathcal{S}^p)$.

A stair $p \in \mathbb{N}$ of π is a position where for all $i \in \mathbb{N}$ with $p \le i$ and each stack $j \in [1..n]$, $|\mathcal{S}_j^p| \le |\mathcal{S}_j^i|$.

Be aware, that by this definition there is no push-pop-pair (p', p'') such that $p' < p < p''$.

3.2 Finite state game

We construct the finite state game FSG equivalent to the multi-pushdown game G_P. It consists of positions of the form

$$(\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}).$$

They represent positions of G_P with additional information. The positions contain

- q, the state of the configuration.
- c, the context of the computation.
- d, the stack holding the current context.
- $\overline{\gamma} : [1..n] \to \Gamma$, the current top of stack symbols.
- $\overline{\mathcal{P}} : [1..n] \to \mathbb{P}$, a prediction set for each stack. This is explained later.
- $\overline{m} : [1..n] \to [1..max]$, the maximal parity seen for each stack since the the last unmatched push of that stack.

We will define a set of predictions $\mathbb{P}$. Each element of $\mathbb{P}$ describes a popping scenario. We partition the set in smaller sets $\mathbb{P} = \cup_{r\in[1..n],j\in[1..k]}\mathbb{P}_{r,j}$. An element of $\mathbb{P}_{r,j}$ describes a situation where the top of stack symbol from stack r is popped during context j.

Intuitivly, when a player wants to use a pushing transition for stack r, instead Eve will guess a set $\overline{\mathcal{P}}_r \subseteq \cup_{j\in[1..k]}\mathbb{P}_{r,j}$. After Eve suggests such a prediction set, Ana may either choose a popping scenario from the set and skip the computation in between, or actually perform the pushing transition. In this case, either the pushed symbol is never popped, or when it is popped, one of the players is determined as winner of this play by the use of the current prediction sets $\overline{\mathcal{P}}$.

A popping scenario contains more than just the top of stack symbols, the state and the context after popping. It also contains the maximal parity for each stack since their last unmatched push and a prediction set for each stack.

Such a popping scenario can be thought of as an abstraction of the computation from the pushing position up to the popping position. Other stacks can have pushed or popped symbols in the mean time, creating different prediction sets for them. A popping scenario thus also contains a prediction set for each other stack.

Definition 20. Let $q, \overline{\gamma}, \overline{m}$ range over $Q, \Gamma^n, [1..max]^n$. For each $r \in [1..n]$, $j \in$

$[3..k-1]$ define the prediciton sets

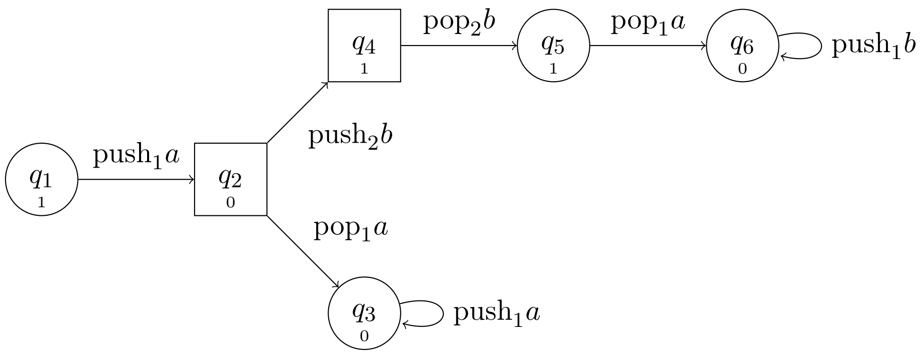
$$\begin{aligned}
\mathbb{P}_{r,k} &= \{(\overline{\mathcal{P}}_1, \dots, \overline{\mathcal{P}}_{r-1}, (q, k, \overline{\gamma}, \overline{m}), \overline{\mathcal{P}}_{r+1}, \dots, \overline{\mathcal{P}}_n) \mid \overline{\mathcal{P}}_1 = \dots = \overline{\mathcal{P}}_n = \varnothing\} \\
\mathbb{P}_{r,1} &= \{(\overline{\mathcal{P}}_1, \dots, \overline{\mathcal{P}}_{r-1}, (q, 1, \overline{\gamma}, \overline{m}), \overline{\mathcal{P}}_{r+1}, \dots, \overline{\mathcal{P}}_n) \mid \overline{\mathcal{P}}_1 = \dots = \overline{\mathcal{P}}_n = \varnothing\} \\
\mathbb{P}_{r,2} &= \{(\overline{\mathcal{P}}_1, \dots, \overline{\mathcal{P}}_{r-1}, (q, 2, \overline{\gamma}, \overline{m}), \overline{\mathcal{P}}_{r+1}, \dots, \overline{\mathcal{P}}_n) \mid \overline{\mathcal{P}}_1 = \dots = \overline{\mathcal{P}}_n = \varnothing\} \\
\mathbb{P}_{r,c} &= \{(\overline{\mathcal{P}}_1, \dots, \overline{\mathcal{P}}_{r-1}, (q, c, \overline{\gamma}, \overline{m}), \overline{\mathcal{P}}_{r+1}, \dots, \overline{\mathcal{P}}_n) \\
&\qquad \mid \overline{\mathcal{P}}_1 \subseteq \cup_{c'=c+1}^{k} \mathbb{P}_{1,c'}, \dots, \overline{\mathcal{P}}_n \subseteq \cup_{c'=c+1}^{k} \mathbb{P}_{n,c'}\} \\
&\qquad \text{for } 3 \le c < k \\
\mathbb{P}_r &= \bigcup_{c=1}^{k} \mathbb{P}_{r,c}
\end{aligned}$$

An element of $\overline{\mathcal{P}} \in \mathbb{P}_{r,c}$ describes a situation where the topmost symbol from stack r is popped in context c. It contains the tuple $(q, c, \overline{\gamma}, \overline{m})$, containting information of the configuration after popping:

- q the state
- c the context
- $\overline{\gamma}$ the top of stack symbols
- $\overline{m}$ the maximal parity seen for each top of stack symbol since its respective push operation.

It also contains $\overline{\mathcal{P}}_1, \dots, \overline{\mathcal{P}}_{r-1}, \overline{\mathcal{P}}_{r+1}, \dots, \overline{\mathcal{P}}_n$, a guarantee for the prediction sets that were created during computation up to this popping scenario for each top of stack symbol. We call them a guarantee, because the prediction set for another stack j at the popping position may actually contain more popping scenarios than $\overline{\mathcal{P}}_j$. That prediction set was made before the popping scenario $\overline{\mathcal{P}}$ happened. Thus, there might be popping scenarios for that top of stack symbol that don't include $\overline{\mathcal{P}}$ in their play.

Example 21. Recall example 12. We used the following multi-pushdown game.



Assume a play in configuration $(q_1, \perp, \perp)$ in a context $c > 2$ on stack 1 (We want the context to be greater than 2 for demonstration purposes). The next transition $(q_1, 1, a, q_2)$ is a pushing transition. In the corresponding FSG, Eve has to push a prediction set. They do not know whether Ana wants to continue the play to either q_3 or q_4. Eve has to collect all possible popping scenarios for the symbol that is being pushed. Looking at the transitions, it is immediate that there are precisly two popping scenarios. One is the transition $(q_2, a, 1, q_3)$ and the other transition is $(q_5, a, 1, q_6)$ for another play.
Eve then proposes a prediction set $\mathcal{P}$ with two elements $\overline{\mathcal{P}}^1$ and $\overline{\mathcal{P}}^2$ where they include the correct parameters for

$$\overline{\mathcal{P}}^1_1 = (q_3, c, (\perp, \perp), (0, 0))$$
$$\overline{\mathcal{P}}^2_1 = (q_6, c + 2, (\perp, \perp), (1, 1)).$$

As discussed, Eve also has to propose guarantees for the prediction sets for stack 2 in each scenario. Since in both scenarios, the top of stack symbol of stack 2 is $\perp$ which is never popped, the sets $\overline{\mathcal{P}}^1_2 = \overline{\mathcal{P}}^2_2 = \varnothing$ are empty.
Assume Ana decides not to skip to one of the predicted scenarios. Instead, we remember $\mathcal{P}$ for stack 1 and the game continues in q_2. Let Ana choose the transition $(q_2, 2, b, q_4)$. Eve has to make a new prediction, this time for stack 2. There is only one popping scenario $\overline{\mathcal{P}}'$ for this symbol:

$$\overline{\mathcal{P}}'_2 = (q_5, c + 1, (a\perp, \perp), (1, 1))$$

and the prediction guarantee for the current stack 1 top of stack symbol. Note that Eve now knows more precisely about the possible popping scenarios of the pushed a. They can reduce the prediction set for that stack by setting

$$\overline{\mathcal{P}}'_1 = \{\overline{\mathcal{P}}^2\}.$$

Be aware that they cannot even add the prediction $\overline{\mathcal{P}}^1$ to this set: Since $\overline{\mathcal{P}}'_2 \in \mathbb{P}_{2,c+1}$ is a scenario for popping in context $c + 1$ and $\overline{\mathcal{P}}^1 \in \mathbb{P}_{1,c}$ is a scenario for popping in context c, the construction does not allow that scenario as a guarantee (see Definition 20).
Assume Ana again does not skip to the proposed popping scenario but remembers the prediction set $\{\overline{\mathcal{P}}\}$ for stack 2. Note that the remembered prediciton set for stack 1 is still $\mathcal{P} = \{\overline{\mathcal{P}}^1, \overline{\mathcal{P}}^2\}$. The next transition is a popping transition $(q_4, b, 2, q_5)$ and the finite state game is supposed to find a winner for the play. It will see that the prediction set for stack 2 is $\{\overline{\mathcal{P}}\}$. The context, the resulting state after popping, the top of stack symbols and the maximal parities seen for each stack are correct in $\overline{\mathcal{P}}$. Also, the given guarantee $\overline{\mathcal{P}}'_1$ for stack 1 is fitting. The current remembered prediction is still $\mathcal{P} = \{\overline{\mathcal{P}}^1, \overline{\mathcal{P}}^2\} \supseteq \{\overline{\mathcal{P}}^2\} = \overline{\mathcal{P}}'_1$. In other words, it contained the popping scenarios which were guaranteed to exist by the popping scenario $\overline{\mathcal{P}}'$.

3.2.1 Construction

We construct the finite state game for G_P. For a position $(\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})$, and a stack r, define

$$c' = \begin{cases} c & d = r \\ c+1 & d \neq r \end{cases}.$$

The following moves are only introduced, if $c' \leq k$.

1. $(q, r, q') \in \delta_{int}$:
   $(\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}) \to (\text{Check}, q', c', d', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}')$ where
   a) $\overline{m}'_j = \max\{\overline{m}_j, \Omega(q')\}$ for each $1 \leq j \leq n$
   b) $d' = r$

2. $(q, r, s, q') \in \delta_{push}$:
   a) $(\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}) \to (\text{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s)$
   b) $(\text{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s) \to (\text{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P})$ where $\mathcal{P} \subseteq \mathbb{P}_r$.
   c) $(\text{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}) \to (\text{Check}, q', c', r, \overline{\gamma}[s/r], \overline{\mathcal{P}}[\mathcal{P}/r], \overline{m}')$ where for each stack $j \neq r$, $\overline{m}'_j = \max\{\overline{m}_j, \Omega(q')\}$ and $\overline{m}'_r = \Omega(q')$.
   d)
   - $(\text{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}) \to (\text{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}', \overline{m}', \overline{m}_r)$ for each $\overline{\mathcal{P}}'' \in \mathcal{P}$, where $\overline{\mathcal{P}}''_r = (q'', c'', \overline{\gamma}', \overline{m}')$ with $c'' \neq 2$ and $\overline{\mathcal{P}}' = \overline{\mathcal{P}}''[\overline{\mathcal{P}}_r/r]$.
   - $(\text{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}) \to (\text{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}, \overline{m}', m_r)$ for each $\overline{\mathcal{P}}'' \in \mathcal{P}$ where $\overline{\mathcal{P}}''_r = (q'', 2, \overline{\gamma}', \overline{m}')$.
   e) $(\text{Jump}_r, q, c'', \overline{\gamma}', \overline{\mathcal{P}}', \overline{m}', \overline{m}_r) \to (\text{Check}, q'', c'', r, \overline{\gamma}, \overline{\mathcal{P}}', \overline{m}'')$ where

$$\overline{m}''_j = \begin{cases} \max\{\Omega(q), t'_r, \overline{m}_r\} & j = r \\ \max\{\Omega(q), t'_j\} & j \neq r \end{cases}$$

3. $(q, s, r, q') \in \delta_{pop}$:
   - $(\text{Check}, q, c, d, \overline{\gamma}[s/r], \overline{\mathcal{P}}, \overline{m}) \to \text{Evewin}$ if $\overline{\mathcal{P}}' \in \overline{\mathcal{P}}_r$
   - $(\text{Check}, q, c, d, \overline{\gamma}[s/r], \overline{\mathcal{P}}, \overline{m}) \to \text{Anawin}$ if $\overline{\mathcal{P}}' \notin \overline{\mathcal{P}}_r$

   where

$$\begin{aligned} &\overline{\mathcal{P}}' = \overline{\varnothing}[(q', c', \overline{\gamma}, \overline{m})/r] && \text{if } c' = k \text{ or } c' = 2 \\ &\overline{\mathcal{P}}'_j \subseteq \overline{\mathcal{P}}_j, \quad \overline{\mathcal{P}}'_r = (q', c', \overline{\gamma}, \overline{m}) && \text{for all } j \neq r, \text{ if } 2 \neq c' < k \end{aligned}$$

Together with the parity assignment

$$\begin{aligned}
\Omega_{\mathrm{FSG}}((\mathrm{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})) &= \Omega(q) \\
\Omega_{\mathrm{FSG}}((\mathrm{Push}_r, \dots)) = \Omega_{\mathrm{FSG}}((\mathrm{Claim}_r, \dots)) &= 0 \\
\Omega_{\mathrm{FSG}}((\mathrm{Jump}_r, q, c'', \overline{\gamma}', \overline{\mathcal{P}}', \overline{m}', m_r)) &= \overline{m}'_r \\
\Omega_{\mathrm{FSG}}(\mathrm{Evewin}) &= 0 \\
\Omega_{\mathrm{FSG}}(\mathrm{Anawin}) &= 1
\end{aligned}$$

this defines a parity game $\mathrm{FSG} = (V_{\mathrm{FSG}}, E_{\mathrm{FSG}}, \Omega_{\mathrm{FSG}})$.

Constructional error in Seth's construction Seth's construction [13] is very similar to the above one. However, in the transitions introduced for popping a symbol and determining a winner of the play, the guarantees had to equal the current prediction in the position, i.e. given a transition rule $(q, s, r, q') \in \delta_{pop}$, the following moves were introduced

- $(\mathrm{Check}, q, c, d, \overline{\gamma}[s/r], \overline{\mathcal{P}}, \overline{m}) \to$ Evewin if $\overline{\mathcal{P}}' \in \overline{\mathcal{P}}_r$
- $(\mathrm{Check}, q, c, d, \overline{\gamma}[s/r], \overline{\mathcal{P}}, \overline{m}) \to$ Anawin if $\overline{\mathcal{P}}' \notin \overline{\mathcal{P}}_r$

where

$$\begin{aligned}
&\overline{\mathcal{P}}' = \overline{\varnothing}[(q', c', \overline{\gamma}, \overline{m})/r] && \text{if } c' = k \text{ or } c' = 2 \\
&\overline{\mathcal{P}}'_j = \overline{\mathcal{P}}_j, \quad \overline{\mathcal{P}}'_r = (q', c', \overline{\gamma}, \overline{m}) && \text{for all } j \neq r, \text{ if } 2 \neq c' < k
\end{aligned}$$

The constructional difference is $\overline{\mathcal{P}}'_j = \overline{\mathcal{P}}_j$ instead of $\overline{\mathcal{P}}'_j \subseteq \overline{\mathcal{P}}_j$. In example 21, we can see that Eve has no way to achieve this equality in that particular multi-pushdown game. Even though Eve possesses a winning strategy in the multi-pushdown game (See example 12), Ana would possess a winning strategy in the resulting finite state game.

3.2.2 Strategy automaton

To transport a strategy σ of FSG to G_P, we introduce a strategy automaton $\mathbb{S}_\sigma$. Intuitivly, this is an multi-pushdown system with the same number of stacks as P. At any point in the game, the stack heights of $\mathbb{S}_\sigma$ are identical to the stack heights of P. However, we will not use the transitions of a common mpds. Instead, each transition will change all the top of each stack at once. Also, the stack alphabet of $\mathbb{S}_\sigma$ is $\Gamma \times 2^{\mathbb{P}} \times [1..max]$ and has another state space.

The strategy automaton remembers information of the finite state game in the following sense. If the finite state game would be in a position $(\mathrm{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})$, then the strategy automaton is in state (q, c, d) and the top of stack tuple of each stack r is $(\overline{\gamma}_r, \overline{\mathcal{P}}_r, \overline{m}_r)$. At the same time, it mimics moves of G_P. The automaton can use the additional stack information to choose the next move by using σ from the position $(\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})$ if it belongs to Eve. Otherwise, it just mimics the move made by Ana. Below is listed, how $\mathbb{S}_\sigma$ updates upon a move in G_P. This is independent to whichever player makes the move. Be aware, that the configuration of $\mathbb{S}_\sigma$ can be used to derive a move for Eve, when they possess a strategy in FSG.

Definition 22. Given a strategy σ for Eve in FSG, the *strategy automaton* $\mathbb{S}_\sigma$ is a tuple $\mathbb{S}_\sigma = (Q \times [0..k] \times [0..n], \Gamma \times 2^{\mathbb{P}} \times [1..max], \mapsto, n)$ where

- $Q \times [0..k] \times [0..n]$ is the state space,
- $\Gamma \times 2^{\mathbb{P}} \times [1..max]$ is the stack alphabet and
- $\mapsto$ is a transition relation, that is defined on the set of configurations.

A configuration of $\mathbb{S}_\sigma$ is $((q, c, d), \overline{\mathcal{R}})$, where $\overline{\mathcal{R}} : [1..n] \to (\Gamma \times 2^{\mathbb{P}} \times [1..max])^*$ are the stack contents. For each stack j,

$$\overline{\mathcal{R}}_j = {}^j\mathcal{R}_{|\overline{\mathcal{R}}_j|}\, {}^j\mathcal{R}_{|\overline{\mathcal{R}}_j|-1} \dots {}^j\mathcal{R}_1,$$

where ${}^j\mathcal{R}_{|\overline{\mathcal{R}}_j|}$ is the top of stack symbol. Since each stack symbol ${}^j\mathcal{R}_i$ is a tuple, we denote them by ${}^j\mathcal{R}_i = ({}^j\gamma_i, {}^j\mathcal{P}_i, {}^jm_i)$.

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:

$$\begin{aligned}\overline{\mathcal{R}}_j &= {}^j\mathcal{R}_{|\overline{\mathcal{R}}_j|}\, {}^j\mathcal{R}_{|\overline{\mathcal{R}}_j|-1} \dots {}^j\mathcal{R}_1\\ &= {}^j\mathcal{R}_{\uparrow}\, {}^j\mathcal{R}_{\uparrow-1} \dots {}^j\mathcal{R}_1.\end{aligned}$$

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

$${}^j\gamma_{\uparrow} = {}^j\gamma_{|\overline{\mathcal{R}}_j|} \qquad {}^j\mathcal{P}_{\uparrow} = {}^j\mathcal{P}_{|\overline{\mathcal{R}}_j|} \qquad {}^jm_{\uparrow} = {}^jm_{|\overline{\mathcal{R}}_j|}.$$

We call the set of all configurations $C_{\mathbb{S}_\sigma}$.

Definition 23. We define a function $F : C_{\mathbb{S}_\sigma} \to V_{\text{FSG}}$ mapping configurations of $\mathbb{S}_\sigma$ to positions in FSG.

$$F(((q, c, d), \mathcal{R})) = (\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})$$

where $\overline{\gamma}_j = {}^j\gamma_{\uparrow}$, $\overline{\mathcal{P}}_j = {}^j\mathcal{P}_{\uparrow}$ and $\overline{m}_j = {}^jm_{\uparrow}$.

Definition 24. We define the transition relation $((q, c, d), \overline{\mathcal{R}}) \overset{\tau}{\mapsto} ((q', c', r), \overline{\mathcal{R}}')$, where

$$c' = \begin{cases} c & d = r \\ c+1 & d \neq r \end{cases} \qquad \text{and} \qquad \overline{\gamma}'_j = {}^j\gamma'_{\uparrow}.$$

Let $F(((q, c, d), \overline{\mathcal{R}})) = (\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})$. A transition is only introduced, if $c' \leq k$ and one of the following is true.

1. $\tau = (q, r, q') \in \delta_{int}$ and each of the following
   a) $own(q) = \text{Ana}$ or $\sigma((\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})) = (\text{Check}, q', c', r, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}')$
   b) $\overline{\mathcal{R}} = \overline{\mathcal{R}}'$, except
      - for each stack j, ${}^j m'_{\uparrow} = \max\{\Omega(q'), {}^j m_{\uparrow}\}$.
2. $\tau = (q, r, s, q') \in \delta_{push}$ and each of the following
   a) $own(q) = \text{Ana}$ or $\sigma((\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})) = (\text{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s)$
   b) $\overline{\mathcal{R}} = \overline{\mathcal{R}}'$, except
      - for each stack $j \neq r$, ${}^j m'_{\uparrow} = \max\{\Omega(q'), {}^j m_{\uparrow}\}$,
      - $\overline{\mathcal{R}}'_r = (s, \mathcal{P}, \Omega(q'))\overline{\mathcal{R}}_r$, where

$$\sigma((\text{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s)) = (\text{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}).$$

3. $\tau = (q, s, r, q') \in \delta_{pop}$ and each of the following
   a) $own(q) = \text{Ana}$ or $\sigma((\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})) = \text{Evewin}$
   b) there is $\overline{\mathcal{P}} \in {}^r\mathcal{P}_{\uparrow}$ with $\overline{\mathcal{P}}_r = (q', c', \overline{\gamma}', \overline{m})$ such that for each stack $j \neq r$,
      - $c' \neq 2$ and $\overline{\mathcal{P}}_j \subseteq {}^j\mathcal{P}_{\uparrow}$ or
      - $c' = 2$ and $\overline{\mathcal{P}}_j = \varnothing$
   c) $\overline{\mathcal{R}} = \overline{\mathcal{R}}'$, except
      - for each $j \neq r$,
        - ${}^j m'_{\uparrow} = \max\{\Omega(q'), {}^j m_{\uparrow}\}$
        - ${}^j\mathcal{P}'_{\uparrow} = \begin{cases} \overline{\mathcal{P}}_j & c' \neq 2 \\ {}^j\mathcal{P}_{\uparrow} & c' = 2 \end{cases}$
      - $\overline{\mathcal{R}}'_r = ({}^r\gamma_{\uparrow-1}, {}^r\mathcal{P}_{\uparrow-1}, {}^r m'_{\uparrow}){}^r\mathcal{R}_{\uparrow-2}{}^r\mathcal{R}_{\uparrow-3} \ldots {}^r\mathcal{R}_1$ where ${}^r m'_{\uparrow} = \max\{{}^r m_{\uparrow}, {}^r m_{\uparrow-1}, \Omega(q')\}$.
4. $\tau = (q, s, r, q') \in \delta_{pop}$ and each of the following
   a) $own(q) = \text{Ana}$ or $\sigma((\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})) = \text{Anawin}$
   b) there is **no** $\overline{\mathcal{P}} \in {}^r\mathcal{P}_{\uparrow}$ with $\overline{\mathcal{P}}_r = (q', c', \overline{\gamma}', \overline{m})$ such that for each stack $j \neq r$,
      - $c' \neq 2$ and $\overline{\mathcal{P}}_j \subseteq {}^j\mathcal{P}_{\uparrow}$ or
      - $c' = 2$ and $\overline{\mathcal{P}}_j = \varnothing$
   c) $\overline{\mathcal{R}} = \overline{\mathcal{R}}'$, except
      - for each $j \neq r$, ${}^j m'_{\uparrow} = \max\{\Omega(q'), {}^j m_{\uparrow}\}$

- $\overline{\mathcal{R}}'_r = ({}^r\gamma_{\uparrow-1}, {}^r\mathcal{P}_{\uparrow-1}, {}^r{m'}_{\uparrow}){}^r\mathcal{R}_{\uparrow-2}{}^r\mathcal{R}_{\uparrow-3}\dots{}^r\mathcal{R}_1$
  where ${}^r{m'}_{\uparrow} = \max\{{}^rm_{\uparrow}, {}^rm_{\uparrow-1}, \Omega(q')\}$.

The strategy automaton is an enhanced version of the multi-pushdown system P. As such, its stack contents share the same height and only the elements in the top of stack tuples are changed during a transition.

Lemma 25. *Let* $\eta = ((q,c,d),\overline{\mathcal{R}})$ *and* $\eta' = ((q',c',d'),\overline{\mathcal{R}}')$ *with* $\eta \overset{\tau}{\mapsto} \eta'$ *in* $\mathbb{S}_\sigma$. *For each stack* $j \in [1..n]$, *let* $\mathrm{sh}_j = \min\{|\overline{\mathcal{R}}_j|, |\overline{\mathcal{R}}'_j|\}$.
For each stack $j \in [1..n]$, ${}^j\mathcal{P}'_{\mathrm{sh}_j} \subseteq {}^j\mathcal{P}_{\mathrm{sh}_j}$ *and for each* $u \in [1..\mathrm{sh}_j - 1]$, ${}^j\mathcal{R}_u = {}^j\mathcal{R}'_u$.

Proof. By the construction of $\mathbb{S}_\sigma$, for all stacks $j \in [1..n]$, for each $u \in [1..\mathrm{sh}_j - 1]$, ${}^j\mathcal{R}_u = {}^j\mathcal{R}'_u$.

Case 1 ($\tau = (q, r, q') \in \delta_{int}$)*:* By construction, for each stack j, $\mathrm{sh}_j = |\overline{\mathcal{R}}_j| = |\overline{\mathcal{R}}'_j|$ and ${}^j\mathcal{P}_{\mathrm{sh}_j} = {}^j\mathcal{P}'_{\mathrm{sh}_j}$.

Case 2 ($\tau = (q, r, s, q') \in \delta_{push}$)*:* By construction for each stack $j \neq r$, $\mathrm{sh}_j = |\overline{\mathcal{R}}_j| = |\overline{\mathcal{R}}'_j|$ and ${}^j\mathcal{P}_{\mathrm{sh}_j} = {}^j\mathcal{P}'_{\mathrm{sh}_j}$.
For stack r, $\mathrm{sh}_r = |\overline{\mathcal{R}}_r| = |\overline{\mathcal{R}}'_r| - 1$ and ${}^r\mathcal{R}_{\mathrm{sh}_r} = {}^r\mathcal{R}'_{\mathrm{sh}_r}$.

Case 3 ($\tau = (q, s, r, q') \in \delta_{pop}$)*:* For stack r, $\mathrm{sh}_r = |\overline{\mathcal{R}}_r| - 1 = |\overline{\mathcal{R}}'_r|$ and ${}^r\mathcal{P}'_{\mathrm{sh}_r} = {}^r\mathcal{P}_{\mathrm{sh}_r}$.
Case 3.1 (the transition is by condition 3 of $\mathbb{S}_\sigma$)): By construction for each stack $j \neq r$, $\mathrm{sh}_j = |\overline{\mathcal{R}}_j| = |\overline{\mathcal{R}}'_j|$. If $c' = 2$, ${}^j\mathcal{P}'_{\mathrm{sh}_j} = {}^j\mathcal{P}_{\mathrm{sh}_j}$. Otherwise, ${}^j\mathcal{P}'_{\mathrm{sh}_j} \subseteq {}^j\mathcal{P}_{\mathrm{sh}_j}$.
Case 3.2 (the transition is by condition 4 of $\mathbb{S}_\sigma$)): By construction for each stack $j \neq r$, $\mathrm{sh}_j = |\overline{\mathcal{R}}_j| = |\overline{\mathcal{R}}'_j|$ and ${}^j\mathcal{P}'_{\mathrm{sh}_j} = {}^j\mathcal{P}_{\mathrm{sh}_j}$.

□

A run of the strategy automaton is a sequence $((q_{\mathrm{init}}, 0, 0), (\bot, \varnothing, \Omega(q_{init})^n) = \eta_0 \overset{\tau_0}{\longmapsto} \eta_1 \overset{\tau_1}{\longmapsto} \dots$. We can use the strategy automaton to define a strategy ν on (P, Ω) for starting positions of the form $(q_{\mathrm{init}}, \bot^n)$. We define ν inductivly on the play prefix. As $\mathbb{S}_\sigma$ is an enhanced version of the multi-pushdown system P of the game G_P, we require the invariants stated by the next lemma that hold between the play π of G_P compliant with ν and the run η of $\mathbb{S}_\sigma$.

Lemma 26. *At any position* $p \in \mathbb{N}$, *if* $\eta_p = ((q,c,d),\mathcal{R})$, *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. π_p *is in context* c *of stack* d.
3. - *If* $\mathrm{lup}^\pi_j(p)$ *is defined,* $\max_{u \in [\mathrm{lup}^\pi_j(p)..p]}\{\Omega(q^u)\} = {}^jm_\uparrow$.
   - *If* $\mathrm{lup}^\pi_j(p)$ *is undefined,* $\max_{u \in [0..p]}\{\Omega(q^u)\} = {}^jm_\uparrow$.

Proof.

Base Case (π_0, η_0). $\pi_0 = (q_{\text{init}}, \bot^n)$, $\eta_0 = ((q_{\text{init}}, 0, 0), (\bot, \varnothing, \Omega(q_{init}))^n)$.

Inductive Case (π_i to π_{i+1}, η_i to η_{i+1}). By induction, there is a play prefix $\pi_0 \xmapsto{\tau_0} \dots \xmapsto{\tau_{i-1}} \pi_i = (q, \mathcal{S})$ of (P, Ω). The strategy automaton has a run prefix $\eta_0 \xmapsto{\tau_1} \dots \xmapsto{\tau_{i-1}} \eta_i = ((q, c, d), \mathcal{R})$, where c, d and $\mathcal{R}$ are as by the lemma.

In case of $own(\eta_i) =$ Eve, all transitions enabled in η_i belong to a single transition $\pi_i \xmapsto{\tau_i} \pi_{i+1}$. We extend ν by setting $\nu(\pi_0 \dots \pi_i) = \pi_{i+1}$.

In the other case, for every transition τ_i enabled in π_i, the same is enabled in $\mathbb{S}_\sigma$. Let $\pi_i \xmapsto{\tau_i} \pi_{i+1}$ for some τ_i enabled in η_i. We continue η as described by its definition $\eta_i = ((q, c, d), \mathcal{R}) \xmapsto{\tau_i} ((q', c', d'), \mathcal{R}') = \eta_{i+1}$.

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

The second condition is fulfilled by construction of $\mathbb{S}_\sigma$: By induction, π_i is in context c of stack d. If $\tau_i \in \delta_d$, then $c' = c$ and $d' = d$. This is compliant with the lemma as π_{i+1} is in the same context of the same stack. Otherwise, $c' = c + 1$ and $d' = r$ where $\tau_i \in \delta_r$, which again fits context and stack of position π_{i+1}.

For all stacks $j \neq r$, the third condition is also fulfilled by construction. By induction the maximal parity seen since position $\text{lup}^\pi_j(i)$ 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 ${}^j m'_\uparrow$ appropriatly 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 thrid 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, $\overline{\mathcal{R}}'_r = (s, \mathcal{P}, \Omega(q'))\overline{\mathcal{R}}_r$, such that s is the new additional symbol, meeting 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 $\mathbb{S}_\sigma$ contains multiple possible transitions for τ_i. These only differ in the prediction sets ${}^j\mathcal{P}_\uparrow$ for each stack j. The lemma does not state any conditions on the prediction 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, $(s, \mathcal{P}, \Omega(q'))\overline{\mathcal{R}}'_r = \overline{\mathcal{R}}_r$, removing the top most symbol from the stack, meeting the stack symbols of $\mathcal{S}_r$.

For the third 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 $\text{lup}^\pi_r(i + 1) = \text{lup}^\pi_r(\text{lup}^\pi_r\, i)$.

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

$\square$

The next lemma ensures that the strategy automaton $\mathbb{S}_\sigma$ and the mapping F of configurations from $\mathbb{S}_\sigma$ to positions in FSG 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 $F(\eta) \mapsto \cdots \mapsto F(\eta')$ compliant with σ in FSG.

Lemma 27. *Let σ be a strategy for Eve in FSG and $\mathbb{S}_\sigma$ the strategy automaton. Let $\eta = \eta_0 \overset{\tau_0}{\longmapsto} \eta_1 \overset{\tau_1}{\longmapsto} \dots$ be a computation of $\mathbb{S}_\sigma$ starting in a stair η_0.*

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

$$\eta_i = ((q^i, c^i, d^i), \overline{\mathcal{R}}^i) \qquad\qquad F(\eta_i) = (\mathit{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i)$$

where for each stack j, $\overline{\mathcal{R}}^i_j = ({}^j\gamma^i_\uparrow, {}^j\mathcal{P}^i_\uparrow, {}^jm^i_\uparrow)({}^j\gamma^i_{\uparrow-1}, {}^j\mathcal{P}^i_{\uparrow-1}, {}^jm^i_{\uparrow-1}) \dots ({}^j\gamma^i_1, {}^j\mathcal{P}^i_1, {}^jm^i_1)$.

Let $i \in \mathbb{N}$. Let $\eta_i \overset{\tau_i}{\longmapsto} \eta_{i+1}$ be a transition of $\mathbb{S}_\sigma$ that is not by condition 4.

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

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

$$\begin{aligned} F(\eta_i) = (\mathit{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i) \mapsto& \\ &(\mathit{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^{i+1}) = F(\eta_{i+1}) \end{aligned}$$

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

$$\begin{aligned} F(\eta_i) &= (\mathit{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i) \\ &\mapsto (\mathit{Push}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s) \\ &\mapsto (\mathit{Claim}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s, {}^j\mathcal{P}^{i+1}_\uparrow) \\ &\mapsto (\mathit{Check}, q^{i+1}, c^{i+1}, r, \overline{\gamma}^i[s/r], \overline{\mathcal{P}}^i[{}^j\mathcal{P}^{i+1}_\uparrow/r], \overline{m}^{i+1}) = F(\eta_{i+1}) \end{aligned}$$

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

$$\begin{aligned} F(\eta_t) &= (\mathit{Check}, q^t, c^t, d^t, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t) \\ &\mapsto (\mathit{Push}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s) \\ &\mapsto (\mathit{Claim}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s, {}^r\mathcal{P}^{t+1}_\uparrow) \\ &\mapsto (\mathit{Jump}_r, q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^i, t^t_r) \\ &\mapsto (\mathit{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^{i+1}) = F(\eta_{i+1}) \end{aligned}$$

Proof. For any of the following, the correctness of q^{i+1}, c^{i+1} and d^{i+1} is immediate and therefore skipped. Also be aware that both, FSG and $\mathbb{S}_\sigma$, only introduce transitions if $c^{i+1} \leq k$.

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

$$\begin{aligned} F(\eta_i) = (\text{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i) &\mapsto (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}') \\ &\overset{!}{=} (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^{i+1}) = F(\eta_{i+1}) \end{aligned}$$

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

$$\overline{m}_j^{i+1} = {}^j m_\uparrow^{i+1} = \max\{{}^j m_\uparrow^i, \Omega(q^{i+1})\} = \max\{\overline{m}_j^i, \Omega(q^{i+1})\} = \overline{m}'_j.$$

Also, by construction of $\mathbb{S}_\sigma$, the transition $\eta_i \xmapsto{\tau_i} \eta_{i+1}$ only exists, if $own(q^i) = \text{Ana}$ or $\sigma((\text{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i)) = (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}')$. The transition is compliant with σ.

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

$$\begin{aligned} F(\eta_i) &= (\text{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i) \\ &\mapsto (\text{Push}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s) \\ &\mapsto (\text{Claim}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s, {}^r\mathcal{P}_\uparrow^{i+1}) \\ &\mapsto (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^i[s/r], \overline{\mathcal{P}}^i[{}^r\mathcal{P}_\uparrow^{i+1}/r], \overline{m}') \\ &\overset{!}{=} (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^{i+1}) = F(\eta_{i+1}) \end{aligned}$$

Remains to show the equality of the last two check states. By definition of $\mathbb{S}_\sigma$, $\overline{\mathcal{R}}^i = \overline{\mathcal{R}}^{i+1}$, except

- for each $j \neq r$, ${}^j m_\uparrow^{i+1} = \max\{\Omega(q^{i+1}), {}^j m_\uparrow^i\}$,
- $\overline{\mathcal{R}}_r^{i+1} = (s, {}^r\mathcal{P}_\uparrow^{i+1}, \Omega(q^{i+1}))\overline{\mathcal{R}}_r$, where

$$\sigma((\text{Push}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s)) = (\text{Claim}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s, {}^r\mathcal{P}_\uparrow^{i+1}).$$

This immediatly yields $\overline{\gamma}^i[s/r] = \overline{\gamma}^{i+1}$ and $\overline{\mathcal{P}}^i[{}^r\mathcal{P}_\uparrow^{i+1}/r] = \overline{\mathcal{P}}^{i+1}$.

Also, $\overline{m}'_r = \Omega(q^{i+1}) = {}^r m_\uparrow^{i+1}$ and for each stack $j \neq r$:

$$m' = \max\{\Omega(q^{i+1}), \overline{m}_j^i\} = \max\{\Omega(q^{i+1}), {}^j m_\uparrow^i\} = {}^j m_\uparrow^{i+1}.$$

Since the transition $\eta_i \xmapsto{\tau_i} \eta_{i+1}$ exists in $\mathbb{S}_\sigma$, by its construction,

$$\sigma((\text{Push}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s)) = (\text{Claim}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s, {}^j\mathcal{P}_\uparrow^{i+1}).$$

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

$$\sigma((\text{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i)) = (\text{Push}_r, c^{i+1}, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i, q^{i+1}, s).$$

Thus, the transitions exist in FSG 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 FSG, τ_t causes the existence of the transitions

$$\begin{aligned} F(\eta_t) &= (\text{Check}, q^t, c^t, d^t, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t) \\ &\mapsto (\text{Push}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s) \\ &\mapsto (\text{Claim}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s, {}^r\mathcal{P}_{\uparrow}^{t+1}) \end{aligned}$$

which are compliant with σ, as discussed in the previous case. Since (t, i) is a push-pop-pair, for any position $t < p < i$, $|\overline{\mathcal{R}}_r^{t+1}| = |\overline{\mathcal{R}}_r^i| \leq |\overline{\mathcal{R}}_r^p|$. Then, by repetitive use of Lemma 25, ${}^r\mathcal{P}_{\uparrow}^i \subseteq {}^r\mathcal{P}_{\uparrow}^{t+1}$.

Next, we show that

$$(\text{Claim}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s, {}^r\mathcal{P}_{\uparrow}^{t+1}) \mapsto (\text{Jump}_r, q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^i, t_r^t)$$

is a valid transition in FSG. Since the transition $\eta_i \xrightarrow{\tau_i} \eta_{i+1}$ is not by condition 4, it must be by condition 3 of $\mathbb{S}_\sigma$. So either

- $c^{i+1} = 2$ and $\varnothing[(q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{m}^i)/r] \in \overline{\mathcal{P}}_r^i$ or
- $c^{i+1} \neq 2$ and $\mathbb{S}_\sigma$ finds a $\overline{\mathcal{P}}[(q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{m}^i)/r] \in \overline{\mathcal{P}}_r^i$ such that for each stack $j \neq r$, $\overline{\mathcal{P}}_j \subseteq \overline{\mathcal{P}}_j^i$.

By construction of FSG, there is the transition

$$\begin{aligned} &(\text{Claim}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s, {}^r\mathcal{P}_{\uparrow}^{t+1}) \\ &\quad\mapsto \begin{cases} (\text{Jump}_r, q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}[\overline{\gamma}_r^t/r], \overline{\mathcal{P}}[\overline{\mathcal{P}}_r^t/r], \overline{m}^i, \overline{m}_r^t) & c^{i+1} \neq 2 \\ (\text{Jump}_r, q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}[\overline{\gamma}_r^t/r], \overline{\mathcal{P}}^t, \overline{m}^i, \overline{m}_r^t) & c^{i+1} = 2 \end{cases} \\ &\quad\quad\quad \overset{!}{=} (\text{Jump}_r, q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^i, t_r^t) \end{aligned}$$

To find the last equation, we need to identify $\overline{\gamma}^{i+1}[\overline{\gamma}_r^t/r] = \overline{\gamma}^{i+1}$ and

- if $c^{i+1} = 2$, $\overline{\mathcal{P}}^t = \overline{\mathcal{P}}^{i+1}$ or
- if $c^{i+1} \neq 2$, $\overline{\mathcal{P}}[\overline{\mathcal{P}}_r^t/r] = \overline{\mathcal{P}}^{i+1}$.

Remember that $\tau_i = (q^i, s, r, q^{i+1})$ is a popping transition with

$$\eta_i = ((q^i, c^i, d^i), \overline{\mathcal{R}}^i) \xrightarrow{\tau_i} ((q^{i+1}, c^{i+1}, d^{i+1}), \overline{\mathcal{R}}^{i+1}) = \eta_{i+1},$$

that used condition 3 of the strategy automaton conditions for a transition with the prediction $\overline{\mathcal{P}} \in {}^r\mathcal{P}_{\uparrow}^i = \overline{\mathcal{P}}_r^i$. Since (t, i) is a push-pop-pair, for all positions $t+1 \leq p \leq i$,

$$|\overline{\mathcal{R}}_r^t| = |\overline{\mathcal{R}}_r^{t+1}| - 1 = |\overline{\mathcal{R}}_r^i| - 1 = |\overline{\mathcal{R}}_r^{i+1}| < |\overline{\mathcal{R}}_r^p|.$$

As $\mathbb{S}_\sigma$ does not change the symbol of its tuples and by repetitive use of Lemma 25,

$$\overline{\gamma}_r^{i+1} = {}^r\gamma_\uparrow^{i+1} = {}^r\gamma_\uparrow^t = \overline{\gamma}_r^t \qquad \text{and} \qquad \overline{\mathcal{P}}_r^t = {}^r\mathcal{P}_\uparrow^t = {}^r\mathcal{P}_{\uparrow-1}^{t+1} = {}^r\mathcal{P}_{\uparrow-1}^i = {}^r\mathcal{P}_\uparrow^{i+1} = \overline{\mathcal{P}}_r^{i+1}.$$

Since $\mathbb{S}_\sigma$ used $\overline{\mathcal{P}}$ for its condition 3 transition, by its construction: For all stacks $j \neq r$, $\overline{\mathcal{P}}_j^{i+1} = {}^j\mathcal{P}_\uparrow^{i+1} = \overline{\mathcal{P}}_j$. This completes the case of $c^{i+1} \neq 2$.

For the case of $c^{i+1} = 2$, it remains to show that for each stack $j \neq r$, $\overline{\mathcal{P}}_j^t = \overline{\mathcal{P}}_j^{i+1}$. Observe, that the push-pop-pair (t, i) is within one context, namely $c^t = c^{t+1} = \cdots = c^i = c^{i+1} = 2$ and $d^t = d^{t+1} = \cdots = d^i = d^{i+1} = r$ (If the symbol was pushed in the first context, the second context would be another stack). By construction of $\mathbb{S}_\sigma$, during the computation $\eta_t \mapsto \cdots \mapsto \eta_{i+1}$ in context 2, the top of stack prediction sets for all stacks $j \neq r$ do not change:

$$\overline{\mathcal{P}}_j^t = {}^j\mathcal{P}_\uparrow^t = {}^j\mathcal{P}_\uparrow^{t+1} = \cdots = {}^j\mathcal{P}_\uparrow^i = {}^j\mathcal{P}_\uparrow^{i+1} = \overline{\mathcal{P}}_j^{i+1}.$$

By construction of FSG, the last transition also exists:

$$\begin{aligned}(\text{Jump}_r, q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^i, t_r^t) &\mapsto (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}') \\ &\overset{!}{=} (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^{i+1}) = F(\eta_{i+1})\end{aligned}$$

Remains to show $\overline{m}' = \overline{m}^{i+1}$. For any stack $j \neq r$, by construction of FSG and $\mathbb{S}_\sigma$,

$$\overline{m}_j^{i+1} = {}^jm_\uparrow^{i+1} = \max\{{}^jm_\uparrow^i, \Omega(q^{i+1})\} = \max\{\overline{m}_j^i, \Omega(q^{i+1})\} = \overline{m}_j'.$$

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

$$|\overline{\mathcal{R}}_r^t| = |\overline{\mathcal{R}}_r^{t+1}| - 1 = |\overline{\mathcal{R}}_r^i| - 1 = |\overline{\mathcal{R}}_r^{i+1}| < |\overline{\mathcal{R}}_r^p|.$$

Repetitive use of Lemma 25 leads to ${}^rm_\uparrow^t = {}^rm_{\uparrow-1}^{t+1} = {}^rm_{\uparrow-1}^i$. Finally,

$$\begin{aligned}\overline{m}_r^{i+1} &= \max\{{}^rm_\uparrow^i, {}^rm_{\uparrow-1}^i, \Omega(q^{i+1})\} \\ &= \max\{{}^rm_\uparrow^i, {}^rm_\uparrow^t, \Omega(q^{i+1})\} = \max\{\overline{m}_r^i, \overline{m}_r^t, \Omega(q^{i+1})\} = \overline{m}_r'.\end{aligned}$$

The states

$$(\text{Claim}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s, {}^r\mathcal{P}_\uparrow^i), (\text{Jump}_r, q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^i, t_r^t)$$

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

□

3.2.3 Correctness

With the strategy automaton we have the tools at hand to show the equivalence of G_P and FSG for starting positions of G_P with empty stacks.

Theorem 28. *Eve possesses a winning strategy from $(q_{init}, \bot^n)$ in G_P if and only if they possess a winning strategy from $(\mathit{Check}, q_{init}, 0, 0, \bot^n, \varnothing^n, 0^n)$ in FSG.*

We show this in both directions. First we assume the existence of a winning strategy for Eve in FSG and use the strategy automaton to find a winning strategy for Eve in G_P.

Then we assume the existence of a winning strategy for Eve in G_P and show how to construct a winning strategy for Eve in FSG.

Transferring a winning strategy from FSG to G_P

Let σ be a winning strategy for Eve in FSG from $(\text{Check}, q_{init}, 0, 0, \bot^n, \varnothing^n, 0^n)$. We use $\mathbb{S}_\sigma$ to derive a strategy ν for Eve in G_P as described above (Lemma 26). 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 FSG compliant with σ from $\rho_0 = (\text{Check}, q_{init}, 0, 0, \bot^n, \varnothing^n, 0^n)$ that is losing for Eve.

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

$$\eta_i = ((q^i, c^i, d^i), \overline{\mathcal{R}}^i), \qquad F(\eta_i) = (\text{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i).$$

where for each stack j, $\overline{\mathcal{R}}^i_j = ({}^j\gamma^i_\uparrow, {}^j\mathcal{P}^i_\uparrow, {}^jm^i_\uparrow)({}^j\gamma^i_{\uparrow-1}, {}^j\mathcal{P}^i_{\uparrow-1}, {}^jm^i_{\uparrow-1}) \dots ({}^j\gamma^i_1, {}^j\mathcal{P}^i_1, {}^jm^i_1)$.

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 the 4th condition of $\mathbb{S}_\sigma$ for having a transition. We handle that case later.

Lemma 29. *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 $p \le i$, there is a play $\rho^i = \rho^i_1 \mapsto \dots \mapsto \rho^i_{\psi(i)}$ of length $\psi(i)$ compliant with σ in FSG from $\rho^i_1 = F(\eta_p)$ to $\rho^i_{\psi(i)} = F(\eta_i)$ 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, $F(\eta_p) = F(\eta_i)$ and $\Omega(q^p) = \Omega(F(\eta_p))$.

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

We create a play in FSG from $F(\eta_p)$ to $F(\eta_{i+1})$ 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 27, the following transition is compliant with σ.

$$F(\eta_i) = (\text{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i) \mapsto \\ (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}^{i+1}) = F(\eta_{i+1})$$

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 $F(\eta_i)$. By Lemma 27, the following transitions are compliant with σ.

$$\begin{aligned}F(\eta_i) &= (\text{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i) \\ &\mapsto (\text{Push}_r, c^{i+1}, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q^{i+1}, s) \\ &\mapsto (\text{Claim}_r, c^{i+1}, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q^{i+1}, s, \mathcal{P}^{i+1}_{r,1}) \\ &\mapsto (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}[s/r], \overline{\mathcal{P}}[\mathcal{P}^{i+1}_{r,1}/r], \overline{m}^{i+1}) = F(\eta_{i+1})\end{aligned}$$

By induction,

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

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 $p \le t < i$. 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 condition 4 of $\mathbb{S}_\sigma$'s transition conditions, by Lemma 27, the following transitions are compliant with σ.

$$\begin{aligned}F(\eta_t) &= (\text{Check}, q^t, c^t, d^t, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t) \\ &\mapsto (\text{Push}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s) \\ &\mapsto (\text{Claim}_r, c^{t+1}, \overline{\gamma}^t, \overline{\mathcal{P}}^t, \overline{m}^t, q^{t+1}, s, {}^r\mathcal{P}^{t+1}_{\uparrow}) \\ &\mapsto (\text{Jump}_r, q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^i, \overline{m}^t_r) \\ &\mapsto (\text{Check}, q^{i+1}, c^{i+1}, d^{i+1}, \overline{\gamma}^{i+1}, \overline{\mathcal{P}}^{i+1}, \overline{m}^{i+1}) = F(\eta_{i+1})\end{aligned}$$

Since (t, i) is push-pop-pair, $\mathrm{lup}_r^\eta(i) = t$. By Lemma 26,

$$\overline{m}_r^i = \max_{u \in [\mathrm{lup}_r^\eta(i)..i]} \{\Omega(q^u)\} = \max_{u \in [t..i]} \{\Omega(q^u)\}.$$

By induction,

$$\begin{aligned}\max_{u \in [p..i+1]} \{\Omega(q^u)\} &= \max \left\{ \max_{u \in [p..t]} \{\Omega(q^u)\}, \max_{u \in [t..i]} \{\Omega(q^u)\}, \Omega(q^{i+1}) \right\} \\ &= \max \left\{ \max_{u \in [1..\psi(t)]} \{\Omega(\rho_u^t)\}, \overline{m}_r^i, \Omega(\rho_{\psi(i+1)}^t) \right\} \\ &= \max_{u \in [1..\psi(i+1)]} \{\Omega(\rho_u^{i+1})\}\end{aligned}$$

□

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 FSG that is winning for Ana and is compliant with σ. We need a function $\phi : \mathcal{ST} \to \mathbb{N}$, such that for any $p \in \mathcal{ST}$, $F(\eta_p) = \rho_{\phi(p)}$. 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 $\rho_0 = (\text{Check}, q_{init}, 0, 0, \perp^n, \varnothing^n, 0^n) = F(\eta_0)$, 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

$$\begin{aligned}\rho_0 = F(\eta_0) \mapsto \dots \mapsto \rho_{\phi(p_1)} = F(\eta_1) \mapsto \dots \mapsto \rho_{\phi(p_2)} = F(\eta_2) \mapsto \dots \\ \dots \mapsto \rho_{\phi(p_i)} = F(\eta_{p_i})\end{aligned}$$

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

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

Since η_{p_i} is a stair and $F(\eta_{p_i}) = \rho_{\phi(p_i)}$, by Lemma 29, we can find a position for $\phi(p_{i+1})$ and continue ρ by some transitions

$$F(\eta_{p_i}) = \rho_{\phi(p_i)} \mapsto \cdots \mapsto \rho_{\phi(p_{i+1})} = F(\eta_{p_{i+1}}),$$

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 condition 4 of the strategy automaton. Let there is a minimal position $i \in \mathbb{N}$, such that $\eta_i \xmapsto{\tau_i} \eta_{i+1}$ is due to condition 4 of $\mathbb{S}_\sigma$'s transition conditions. Be aware, that the induction in Lemma 29 still works up to position i. Thus, there is a play ρ compliant with σ from $(\text{Check}, q_{init}, 0, 0, \perp^n, \varnothing^n, 0^n)$, which is a stair, to $\psi(\eta_i)$. Since $\mathbb{S}_\sigma$ had a transition for τ_i, either $own(\psi(\eta_i)) = \text{Ana}$ or σ chose the transition introduced to FSG caused by τ_i. In either case, the following transition is compliant with σ:

$$\psi(\eta_i) = (\text{Check}, q^i, c^i, d^i, \overline{\gamma}^i, \overline{\mathcal{P}}^i, \overline{m}^i) \mapsto \text{Anawin}.$$

Because τ_i used condition 4, we know that there is no prediction $\overline{\mathcal{P}} \in \overline{\mathcal{P}}^i_r$, with $\overline{\mathcal{P}}_r = (q^{i+1}, c^{i+1}, \overline{\gamma}^{i+1}, \overline{m}^i)$ such that for each stack $j \neq r$,

- $c' \neq 2$ and $\overline{\mathcal{P}}_j \subseteq {}^j\mathcal{P}_\uparrow$ or
- $c' = 2$ and $\overline{\mathcal{P}}_j = \varnothing$.

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.

Transferring a winning strategy from G_P to FSG

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

Definition 30. Given a strategy ν for Eve in G_P, a play prefix $\pi_{..l} = \pi_0 \xmapsto{\tau_0} \dots \xmapsto{\tau_{l-1}} \pi_l$ 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 a game prediction set $\mathcal{P}^{\pi_{..l},\nu,p} \subseteq \mathbb{P}_r$ recursivly:
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 an element $\overline{\mathcal{P}}$ to $\mathcal{P}^{\pi_{..l},\nu,p}$:

Let c be the context of $\pi_{..l'}$ at position $t+1$, q'' be the state and $\overline{\gamma}$ be the top of stack symbols at π_{t+1}. For each stack $j \in [1..n]$, let $t_j = \text{lup}^j_{\pi_{..t}}(t)$. Let $\overline{m}$ be such that

$$\overline{m}_j = \begin{cases} \max_{u\in[t_j+1..t]}\{\Omega(\pi_u)\} & t_j \neq \perp \\ \max_{u\in[0..t]}\{\Omega(\pi_u)\} & t_j = \perp \end{cases}$$

We add $\overline{\mathcal{P}}$ to $\mathcal{P}^{\pi_{..l},\nu,p}$, where $\overline{\mathcal{P}}_r = (q'', c, \overline{\gamma}, \overline{m})$ and for each stack $j \neq r$,

$$\overline{\mathcal{P}}_j = \begin{cases} \varnothing & t_j = \bot \text{ or } c = 2 \\ \mathcal{P}^{\pi_{..t+1},\nu,t_j} & t_j \neq \bot \text{ and } c \neq 2 \end{cases}$$

Be aware, that this construction is finite and the result is an actual prediction: The sets $\mathcal{P}^{\pi_{..t+1},\nu,t_j}$ contain predictions with context greater than c by construction (there is no play continuation of $\pi_{..t+1}$, where a pop can occur in context c for any stack other than r). Furthermore, this construction is finite as $\mathbb{P}_i$ is finite.

For a play prefix $\pi_{..l}$ and its continuation $\pi_{..l'}$, i.e. $l < l'$, it is immediate that $\mathcal{P}^{\pi_{..l'},\nu,p} \subseteq \mathcal{P}^{\pi_{..l},\nu,p}$. 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 FSG compliant with σ, we maintain the play prefix of G_P. In order to keep the construction short, we define the strategy and an invariant (Lemma 31) 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 31. *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 $\rho_p = (Check, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})$, then $\pi_{\psi(p)}$ is in state q, in context c of stack d and the top of stack symbols are $\overline{\gamma}$. Let $t_j = \text{lup}_j^{\pi_{..\psi(p)}}(\psi(p))$. For every stack j with $t_j \neq \bot$, $\mathcal{P}^{\pi_{..\psi(p)},\nu,t_j} \subseteq \overline{\mathcal{P}}_j$.*
- *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. $\pi_0 = (q_{init}, \bot^n)$, $\rho_0 = (\text{Check}, q_{init}, 0, 0, \bot^n, \varnothing^n, \Omega(q_{init})^n)$. $\psi(p_1) = \psi(0) = 0$.

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 31.

If $own(\rho_{p_i}) = \text{Eve}$, we need to construct σ for $\rho_{p_i} = (\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})$. In that case, let $\pi_{\psi(p_i)} \overset{\tau}{\hookrightarrow} \nu(\pi_{\psi(p_i)})$ be the transition used by ν in G_P. If $own(\rho_{p_i}) = \text{Ana}$, Ana takes some transition in FSG that was introduced by some transition τ enabled in $\pi_{\psi(p_i)}$.

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

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

$$\rho_{p_i} = (\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}) \overset{\tau}{\mapsto} (\text{Check}, q', c', i, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}') = \rho_{p_{i+1}}$$

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

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

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

To the invariant:

- This is compliant with ν.
- State, context and stack conditions are fulfilled by construction, as well as the top of stack condition. The prediction sets did not change, thus

$$\mathcal{P}^{\pi_{..\psi(p_{i+1})},\nu,t_j} \subseteq \mathcal{P}^{\pi_{..\psi(p_i)},\nu,t_j} \subseteq \overline{\mathcal{P}}_j.$$

- for the parity condition, we get, that

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

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

$$\rho_{p_i} = (\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}) \mapsto (\text{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s)$$

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 FSG. To define their strategy, we use the game prediction from above.

$$(\text{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s) \mapsto (\text{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}^{\pi_{..\psi(p_i)+1},\nu,\psi(p_i)})$$

Case 2.1 (Ana continues to $(\text{Check}, q', c', r, \overline{\gamma}[s/r], \overline{\mathcal{P}}[\mathcal{P}^{\pi_{..\psi(p_i)+1},\nu,\psi(p_i)}/r], \overline{m}')$): The transitions in FSG up to p_{i+1} are

$$\begin{aligned}\rho_{p_i} &= (\text{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}) \\ &\mapsto (\text{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s) \\ &\mapsto (\text{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}^{\pi_{..\psi(p_i)+1},\nu,\psi(p_i)}) \\ &\mapsto (\text{Check}, q', c', r, \overline{\gamma}[s/r], \overline{\mathcal{P}}[\mathcal{P}^{\pi_{..\psi(p_i)+1},\nu,\psi(p_i)}/r], \overline{m}') = \rho_{p_{i+1}}\end{aligned}$$

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, context and stack 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 $t_j = \mathrm{lup}_j^{\pi_{..\psi(p_i)}}(\psi(p_i)) = \mathrm{lup}_j^{\pi_{..\psi(p_{i+1})}}(\psi(p_{i+1}))$, thus

  $$\mathcal{P}^{\pi_{..\psi(p_{i+1})},\nu,t_j} \subseteq \mathcal{P}^{\pi_{..\psi(p_i)},\nu,t_j} \subseteq \overline{\mathcal{P}}_j = \overline{\mathcal{P}}[\mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}/r]_j.$$

  For stack r, we have $\mathrm{lup}_j^{\pi_{..\psi(p_{i+1})}}(\psi(p_{i+1})) = \psi(p_i)$. Adequately,

  $$\mathcal{P}^{\pi_{..\psi(p_{i+1})},\nu,\psi(p_i)} = \overline{\mathcal{P}}[\mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}/r]_r.$$

- for the parity condition, we have

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

Case 2.2 (Ana continues to $(\mathrm{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}'[\overline{\mathcal{P}}_r/r], \overline{m}', \overline{m}_r)$, where $c'' \neq 2$): The transitions in FSG up to p_{i+1} are

$$\begin{aligned}\rho_{p_i} &= (\mathrm{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})\\ &\mapsto (\mathrm{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s)\\ &\mapsto (\mathrm{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)})\\ &\mapsto (\mathrm{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}'[\overline{\mathcal{P}}_r/r], \overline{m}', \overline{m}_r)\\ &\mapsto (\mathrm{Check}, q'', c'', r, \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}'[\overline{\mathcal{P}}_r/r], \overline{m}'') = \rho_{p_{i+1}}\end{aligned}$$

We set $p_{i+1} = p_i + 4$. Since $c'' \neq 2$ it must be that $\overline{\mathcal{P}}' \in \mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$ in order for

$$(\mathrm{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}) \mapsto (\mathrm{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}'[\overline{\mathcal{P}}_r/r], \overline{m}', \overline{m}_r)$$

to exist, where $\overline{\mathcal{P}}'_r = (q'', c'', \overline{\gamma}', \overline{m}')$. By construction of $\mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, 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 $\mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, this is compliant with ν.
- By construction of $\mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, π_{t+1} is in state q'' in context c'' of stack r with the top of stack symbols being $\overline{\gamma}'[\overline{\gamma}_r/r]$.

  Furthermore, for each stack $j \neq r$, since $\overline{\mathcal{P}}' \in \mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, with $t_j = \mathrm{lup}_j^{\pi_{..t}}(t)$,

  $$\overline{\mathcal{P}}'[\overline{\mathcal{P}}_r/r]_j = \overline{\mathcal{P}}'_j = \begin{cases}\mathcal{P}^{\pi_{..t},\nu,t_j} & t_j \neq \bot\\ \varnothing & t_j = \bot\end{cases}.$$

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

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

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

$$\mathcal{P}^{\pi_{..\psi(p_{i+1})},\nu,t_r} \subseteq \mathcal{P}^{\pi_{..\psi(p_i)},\nu,t_r} \subseteq \overline{\mathcal{P}}_r = \overline{\mathcal{P}}'[\overline{\mathcal{P}}_r/r]_j.$$

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

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

Together, we arrive at:

$$\begin{aligned}\max_{u\in[p_i..p_{i+1}]}\{\Omega(\rho_u)\} = \max\{&\rho_{p_i}, \rho_{p_{i+1}}, (\mathrm{Push}_r, \dots), (\mathrm{Claim}_r, \dots),\\ &(\mathrm{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}'[\overline{\mathcal{P}}_r/r], \overline{m}', \overline{m}_r)\}\\ = \max\{&\rho_{p_i}, \rho_{p_{i+1}}, \overline{m}_r\} = \max_{u\in[\psi(p_i)..\psi(p_{i+1})]}\{\Omega(\pi_u)\}.\end{aligned}$$

Case 2.3 (Ana continues to $(\mathrm{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}, \overline{m}', \overline{m}_r)$, where $c'' = 2$): The transitions in FSG up to p_{i+1} are

$$\begin{aligned}\rho_{p_i} &= (\mathrm{Check}, q, c, d, \overline{\gamma}, \overline{\mathcal{P}}, \overline{m})\\ &\mapsto (\mathrm{Push}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s)\\ &\mapsto (\mathrm{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)})\\ &\mapsto (\mathrm{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}, \overline{m}', \overline{m}_r)\\ &\mapsto (\mathrm{Check}, q'', c'', r, \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}, \overline{m}'') = \rho_{p_{i+1}}\end{aligned}$$

We get, that $p_{i+1} = p_i + 4$. Since $c'' = 2$ it must be that $\overline{\mathcal{P}}' \in \mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$ in order for

$$(\mathrm{Claim}_r, c', \overline{\gamma}, \overline{\mathcal{P}}, \overline{m}, q', s, \mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}) \mapsto (\mathrm{Jump}_r, q'', c'', \overline{\gamma}'[\overline{\gamma}_r/r], \overline{\mathcal{P}}, \overline{m}', \overline{m}_r)$$

to exist, where $\overline{\mathcal{P}}'_r = (q'', c'', \overline{\gamma}', \overline{m}')$ and for each other stack $j \neq r$, $\overline{\mathcal{P}}'_j = \varnothing$. Thus, by construction of $\mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, 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 $\psi(p_i) < t < l$. Be aware, that this play continuation does not change the context of the play, since a pop-transition in context 2 can only occur after a push transition on the active stack, which means, it was also pushed in context 2.

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

To the invariant:

- By construction of $\mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, this is compliant with ν.

- By construction of $\mathcal{P}^{\pi_{..\psi(p_i)},\nu,\psi(p_i)}$, π_{t+1} is in state q'' in context $c''=2$ of stack r with the top of stack symbols being $\overline{\gamma}'[\overline{\gamma}_r/r]$.

  Furthermore, by the fact that the context has not switched since $\pi_{\psi(p_i)}$ and that $(\psi(p_i),t)$ is push-pop-pair, for each stack $j\in[1..n]$, $t_j=\mathrm{lup}_j^{\pi_{..\psi(p_i)}}(\psi(p_i))=\mathrm{lup}_j^{\pi_{..\psi(p_{i+1})}}(\psi(p_{i+1}))$ and thus

$$\mathcal{P}^{\pi_{..\psi(p_{i+1})},\nu,t_j}\subseteq\mathcal{P}^{\pi_{..\psi(p_i)},\nu,t_j}\subseteq\overline{\mathcal{P}}_j.$$

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

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

  Together, we arrive at:

$$\begin{aligned}\max_{u\in[p_i..p_{i+1}]}\{\Omega(\rho_u)\}&=\max\{\rho_{p_i},\rho_{p_{i+1}},(\mathrm{Push}_r,\dots),(\mathrm{Claim}_r,\dots),\\&\qquad(\mathrm{Jump}_r,q'',c'',\overline{\gamma}'[\overline{\gamma}_r/r],\overline{\mathcal{P}},\overline{m}',\overline{m}_r)\}\\&=\max\{\rho_{p_i},\rho_{p_{i+1}},\overline{m}_r\}=\max_{u\in[\psi(p_i)..\psi(p_{i+1})]}\{\Omega(\pi_u)\}.\end{aligned}$$

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

$$\begin{aligned}\rho_{p_i}&=(\mathrm{Check},q,c,d,\overline{\gamma},\overline{\mathcal{P}},\overline{m})\mapsto\mathrm{Evewin}\quad\text{or}\\\rho_{p_i}&=(\mathrm{Check},q,c,d,\overline{\gamma},\overline{\mathcal{P}},\overline{m})\mapsto\mathrm{Anawin}.\end{aligned}$$

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 $\mathcal{P}^{\pi_{..\psi(p_i)},\nu,t_r}$, there is a prediction $\overline{\mathcal{P}}'\in\mathcal{P}^{\pi_{..\psi(p_i)},\nu,t_r}\subseteq\overline{\mathcal{P}}_r$, such that:

$$\begin{aligned}&\overline{\mathcal{P}}'=\overline{\varnothing}[(q',c',\overline{\gamma},\overline{m})/i] &&\text{if } c'=k\text{ or }c'=2\\&\left.\begin{aligned}\overline{\mathcal{P}}'_j&=\mathcal{P}^{\pi_{..\psi(p_i)+1},\nu,t_j}\subseteq\overline{\mathcal{P}}_j\\\overline{\mathcal{P}}'_i&=(q',c',\overline{\gamma},\overline{m})\end{aligned}\right. &&\text{for all }j\neq i,\text{ if }2\neq c'<k\end{aligned}$$

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)*:* By Lemma 31, this cannot 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.

□

4 Lower bound

This chapter is dedicated to proving a lower bound on the complexity of reachability multi-pushdon games. As reachability games can easily be formulated as parity games, it is sufficient to show the lower bound for reachability multi-pushdown games. In fact, we will transform a Turing machine into a state reachability multi-pushdown game.

Theorem 32. $(k+2)$*-context bounded multi-pushdown games are* k*-EXPTIME hard.* k*-phase bounded multi-pushdown games are* k*-EXPTIME hard.*

The task of this chapter is to transform a k-EXPTIME Turing machine and an input word into a multi-pushdown game such that Eve possesses a winning strategy from a certain position if and only if the input word would be accepted by the Turing machine.

Let us draw a rough outline of the steps taken in order to accomplish the above. The first aspect is turning the Turing machine into a game. Due to the alternating space – deterministic time theorem about Turing machines, we reduce from an alternating Turing machine with $(k-1)$-EXP space bound. Alternating Turing machines are easily transformed into a game on the Turing machines configuration graph, as seen in the next section (4.1.1) about Turing machines. Intuitivly, the successors of configurations in existential branching state are chosen by Eve and successors of configurations in universal branching state by Ana. A winning strategy for Eve then results in the fact that each branch from a configuration in universal branching state yields a computation accepted by the Turing machine (And a winning strategy for Ana finds a branch violating the acceptance).

The next step is to construct a multi-pushdown game that simulates the above alternating Turing machine game. There are three core ideas involved in the construction.

The first idea is storing a branch of the computation on one stack. This stack will contain a computation branch prefix of the alternating Turing machine, which represents a play in the alternating Turing machine game. Whenever Eve (Ana) would choose to make a move to the next successing configuration, they instead push the desired configuration onto the stack holding the current prefix.

The second and third idea are used to keep the state space of the multi-pushdown game small. The configurations we are handling are sized $(k-1)$-EXP in the input of the Turing machine. But for a PTIME reduction, we need the resulting multi-pushdown game to be adequatly small. In particular, we can not simply use the state space to ensure that each player is indeed pushing (successing) configurations.

Instead, we find a solution in the second idea that is called Guess & Check: Since we are only interested in winning strategies for each player, we use Eve as an oracle to guess something and Ana as a checker, either disbelieving and entering a checking routine to invalidate the Guess or believing and continuing the play. The checking routine is a part of the game that will determine the winner of any play entering it. It should hold that when Eve guessed correctly they will possess a winning strategy within this routine. Vice versa, if they guessed incorrectly, Ana should possess a winning strategy in this routine. A winning strategy for Eve now always guesses correctly as guessing incorrectly contradicts the strategy to be winning.

For our use case of Guess & Check, we give Eve a transition enabled in the topmost configuration of the stack holding the computation prefix. Eve makes a Guess by pushing any sequence onto this stack. They guess correctly, if they push precisely the successing configuration reached by taking the stated Turing machine transition from the previous top of stack configuration. For the Check part, the multi-pushdown game needs to check whether the pushed sequence by Eve was indeed the desired successing configuration. Finding a routine with this property is the trickiest part in the construction and involves the third idea.

The third core idea is a layered indexing system. Each letter of the configurations will be succeded by an index containing the position of the letter within the configuration. The intuitive idea behind this is that when the configuration has size $(k-1)$-EXP in the input, then each index has only length $(k-2)$-EXP. Using this indexing system in a layered fashion, which means that each of the indecies is indexed itself, results in the smallest indecies being of polynomial length. So for the smallest indecies we can construct gadgets in our multi-pushdown game to check properties like equality of the indecies. We then use induction to create gadgets that check the same properties on higher level indexed words. In fact, the mechanism is powerful enough to check for two indexed words on a stack and any first order relation (defined in section 4.2) whether the words are in relation or not. The successor relation of a Turing machine is a first order relation. This completes the checking routine.

4.1 Turing machines

Turing machines are a generic model for computation used for the definition of complexity classes. Since we plan to build a reduction from the acceptance problem for alternating Turing machines with $(k-1)$-EXP space bound, we need to settle notation for Turing machines and their computations.

Definition 33. An *alternating Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \Delta, q_{init})$, where Q is a set of states, Γ is the tape alphabet containing $\#$, the empty cell symbol, $\Sigma \subseteq \Gamma \setminus \{\#\}$ is the input alphabet, $q_{init} \in Q$ is the initial state and $\Gamma \subseteq Q \times \Gamma \times Q \times \Gamma \times \{N, L, R\}$ is the transition relation. The set of states is partitioned into the existential and universal branching states by a branching assignment $\mathrm{br} : Q \to \{\exists, \forall\}$. A Turing machine may have a *space bound* $sb : \mathbb{N} \to \mathbb{N}$.

For the definition of configurations, we introduce a copy of the original tape alphabet $\overline{\Gamma} = \Gamma \cup (\Gamma \times \{\dagger\})$. We represent tape cells by symbols from $\overline{\Gamma}$. A symbol $(s, \dagger)$ corresponds to a cell with content s with the head of the Turing machine is at this position and s to a cell with content s and the head of the Turing machine is absent at this position. A configuration of M on input w of length $n = |w|$ is of the form (q, c), where $c \in \overline{\Gamma}^{sb(n)}$ and only one symbol in c is from $(\Gamma \times \{\dagger\})$. The set of configurations then is $C_M = Q \times \{c \in \overline{\Gamma}^{sb(n)} \mid c$ contains precisely one letter $(s, \dagger), s \in \Gamma\}$.

Definition 34. Let $(q, c), (q', c') \in C_M$ be configurations and let $p \in [0..sb(n) - 1]$ be the position of the head in (q, c), i.e. $c_p = (s, \dagger)$ for some $s \in \Gamma$.
The configurations are in the *successor* relation $\mapsto \subseteq C_M \times \Delta \times C_M$, denoted by $(q, c) \overset{\delta}{\mapsto} (q', c')$, if one of the following is true

- $\delta = (q, s, q', s', N)$ and $c'_p = (s', \dagger)$
- $\delta = (q, s, q', s', L)$ and $p > 0$ and $c'_p = s'$ and $c'_{p-1} = (s'', \dagger)$ where $s'' = c_{p-1}$
- $\delta = (q, s, q', s', R)$ and $p < sb(n) - 1$ and $c'_p = s'$ and $c'_{p+1} = (s'', \dagger)$ where $s'' = c_{p+1}$

and for all other positions u, $c'_u = c_u$.
We may omit δ in the notation, if it is not of importance.

We define a set of leaf configurations Leaf($\Box$) for each branching type $\Box \in \{\exists, \forall\}$, which do not offer any successing configuration:

$$\text{Leaf}(\Box) = \{(q, c) \in C_M \mid \text{br}(q) = \Box \text{ and } \neg\exists(q', c') \in C_M.(q, c) \mapsto (q'c')\}.$$

These form the acceptance condition for a computation.
A computation begins with the input word on the tape and the head of the Turing machine on the right most symbol. The Turing machine may only use space according to its space bound to the left of the input word. Any other conventions would perform as well. They are only needed to identify a unique starting configuration.

Definition 35. A *computation tree* of M on input w of length n is a prefix closed, nonempty set $T \subset \mathbb{N}^\omega$ with a labelling $l : T \to C_M$ such that

- $l(\varepsilon) = (q_{init}, c_{init})$, where $c_{init} = \#^{sb(n)-n} w_0 \dots w_{n-2}(w_{n-1}, \dagger)$,
- if $l(t) = (q, c)$ with $\text{br}(q) = \exists$ and $(q, c) \notin \text{Leaf}(\exists)$, then $t.0 \in T$ is the only child of t with some labelling $l(t.0)$ such that $l(t) \mapsto l(t.0)$.
  If $(q, c) \in \text{Leaf}(\exists)$, t is a leaf.
- if $l(t) = (q, c)$, with $\text{br}(q) = \forall$, let $\{(q_0, c_0), \dots, (q_x, c_x)\} = \{(q', c') \in C_M \mid l(t) \mapsto (q', c')\}$. Then, $t.0, \dots, t.x - 1 \in T$ are the only childs of t such that $l(t.y) = (q_y, c_y)$.
  If $(q, c) \in \text{Leaf}(\forall)$, this construction results in t being a leaf.

A computation tree T is called accepting, if it is finite and all leaves t of T have a labelling $l(t) \in \text{Leaf}(\forall)$. An input word w is called accepting, if there is an accepting computation tree on input w. The language of M is $L(M) = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}$. M is called decider if all computation trees are finite for all inputs.

4.1.1 A game equivalent to an alternating Turing machine

We can resemble the acceptance problem of an alternating Turing machine by the means of a game. The key principle is that each player chooses a successing configuration for configurations that are in their respective branching state.

Let $M = (Q, \Sigma, \Gamma, \Gamma, q_{init})$ be an alternating decider with space bound $sb : \mathbb{N} \to \mathbb{N}$ and branching assignment $\text{br} : Q \to \{\exists, \forall\}$. The game $G_M = (C_M, \mapsto, \text{Leaf}(\forall))$ with ownership assignment

$$own((q, c)) = \begin{cases} \text{Eve} & \text{if } \text{br}(q) = \exists \\ \text{Ana} & \text{if } \text{br}(q) = \forall \end{cases}$$

is equivalent to the Turing machine in the following sense.

Theorem 36. *$w \in L(M)$ if and only if Eve possesses a winning strategy in G_M from (q_{init}, c_{init}).*

Be aware that M is a decider. Any computation tree of it is finite. In particular, any play in G_M visits some position that does not offer a next move. As discussed in the section for games, the owner of such position looses the play. The reachability set $\text{Leaf}(\forall)$ for Eve is precisely the set of positions where Ana cannot make another move.

Proof. Assume Eve possesses a winning strategy σ_{Eve} from (q_{init}, c_{init}). We want to create a computation tree T dependent on σ_{Eve} with the following property. For each node $t = t_0 \dots t_m \in T$ the play prefix $l(t_0)l(t_0t_1)\dots l(t_0 \dots t_m)$ is compliant with strategy σ_{Eve}.

We construct the tree inductivly as follows. It is immediate that it is a computation tree of M on input w.

- Initially, set $T = \{\varepsilon\}$ with $l(\varepsilon) = (q_{init}, c_{init})$. This is the prefix of every play starting in (q_{init}, c_{init}).
- If $t \in T$ is currently a leaf and $l(t) = (q, c)$ with $\text{br}(q) = \exists$, then add $t.0$ to T with labelling $l(t.0) = \sigma_{\text{Eve}}(l(t))$. This exists, since σ_{Eve} is a winning strategy. By induction, because $t \in T$, the play prefix $l(t_0)l(t_0t_1)\dots l(t)$ is compliant with strategy σ_{Eve}. The play prefix $l(t_0)l(t_0t_1)\dots l(t)l(t.0)$ is also compliant with strategy σ_{Eve}.

- If $t \in T$ is currently a leaf and $l(t) = (q, c)$ with $\mathrm{br}(q) = \forall$, add the nodes $t.0, \dots, t.x$ to T, where $\{(q_0, c_0), \dots, (q_x, c_x)\} = \{(q', c') \in C_M \mid l(t) \mapsto (q', c')\}$. Label them by $l(t.y) = (q_y, c_y)$. Since $t \in T$, the play prefix $l(t_0)l(t_0t_1)\dots l(t)$ is compliant with strategy σ_{Eve}. Any play prefix $l(t_0)l(t_0t_1)\dots l(t)l(t.y)$, by $own(l(t)) \neq \mathrm{Eve}$, is also compliant with strategy σ_{Eve}.

This computation tree's leaves are labelled by configurations from $\mathrm{Leaf}(\forall)$. Towards contradiction, assume that there is a leaf t with $l(t) = \mathrm{Leaf}(\exists)$. By the construction of the tree, $t.0$ is being added to T with labelling $l(t.0) = \sigma_{\mathrm{Eve}}(l(t))$. This contradicts t being a leaf. As discussed earlier, each play in G_M is finite, and so is T. Therefore, T is accepting and $w \in L(M)$.

Now assume that Ana possesses a winning strategy σ_{Ana} from (q_{init}, c_{init}). Towards contradiction, assume there is an accepting computation tree T of M on input w such that all leafs t have a label $l(t) \in \mathrm{Leaf}(\forall)$. We construct a play $\pi = \pi_0\pi_1\dots$ compliant with strategy σ_{Ana} inductivly with the following property. For every play prefix $\pi_0 \dots \pi_m$, there is $t_0 \dots t_{m-1} = t \in T$ with $\pi_0 \dots \pi_m = l(\varepsilon)l(t_0)l(t_0t_1)\dots l(t)$.

- $\pi_0 = l(\varepsilon) = (q_{init}, c_{init})$.
- $\pi_0 \dots \pi_m = l(\varepsilon)l(t_0)l(t_0t_1)\dots l(t)$ for some $t_0 \dots t_{m-1} = t \in T$. Continue the play to π_{m+1} where

  Case 1 ($own(\pi_m) = \exists$)*:* $\pi_{m+1} = l(t.0)$

  Case 2 ($own(\pi_m) = \forall$)*:* $\pi_{m+1} = \sigma_{\mathrm{Ana}}(l(t))$. Since T is a computation tree and $l(t) \mapsto \sigma_{\mathrm{Ana}}(l(t))$, there is y, s.t. $\sigma_{\mathrm{Ana}}(l(t)) = l(t.y)$.

It is immediate that π is compliant with σ_{Ana}. Since T is accepting, the play ends in some leaf $t \in T$ with $l(t) \in \mathrm{Leaf}(\forall)$ which would be loosing for Ana. This contradicts σ_{Ana} being a winning strategy. Therefore, no accepting computation tree exists and $w \notin L(M)$. □

4.2 First order relations

First order relations are defined by the use of a first order formula. The formula is evaluated over a structure of two words of same length. Given two words of same length and a first order formula, the corresponding first order relation puts those words in relation if they satisfy the formula.

The syntax of a first order formula consists of terms and formulas.

Definition 37. Let Σ be an alphabet and y_1, y_2 be variables and let $s \in \Sigma$ be symbol.
A *term* is defined by

$$t ::= s \mid s_1(y) \mid s_2(y).$$

Let t_1, t_2 be terms.
A *first order formula* is defined by

$$\varphi ::= y_1 \leq y_2 \mid t_1 = t_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \exists y.\varphi \mid \forall y.\varphi.$$

A formula φ is *closed*, if it contains no free variables, i.e. each variable is bound by a quantor.

To evaluate a formula under a structure of two words, we need a valuation binding the open variables of the formula. The domain for variables is the set of word positions. This way, each variable points to a position within the two words. The semantics of the formula are defined by structural induction. Terms represent single symbols. They are either a constant symbol or a symbol from within one of the two words at the position of a given variable. Formulas compare terms and variables. They are closed under boolean combination and first order quantification.

Definition 38. A first order formula φ is evaluated under a structure of two words $v_1, v_2 \in \Sigma^*$ of same length $|v_1| = |v_2| = n$ and a valuation $\mathrm{val} : \mathrm{Var} \to [0..n-1]$. We define the *semantics* of formulas and terms under such a structure.

$$\begin{aligned}
[\![t_1 = t_2]\!]^{\mathrm{val}}_{v_1,v_2} &:= [\![t_1]\!]^{\mathrm{val}}_{v_1,v_2} = [\![t_2]\!]^{\mathrm{val}}_{v_1,v_2} & [\![s]\!]^{\mathrm{val}}_{v_1,v_2} &:= s\\
[\![y_1 \leq y_2]\!]^{\mathrm{val}}_{v_1,v_2} &:= \mathrm{val}(y_1) \leq \mathrm{val}(y_2) & [\![s_1(y)]\!]^{\mathrm{val}}_{v_1,v_2} &:= v_{1\,\mathrm{val}(y)}\\
[\![\varphi_1 \wedge \varphi_2]\!]^{\mathrm{val}}_{v_1,v_2} &:= [\![\varphi_1]\!]^{\mathrm{val}}_{v_1,v_2} \wedge [\![\varphi_2]\!]^{\mathrm{val}}_{v_1,v_2} & [\![s_2(y)]\!]^{\mathrm{val}}_{v_1,v_2} &:= v_{2\,\mathrm{val}(y)}\\
[\![\varphi_1 \vee \varphi_2]\!]^{\mathrm{val}}_{v_1,v_2} &:= [\![\varphi_1]\!]^{\mathrm{val}}_{v_1,v_2} \vee [\![\varphi_2]\!]^{\mathrm{val}}_{v_1,v_2}\\
[\![\neg\varphi_1]\!]^{\mathrm{val}}_{v_1,v_2} &:= \neg[\![\varphi_1]\!]^{\mathrm{val}}_{v_1,v_2}\\
[\![\exists y.\varphi_1]\!]^{\mathrm{val}}_{v_1,v_2} &:= \exists p.[\![\varphi_1]\!]^{\mathrm{val}[p/y]}_{v_1,v_2}\\
[\![\forall y.\varphi_1]\!]^{\mathrm{val}}_{v_1,v_2} &:= \forall p.[\![\varphi_1]\!]^{\mathrm{val}[p/y]}_{v_1,v_2}
\end{aligned}$$

A closed formula φ is satisfied by v_1, v_2, denoted by $v_1, v_2 \models \varphi$, if $[\![\varphi]\!]^{\varnothing}_{v_1,v_2} = \mathit{true}$.

Now, we can use first order formulas to define relations between words of the same length.

Definition 39. A *first order relation* $\sim$ is a relation $\sim \subseteq \bigcup_{i\in\mathbb{N}}(\Sigma^i \times \Sigma^i)$ with a formula φ such that for all words v_1, v_2, $v_1 \sim v_2$ if and only if $v_1, v_2 \models \varphi$.
We call a first order relation *simple* if s_2 is not occuring in the formula. Note, that simple first order relations are unary relations and can be evaluated on a structure of a single word.

Example 40. The relations $=, <, (+1)$ over the most significant bit first encoding of numbers in $\mathbb{N}$ are first order relations with the following formulas.

$$\begin{array}{rll}
=: & \forall y. & s_1(y) = s_2(y) \\
<: & \exists y_1 \forall y_2. & (y_2 < y_1) \Rightarrow s_1(y_2) = s_2(y_2) \\
 & & \wedge\, (y_1 = y_2) \Rightarrow s_1(y_2) = 0 \wedge s_2(y_2) = 1 \\
(+1): & \exists y_1 \forall y_2. & (y_2 < y_1) \Rightarrow s_1(y_2) = s_2(y_2) \\
 & & \wedge\, (y_1 = y_2) \Rightarrow s_1(y_2) = 0 \wedge s_2(y_2) = 1 \\
 & & \wedge\, (y_1 < y_2) \Rightarrow s_1(y_2) = 1 \wedge s_2(y_2) = 0
\end{array}$$

To transform a first order formula into a game in the next section, we want the first order formula to be in normal form.

Definition 41. A first order formula is in *normal form*, if it is of the form

$$Q_1 y_1 \ldots Q_m y_m . \varphi,$$

where φ does not contain any quantifiers.
For each first order formula there is an equivalent first order formula in normal form.

4.2.1 A game equivalent to a first order formula

In this section, we transform a closed first order formula in normal form φ into a game. We want a winning strategy for Eve if the formula is satisfied. More formal, given a closed first order formula $Q_1 y_1 \ldots Q_m y_m . \varphi$ in normal form and a structure v_1, v_2, we create a game such that Eve possesses a winning strategy from a starting position if and only if $v_1, v_2 \vDash Q_1 y_1 \ldots Q_m y_m . \varphi$.

The general idea is that each player takes the role of chosing a valuation for the variables preceeded by their respective quantors. They begin with the outermost quantors and build up a valuation val along the play. The winner of a play is then determined by $[\![\varphi]\!]^{\text{val}}_{v_1,v_2}$.

Let $Q_1 y_1 \ldots Q_m y_m . \varphi$ be a closed first order formala in normal form, i.e. φ does not contain quantors. Let v_1, v_2 be words from Σ^n.

We construct the following game with positions of the form $(\text{Quant}, i, \text{val})$. Such a position means, that the players have already chosen a valuation for the first $i-1$ quantors. The choices are stored in val, which is a partial valuation for the first $i-1$ variables. The next position will determine the valuation for variable y_i. Thus, $(\text{Quant}, i, \text{val})$ should have its owner depend on Q_i.

The starting position is $(\text{Quant}, 1, \bot^{\{y_1,\ldots,y_m\}})$.
We set the ownership by $own((\text{Quant}, i, \text{val})) = \text{Eve}$ if $Q_i = \exists$ and $own((\text{Quant}, i, \text{val})) = \text{Ana}$ if $Q_i = \forall$. Also, $own((\text{Quant}, m+1, \text{val})) = \text{Eve}$. The game consists of the following moves:

$$\begin{array}{lr}
((\text{Quant}, i, \text{val}), (\text{Quant}, i+1, \text{val}[p/y_i])) & p \in [0..n-1], \quad 1 \leq i \leq m \\
((\text{Quant}, m+1, \text{val}), \text{Evewin}) & [\![\varphi]\!]^{\text{val}}_{v_1,v_2} = true \\
((\text{Quant}, m+1, \text{val}), \text{Anawin}) & [\![\varphi]\!]^{\text{val}}_{v_1,v_2} = false
\end{array}$$

Together, the above defines a game $G_{v_1,v_2,\sim} = (V_{v_1,v_2,\sim}, E_{v_1,v_2,\sim}, \{\text{Evewin}\})$ where $\sim$ is the first order relation for $Q_1 y_1 \dots Q_m y_m.\varphi$.

Lemma 42. *Eve possesses a winning strategy in $G_{v_1,v_2,\sim}$ from the position* $(\text{Quant}, 1, \bot^{\{y_1,\dots,y_m\}})$ *if and only if $v_1 \sim v_2$, i.e. $v_1, v_2 \vDash Q_1 y_1 \dots Q_m y_m.\varphi$.*

Proof. We will prove a stronger statement by induction:
Given a partial valuation $\text{val} : \{y_1, \dots, y_i\} \to [0..n-1]$ on the first i variables, Eve possesses a winning strategy from position $(\text{Quant}, i+1, \text{val})$ if and only if

$$[\![Q_{i+1} y_{i+1} \dots Q_m y_m.\varphi]\!]^{\text{val}}_{v_1,v_2} = \mathit{true}.$$

Base Case ($i = m$)**.** Let $\text{val} : \{y_1, \dots, y_m\} \to [0..n-1]$. By construction, Eve possesses a winning strategy from position $(\text{Quant}, m+1, \text{val})$ if and only if $[\![\varphi]\!]^{\text{val}}_{v_1,v_2} = \mathit{true}$.

Inductive Case ($i < m$)**.** By induction: Given any valuation $\text{val} : \{y_1, \dots, y_{i+1}\} \to [0..n-1]$, Eve possesses a winning strategy from $(\text{Quant}, i+2, \text{val})$ if and only if

$$[\![Q_{i+2} y_{i+2} \dots Q_m y_m.\varphi]\!]^{\text{val}}_{v_1,v_2} = \mathit{true}.$$

Let $\text{val} : \{y_1, \dots, y_i\} \to [0..n-1]$ be some valuation on the first i variables.
Assume $[\![Q_{i+1} y_{i+1} \dots Q_m y_m.\varphi]\!]^{\text{val}}_{v_1,v_2} = \mathit{true}$.

Case 1 ($Q_{i+1} = \exists$)*:* There is $u \in [0..n-1]$ s.t. $[\![Q_{i+2} y_{i+2} \dots Q_m y_m.\varphi]\!]^{\text{val}[u/y_{i+1}]}_{v_1,v_2} = \mathit{true}$. By induction, there is a winning strategy σ from $(\text{Quant}, i+2, \text{val}[u/y_{i+1}])$ for Eve. If we set $\sigma((\text{Quant}, i+1, \text{val})) = (\text{Quant}, i+2, \text{val}[u/y_{i+1}])$, we arrive at a winning strategy for Eve from $(\text{Quant}, i+1, \text{val})$.

Case 2 ($Q_{i+1} = \forall$)*:* For all positions $u \in [0..n-1]$, $[\![Q_{i+2} y_{i+2} \dots Q_m y_m.\varphi]\!]^{\text{val}[u/y_{i+1}]}_{v_1,v_2} = \mathit{true}$. By induction, there is a winning strategy σ from $(\text{Quant}, i+2, \text{val}[u/y_{i+1}])$ for Eve. Thus, σ is winning for Eve from $(\text{Quant}, i+1, \text{val})$.

Now assume $[\![Q_{i+1} y_{i+1} \dots Q_m y_m.\varphi]\!]^{\text{val}}_{v_1,v_2} = \mathit{false}$.

Case 1 ($Q_{i+1} = \exists$)*:* For all positions $u \in [0..n-1]$, $[\![Q_{i+2} y_{i+2} \dots Q_m y_m.\varphi]\!]^{\text{val}[u/y_{i+1}]}_{v_1,v_2} = \mathit{false}$. By induction, there is a winning strategy σ from $(\text{Quant}, i+2, \text{val}[u/y_{i+1}])$ for Ana. Thus, σ is winning for Ana from $(\text{Quant}, i+1, \text{val})$.

Case 2 ($Q_{i+1} = \forall$)*:* There is $u \in [0..n-1]$, s.t. $[\![Q_{i+2} y_{i+2} \dots Q_m y_m.\varphi]\!]^{\text{val}[u/y_{i+1}]}_{v_1,v_2} = \mathit{false}$. By induction, there is a winning strategy σ from $(\text{Quant}, i+2, \text{val}[u/y_{i+1}])$ for Ana. If we set $\sigma((\text{Quant}, i+1, \text{val})) = (\text{Quant}, i+2, \text{val}[u/y_{i+1}])$, we arrive at a winning strategy for Ana from $(\text{Quant}, i+1, \text{val})$.

□

4.3 Layered indexing

We use a layered indexing system similar to the one used in [6]. We havily rely on its structure for the check part of the final multi-pushdown game. The goal of the indexing system is to transform k-EXP sized configurations of Turing machines into indexed configurations with small indices that can be compared using only an polynomial amount of states in the input length of the Turing machine.

Intuitivly, an indexed word is the same word, but each letter is succeded by an index. An index contains the position of the letter in form of a most significant bit first encoding. Each index is shorter than the original indexed word by one exponent. In most cases, this will still be non-polynomial in the input length, which leads to indexing each index itself, resulting in a recursive indexing system.

Definition 43. We define the k-EXP function $\exp_k$

$$\exp_0(n) = n \qquad \exp_{k+1}(n) = 2^{\exp_k(n)}.$$

Configurations (q, c) of the Turing machine will be of the size $sb(n) = \exp_k(n^c)$ in the input length n for some $c \in \mathbb{N}$. We introduce $max_k = \exp_k(n^c) - 1$ for a shorter notation: $c = c_0 c_1 \dots c_{max_k}$.

To not confuse the different layers of indecies, we introduce index alphabets for each layer of indecies.

$$\Sigma_k = \{0_k, 1_k\} \qquad \Sigma_{<k} = \bigcup_{1 \leq i < k} \Sigma_k \qquad \Sigma_{\leq k} = \Sigma_{<k} \cup \Sigma_k$$

Definition 44. msbf is the function assigning a number its *most significant bit encoding* over a given binary alphabet Σ_k. For $n \in \mathbb{N}$, $\mathrm{msbf}(n) = n_0 \dots n_m$, such that $n = \sum_{i=0}^{m} n_i 2^{m-i}$.

Definition 45. A k-indexing $\mathrm{Ind}_k(v)$ of a word $v = v_0 \dots v_{max_k}$ is inductivly defined as

$$\begin{aligned}
\mathrm{Ind}_0(v) &= v \\
\mathrm{Ind}_{k+1}(v) &= v_0 x_0 \dots v_{max_{k+1}} x_{max_{k+1}} \\
&\qquad \text{where } x_i = \mathrm{Ind}_k(\mathrm{msbf}(i)) \text{ with } \mathrm{msbf}(i) \in \Sigma_{k+1}^{max_k+1}.
\end{aligned}$$

Be aware, that we require each most significant bit encoding for $\mathrm{msbf}(i)$ to have the same length $max_k + 1$.

4.4 Comparing indexed words with mpdg

We construct three different gadgets for the check part of the multi-pushdown system: k-Equality(Σ), k-ValidIndexing(Σ) and k-$\sim$-Check(Σ, Υ). Each is a multi-pushdown game with two stacks, that contains a winning strategy for Eve if its top of stack contents fulfill certain conditions.

The k-Equality(Σ) gadget expects both stacks to have a k-indexing at the top. Eve possesses a winning strategy, if they index the same word.

The k-ValidIndexing(Σ) gadget contains a winning strategy for Eve if there is a k-indexing at the top of stack 1.

The k-$\sim$-Check(Σ, Υ) gadget expects two k-indexings on stack 1 which are marked by symbols from Υ. Eve possesses a winning strategy, if the two indexed words are in relation by $\sim$ which is a first order relation.

To define these conditions more formally, let Λ be the alphabet of all stack content symbols of the mpdg instantiating one of the following gadgets. Informally, this is the alphabet of all possibly occuring stack symbols. Let $\gamma_1, \gamma_2, \gamma_3 \in \Lambda^*$.

- k-**Equality**(Σ): Let $v, v' \in \Sigma^{max_k+1}$ and $w_1 = \text{Ind}_k(v)$, $w_2 = \text{Ind}_k(v')$. Let $\sigma_1, \sigma_2 \in \Lambda \setminus (\Sigma_{\leq k} \cup \Sigma)$. From a configuration $(\mathit{start}, w_1\sigma_1\gamma_1, w_2\sigma_2\gamma_2)$, Eve possesses a winning strategy for k-Equality(Σ) if and only if $v = v'$.

- k-**ValidIndexing**(Σ): Let $w \in (\Sigma \cup \Sigma_{\leq k})^*$ and $\sigma \in \Lambda \setminus (\Sigma \cup \Sigma_{\leq k})$. From a configuration $(\mathit{start}, w\sigma\gamma_1, \gamma_2)$, Eve possesses a winning strategy for the gadget k-ValidIndexing(Σ) if and only if w is a k-indexing $w = \text{Ind}_k(v)$ for some $v \in \Sigma^{max_k+1}$.

- k-$\sim$-**Check**(Σ, Υ): Let $v, v' \in \Sigma^{max_k+1}$ and $w_1 = \text{Ind}_k(v)$, $w_2 = \text{Ind}_k(v')$. Let $\sigma \in \Upsilon$ and $\sim$ a first order relation. Let $\gamma_1 \in (\Lambda \setminus \Upsilon)^*$. From a configuration $(\mathit{start}, w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, \gamma_3)$ or $(\mathit{start}, w_1\sigma w_2\sigma_2\gamma_2, \gamma_3)$, Eve possesses a winning strategy for k-$\sim$-Check(Σ, Υ) if and only if $v \sim v'$.

We construct these by induction. We start with the gadget for 0-indexings and then show how to construct them for k-indexings given the $k-1$ gadgets already exist.

The k-ValidIndexing(Σ) and k-$\sim$-Check(Σ, Υ) gadget's inductive (k) construction depend on each other's $k-1$ gadgets. We split the construction parts for understandibility. Be aware, that by induction we can use both $k-1$ gadgets in the construction of each gadget.

We use the term "instantiate *gadgetName* on starting state *stateName*". This means that we disjointly add the states of the named gadget whilest adding the transition rules available for *start* to the state *stateName*. If the same gadget was instantiated with the same parameters on the same stack previously in the desired mpdg, we do not add new states, but only add the transition rules available for *start* to the state *stateName*.

Since we introduced the indexing idea in order to keep the amount of states in the final construction small, we are interested in the amount of states used by each gadget. We will only count the states of directly that gadget. Not the states of other gadgets instantiated by it. We will later add up all states possibly occuring by ranging over all possible instances for gadgets.

During the construction, there will be positions that do not offer a move. This is intentional in order to reduce the amount of transition rules. Let $\square \in \{\text{Eve}, \text{Ana}\}$ and $\blacksquare$ their opponent. For each position owned by $\square$ in some state q, each gadget also contains the transition rules $(q, r, \blacksquare\text{win})$ for every stack r. This way, there are no positions that do not offer a move and positions that would not offer one by construction are loosing for their owner.

4.4.1 k-Equality(Σ)

Given a k-indexing at the top of both stacks, this checking gadget's purpose is to give Eve a winning strategy if the indexed words on both stacks are the same.

Theorem 46. *Let $v, v' \in \Sigma^{max_k+1}$ and $w_1 = \text{Ind}_k(v)$, $w_2 = \text{Ind}_k(v')$. Eve possesses a strategy from $(\text{start}, w_1\sigma_1\gamma_1, w_2\sigma_1\gamma_1)$ if and only if $v = v'$.*
The maximal number of contexts or phases of any play is at most $k+2$. If the previous context or phase was on stack 1, it only takes at most $k+1$ additional contexts or phases.
The number of states of this gadget (not counting the states of additionally instantiated gadgets) is polynomially in the size of Σ and max_0.

Base Case ($k = 0$)**.** The starting configuration is $(start, w_1\sigma_1\gamma_1, w_2\sigma_1\gamma_1)$, where $w_1 = \text{Ind}_0(v) = v$ and $w_2 = \text{Ind}_0(v') = v'$. We need to check $v = v'$ and give Eve a winning strategy if that is the case.

The basic principle is to let Ana choose a violating position $0 \leq p < max_0$ in the words, where they claim $v_p \neq v'_p$. We store v_p in the state space and deterministically check the same position of the other word. If Ana cannot find a violating position, Eve wins the play. If there is no violating position, Eve possesses a winning strategy.

All the states belong to Ana. For all $s, s' \in \Sigma$ and $0 \leq p' < p \leq max_0$, introduce the following transition rules.

$$\begin{array}{ll}
((\text{pos}, p), s, 1, (\text{pos}, p+1)) & p < max_0 \\
(start, 1, (\text{pos}, 0)) & \\
((\text{pos}, p), s, 1, (\text{find}, p, s, 0)) & \\
((\text{find}, p, s, p'), s', 2, (\text{find}, p, s, p'+1)) & \\
((\text{find}, p, s, p), s, 2, \text{Evewin}) & \\
((\text{find}, p, s, p), s', 2, \text{Anawin}) & s \neq s'
\end{array}$$

Number of contexts/phases: The first context (phase) is on stack 1, visiting the second context (phase) on stack 2 with the popping transition $((\text{find}, p, s, p'), s', 2, (\text{find}, p, s, p'+1))$. No further contexts are visited. If the previous context was stack 1, the number of additional contexts is one.

Number of states: The number of states is limited by max_0 and Σ. It is polynomial in both.

Lemma 47. *Eve possesses a winning strategy from $(\text{start}, v\sigma_1\gamma_1, v'\sigma_2\gamma_2)$, if and only if $v = v'$.*

Proof. Assume Eve possesses a winning strategy σ_{Eve} from $(\mathit{start}, v\sigma_1\gamma_1, v'\sigma_2\gamma_2)$. We show that $v = v'$ by taking a look at the plays compliant with σ_{Eve}. By construction, Ana chooses the transition $((\text{pos}, p), v_p, 1, (\text{find}, p, v_p, 0))$ for some p and stores the symbol v_p in the state. From this position, the play deterministically continues through the states $(\text{find}, p, v_p, 0), \dots, (\text{find}, p, v_p, p)$ and removes the symbols $v'_0 \dots v'_{p-1}$ from stack 2, leaving v'_p as its topmost stack symbol. Then, since σ_{Eve} is a winning strategy, the next transition has to be $((\text{find}, p, v_p, p), v_p, 2, \text{Evewin})$, which proves $v_p = v'_p$. As this play is compliant with the strategy for any of Ana's choice of p, we arrive at $v = v'$.

Assume Ana possesses a winning strategy. Let there be a play compliant with that strategy. Analogously to the above, it contains a position in state (find, p, v_p, p). Since the strategy is winning for Ana, the play's next transition has to be $((\text{find}, p, v_p, p), v'_p, 2, \text{Anawin})$, where $v_p \neq v'_p$, proving $v \neq v'$. □

Inductive Case $(k > 0)$**.** The starting configuration is $(\mathit{start}, w_1\sigma_1\gamma_1, w_2\sigma_2\gamma_2)$, where $w_1 = \text{Ind}_k(v)$ and $w_2 = \text{Ind}_k(v')$ for some $v, v' \in \Sigma^{max_k}$. They are of the form

$$w_1 = v_0x_0 \dots v_{max_k}x_{max_k}, \quad w_2 = v'_0x_0 \dots v'_{max_k}x_{max_k}.$$

We repeat the principle of the base case, but instead of remembering the violating position p in the state space, we use the indexings.

We start by letting Ana remove a sequence in $(\Sigma\Sigma^*_{\leq k})^*$ from stack 1. This corresponds to choosing a violating position, as it leaves v_px_p on top of stack 1 for some p. We then store the symbol v_p in the state. The next part is to find the same position in stack 2. For this, we use Guess & Check: Eve has to pop a sequence of $(\Sigma\Sigma^*_{\leq k})^*$ from stack 2, leaving $v'_{p'}x_{p'}$ on top of stack 2. After storing $v'_{p'}$ in the state, Ana may now choose to

1. Believe Eve's choice to be $p' = p$. Then, according to v_p and $v'_{p'}$ we proceed as in the base case. I.e. if $v_p = v'_{p'}$, Eve wins and vice versa.

2. Disbelieve Eve's choice and claim that the two top of stack indexings don't contain the same position. I.e. $x_p = \text{Ind}_{k-1}(\text{msbf}(p))$ and $x_{p'} = \text{Ind}_{k-1}(\text{msbf}(p'))$ and $p \neq p'$. To check this, we use $(k-1)$-Equality(Σ_k), as the indices are $(k-1)$-indexings themselves.

Instantiate $(k-1)$-Equality(Σ_k) on stack 2 with starting state disbelievePos. Create the following transition rules for all $s, s' \in \Sigma$. The states owned by Eve are (reproducePos, s). All other states belong to Ana.

$$
\begin{aligned}
&(\mathit{start}, (\Sigma\Sigma_{\leq k}^{*})^{*}, 1, \text{violatingPos})\\
&(\text{violatingPos}, s, 1, (\text{reproducePos}, s))\\
&((\text{reproducePos}, s), (\Sigma\Sigma_{\leq k}^{*})^{*}s', 2, (\text{claimPos}, s, s'))\\
&((\text{claimPos}, s, s'), 2, (\text{believe}, s, s'))\\
&((\text{claimPos}, s, s'), 2, \text{disbelievePos})\\
&((\text{believe}, s, s), 2, \text{Evewin})\\
&((\text{believe}, s, s'), 2, \text{Evewin}) && s \neq s'
\end{aligned}
$$

Number of contexts/phases: Any play starts in the first context (phase) on stack 1. The second context (phase) is introduced by the popping transition ((reproducePos, s), $(\Sigma\Sigma_{\leq k}^{*})^{*}$, 2, (claimPos, s)) on stack 2. Then, either the play ends within this context (phase) or it continues in $(k-1)$-ValidIndexing(Σ_k) instanciated on stack 2. By induction (Theorem 46) and since the play's context (phase) is already on stack 2, this takes up to an additional k contexts (phases). Together this gadget takes up to $k+2$ contexts (phases). If the context (phase) was already on stack 1, it only introduces up to $k+1$ additional contexts (phases).

Number of states: Remember that we only count states belonging to this gadget, not its subgadgets. These are limited by the size of Σ polynomially. Note that they are constant in max_0. The transitions with regular expressions also add only a constant amount of transitions for each transition rule, because the underlying NFA has a constant amount of states.

Lemma 48. *Let $v, v' \in \Sigma^{max_k+1}$ and $w_1 = \text{Ind}_k(v)$, $w_2 = \text{Ind}_k(v')$. Eve possesses a winning strategy from* (start, $w_1\sigma_1\gamma_1, w_2\sigma_2\gamma_2$) *if and only if $v = v'$.*

Proof. Let Eve possess a winning strategy. As in the base case, let us look at the possible plays compliant with that strategy. By construction, Ana removed a sequence of $(\Sigma\Sigma_{\leq k}^{*})^{*}$ from w_1. If they removed w_1 completly or they did not remove one of the indexings x_p completly, they would end in a position without moves and the play would enter Evewin. Look at a play where that is not the case. Remember that $w_1 = v_0x_0 \dots v_{max_k}x_{max_k}$. Thus, the remaining content of stack 1 in state (reproducePos, v_p) is $x_p \dots v_{max_k}x_{max_k}\sigma_1\gamma_1$. Then, according to their strategy, Eve removes a sequence of $(\Sigma\Sigma_{\leq k}^{*})^{*}s'$ from w_2. By induction, and because the top of stack 1 is x_p, Eve possesses a winning strategy from state disbelievePos if and only if the top of stack content of stack 2 is x_p. Since Eve's strategy is winning, they removed a sequence, such that the resulting position fulfills that condition. Since w_2 is a k-indexing, there is only one such position: ((claimPos, v_p, v'_p), $x_p \dots v_{max_k}x_{max_k}\sigma_1\gamma_1$, $x_p \dots v'_{max_k}x_{max_k}\sigma_2\gamma_2$). The play's next position can be in state disbelievePos or (believe, v_p, v'_p). Look at the play continuing to (believe, v_p, v'_p). It must be in state Evewin, since Eve's strategy is winning. This states $v_p = v'_p$. This play is compliant with Eve's strategy for any p. It follows that $v = v'$.

Assume Ana possesses a winning strategy. We construct a play compliant with that strategy. The play visits position $((\text{reproducePos}, v_p), x_p \dots v_{max_k} x_{max_k} \sigma_1 \gamma_1, w_2 \sigma_2 \gamma_2)$. Since the strategy is winning for Ana, it has neither removed w_1 completly nor is there a remainder of some indexing x_p on top of stack 1 (in both cases, Ana would loose in the result of no possible moves). Then, Eve removes any sequence of $(\Sigma\Sigma_{\leq k}^*)^* s'$ from w_2. Look at the play, where the resulting position $((\text{claimPos}, v_p, v'_p), x_p \dots v_{max_k} x_{max_k} \sigma_1 \gamma_1, x_p \dots v'_0 x_0 \sigma_2 \gamma_2)$. If the next move would be $((\text{claimPos}, v_p, v'_p), 2, \text{disbelievePos})$, the resulting position is losing for Ana by induction (both stacks have the same top of stack $(k-1)$-indexing $x_p = \text{Ind}_{k-1}(\text{msbf}(p)))$. Thus, the only possible winning move can be $((\text{claimPos}, v_p, v'_p), 2, (\text{believe}, v_p, v'_p))$. Since the strategy is winning, the next move must be $((\text{believe}, v_p, v'_p), 2, \text{Anawin})$, which proves $v_p \neq v'_p$ and $v \neq v'$. □

4.4.2 k-ValidIndexing(Σ)

Given a sequence from $(\Sigma \cup \Sigma_{\leq k})^*$ on top of stack 1, this checking gadget gives Eve a winning strategy if the sequence is actually a k-indexing for any word v.

Theorem 49. *Let w be a sequence from $(\Sigma \cup \Sigma_{\leq k})^*$.*
Eve possesses a winning strategy from $(\text{start}, w\sigma\gamma_1, \gamma_2)$ *if and only if w is a k-indexing* $\text{Ind}_k(v)$ *for some $v \in \Sigma^{max_k+1}$.*
Any play has at most $k+2$ many contexts. If the previous context was already on stack 1, it takes at most $k+1$ additional contexts.
Any play has at most $k+1$ many phases. If the previous phase was already on stack 1, it takes at most k additional phases.
The number of states of this gadget (not counting states of subgadgets) is polynomial in max_0.

Base Case ($k = 0$)**.** A valid 0-indexing $\text{Ind}_0(v)$ is just a sequence $v \in \Sigma^{max_0+1}$. All states belong to Eve. Note, that Eve looses due to not having a move, if w is not actually a 0-indexing.

$$(start, \Sigma^{max_0+1}(\Lambda \setminus \Sigma), 1, \text{Evewin})$$

Number of contexts/phases: This takes up to one context or phase. If the previous context or phase was on stack 1, this takes no additional context or phase.

Number of states: The gadget needs max_0 states in the transition, because the regular expression contains max_0. In particular, this is polinomial in max_0.

Inductive Case ($k > 0$)**.** Assume the starting configuration $(start, w\sigma\gamma_1, \gamma_2)$ with $w \in (\Sigma \cup \Sigma_{\leq k})^*$. Then, w is a valid k-indexing, if

- $w = v_0 x_0 \dots v_m x_m \in (\Sigma\Sigma_{\leq k}^*)^+$
- Each x_p is a valid $(k-1)$-indexing.
- x_0 satisfies the first order formula $\forall y, s_1(y) = 0_k$, i.e. it indexes msbf(0).

- x_m satisfies the first order formula $\forall y, s_1(y) = 1_k$, i.e. it indexes $\text{msbf}(max_k)$.
- For each $0 \leq p < max_k$, if $x_p = \text{Ind}_{k-1}(\text{msbf}(i))$ and $x_{p+1} = \text{Ind}_{k-1}(\text{msbf}(i'))$, then $i' = i + 1$. I.e. $(\text{msbf}(i), \text{msbf}(i')) \in (+1)$, where $(+1)$ is the first order relation from Example 40.

The general idea is to let Ana choose, which of the conditions is violated. State *start* is owned by Ana. The following transition rules embody Ana's choice.

$$\begin{aligned}
&(start, 1, \text{sequenceCheck})\\
&(start, \Sigma(\Sigma^*_{\leq k}\Sigma)^*, 1, \text{smallerIndexCheck})\\
&(start, \Sigma, 1, \text{minCheck})\\
&(start, 1, \text{maxCheck})\\
&(start, \Sigma(\Sigma^*_{\leq k}\Sigma)^*, 1, \text{successorCheck})
\end{aligned}$$

In the first case, Eve has to prove that w is from $(\Sigma\Sigma^*_{\leq k})^+$. State sequenceCheck belongs to Eve and there is the transition rule

$$(\text{sequenceCheck}, (\Sigma\Sigma^*_{\leq k})^+(\Lambda \setminus (\Sigma \cup \Sigma_{\leq k})), 1, \text{Evewin}).$$

Be aware that σ is the first symbol from $\Lambda \setminus (\Sigma \cup \Sigma_{\leq k})$ on stack 1. If w is not a sequence of the desired form, Eve gets stuck and the play is won by Ana.

In the second case, Ana also finds a violating position p by removing a sequence from $\Sigma(\Sigma^*_{\leq k}\Sigma)^*$. Then, we use $(k-1)$-ValidIndexing(Σ_k) to check whether x_p is a $(k-1)$-indexing: Instantiate the gadget $(k-1)$-ValidIndexing(Σ_k) on stack 1 with starting state smallerIndexCheck.

For the third case, we remove v_0 and check the first indexing for whether it satisfies $\forall y, s_1(y) = 0_k$. Instantiate $(k-1)$-(0)-Check(Σ_k) on stack 1 with starting state minCheck, where (0) is the simple first order relation for above formula.

For the fourth case, we let Eve find the indexing x_{max_k}. We do so by another use of Guess & Check: Eve removes a sequence of $\Sigma(\Sigma^*_{\leq k}\Sigma)^*$. Then, Ana may believe them or check whether the next indexing is indeed the last one before finding the delimiter symbol σ on stack 1. If they believe, we use $(k-1)$-(1)-Check(Σ_k) on x_{max_k} analogue to the second case. Instantiate $(k-1)$-(1)-Check(Σ_k) on stack 1 with starting state actualMaxCheck, where (1) is the simple first order relation for the formula $\forall y, s_1(y) = 1_k$. The owner of maxCheck is Eve. All other states belong to Ana.

$$\begin{aligned}
&(\text{maxCheck}, \Sigma(\Sigma^*_{\leq k}\Sigma)^*, 1, \text{found})\\
&(\text{found}, 1, \text{actualMaxCheck})\\
&(\text{found}, 1, \text{disbelieve})\\
&(\text{disbelieve}, \Sigma^*_{\leq k}(\Lambda \setminus (\Sigma \cup \Sigma_{\leq k})), 1, \text{Evewin})\\
&(\text{disbelieve}, \Sigma^*_{\leq k}\Sigma, 1, \text{Anawin})
\end{aligned}$$

In the fifth case, Ana also finds a violating position p by removing a sequence from $\Sigma(\Sigma^*_{<k}\Sigma)^*$. To check, whether the indecies are successing, we use $(k-1)$-$(+1)$-Check(Σ_k). Instantiate $(k-1)$-$(+1)$-Check(Σ_k) on stack 1 in state successorCheck.

Number of contexts: All transition rules above are on stack 1, which provide the first context of any play. Any play can continue to

- $(k-1)$-ValidIndexing(Σ_k) instantiated on stack 1. Since the context of the play is already on stack 1, by induction (Theorem 49) the play needs up to $(k-1)+1 = k$ additional contexts. Together, $k+1$ contexts. If the previous context was on stack 1, the transitions before entering $(k-1)$-ValidIndexing(Σ_k) do not introduce a new context and the play introduces only up to k additional contexts.

- At multiple occasions, the play can enter either $(k-1)$-(0)-Check(Σ_k), $(k-1)$-(1)-Check(Σ_k) or $(k-1)$-$(+1)$-Check(Σ_k), each instantiated on stack 1. By induction (Theorem 51) the play then takes up to $(k-1)+2 = k+1$ additional contexts, adding up to $k+2$ contexts. If the previous context was on stack 1, the transitions before entering $(k-1)$-$\sim$-Check(Σ_k) do not introduce a new context and the play introduces only up to $k+1$ additional contexts.

In total, this is $k+1$ additional contexts, if the previous context was on stack 1 and $k+2$ contexts otherwise.

Number of phases: The transitions above also contain pop transitions for stack 1 before entering $(k-1)$-ValidIndexing(Σ_k) or $(k-1)$-1-Check(Σ_k). Thus there are plays, where the above transitions introduce a first phase on stack 1. Afterwards, the play can continue to

- $(k-1)$-ValidIndexing(Σ_k) instantiated on stack 1. Since the play is already in a phase for stack 1, by induction (Theorem 49) the play needs up to $k-1$ additional phases. Together, k many phases. If the previous phase was on stack 1, this reduces to $k-1$ many phases.

- At multiple occasions, the play can enter either $(k-1)$-(0)-Check(Σ_k), $(k-1)$-(1)-Check(Σ_k) or $(k-1)$-$(+1)$-Check(Σ_k), each instantiated on stack 1. By induction (Theorem 51) the play then takes up to $k-1$ additional phases, adding up to k phases. If the previous phase was on stack 1, this reduces to $k-1$ many phases.

In the inductive case thus, the play is takes actually less phases than the lemma suggests. However, since the base case does comply with the phases as given by the theorem and this gadget is not the limiting factor of the construction, we stay with those values even in the inductive case.

Number of states: This gadget has a constant number of states. Remember that we do not count states from subgadgets.

Lemma 50. *Let w be a sequence from $(\Sigma\Sigma_{<k})^*$. Eve possesses a winning strategy from $((\text{start}, w\sigma\gamma_1), \gamma_2, i)$f and only if w is a valid k-indexing* $\text{Ind}_k(v)$ *for some $v \in \Sigma^{max_k+1}$.*

Proof. Assume $w = \mathrm{Ind}_k(v) = v_0 x_0 \dots v_{max_k} x_{max_k}$ is a k-indexing. The following strategy σ_{Eve} is winning for Eve.

$$\sigma_{\mathrm{Eve}}((\mathrm{maxCheck}, w\sigma\gamma_1, \gamma_2)) = (\mathrm{found}, x_{max_k}\sigma\gamma_1, \gamma_2)$$
$$\sigma_{\mathrm{Eve}}((\mathrm{sequenceCheck}, w\sigma\gamma_1, \gamma_2)) = (\mathrm{Evewin}, \gamma_1, \gamma_2))$$

These are compliant with the transition rules. Since w is a k-indexing, it is a sequence from $(\Sigma\Sigma^*_{\leq k})^+$.

Let there be a play beginning in $(\mathit{start}, w\sigma\gamma_1, \gamma_2)$ compliant with σ_{Eve}.

Case 1 (the next position is in state *sequenceCheck*)*:* The position is $(\mathrm{sequenceCheck}, w\sigma\gamma_1, \gamma_2)$ By σ_{Eve}, the next position is in state Evewin. The play is winning for Eve.

Case 2 (the next position is in state *smallerIndexCheck*)*:* The successing position is of the form $(\mathrm{smallerIndexCheck}, x_p v_{p+1} x_{p+1} \dots v_{max_k} x_{max_k}\sigma\gamma_1, \gamma_2)$. Since w is a k-indexing, $x_p = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p))$. The play continues in $(k-1)$-ValidIndexing(Σ_k). By induction, Eve possesses a winning strategy there, because the top of stack sequence is $x_p \in (\Sigma_k \cup \Sigma_{\leq(k-1)})$, a valid $(k-1)$-indexing.

Case 3 (the next position is in state *minCheck*)*:* The successing position is of the form $(\mathrm{minCheck}, x_0 \dots v_{max_k} x_{max_k}\sigma\gamma_1, \gamma_2)$. Since $w = x_0 \dots v_{max_k} x_{max_k}$ is a k-indexing, $x_0 = \mathrm{Ind}_{k-1}(\mathrm{msbf}(0))$. The play continues in $(k-1)$-(0)-Check(Σ_k). By induction, Eve possesses a winning strategy there, because the top of stack 1 is x_0, a $(k-1)$-indexing of msbf(0).

Case 4 (the next position is in state *maxCheck*)*:* The successing position is of the form $(\mathrm{maxCheck}, v_0 x_0 \dots v_{max_k} x_{max_k}\sigma\gamma_1, \gamma_2)$. By strategy σ_{Eve}, it further continues to $(\mathrm{found}, x_{max_k}\sigma\gamma_1, \gamma_2)$. The next position is either $(\mathrm{disbelieve}, x_{max_k}\sigma\gamma_1, \gamma_2)$, which continues to state Evewin, since the other transition rule is not enabled. Or the next position is $(\mathrm{actualMinCheck}, x_{max_k}\sigma\gamma_1, \gamma_2)$ and continues in $(k-1)$-(1)-Check(Σ_k). By induction, Eve has winning strategy from there, because the top of stack 1 is $x_{max_{k-1}}$, a $(k-1)$-indexing of $\mathrm{msbf}(max_{k-1})$.

Case 5 (the next position is in state *successorCheck*)*:* The successing position is of the form $(\mathrm{maxCheck}, x_p v_{p+1} x_{p+1} \dots v_{max_k} x_{max_k}\sigma\gamma_1, \gamma_2)$. Since w is a k-indexing, $x_p = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p))$ and $x_{p+1} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p{+}1))$. The play continues in $(k{-}1)$-$(+1)$-Check(Σ_k). By induction, Eve possesses a winning strategy from there, because $x_p = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p))$ and $x_{p+1} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p+1))$ and they are delimited by $v_{p+1} \in \Sigma_k$.

Assume $w \in (\Sigma \cup \Sigma_{<k})^*$ is not a k-indexing. Then, it violates one of the above conditions.

Case 1 ($w \notin (\Sigma\Sigma^*_{<k})^+$)*:* The following strategy σ_{Ana} is winning for Ana.

$$\sigma_{\mathrm{Ana}}((\mathit{start}, w\sigma\gamma_1, \gamma_2)) = (\mathrm{sequenceCheck}, w\sigma\gamma_1, \gamma_2)$$

A play compliant with this strategy visits $(\mathrm{sequenceCheck}, w\sigma\gamma_1, \gamma_2)$. This position is owned by Eve, who has no transition available, since $w \notin (\Sigma\Sigma^*_{\leq k})^+$. Thus, the play is winning for Ana.

For the other cases, assume $w = v_0x_0 \dots v_mx_m$ is a sequence from $(\Sigma\Sigma^*_{\leq k})^m$ that is still not a k-indexing. Be aware that even though the notation is close to the above, the parts $x_0, \dots, x_m$ are not neccessarily $(k-1)$-indexings. They are merely sequences from $\Sigma^*_{\leq k}$.

Case 2 (there is an index p, s.t. x_p is not a $(k-1)$-indexing)*:* The following strategy σ_{Ana} is winning for Ana.

$$\sigma_{\text{Ana}}((\textit{start}, w\sigma\gamma_1, \gamma_2)) = (\text{smallerIndexCheck}, x_pv_{p+1}x_{p+1}\dots v_mx_m\sigma\gamma_1, \gamma_2)$$

The play visits $(\text{smallerIndexCheck}, x_pw_{p+1}x_{p+1}\dots w_mx_m\sigma\gamma_1, \gamma_2)$ immediatly and continues in $(k-1)$-ValidIndexing(Σ_k). By induction, Ana possesses a winning strategy from there, because the top of stack sequence of stack 1 is x_p, wich is not a $(k-1)$-indexing.

Case 3 ($x_0 \neq \text{Ind}_{k-1}(0_k^{max_{k-1}})$)*:* The following strategy σ_{Ana} is winning for Ana.

$$\sigma_{\text{Ana}}((\textit{start}, w\sigma\gamma_1, \gamma_2)) = (\text{maxCheck}, x_0v_1x_1\dots v_mx_m\sigma\gamma_1, \gamma_2)$$

A play compliant with this strategy visits $(\text{maxCheck}, x_0v_1x_1\dots v_mx_m\sigma\gamma_1, \gamma_2)$ immediatly and continues in $(k-1)$-(0)-Check(Σ_k), By induction, Ana possesses a winning strategy from there, because the top of stack sequence is x_0, which is a $(k-1)$-indexing of another word than msbf(0).

Case 4 ($x_m \neq \text{Ind}_{k-1}(1_k^{max_{k-1}})$)*:* The following strategy σ_{Ana} is winning for Ana.

$$\sigma_{\text{Ana}}((\textit{start}, w\sigma\gamma_1, \gamma_2)) = (\text{maxCheck}, w\sigma\gamma_1, \gamma_2)$$

$$\sigma_{\text{Ana}}((\text{found}, w'\sigma\gamma_1, \gamma_2)) = \begin{cases} ((\text{actualMinCheck}, w'\sigma\gamma_1), \gamma_2, &)\text{if } w' = x_m \\ (\text{disbelieve}, w'\sigma\gamma_1, \gamma_2) & \text{if } w' \neq x_m \end{cases}$$

Let there be a play compliant with this strategy. After $(\text{maxCheck}, w\sigma\gamma_1, \gamma_2)$, it visits a position $(\text{found}, w'\sigma\gamma_1, \gamma_2)$, where $w' = x_pv_{p+1}x_{p+1}\dots v_mx_m$ for some p.
Case 4.1 ($p \neq m$): Then, $w' \neq x_m$. The play continues to $(\text{disbelieve}, w'\sigma\gamma_1, \gamma_2)$ and the only transition available is $(\text{disbelieve}, \Sigma^*_{\leq k}\Sigma, 1, \text{Anawin})$, since $v_{p+1} \in \Sigma$.
Case 4.2 ($p \neq m$): Then, $w' = x_m$. The play continues in $(k-1)$-1-Check(Σ_k). By induction, Ana possesses a winning strategy from there, because the top of stack sequence is x_m, which is a $(k-1)$-indexing of another word than msbf(max_k).

Case 5 (there is an p, s.t. $x_p = \text{Ind}_{k-1}(\text{msbf}(i))$ and $x_{p+1} = \text{Ind}_{k-1}(\text{msbf}(i'))$, but $i+1 \neq i'$)*:* The following strategy σ_{Ana} is winning for Ana.

$$\sigma_{\forall}((\textit{start}, w\sigma\gamma_1, \gamma_2)) = (\text{successorCheck}, x_pv_{p+1}x_{p+1}\dots v_mx_m\sigma\gamma_1, \gamma_2)$$

A play compliant with this strategy visits $(\text{successorCheck}, x_pv_{p+1}x_{p+1}\dots v_mx_m\sigma\gamma_1, \gamma_2)$ immediatly and continues in $(k-1)$-$(+1)$-Check(Σ_k). By induction, Ana possesses a winning strategy from there, because $x_p = \text{Ind}_{k-1}(\text{msbf}(i))$ and $x_{p+1} = \text{Ind}_{k-1}(\text{msbf}(i'))$ are $(k-1)$-indexings and they are delimited by $v_{p+1} \in \Sigma_k$, however $i' \neq i+1$, i.e. $(\text{msbf}(i), \text{msbf}(i')) \notin (+1)$.

□

4.4.3 k-$\sim$-Check(Σ, Υ)

This gadget is build for a first order relation $\sim$. Its purpose is to check for two indexed words v, v', if $v \sim v'$ holds. Eve gains a winning strategy in the gadget if that is the case. In the starting configuration, the indexings are stored on stack 1. The first is on top of the stack and the other is marked by a delimiter symbol from Υ.

Theorem 51. *Let $v, v' \in \Sigma^{max_k}$ and $w_1 = \mathrm{Ind}_k(v)$, $w_2 = \mathrm{Ind}_k(v')$. Let $\sigma \in \Upsilon$ and $\sim$ a first order relation. Let $\gamma_1 \in (\Lambda \setminus \Upsilon)^*$.*
Eve possesses a winning strategy from $(\mathrm{start}, w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, \gamma_3)$ if and only if $v \sim v'$.
Any play takes at most $k+2$ contexts. If the previous context was on stack 2, it takes up to $k+1$ additional contexts.
Any play takes at most $k+1$ phases. If the previous phase was on stack 1, it takes up to k additional phases.
The number of states of this gadget (not counting the states of subgadgets) is polynomial in the size of Σ and max_0.

Let $\sim$ be a first order relation with first order formula $Q_1y_1 \ldots Q_my_m.\varphi$ in normal form.

The general idea is to simulate the game $G_{v,v',\sim}$ from section 4.2.1. In that game, each player took the role of choosing a valuation for each variable y_l, where Q_l was their respective quantor. We stored the valuation for that variable in the position and continued with the next quantor.

Positions of mpdg consist of the state and the stack contents. In the base case, the set of positions is small enough to store the valuation in the state space. In the inductive case, we will use stack 2 to store the chosen valuation for each variable. Actually, we will let the respective player choose single digits of the most significant bit encoding of the desired position, while Eve is responsible for indexing each digit with a correct $(k-2)$-indexing. The result will be a $(k-1)$-indexing of the most significant bit encoding of the chosen valuation for the current variable.

In order to evaluate the formula under the complete valuation, we need to identify the semantics of the terms $s_1(y)$ and $s_2(y)$ for any occuring variable. In order to keep the number of contexts and phases small, we use another instance of Guess & Check to find these. Eve will guess the letters from v, v' at the chosen valuation for each variable y. Ana will either believe them, or enter a checking routine that is similar to what the gadget k-Equality(Σ) does in its inductive case. In the inductive case, we also need to address that the valuation is not stored in the state space. We instead create a variable ordering in the state space.

Base Case ($k = 0$)**.** The starting configuration is $(\mathit{start}, w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, \gamma_3)$, where $w_1 = \mathrm{Ind}_0(v) = v$ and $w_2 = \mathrm{Ind}_0(v') = v'$. We want to simulate the game $G_{v,v',\sim}$ from section 4.2.1. Similar to $G_{v,v',\sim}$, we maintain a partial valuation val : $\{y_1, \ldots, y_m\} \to [1..max_0]$ of the variables in the state space. After the valuation is complete, Eve guesses $s_1, s_2 : \{y_1, \dots, y_m\} \to \Sigma$, the letters of the words at the variable positions as chosen by the valuation. These are used to evaluate the formula's terms $s_1(y), s_2(y)$ instead of using the whole structure v, v'.

Definition 52. For functions $s_1, s_2 : \{y_1, \dots, y_m\} \to \Sigma$ we define new semantics $[\![\varphi]\!]^{\mathrm{val}}_{s_1,s_2}$. They are the same as the previous semantics except for

$$[\![s_1(y)]\!]^{\mathrm{val}}_{s_1,s_2} := s_1(y), \quad [\![s_2(y)]\!]^{\mathrm{val}}_{s_1,s_2} := s_2(y).$$

For a valuation val and a word v, we define the function $s^{\mathrm{val},v} : \{y_1, \dots, y_m\} \to \Sigma$ by $s^{\mathrm{val},v}(y) = v_{\mathrm{val}(y)}$.
Expectedly, $[\![\varphi]\!]^{\mathrm{val}}_{s^{\mathrm{val},v}, s^{\mathrm{val},v'}} = [\![\varphi]\!]^{\mathrm{val}}_{v,v'}$.

After Eve chose s_1 and s_2, Ana has the option to disbelieve that $s_1(y) = s^{\mathrm{val},v}(y)$ or $s_2(y) = s^{\mathrm{val},v'}(y)$ for any variable.

For all $p \in [0..max_0]$, $j, l \in [1..m]$ and $s_1, s_2 : \{y_1, \dots, y_m\} \to \Sigma$ and $s \in \Sigma$, create the following transitions rules.

$$\begin{array}{ll}
(\mathit{start}, 1, (\mathrm{Quant}, 1, \bot^{\{y_1,\dots,y_m\}})) & \\
((\mathrm{Quant}, l, \mathrm{val}), 1, (\mathrm{Quant}, l+1, \mathrm{val}[p/y_l])) & \\
((\mathrm{Quant}, m+1, \mathrm{val}), 1, (\mathrm{val}, s_1, s_2)) & \\
((\mathrm{val}, s_1, s_2), 1, (\mathrm{bel}, \mathrm{val}, s_1, s_2)) & \\
((\mathrm{val}, s_1, s_2), 1, (\mathrm{dis}, 1, y_j, \mathrm{val}, s_1)) & \\
((\mathrm{val}, s_1, s_2), 1, (\mathrm{dis}, 2, y_j, \mathrm{val}, s_2)) & \\
((\mathrm{bel}, \mathrm{val}, s_1, s_2), 1, \mathrm{Evewin}) & \text{if } [\![\varphi]\!]^{\mathrm{val}}_{s_1,s_2} = \mathit{true} \\
((\mathrm{bel}, \mathrm{val}, s_1, s_2), 1, \mathrm{Anawin}) & \text{if } [\![\varphi]\!]^{\mathrm{val}}_{s_1,s_2} = \mathit{false}
\end{array}$$

For disbelieving s_1 or s_2, we get the following transitions.

$$\begin{array}{ll}
((\mathrm{dis}, 1, y_j, \mathrm{val}, s_1), \Sigma^{\mathrm{val}(y_j)}, 1, \mathrm{Evewin}) & s_1(y_j) = s \\
((\mathrm{dis}, 1, y_j, \mathrm{val}, s_1), \Sigma^{\mathrm{val}(y_j)}, 1, \mathrm{Anawin}) & s_1(y_j) \neq s \\
((\mathrm{dis}, 2, y_j, \mathrm{val}, s_2), (\Lambda \setminus \Upsilon)^* \Upsilon \Sigma^{\mathrm{val}(y_j)} s, 1, \mathrm{Evewin}) & s_2(y_j) = s \\
((\mathrm{dis}, 2, y_j, \mathrm{val}, s_2), (\Lambda \setminus \Upsilon)^* \Upsilon \Sigma^{\mathrm{val}(y_j)} s, 1, \mathrm{Anawin}) & s_2(y_j) \neq s
\end{array}$$

The above transition rules define the game $G_{0,\sim} = (V_{0,\sim}, E_{0,\sim}, \{(\mathrm{Evewin})\})$.

Number of contexts/phases: Since all transitions are for stack 1, the plays can have at most one context or phase.

Number of states: The number of states in this gadget is determined by the number of possible valuations, and the number of functions s_1, s_2. The number of possible valuations is $(max_0 + 1)^m$. The number of functions for s_1, s_2 is $|\Sigma|^m$. In total, the number of states is polynomial in max_0 and the size of Σ, since the number of quantors m is a constant for each first order relation.

Lemma 53. *Eve possesses a winning strategy from* $(\mathrm{start}, v\sigma_1\gamma_1\sigma v'\sigma_2\gamma_2, \gamma_3)$ *(and from* $(\mathrm{start}, v\sigma v'\sigma_2\gamma_2, \gamma_3)$*) if and only if* $v \sim v'$*.*

Proof. In the following, we omit the stack contents of game positions, if they are not of interest.

We will prove this by showing that the game $G_{0,\sim}$ from $(start, v\sigma_1\gamma_1\sigma v'\sigma_2\gamma_2, \gamma_3)$ is a simulation of the game $G_{v,v',\sim}$ from section 4.2.1 via the mapping $f : V_{v,v',\sim} \to V_{0,\sim}$:

$$f((\mathrm{Quant}, l, \mathrm{val})) = ((\mathrm{Quant}, l, \mathrm{val}), v\sigma_1\gamma_1\sigma_2 v'\sigma_3\gamma_2, \gamma_3)$$
$$f(\Box\mathrm{win}) = (\Box\mathrm{win}, v\sigma_1\gamma_1\sigma_2 v'\sigma_3\gamma_2, \gamma_3)$$

Remember the conditions neccessary for a reachability game to be simulated (Section 2.2.1). Condition 1 (the ownership condition) and condition 2 (the winning area condition) for this simulation are clearly fulfilled. Be aware, that the reachability set in $G_{0,\sim}$ is $\{\mathrm{Evewin}\} \times (\Lambda^*)^*$. Any play visiting state Evewin never returns to any other state. Condition 4 and 5 for simulation (a play visiting a successor may not visit the reachability set) are fulfilled trivially. Condition 3 (the strategy requirements) is trivially fulfilled for the positions $(\mathrm{Quant}, l, \mathrm{val})$ where $l < m + 1$, since the moves in $G_{0,\sim}$ and $G_{v,v',\sim}$ for these positions are isomorph.

We need to show that each player possesses a strategy from $((\mathrm{Quant}, m + 1, \mathrm{val}), v\sigma v'\sigma_2\gamma_2, \gamma_3)$ to

$$\mathrm{Anawin, if } [\![\varphi]\!]^{\mathrm{val}}_{v,v'} = false$$
$$\mathrm{Evewin, if } [\![\varphi]\!]^{\mathrm{val}}_{v,v'} = true$$

Since the outgoing transitions from $(\mathrm{bel}, \mathrm{val}, s_1, s_2)$ represent exactly that behaviour, it is sufficient to show that each player possesses a strategy from $(\mathrm{Quant}, m+1, \mathrm{val}))$ to $(\mathrm{bel}, \mathrm{val}, s^{\mathrm{val},v}, s^{\mathrm{val},v'})$, knowing that $[\![\varphi]\!]^{\mathrm{val}}_{v,v'} = [\![\varphi]\!]^{\mathrm{val}}_{s^{\mathrm{val},v}, s^{\mathrm{val},v'}}$.

For Eve, this strategy is

$$\sigma_{\mathrm{Eve}}(((\mathrm{Quant}, m + 1, \mathrm{val}), v\sigma_1\gamma_1\sigma_2 v'\sigma_3\gamma_2, \gamma_3)) = (\mathrm{val}, s^{\mathrm{val},v}, s^{\mathrm{val},v'})$$

Any play compliant with that strategy is either winning for Eve or visits a position in state $(\mathrm{bel}, \mathrm{val}, s^{\mathrm{val},v}, s^{\mathrm{val},v'})$. If Ana would choose to use the transition $((\mathrm{val}, s^{\mathrm{val},v}, s^{\mathrm{val},v'}), 1, (\mathrm{dis}, 1, y_l, \mathrm{val}, s^{\mathrm{val},v}))$, the next transition determines the winner in $((\mathrm{dis}, 1, y_l, \mathrm{val}, s^{\mathrm{val},v}), \Sigma^{\mathrm{val}(y_l)} v_{\mathrm{val}(y_l)}, 1, \mathrm{Evewin})$, where only $v_0 \ldots v_{\mathrm{val}(y_l)-1}$ can be chosen for $\Sigma^{\mathrm{val}(y_l)}$. Be aware that this leads indeed to state Evewin, since $s^{\mathrm{val},v}(y_l) = v_{\mathrm{val}(y_l)}$. We can apply the same arguments, if Ana uses transition $((\mathrm{val}, s^{\mathrm{val},v}, s^{\mathrm{val},v'}), 1, (\mathrm{dis}, 2, y_l, \mathrm{val}, s^{\mathrm{val},v'}))$. The only remaining transition from state $(\mathrm{val}, s^{\mathrm{val},v}, s^{\mathrm{val},v'})$ is to the desired position $((\mathrm{bel}, \mathrm{val}, s^{\mathrm{val},v}, s^{\mathrm{val},v'}), v\sigma v'\sigma_2\gamma_2, \gamma_3)$.

For Ana, the strategy is

$$\sigma_{\text{Ana}}(((\text{val}, s_1, s_2), w_1\sigma_1\gamma_1\sigma_2 w_2\sigma_3\gamma_2, \gamma_3)) = \begin{cases} (\text{dis}, 1, y_l, \text{val}, s_1) & \text{if } s_1(\text{val}(y_l)) \neq v_{\text{val}(y_l)} \\ (\text{dis}, 2, y_l, \text{val}, s_2) & \text{if } s_2(\text{val}(y_l)) \neq v'_{\text{val}(y_l)} \\ (\text{bel}, \text{val}, s_1, s_2) & \text{if } s_1 = s^{\text{val},v} \text{ and} \\ & \quad s_2 = s^{\text{val},v'} \end{cases}$$

For analouge arguments to those for Eve, this strategy is either winning for Ana or visits $(\text{bel}, \text{val}, s^{\text{val},v}, s^{\text{val},v'})$. □

Inductive Case ($k > 0$)**.** The inductive case stores the valuation for the variables not in the state space. To determine the semantics $[\![\varphi]\!]^{\text{val}}_{v,v'}$ requires us to evaluate formulas of the form $y \leq y'$. We want to store sufficient information in the state space to evaluate these parts of the formula.

We solve this problem by storing variable orders in the state space.

Definition 54. A *variable order* Ψ is a sequence $y_{l_1}\ \theta_1\ y_{l_2}\ \theta_2\ \ldots\ \theta_{m-1}\ y_{l_m}$ where the variables are ordered by $\theta_1, \ldots, \theta_{m-1} \in \{=, <\}$.
We call Ψ *induced* by a valuation val, if for all variables y, y' with $\text{val}(y) \leq \text{val}(y')$, if there are indecies $l_j < l_{j'}$ with $y = y_{l_j}$ and $y' = y_{l_{j'}}$. I.e. Ψ contains a sequence $y\ \theta_j\ \ldots\ \theta_{j'-1}\ y'$, transitivly implying $y \leq y'$.

We introduce semantics to first order formulas under variable orders and the letter functions from the base case.

Definition 55. Let s_1, s_2 be functions $s_1, s_2 : \{y_1, \ldots, y_m\} \to [0..max_k]$. We define semantics $[\![\varphi]\!]^{\Psi}_{s_1,s_2}$ for quantor free formulas. The boolean and constant semantics are identical to the previous semantics. For the other terms and formulas,

$$\begin{aligned} [\![s_1(y)]\!]^{\Psi}_{s_1,s_2} &:= s_1(y) \\ [\![s_2(y)]\!]^{\Psi}_{s_1,s_2} &:= s_2(y) \\ [\![t_1 = t_2]\!]^{\Psi}_{s_1,s_2} &:= [\![t_1]\!]^{\Psi}_{s_1,s_2} = [\![t_2]\!]^{\Psi}_{s_1,s_2} \\ [\![y \leq y']\!]^{\Psi}_{s_1,s_2} &:= \begin{cases} true & y \leq y' \text{ by } \Psi \\ false & y > y' \text{ by } \Psi \end{cases}. \end{aligned}$$

Expectedly, given a valuation val and a variable order Ψ induced by val, $[\![\varphi]\!]^{\text{val}}_{v,v'} = [\![\varphi]\!]^{\Psi}_{s^{\text{val},v}, s^{\text{val},v'}}$.

The starting configuration is $(start, w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, \gamma_3)$ where $w_1 = \text{Ind}_k(v)$ and $w_2 = \text{Ind}_k(v')$. They are of the form

$$w_1 = v_0 x_0 \ldots v_{max_k} x_{max_k}, \qquad w_2 = v'_0 x_0 \ldots v'_{max_k} x_{max_k}.$$

Again we want to simulate the game $G_{v,v',\sim}$. We repeat the base case whilest storing the valuation on stack 2 instead of using the state space.

To create the valuation for a variable y_l, the players shall push a $(k-1)$-indexing of position a position $p_l \in [0..max_k]$ on stack 2. Like in the base case, Q_l determines the player responsible for choosing the position p_l.

The sequence to push is $\mathrm{Ind}_{k-1}(\mathrm{msbf}(p_l)) = p_{l0} z_0 \dots p_{l\,max_{k-1}} z_{max_{k-1}}$. The player responsible for choosing the position p_l pushes the symbols $p_{l0}, \dots, p_{l\,max_{k-1}} \in \Sigma_k$. In between, Eve is responsible for pushing the correct $(k-2)$-indexings $z_0, \dots, z_{max_{k-1}}$. At the end of pushing the valuation for a single variable, Ana may choose to either disbelieve the validity of the final sequence via $(k-1)$-ValidIndexing(Σ_k) or continue the play with the valuation of the next variable.

Afterwards, the whole valuation is stored on stack 2. To find the correct variable order, we use another instance of Guess & Check. Eve simply suggests a variable order and puts it in the state. Ana may disbelieve the variable order and enter a checking routine. If Ana believes the variable order, we enter Evewin or Anawin dependening on the evaluation of the formula under Ψ and the s_1, s_2 functions.

We construct this gadget by iterativly giving parts of the transition rules. Each time follows a lemma stating existing strategies within these parts. At the end follows a lemma stating the actual correctness of the whole gadget.

Instantiate the gadget $(k-1)$-ValidIndexing(Σ_k) on stack 2 with starting state disbelieveValidity.

The states $(\mathrm{pushDigit}, l, s_1, s_2)$, $(\mathrm{pushLastDigit}, l, s_1, s_2)$ belong to Eve, if $Q_l = \exists$. Respectivly, they are owned by Ana, if $Q_l = \forall$. The states $(\mathrm{believeValidity}, l, s_1, s_2)$, $(\mathrm{pushIndexing}, l, s_1, s_2)$ belong to Eve and all other states belong to Ana. For all $1 \le l < m$, create the following transisiton rules:

$$\begin{aligned}
&(\mathrm{start}, 2, (\mathrm{Quant}, 1, \bot^{\{y_1,\dots,y_m\}}, \bot^{\{y_1,\dots,y_m\}}))\\
&((\mathrm{Quant}, l, s_1, s_2), 2, (\mathrm{pushIndexing}, l, s_1, s_2))\\
&((\mathrm{pushIndexing}, l, s_1, s_2), 2, \Sigma^*_{<k}, (\mathrm{pushDigit}, l, s_1, s_2))\\
&((\mathrm{pushIndexing}, l, s_1, s_2), 2, \Sigma^*_{<k}, (\mathrm{pushLastDigit}, l, s_1, s_2))\\
&((\mathrm{pushDigit}, l, s_1, s_2), 2, \Sigma_k, (\mathrm{pushIndexing}, l, s_1, s_2))\\
&((\mathrm{pushLastDigit}, l, s_1, s_2), 2, \Sigma_k, (\mathrm{Pushed}, l, s_1, s_2))\\
&((\mathrm{Pushed}, l, s_1, s_2), 2, \mathrm{disbelieveValidity})\\
&((\mathrm{Pushed}, l, s_1, s_2), 2, (\mathrm{believeValidity}, l, s_1, s_2))
\end{aligned}$$

Lemma 56. *Let x_{p_l} be a $(k-1)$-indexing of $\mathrm{msbf}(p_l) = p_{l0} \dots p_{l\,max_{k-1}} \in \Sigma_k^*$. Let $\square = own((Quant, l, s_1, s_2))$ and $\blacksquare$ their opponent. For readability, let $\lambda_1 = w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2$ and $\lambda_2 = y_{l-1}x_{p_{l-1}} \dots y_1 x_{p_1}\gamma_3$.*
For every $p_l \in [0..max_k]$, $\square$ possesses a strategy from

$$\begin{aligned}
&((\mathrm{pushIndexing}, l, s_1, s_2), \lambda_1, \lambda_2)\\
\textit{to } &((\mathrm{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2)
\end{aligned}$$

Player $\blacksquare$ *possesses a strategy from*

$$\begin{aligned}&((\text{pushIndexing}, l, s_1, s_2), \lambda_1, \lambda_2)\\ \textit{to}\ &\{((\text{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2) \mid p_l \in [0..max_k]\}\end{aligned}$$

Proof. We provide the strategies $\sigma_{\text{Eve}}, \sigma_{\text{Ana}}$ for each player. The strategy for Eve contains

$$\sigma_{\text{Eve}}(((\text{pushIndexing}, l, s_1, s_2), \lambda_1, \lambda_2)) = ((\text{pushDigit}, l, s_1, s_2), \lambda_1, z_{max_{k-1}}\lambda_2)$$

where $z_{max_{k-1}} = \text{Ind}_{k-2}(\text{msbf}(max_{k-1})$. Also,

$$\sigma_{\text{Eve}}(((\text{pushIndexing}, l, s_1, s_2), \lambda_1, p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2)) =$$
$$\begin{cases}((\text{pushDigit}, l, s_1, s_2), \lambda_1, z_u p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2) & \text{if } u \neq 0\\ ((\text{pushLastDigit}, l, s_1, s_2), \lambda_1, z_u p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2) & \text{if } u = 0\end{cases}$$

where $z_u = \text{Ind}_{k-2}(\text{msbf}(u))$ ($\text{msbf}(u) \in \Sigma_{k-1}^*$).

The strategy for $\square$ contains

$$\sigma_{\square}(((\text{pushDigit}, l, s_1, s_2), \lambda_1, z_u p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2)) =$$
$$\begin{cases}((\text{pushIndexing}, l, s_1, s_2), \lambda_1, p_{l_u}z_u p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2) & \text{if } 0 \leq u\\ ((\text{pushIndexing}, l, s_1, s_2), \lambda_1, s z_u p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2) & \text{if } u < 0\\ & \text{for } s \in \Sigma_k\end{cases}$$

$$\begin{aligned}\sigma_{\square}(((\text{pushLastDigit}, l, s_1, s_2), \lambda_1, z_u p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2)) =\\ ((\text{Pushed}, l, s_1, s_2), \lambda_1, p_{l_u}z_u p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2)\end{aligned}$$

The strategy for Ana contains

$$\sigma_{\text{Ana}}(((\text{Pushed}, l, s_1, s_2), \lambda_1, x\lambda_2)) =$$
$$\begin{cases}(\text{disbelieveValidity}, \lambda_1, x\lambda_2) & \text{if } x \text{ is not a } k-1\text{-indexing}\\ ((\text{believeValidity}, l, s_1, s_2), \lambda_1, x\lambda_2) & \text{otherwise}\end{cases}$$

To the first point of the lemma, let $p_l \in [0..max_k]$. Let there be a play compliant with $\sigma_{\square}$.

Case 1 ($\square = \text{Eve}$)*:* By σ_{Eve}, the play first visits $((\text{pushDigit}, l, s_1, s_2), \lambda_1, z_{max_{k-1}}\lambda_2)$. From there, whenever a position

$$((\text{pushIndexing}, l, s_1, s_2), \lambda_1, p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2)$$

is visited, strategy σ_{Eve} pushes $z_u = \text{Ind}_{k-2}(\text{msbf}(u))$. Also, from

$$((\text{pushDigit}, l, s_1, s_2), \lambda_1, z_u p_{l_{u+1}}z_{u+1} \dots p_{l_{max_{k-1}}}z_{max_{k-1}}\lambda_2),$$

σ_{Eve} pushes p_{l_u}. This continues up until $u = 0$, where the play continues to

$$((\text{pushLastDigit}, l, s_1, s_2), \lambda_1, z_0 p_{l_1} z_1 \dots p_{l_{max_{k-1}}} z_{max_{k-1}} \lambda_2)$$

and then to $((\text{Pushed}, l, s_1, s_2), \lambda_1, p_{l_0} z_0 p_{l_1} z_1 \dots p_{l_{max_{k-1}}} z_{max_{k-1}} \lambda_2)$.

The stack content of stack 2 is now $p_{l_0} z_0 p_{l_1} z_1 \dots p_{l_{max_{k-1}}} z_{max_{k-1}}$ where each $z_u = \text{Ind}_{k-2}(\text{msbf}(u))$. This meets the definition of a $(k-1)$-indexing. Thus,

$$p_{l_0} z_0 p_{l_1} z_1 \dots p_{l_{max_{k-1}}} z_{max_{k-1}} = x_{p_l}.$$

If Ana continues the play to $(\text{disbelieveValidity}, \lambda_1, x_{p_l}\lambda_2)$, the starting state of $(k-1)$-ValidIndexing(Σ_k), Eve possesses a winning strategy from there by induction (Theorem 49). Otherwise, the play continues to $((\text{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2)$.

Case 2 ($\Box = \text{Ana}$)*:* By construction, the play first visits

$$((\text{pushDigit}, l, s_1, s_2), \lambda_1, z_{max_{k-1}} \lambda_2)$$

for some $z_{max_{k-1}} \in \Sigma^*_{<k}$. By strategy σ_{Ana}, each time the play visits

$$((\text{pushDigit}, l, s_1, s_2), \lambda_1, z_u p_{l_{u+1}} z_{u+1} \dots p_{l_{max_{k-1}}} z_{max_{k-1}} \lambda_2),$$

it pushes p_{l_u}, leading back to

$$((\text{pushIndexing}, l, s_1, s_2), \lambda_1, p_{l_u} z_u p_{l_{u+1}} z_{u+1} \dots p_{l_{max_{k-1}}} z_{max_{k-1}} \lambda_2),$$

which by construction either again visits state $(\text{pushDigit}, l, s_1, s_2)$ or $(\text{pushLastDigit}, l, s_1, s_2)$. If the play never visits a position in state $(\text{pushLastDigit}, l, s_1, s_2)$, it is winning for Ana due to the reachability winning condition. If the play visits

$$((\text{pushLastDigit}, l, s_1, s_2), \lambda_1, z_u p_{l_{u+1}} z_{u+1} \dots p_{l_{max_{k-1}}} z_{max_{k-1}} \lambda_2)$$

for some u, strategy σ_{Ana} pushes p_{l_u} and visits

$$((\text{Pushed}, l, s_1, s_2), \lambda_1, p_{l_u} z_u p_{l_{u+1}} z_{u+1} \dots p_{l_{max_{k-1}}} z_{max_{k-1}} \lambda_2).$$

Be aware, that Eve could have continued this to $u \neq 0$. In this case or if one of the other conditions for a $(k-1)$-indexing is violated, then

$$x = p_{l_u} z_u p_{l_{u+1}} z_{u+1} \dots p_{l_{max_{k-1}}} z_{max_{k-1}} \in \Sigma^*_{\leq k}$$

is not a $(k-1)$-indexing. Then σ_{Ana} continues the play to $(\text{disbelieveValidity}, \lambda_1, x\lambda_2)$ and Ana possesses a winning strategy from there by induction (Theorem 49). Otherwise, x must be a $(k-1)$-indexing, i.e. $x = \text{Ind}_{k-1}(\text{msbf}(p_l)) = x_{p_l}$ and the play continues to $((\text{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2)$.

The proof for the second part of the lemma is analouge to the first. The strategies only differ in that ■ is not choosing the symbols $p_{l0}, \dots, p_{l\,max_{k-1}}$. By construction however, the symbols are from Σ_k. A $(k-1)$-indexing of a word in $\Sigma_k^{max_k+1}$ ensures, that it indexes $\mathrm{msbf}(p_l) = p_{l0} \dots p_{l\,max_{k-1}}$ for some $p_l \in [0..max_k]$. □

By the above lemma, each player possesses a strategy forcing the play to visit a position in state $(\text{believeValidity}, l, s_1, s_2)$, where the top of stack 2 is $x_{p_l} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p_l))$, an $(k-1)$-indexing of the chosen position for $\mathrm{val}(y_l)$.

From there, Eve suggests values for $s_1(y_l)$ and $s_2(y_l)$ (just like the base case, but for reasons about the number of phases, we need this suggestion after each single variable). Ana may disbelieve these values. If they don't, the play continues with the next variable y_{l+1}.

States $(\text{suggest}, l, s_1, s_2)$ belong to Ana.

$$\begin{array}{lr}
((\text{believeValidity}, l, s_1, s_2), 2, (\text{suggest}, l, s_1[s/y_l], s_2[s'/y_l])) & s, s' \in \Sigma \\
((\text{suggest}, l, s_1, s_2), 2, (\text{disVal}, l, I, s_I)) & I \in \{1,2\} \\
((\text{suggest}, l, s_1, s_2), 2, y_l, (\text{Quant}, l+1, s_1, s_2)) & l < m
\end{array}$$

If Ana chooses to believe the suggested symbols for $s_1(y_l)$ and $s_2(y_l)$, they put the symbol y_l on top of the indexing as a marker and continue with the next quantor.

If Ana chooses to disbelieve one ot the values, Eve pops symbols from stack 1 until they claim to have found the position p_l in w_1 or w_2 respectivly. Ana believe the position. and the top of stack symbol is compared to $s_1(y_l)$ or $s_2(y_l)$ respectivly. If Ana disbelieves the position, the play continues to $(k-1)$-Equality(Σ_k) to verify it.

Instantiate $(k-1)$-ValidIndexing(Σ_k) on stack 1 with starting state disbelievePos.

States $(\text{disVal}, l, I, s_I)$, $(\text{believe}, l, I, s_I)$ and state disbelieve belong to Eve. All other states belong to Ana. introduce the following transition rules for $I \in \{1,2\}$.

$$\begin{array}{lr}
((\text{disVal}, l, 1, s_1), (\Sigma\Sigma_{\le k}^*)^*, 1, (\text{checkVal}, l, 1, s_1)) & \\
((\text{disVal}, l, 2, s_2), (\Lambda \setminus \Upsilon)^*\Upsilon(\Sigma\Sigma_{\le k}^*)^*, 1, (\text{checkVal}, l, 2, s_2)) & \\
((\text{checkVal}, l, I, s_I), 1, (\text{believe}, l, I, s_I)) & \\
((\text{checkVal}, l, I, s_I), 1, \text{disbelieve}) & \\
((\text{believe}, l, I, s_I), s_I(y_l), 1, \text{Evewin}) & \\
((\text{believe}, l, I, s_I), s, 1, \text{Anawin}) & s \in \Sigma,\ s \neq s_I(y_l) \\
(\text{disbelieve}, 1, \text{disbelievePos}) &
\end{array}$$

Lemma 57. *Let I be 1 or 2 and let $x_{p_l} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p_l))$ for some position $p_l \in [0..max_k]$.*

Eve possesses a winning strategy from

$$((\text{disVal}, l, I, s_I), w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, x_{p_l} \dots y_1 x_{p_1}\gamma_3)$$

(or $((\text{disVal}, l, I, s_I), w_1\sigma w_2\sigma_2\gamma_2, x_{p_l} \dots y_1 x_{p_1}\gamma_3)$) if and only if $s_1(y_l) = v_{p_l}$, if $I = 1$, or $s_2(y_l) = v'_{p_l}$, if $I = 2$.

Proof. W.l.o.g. $I = 1$. Assume Eve possesses a winning strategy σ_{Eve}. Let us look at a play compliant with σ_{Eve}. The first move from $(\text{disVal}, l, 1, s_1)$ is to

$$\sigma_{\text{Eve}}(((\text{disVal}, l, 1, s_1), w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, x_{p_l} \dots y_1 x_{p_1}\gamma_3)) = \\ ((\text{checkVal}, l, 1, s_1), v_u x_u \dots v_{max_k} x_{max_k}\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, x_{p_l} \dots y_1 x_{p_1}\gamma_3)$$

for some u of the strategy's choice. If the strategy would remove all symbols of w_1, or leave a part of some x_u on top of the stack, Eve has no moves if Ana transitions to $(\text{believe}, l, 1, s_1)$, because the top of stack symbol is not from Σ. In contradiction, σ_{Eve} would not be winning.

Case 1 $(u = p_l)$*:* Look at the play continuing to $(\text{believe}, l, 1, s_1)$. Now, since σ_{Eve} is winning for Eve, the next transition must be $((\text{believe}, l, 1, s_1), s_1(y_l), 1, \text{Evewin})$ Thus, $s_1(y_l) = v_{p_l}$.

Case 2 $(u \neq p_l)$*:* In this case, $x_{p_l} = \text{Ind}_{k-1}(\text{msbf}(p_l)) \neq \text{Ind}_{k-1}(\text{msbf}(u)) = x_u$. Look at the play continuing to disbelieve. I further continues to the position $(\text{disbelievePos}, x_u v_{u+1} x_{u+1} \dots v_{max_k} x_{max_k}\sigma_1\sigma w_2\sigma_2\gamma_2, x_{p_l} \dots y_1 x_{p_1}\gamma_3)$, where it enters gadget $(k-1)$-Equality(Σ_k). By Theorem 46, when $x_u \neq x_{p_l}$, Ana possesses a winning strategy from there, because x_{p_l} and x_u index different words. This contradicts σ_{Eve} being a winning strategy and this case does not occur.

Assume Ana possesses a winning strategy σ_{Ana}. Again, look at a play compliant with that strategy. In this play, Eve removes a sequence such that the resulting position is

$$((\text{checkVal}, l, 1, s_1), v_{p_l} x_{p_l} \dots v_{max_k} x_{max_k}\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, x_{p_l} \dots y_1 x_{p_1}\gamma_3).$$

Be aware, that in this play, Eve removed a sequence

Case 1 (σ_{Ana} moves to state $(\text{believe}, l, 1, s_1)$)*:* Since the strategy is winning for Ana, the next transition must be $((\text{believe}, l, 1, s_1), s, 1, \text{Anawin})$, where $s \neq s_1(y_l)$, proving $v_{p_l} \neq s_1(y_l)$.

Case 2 (σ_{Ana} moves to state disbelieve)*:* Analogue to the above Case 2, this case cannot occur.

Together, Eve possesses a winning strategy from $((\text{disVal}, l, I, s_I), w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, x_{p_l} \dots y_1 x_{p_1}\gamma_3)$ if and only if $s_I(y_l) = v_{I p_l}$. □

The above transitions give each player a strategy enforcing the creation of a valuation on stack 2. At the same time, they create s_1 and s_2 in the state. For these, we know by Lemma 57, that $s_1(y_l) = v_{p_l}$ and $s_2(y_l) = v'_{p_l}$, where stack 2 contains the valuation for y_l in the form of a sequence $y_l x_{p_l}$ with $x_{p_l} = \text{Ind}_{k-1}(\text{msbf}(p_l))$.

When Eve suggests the symbols for the last variable's valuation, the play enters a position

$$((\text{suggest}, m, s_1, s_2), w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3).$$

Again, Ana could disbelieve as covered by Lemma 57.

Otherwise, we start by creating the variable order to evaluate the formula. Again, we do so by Guess & Check. Eve suggests a variable order Ψ and Ana may choose to check for correctness of the variable order by disbelieving a single $y_l \theta y_{l'}$ of Ψ.

States $(\text{guessOrder}, s_1, s_2)$ are owned by Eve. All other states belong to Ana. Create the following transition rules for all $s_1, s_2 : \{y_1, \dots, y_m\} \to \Sigma$ and all variable orders Ψ of $y_1, \dots, y_m$.

$$\begin{array}{ll}
((\text{suggest}, m, s_1, s_2), 2, (\text{guessOrder}, s_1, s_2)) & \\
((\text{guessOrder}, s_1, s_2), 2, (\text{Order}, \Psi, s_1, s_2)) & \\
((\text{Order}, \Psi, s_1, s_2), 2, (\text{disbelieve}, y_l \theta y_{l'})) & \text{where } y_l \theta y_{l'} \text{ is part of } \Psi \\
((\text{Order}, \Psi, s_1, s_2), 2, (\Psi, s_1, s_2)) & \\
((\Psi, s_1, s_2), 2, \text{Evewin}) & \text{if } [\![\varphi]\!]^{\Psi}_{s_1,s_2} = \mathit{true} \\
((\Psi, s_1, s_2), 2, \text{Anawin}) & \text{if } [\![\varphi]\!]^{\Psi}_{s_1,s_2} = \mathit{false}
\end{array}$$

For disbelieving Ψ, instantiate $(k-1)$-θ-Check(Σ_k) on state $(\theta\text{check}, y_l)$ for every $l \in [1..m]$ and $\theta \in \{<, >, =\}$ and add the following transition rules. The states belong to Ana.

$$\begin{array}{ll}
((\text{disbelieve}, y_l\ \theta\ y_{l'}), (\Lambda \setminus \{y_l\})^* y_l, 2, (\theta\text{check}, y_{l'})) & \text{if } l > l' \\
((\text{disbelieve}, y_l = y_{l'}), (\Lambda \setminus \{y_{l'}\})^* y_{l'}, 2, (=\text{check}, y_{l'})) & \text{if } l < l' \\
((\text{disbelieve}, y_l < y_{l'}), (\Lambda \setminus \{y_{l'}\})^* y_{l'}, 2, (>\text{check}, y_{l'})) & \text{if } l < l'
\end{array}$$

Lemma 58. *Let* $x_{p_m} = \text{Ind}_{k-1}(\text{msbf}(p_m)), \dots, x_{p_1} = \text{Ind}_{k-1}(\text{msbf}(p_1))$, $\theta \in \{<, =\}$ *and* $l \neq l' \in [1..m]$. *For readability, let* $\lambda = w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2$.
Eve possesses a winning strategy from $((\text{disbelieve}, y_l\theta y_{l'}), \lambda, y_m x_{p_m} \dots y_1 x_{p_1} \gamma_3)$ *if and only if* $p_l\ \theta\ p_{l'}$.

Proof.

Case 1 $(l > l')$*:* Any play continues to $((\theta\text{check}, y_{l'}), \lambda, x_{p_l} \dots y_1 x_{p_1} \gamma_3)$, from which by induction (Theorem 51), Eve possesses a winning strategy, if and only if $p_l\ \theta\ p_{l'}$.

Case 2 $(l < l')$*:* Any play continues to $((\theta'\text{check}, y_l), \lambda, x_{p_{l'}} \dots y_1 x_{p_1} \gamma_3)$, where

$$\theta' = \begin{cases} = & \text{if } \theta \text{ is } = \\ > & \text{if } \theta \text{ is } < \end{cases}$$

Again by induction, Eve possesses a winning strategy if and only if $p_{l'}\ \theta'\ p_l$, i.e. $p_l\ \theta\ p_{l'}$.

□

All the above transition rules define a game $G_{k,\sim} = (V_{k,\sim}, E_{k,\sim}, \{\text{Evewin}\})$.
Number of contexts/phases: Any play can only take one of the following routes:

- push a part of the valuation on stack 2 but Ana decides to disbelieve one of the indexings with $(k-1)$-ValidIndexing(Σ_k) instantiated on stack 2 (by the use of transition $((\text{Pushed}, l, s_1, s_2), 2, \text{disbelieveValidity}))$.
- push a part of the valuation on stack 2 but Ana decides to disbelieve s_1 or s_2 (by taking a transition $((\text{suggest}, l, s_1, s_2), 2, (\text{disVal}, l, I, s_I))$ for $I \in \{1, 2\}$). Eve pops symbols from stack 1 to find the corresponding position. The play either continues to Evewin or Anawin, or Ana disbelieves the position and continues to $(k-1)$-Equality(Σ_k) instantiated on stack 1.
- push and complete the valuation on stack 2, but Ana decides to disbelieve the suggested variable order. (by taking a transition $((\text{Order}, \Psi, s_1, s_2), 2,$ $(\text{disbelieve}, y_l\theta y_{l'})))$. The play removes symbols stack 2 to find the first variable and then continues in $(k-1)$-θ-Check(Σ_k).

In any case, pushing symbols on stack 2 introduces the first context on stack 2.

In the first case, $(k-1)$-ValidIndexing(Σ_k) takes at most $(k-1)+1$ additional contexts by induction (Theorem 49), adding up to $k+1$ contexts.

In the second case, removing symbols from stack 1 introduces the second context. If the play then visits a winning state, the context does not change anymore. If the play continues to $(k-1)$-Equality(Σ_k), it takes at most an additional k contexts (Theorem 46). This adds up to $k+2$ contexts.

In the third case, $(k-1)$-θ-Check$(\Sigma_k, \{y_1, \dots, y_m\})$ takes up to an additional $(k-1)+1 = k$ contexts by induction (Theorem 51), since the context was on stack 2. This case adds up to $k+1$ contexts.

Overall, the most contexts that can occur are $k+2$. In every case, the first context is introduced on stack 2, If the previous context was on stack 2, this adds only up to $k+1$ contexts.

For the phases, this differs a little.

In the first case, no pop transitions are occuring in the play until we transition to $(k-1)$-ValidIndexing(Σ_k), which by induction (Theorem 49) takes up to $k-1$ additional phases, if the previous phase phase was on stack 2 and up to k otherwise.

In the second case, the pop transitions on stack 1 introduce a new phase on stack one. Then, $(k-1)$-Equality(Σ_k) takes up to $(k-1)+1 = k$ additional phases (Theorem 46). In total, if the previous phase was on stack 1, this takes at most k additional phases and $k+1$ otherwise.

In the third case, the pop transitions on stack 2 introduce a phase. Then, by induction (Theorem 51), $(k-1)$-θ-Check$(\Sigma_k, \{y_1, \dots, y_m\})$ adds up to $k-1$ additional phases, since the phase is on stack 2 already. If the previous phase was on stack 2, this takes at most $k-1$ additional contexts. Otherwise, it takes up to k phases.

In total, plays of this gadget take at most $k+1$ phases. If the previous phase was on stack 1, this reduces to k additional phases.

Number of states: The number of states is dependend on the number quantors m, the number of functions s_1, s_2 and the number of possible variable orders Ψ. Remember that we don't count the states of subgadgets. The number of functions s_1, s_2 is $|\Sigma|^m$. The number of possible variable orders is $m! \cdot 2^m$. In total, the number of states is polynomial in the size of Σ, since the number of quantors m is constant for each first order relation.

Lemma 59. *Let $v, v' \in \Sigma^{max_k}$ and $w_1 = \mathrm{Ind}_k(v)$, $w_2 = \mathrm{Ind}_k(v')$. Let $\sigma \in \Upsilon$, $\sigma_1, \sigma_2 \in \Lambda \setminus (\Upsilon \cup \Sigma \cup \Sigma_{\leq k})$ and $\sim$ a first order relation. Let $\gamma_1 \in (\Lambda \setminus \Upsilon)^*$ and $\gamma_2, \gamma_3 \in \Lambda^*$.*
Eve possesses a winning strategy from $(\mathrm{start}, w_1\sigma_1\gamma_1\sigma w_2\sigma_2\gamma_2, \gamma_3)$ if and only if $v \sim v'$.

Proof. We show that $G_{k,\sim}$ simulates $G_{\sim,v,v'}$ via the mapping $f : V_{\sim,v,v'} \to V_{k,\sim}$.

$$\begin{aligned} f(\mathrm{Quant}l, \mathrm{val}) &= ((\mathrm{Quant}, l, s_1, s_2), w_1\sigma_1\gamma_1\sigma_2 w_2\sigma_3\gamma_2, y_l x_{p_l} \dots y_1 x_{p_1}\gamma_3) \\ f(\mathrm{Quant}(m+1), \mathrm{val}) &= ((\mathrm{guessOrder}, s_1, s_2), w_1\sigma_1\gamma_1\sigma_2 w_2\sigma_3\gamma_2, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3) \\ f(\Box\mathrm{win}) &= \Box\mathrm{win} \end{aligned}$$

where $def(\mathrm{val}) = \{y_1, \dots, y_l\}$, $x_{p_l} = \mathrm{Ind}_{k-1}(\mathrm{val}(y_l))$ and $s_1 = s^{\mathrm{val},v}, s_2 = s^{\mathrm{val},v'}$ restricted to $def(\mathrm{val})$.

For human readability, we shorten the stack contents in the following way:

$$\lambda_1 = w_1\sigma_1\gamma_1\sigma w_2\sigma_3\gamma_2, \quad \lambda_2 = y_l x_{p_l} \dots y_1 x_{p_1}\gamma_3$$

Be aware, that these contain the indexings w_1 and w_2 as well as the indexings

$$x_{p_l} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p_l)), \; \dots, \; x_{p_1} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p_1)).$$

The strategies can act depending on this knowledge.

We need to show that

- for every position $p_l \in [0..max_k]$ and valuation $\mathrm{val} : \{y_1, \dots, y_{l-1}\} \to [0..max_k]$, the respective player for quantor Q_l possesses a strategy from

$$f((\mathrm{Quant}, l, \mathrm{val})) = ((\mathrm{Quant}, l, s_1, s_2), \lambda_1, \lambda_2)$$

  to

$$f((\mathrm{Quant}, l, \mathrm{val}[p_l/y_l])) = ((\mathrm{Quant}, l, s_1[v_{p_l}/y_l], s_2[v'_{p_l}/y_l]), \lambda_1, y_l x_{p_l}\lambda_2),$$

  where $x_{p_l} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p_l))$.

- for every valuation $\mathrm{val} : \{y_1, \dots, y_{l-1}\} \to [0..max_k]$ the opponent of player Q_l possesses a strategy from $f((\mathrm{Quant}, l, \mathrm{val})) = ((\mathrm{Quant}, l, s_1, s_2), \lambda_1, \lambda_2)$ to

$$\begin{aligned} f(\{(\mathrm{Quant}, l, \mathrm{val}[p_l/y_l]) \mid p_l \in [0..max_k]\}) = \\ \{((\mathrm{Quant}, l, s_1[v_{p_l}/y_l], s_2[v'_{p_l}/y_l]), \lambda_1, y_l x_{p_l}\lambda_2) \mid \\ x_{p_l} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p_l)) \text{ for some } p_l \in [1..max_k]\} \end{aligned}$$

- for every valuation val : $\{y_1, \dots, y_m\} \to [0..max_k]$, each player possesses a strategy from

$$f((\mathrm{Quant}, m+1, \mathrm{val})) = ((\mathrm{guessOrder}, s^{\mathrm{val},v}, s^{\mathrm{val},v'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1} \gamma_3)$$

to

$$\begin{aligned} f(\mathrm{Evewin}) &= \mathrm{Evewin} && \text{if } [\![\varphi]\!]^{\mathrm{val}}_{v,v'} = \mathit{true} \\ f(\mathrm{Anawin}) &= \mathrm{Anawin} && \text{if } [\![\varphi]\!]^{\mathrm{val}}_{v,v'} = \mathit{false} \end{aligned}$$

To the first two points, let $f((\mathrm{Quant}, l, \mathrm{val})) = ((\mathrm{Quant}, l, s_1, s_2), \lambda_1, \lambda_2)$ and $p_l \in [0..max_k]$. We define strategies $\sigma_{\mathrm{Eve}}, \sigma_{\mathrm{Ana}}$ for each player. Let $\square = \mathrm{Eve}$, if $Q_l = \exists$, and $\square = \mathrm{Ana}$, if $Q_l = \forall$. Let $\blacksquare$ be their opponent.

Let $x_{p_l} = \mathrm{Ind}_{k-1}(\mathrm{msbf}(p_l))$. By Lemma 56, $\square$ possesses a strategy from

$$\begin{aligned} &((\mathrm{Quant}, l, s_1, s_2), \lambda_1, \lambda_2) \\ \text{to } &((\mathrm{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2). \end{aligned}$$

for each $p_l \in [0..max_k]$. Also, player $\blacksquare$ possesses a strategy from

$$\begin{aligned} &((\mathrm{Quant}, l, s_1, s_2), \lambda_1, \lambda_2) \\ \text{to } &\{((\mathrm{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2) \mid p_l \in [0..max_k]\}. \end{aligned}$$

The strategy for Eve also contains

$$\begin{aligned} &\sigma_{\mathrm{Eve}}(((\mathrm{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2)) = \\ &\qquad\qquad ((\mathrm{suggest}, l, s_1[v_{p_l}/y_l], s_2[v'_{p_l}/y_l]), \lambda_1, x_{p_l}\lambda_2) \end{aligned}$$

The strategy for Ana also contains

$$\begin{aligned} &\sigma_{\mathrm{Ana}}(((\mathrm{suggest}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2)) = \\ &\qquad \begin{cases} ((\mathrm{disVal}, l, 1, s_1), \lambda_1, x_{p_l}\lambda_2) & \text{if } s_1(y_l) \neq v_{p_l} \\ ((\mathrm{disVal}, l, 2, s_2), \lambda_1, x_{p_l}\lambda_2) & \text{if } s_2(y_l) \neq v'_{p_l} \\ ((\mathrm{Quant}, l, s_1, s_2), \lambda_1, y_l x_{p_l}\lambda_2) & \text{otherwise} \end{cases} \end{aligned}$$

To the first point, let there be a play compliant with $\sigma_\square$ starting in $((\mathrm{Quant}, l, s_1, s_2), \lambda_1, \lambda_2)$. By construction, the next position then is $((\mathrm{pushIndexing}, l, s_1, s_2), \lambda_1, \lambda_2)$. By Lemma 56, the play is either winning for $\square$ or it visits the configuration $((\mathrm{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2)$.

Case 1 ($\square$ = Eve)*:* The next position is $((\mathrm{suggest}, l, s_1[v_{p_l}/y_l] s_2[v'_{p_l}/y_l]), \lambda_1, x_{p_l}\lambda_2)$ by strategy σ_{Eve}. Then, the play may either continue to $((\mathrm{disVal}, l, 1, s_1[v_{p_l}/y_l]), \lambda_1, x_{p_l}\lambda_2)$ or $((\mathrm{disVal}, l, 2, s_2[v'_{p_l}/y_l]), \lambda_1, x_{p_l}\lambda_2)$, both winning for player $\square$ = Eve by Lemma 57, since $s_1(y_l) = v_{p_l}$ and $s_2(y_l) = v'_{p_l}$. Or the play continues to

$$f((\mathrm{Quant}, l, \mathrm{val}[p_l/y_l])) = ((\mathrm{Quant}, l, s_1[v_{p_l}/y_l], s_2[v'_{p_l}/y_l])), \lambda_1, y_l x_{p_l}\lambda_2).$$

Case 2 ($\Box = \forall$)*:* The play continues to $((\text{suggest}, l, s_1[s/y_l], s_2[s'/y_l]), \lambda_1, x_{p_l}\lambda_2)$ for some $s, s' \in \Sigma$. If $s \neq v_{p_l}$ or $s' \neq v'_{p_l}$, σ_{Ana} continues the play to $((\text{disVal}, l, 1, s_1[s/y_l]), \lambda_1, x_{p_l}\lambda_2)$ or $((\text{disVal}, l, 2, s_2[s'/y_l]), \lambda_1, x_{p_l}\lambda_2)$ respectivly. Both positions are winning for player $\Box = \forall$ by Lemma 57, since $s_1[s/y_l](y_l) = s \neq v_{p_l}$ or $s_2[s'/y_l](y_l) = s' \neq v'_{p_l}$ respectivly. Otherwise, the play continues to

$$f((\text{Quant}, l, \text{val}[p_l/y_l])) = ((\text{Quant}, l, s_1[v_{p_l}/y_l], s_2[v'_{p_l}/y_l]), \lambda_1, y_l x_{p_l}\lambda_2).$$

To the second point, let there be a play compliant with strategy $\sigma_\blacksquare$ starting in $((\text{Quant}, l, s_1, s_2), \lambda_1, \lambda_2)$. By construction, the next position is $((\text{pushIndexing}, l, s_1, s_2), \lambda_1, \lambda_2)$. By Lemma 56, the play is either winning for player $\blacksquare$ or it visits the configuration $((\text{believeValidity}, l, s_1, s_2), \lambda_1, x_{p_l}\lambda_2)$ for some $p_l \in [0..max_k]$.

Analogue to Case 1 and Case 2 above, the play is either winning for player $\blacksquare$ or continues to

$$f((\text{Quant}, l, \text{val}[p_l/y_l])) = ((\text{Quant}, l, s_1[v_{p_l}/y_l], s_2[v'_{p_l}/y_l]), \lambda_1, y_l x_{p_l}\lambda_2).$$

To the third point, look at the outgoing transition rules of state $(\Psi, s^{\text{val},v}, s^{\text{val},v'})$. We know that $[\![\varphi]\!]^{\text{val}}_{v,v'} = [\![\varphi]\!]^{\Psi}_{s^{\text{val},v}, s^{\text{val},v'}}$, when Ψ is induced by val. It is then sufficient to show that each player possesses a strategy from

$$f((\text{Quant}, m+1, \text{val})) = ((\text{guessOrder}, s^{\text{val},v}, s^{\text{val},v'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3)$$

to state $(\Psi, s^{\text{val},v}, s^{\text{val},v'})$, where Ψ is a variable order induced by val.

For Eve, this strategy is

$$\sigma_{\text{Eve}}(((\text{guessOrder}, s^{\text{val},v}, s^{\text{val},v'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3)) = \\ ((\text{Order}, \Psi, s^{\text{val},v}, s^{\text{val},v'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3)$$

where Ψ is a variable order induced by val. Be aware that val is part of the previous configuration, encoded into the indexings $x_{p_1}, \dots, x_{p_m}$ on stack 2. Let there be a play from $((\text{guessOrder}, s^{\text{val},v}, s^{\text{val},v'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3))$ compliant with σ_{Eve}. The next position is in state $(\text{Order}, \Psi, s^{\text{val},v}, s^{\text{val},v'})$. From there, the play continues either to state $(\text{disbelieve}, y_l \theta y_{l'})$ where $y_l\ \theta\ y_{l'}$ is part of Ψ. Since Ψ is induced by val, for each $y_l\ \theta\ y_{l'}$ in it, $p_l = \text{val}(y_l)\ \theta\ \text{val}(y_{l'}) = p_{l'}$. This is a winning position for Eve by Lemma 58. Otherwise, the play continues to state $(\Psi, s^{\text{val},v}, s^{\text{val},v'})$, where Ψ is the variable order induced by val.

For Ana, this strategy is

$$\sigma_{\text{Ana}}(((\text{Order}, \Psi, s^{\text{val},v}, s^{\text{val},v'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3) = \\ \begin{cases} ((\text{disbelieve}, y_l \theta y_{l'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3) & \text{if } y_l\ \theta\ y_{l'} \text{ in } \Psi \text{ where} \\ & p_l\ \theta\ p_{l'} \text{ does not hold} \\ ((\Psi, s^{\text{val},v}, s^{\text{val},v'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1}\gamma_3) & \text{otherwise} \end{cases}$$

Let there be a play from $((\text{guessOrder}, s_1, s_2), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1} \gamma_3)$ compliant with σ_{Ana}. The next position is in state $(\text{Order}, \Psi, s^{\text{val},v}, s^{\text{val},v'})$ for some variable order Ψ. If Ψ contains a $y_l \theta y_{l'}$, such that not $p_l \theta p_{l'}$, i.e. Ψ is not induced by val, then the next position by σ_{Ana} is $(\text{disbelieve}, y_l \theta y_{l'}))$, which is winning by Lemma 58. Otherwise, the play continues to $((\Psi, s^{\text{val},v}, s^{\text{val},v'}), \lambda_1, y_m x_{p_m} \dots y_1 x_{p_1} \gamma_3)$, where Ψ is induced by val. □

4.5 Simulating a Turing machine with an mpdg

Now we have all tools at hand to construct the multi-pushdown game simulating a Turing machine for the proof of the lower bound (Theorem 32).

Making use of the deterministic time – alternating space theorem for turing machines, we transform an alternating turing machine with $k-1$-EXP space bound with input w into a multi-pushdown game. Any play in the constructed game will not exceed the $(k+2)$'th context or the k'th phase.

The game shall simulate the game from section 4.1.1 which was equivalent to an alternating Turing machine. By simulating that game, Eve will possess a winning strategy if the Turing machine accepts the input. The simulation idea is to store the configurations on a stack. Each player chooses the next transition of configurations in their respective branching state.

The mpdg needs to ensure that actual successing configurations are pushed onto the stack. Due to the immense size of the configurations, we use the gadgets from the previous sections and index the configurations. Given a chosen transition and a top of stack indexing of the current configuration, we use a Guess & Check approach to create the indexing containing the successing configuration on top of it.

Let $M = (Q, \Sigma, \Gamma, \Delta, q_{init})$ be an alternating decider with its space bound $sb \in \exp_{k-1}(n^{\mathcal{O}(1)})$ and branching assignment $\text{br} : Q \to \{\exists, \forall\}$. Let $w \in \Sigma^*$ be an input word for M.

We construct a reachability multi-pushdown game $(P, \{\text{Evewin}\})$ with two stacks that fulfills the following property, which completes the proof for Theorem 32.

Theorem 60. *Eve possesses a winning strategy from* $(\text{start}, \varepsilon, \varepsilon)$ *in* $(P, \{Evewin\})$ *if and only if* $w \in L(M)$.
P has polynomial size in $n = |w|$.
Any play from $(\text{start}, \varepsilon, \varepsilon)$ *has at most* $k+2$ *contexts and at most* k *phases.*

We want $(P, \{\text{Evewin}\})$ to simulate G_M from section 4.1.1. The initial position of G_M is the initial Turing machine configuration (q_{init}, c_{init}). We need to put an indexing of the initial configuration on stack 1, before we can start simulating G_M. This is achieved by using Guess & Check: Eve shall push the $(k-1)$-level indexing of the initial configuration word c_{init} followed by the state q_{init} on stack 1. Then, Ana may check the pushed sequence or believe it and start the simulation.

The following transition rules are introduced. Instantiate $(k-1)$-ValidIndexing($\overline{\Gamma}$) on stack 1 with starting state checkIndexing. Eve owns state *start*, Ana owns state initialConfPushed.

$$(\mathit{start}, 1, (\_\Sigma^*_{\leq k})^* w_0 \Sigma^*_{\leq k} \ldots w_{n-2}\Sigma^*_{\leq k}(w_{n-1}, \dagger)\Sigma^*_{\leq k}\#, \text{initialConfPushed})$$
$$(\text{initialConfPushed}, 1, \text{checkIndexing})$$
$$(\text{initialConfPushed}, 1, q_{init}, (\text{state}, q_{init}))$$

Lemma 61. *Let* $x_{init} = \text{Ind}_{k-1}(\_^{sb(n)-n} w_0 \ldots w_{n-2}(w_{n-1}, \dagger))$.
Each player possesses a strategy from $(\mathit{start}, \varepsilon, \varepsilon)$ *to* $((\text{state}, q_{init}), q_{init}x_{init}\#, \varepsilon))$.

Proof. The strategy σ_{Eve} for Eve is

$$\sigma_{\text{Eve}}((\mathit{start}, \varepsilon, \varepsilon)) = (\text{initialConfPushed}, x_{init}\#, \varepsilon).$$

Let there be any play compliant with σ_{Eve}. It first visits position (initialConfPushed, $q_{init}x_{init}\#, \varepsilon$). The next position is either in state checkIndexing, which is winning for Eve by Theorem 49, since x_{init} is a valid k-indexing. Or the play continues to $((\text{state}, q_{init}), q_{init}x_{init}\#, \varepsilon)$.

The strategy σ_{Ana} for Ana is

$$\sigma_{\text{Ana}}((\text{initialConfPushed}, x\#, \varepsilon) = \begin{cases} (\text{checkIndexing}, x\#, \varepsilon) & \text{if } x \text{ is not a } k\text{-indexing} \\ ((\text{state}, q_{init}), q_{init}x\#, \varepsilon) & \text{if } x \text{ is a } k\text{-indexing} \end{cases}$$

Let there be a play according to σ_{Ana}. The play first visits (initialConfPushed, $q_{init}x\#, \varepsilon$). If x is not a k-indexing, the strategy proceeds to (checkIndexing, $x\#$, ε). By Theorem 49, Ana possesses a winning strategy from there. Otherwise x is a $(k-1)$-indexing:

$$x = \text{Ind}_{k-1}(\_^{sb(n)-n} w_0 \ldots w_{n-2}(w_{n-1}, \dagger)) = \text{Ind}_{k-1}(c_{init}) = x_{init}$$

by construction and the play continues to $((\text{state}, q_{init}), q_{init}x_{init}\#, \varepsilon)$. □

With the initial configuration on stack 1, the actual simulation begins. The play shall create successing configurations of M on stack 1, where Eve chooses the transitions to use from configurations in existential branching states and vice versa for Ana. It is essential that the successor relation $\mapsto$ is a first order relation in the following sense. For each $\delta \in \Delta$, there is a first order formula in normal form φ_δ, that tests for two configuration words, if they are successors by the transition $\delta = (q, -, q', -, -)$. That is, $c_1, c_2 \vDash \varphi_\delta$ if and only if $(q, c_2) \overset{\delta}{\mapsto} (q', c_1)$.

The following formulas φ_δ test two configuration words for being successing by δ.

$$\begin{aligned}
\varphi_{(q,s,q',s',N)} &= \exists y.\forall y'.\ \big(s_1(y) = (s', \dagger) \wedge s_2(y) = (s, \dagger) \\
&\qquad \wedge [y \neq y' \Rightarrow s_1(y') = s_2(y')]\big) \\
\varphi_{(q,s,q',s',L)} &= \exists y_1.\exists y_2.\ \big([y_2 < y_1 \wedge \neg\exists y''.(y_2 < y'' \wedge y'' < y_1)] \\
&\qquad \Rightarrow [s_2(y_1) = (s, \dagger) \wedge s_1(y_1) = s' \wedge \\
&\qquad\quad \bigvee_{s \in \Gamma} s_2(y_2) = s \wedge s_1(y_2) = (s, \dagger)] \\
&\qquad \wedge \forall y'.\ [(y' \neq y_1 \wedge y' \neq y_2) \Rightarrow s_1(y') = s_2(y')]\big) \\
\varphi_{(q,s,q',s',R)} &= \exists y_1.\exists y_2.\ \big([y_1 < y_2 \wedge \neg\exists y''.(y_1 < y'' \wedge y'' < y_2)] \\
&\qquad \Rightarrow [s_2(y_1) = (s, \dagger) \wedge s_1(y_1) = s' \wedge \\
&\qquad\quad \bigvee_{s \in \Gamma} s_2(y_2) = s \wedge s_1(y_2) = (s, \dagger)] \\
&\qquad \wedge \forall y'.\ [(y' \neq y_1 \wedge y' \neq y_2) \Rightarrow s_1(y') = s_2(y')]\big)
\end{aligned}$$

Furthermore, it is neccessary to check whether a given transition δ is enabled. The following simple first order formulas $\varphi^?_\delta$ test a configuration word for whether $\delta = (q, -, q', -, -)$ can **not** be used from that configuration. That means, $c \vDash \varphi^?_\delta$ if and only if there is **no** configuration word c' such that $(q, c) \overset{\delta}{\mapsto} (q', c')$.

$$\begin{aligned}
\varphi^?_{(q,s,q',s',N)} &= \neg\exists y.\ s_1(y) = (s, \dagger) \\
\varphi^?_{(q,s,q',s',L)} &= \neg\exists y_1.\exists y_2.\ y_2 < y_1 \wedge s_1(y_1) = (s, \dagger) \\
\varphi^?_{(q,s,q',s',R)} &= \neg\exists y_1.\exists y_2.\ y_1 < y_2 \wedge s_1(y_1) = (s, \dagger)
\end{aligned}$$

Then, from a state (state, q) and stack 1 containing the previous configuration word on top, the respective player for q's branching state chooses a transition δ. If $\text{br}(q) = \forall$, i.e. Ana choses the transition, Eve may disbelieve that the transition is enabled by using $k-1$-$\varphi^?_\delta$-Check$(\overline{\Gamma}, Q)$. If they don't, or $\text{br}(q) = \exists$, Eve pushes a sequence on stack 1. This should be a $(k-1)$-indexing of successing configuration word by taking transition δ. Ana may disbelieve this in various ways. They may check for

- the pushed sequence being a $(k-1)$-indexing
- the pushed $(k-1)$-indexing to contain a configuration word
- the pushed $(k-1)$-indexing to contain the successing configuration word by the chosen transition δ.

Checking for the sequence being a $(k-1)$-indexing can be done by the gadget $(k-1)$-ValidIndexing$(\overline{\Gamma})$. For checking whether a word is a configuration word, it is sufficient to check for precisely one letter from $\overline{\Gamma}$. The length is already given by the indexing. The following simple first order formula φ_{Con} checks a word for being a configuration word, i.e. $c \vDash \varphi_{Con}$ if and only if c is a configuration word.

$$\varphi_{Con} = \exists y_1.\forall y_2.\big[\bigvee_{s\in\Gamma} s_1(y_1) = (s,\dagger) \wedge (y_1 \neq y_2) \Rightarrow \bigvee_{s\in\Gamma} s_1(y_2) = s\big]$$

Instantiate the following gadgets.

- $(k-1)$-$\varphi^?_\delta$-Check$(\overline{\Gamma}, Q)$ on stack 1 with starting state $(\text{checkEnabled}, \delta)$.
- $(k-1)$-ValidIndexing$(\overline{\Gamma})$ on stack 1 with starting state checkIndexing.
- $(k-1)$-Con-Check$(\overline{\Gamma}, Q)$ on stack 1 with starting state checkConfiguration.
- $(k-1)$-φ_δ-Check$(\overline{\Gamma}, Q)$ on stack 1 with starting state $(\text{checkSuccessor}, \delta)$.

For each $q \in Q$ and $\delta = (q, -, q', -, -) \in \Delta$ introduce the following transition rules. The states (state, q) belong to Eve, if $\text{br}(q) = \exists$, and to Ana, if $\text{br}(q) = \forall$. Furthermore, $own((\text{transition}, \delta))$ = Eve and $own((\text{pushedConf}, \delta))$ = Ana.

$$\begin{array}{ll}
((\text{state}, q), 1, (\text{transition}, \delta)) & \\
((\text{transition}, \delta), q, 1, (\text{checkEnabled}, \delta)) & \text{only, if } \text{br}(q) = \forall \\
((\text{transition}, \delta), 1, (\Gamma\Sigma^*_{\leq k})^*\#, (\text{pushedConf}, \delta)) & \\
((\text{pushedConf}, \delta), 1, \text{checkIndexing}) & \\
((\text{pushedConf}, \delta), 1, \text{checkConfiguration}) & \\
((\text{pushedConf}, \delta), 1, (\text{checkSuccessor}, \delta)) & \\
((\text{pushedConf}, \delta), 1, q', (\text{state}, q')) &
\end{array}$$

These transition rules, together with the ones for pushing the initial configuration, define the mpdg $(P, \{\text{Evewin}\})$.

Number of contexts/phases: First, symbols are pushed on stack 1, initiating a first context. Pushing symbols does not introduce a phase. The phase has no specific stack yet, but the transitions already belong to the first phase. Then the play continues to

- $(k-1)$-ValidIndexing$(\overline{\Gamma})$, which by theorem 49 has at most $(k-1)+2 = k+1$ additional contexts or $k-1$ additional phases, or
- state (state, q_{init}) without changing phase or context.

Whenever a position in state (state, q) is visited, only internal and push transitions of stack 1 have been used. So the play is still in the first context on stack 1 or the first phase without a chosen stack yet. From there, the play can either use transitions until another state (state, q') is reached, not changing context or phase. Or it can enter

- $(k-1)$-$\varphi^?_\delta$-Check$(\overline{\Gamma}, Q)$ which adds up to $(k-1)+2 = k+1$ contexts and up to $k-1$ phases by Theorem 51, or

- $(k-1)$-ValidIndexing($\overline{\Gamma}$) which adds up to $(k-1)+2=k+1$ contexts and up to $k-1$ phases by Theorem 49, or
- $(k-1)$-Con-Check($\overline{\Gamma}, Q$) which adds up to $(k-1)+2=k+1$ contexts and up to $k-1$ phases by Theorem 51, or
- $(k-1)$-φ_δ-Check($\overline{\Gamma}, Q$) which adds up to $(k-1)+2=k+1$ contexts and up to $k-1$ phases by Theorem 51.

Together with the first context or phase, this adds up to $k+2$ contexts and k phases in total.

Number of states: The above states are capped by the number of transitions $|\Delta|$ and the number of states $|Q|$. Now we need to count the number of states introduced by all the gadgets and their subgadgets. Be aware that instantiating the same gadget twice with the same parameters on the same stack did not create duplicate states, but only new transitions into the existing gadget. Thus we can find the number of states by counting the number of gadgets instantiated.

- The equality gadget is not instantiated above. But the other create instances internally. The gadget is only instantiated in the forms $(k-2)$-Equality(Σ_{k-1}), …, 0-Equality(Σ_1). By Theorem 46, the size of a single such gadget is polynomial in the size of $\Sigma_{k-2}, \dots, \Sigma_0$, which are all 2, and max_0 which is polynomial in the length of the input n. This number doubles, since each of these gadget may occur for each stack. In total, the number of states is $2 \cdot (k-1) \cdot \text{poly}(n)$, which is still polynomial in n.
- The valid indexing gadget is instantiated above and exists as subgadget. It is instantiated above twice with the same parameters on the same stack. Internally, its only instances are $(k-2)$-ValidIndexing(Σ_{k-1}), …, 0-ValidIndexing(Σ_1). By Theorem 49, the size of each of these gadgets is polynomial in n. The number of states again doubles due to having two stacks on which they could be instantiated. We arrive at a total of $2 \cdot (k-1) \cdot \text{poly}(n)$ states which again is polynomial in n.
- The $\sim$-check gadget is instantiated above and exists as subgadget. First count the number of instances above. For each transition $\delta \in \Delta$, $(k-1)$-$\varphi_\delta^?$-Check($\overline{\Gamma}, Q$) and $(k-1)$-φ_δ-Check($\overline{\Gamma}, Q$) are created. Also, $(k-1)$-Con-Check($\overline{\Gamma}, Q$) is instantiated once.

  Internally, the gadget is created by the valid indexing gadget for the relations $(0), (1), (+1)$. Here it only exists in the form $(k-2)$-$\sim$-Check($\Sigma_{k-1}, \Lambda \setminus \Sigma_{\leq k-1}$), …, 0-$\sim$-Check($\Sigma_1, \Lambda \setminus \Sigma_{\leq 1}$) where $\sim$ is one of $(0), (1), (+1)$.

  It is created by the $\sim$-check gadget for the relations $(<), (>), (=)$. The gadget exists as subgadget only in the form $(k-2)$-$\sim$-Check($\Sigma_{k-1}, \{y\}$), …, $(k-2)$-$\sim$-Check($\Sigma_{k-1}, \{y\}$) where $\sim$ is one of $(<), (>), (=)$ and y is one of the variables occuring in the respective formulas for the relations $(<), (>), (=)$ or from the formulas $\varphi_\delta, \varphi^?_\delta, \varphi_{Con}$. Be aware that the formulas are of constant size and thus the number of different variables $\#vars$ is of constant size. The number of relations is six. The total number of subgadgets is then $2 \cdot 6 \cdot \#vars \cdot (k-1)$, again counting double for instances on each stack.

By Theorem 51, each $\sim$-check gadget has an amount of states polynomial in the size of $\overline{\Gamma}$ and $\Sigma_{k-1}, \dots, \Sigma_1$ and max_0. All alphabets are of constant size and max_0 is polynomial in the input length n. The total amount of states by the $\sim$-check gadget is

$$([|\Delta| \cdot 2 + 1] + [2 \cdot 3 \cdot (k-1)] + [2 \cdot 6 \cdot \#vars \cdot (k-1)]) \cdot \text{poly}(n)$$

where the first block counts the gadgets instanciated above, the second counts the gadgets instanciated by the valid indexing gadgets and the third counts the gadgets instanciated by the $\sim$-check gadget. In total, since the Turing machine is a constant, the number of transitions is constant. The whole prefactor then is a constant and the number of states is polynomial in the input length n.

Lemma 62. *Eve possesses a winning strategy in $(P, \{\text{Evewin}\})$ from $(start, \varepsilon, \varepsilon)$ if and only if x is accepting by M.*

Proof. To show this, we show that $(P, \{\text{Evewin}\})$ simulates G_M via

$$f((q, c)) = ((\text{state}, q), qx\#\gamma, \varepsilon).$$

Let $own((\text{state}, q)) = \square$ and their opponent $\blacksquare$. Let $(q, c), (q', c')$ be configurations and $x = \text{Ind}_{k-1}(c)$, $x' = \text{Ind}_{k-1}(c')$.

We need to show that

- If $(q, c) \overset{\delta}{\mapsto} (q', c')$, $\square$ possesses a strategy from $((\text{state}, q), qx\#\gamma, \varepsilon)$ to $((\text{state}, q'), q'x'\#qx\#\gamma, \varepsilon)$.
- Player $\blacksquare$ possesses a strategy from $((\text{state}, q), qx\#\gamma, \varepsilon)$ to the set of positions $\{((\text{state}, q'), q'x'\#qx\#\gamma, \varepsilon) \mid (q, c) \mapsto (q', c')\}$.

For $(q, -, q', -, -) = \delta \in \Delta$, define the following strategies.

$$\sigma_\square(((\text{state}, q), qx\#\gamma, \varepsilon)) = ((\text{transition}, \delta), qx\#\gamma, \varepsilon)$$

$$\sigma_{\text{Eve}}(((\text{transition}, \delta), x\#\gamma, \varepsilon)) = \begin{cases} ((\text{pushedConf}, \delta), x'\#qx\#\gamma, \varepsilon) & x' = \text{Ind}_{k-1}(c'), \text{ if } c' \\ & \text{exists with } (q, c) \overset{\delta}{\mapsto} (q', c') \\ ((\text{checkEnabled}, \delta), x\#\gamma, \varepsilon) & \text{otherwise} \end{cases}$$

$$\sigma_{\text{Ana}}(((\text{pushedConf},\delta),x'\#qx\#\gamma,\varepsilon)) =$$
$$\begin{cases} (\text{checkIndexing},x'\#qx\#\gamma,\varepsilon) & \text{if } x' \text{ is not a } k\text{-indexing}\\ (\text{checkConfiguration},x'\#qx\#\gamma,\varepsilon) & \text{if } x' = \text{Ind}_{k-1}(c') \text{ but}\\ & c' \text{ is not a configuration word}\\ ((\text{checkSuccessor},\delta),x'\#qx\#\gamma,\varepsilon) & \text{if } x' = \text{Ind}_{k-1}(c') \text{ for a configuration}\\ & \text{word } c' \text{ but not } (q,c) \overset{\delta}{\mapsto} (q',c')\\ ((\text{state},q'),q'x'\#qx\#\gamma,\varepsilon) & \text{otherwise} \end{cases}$$

Let $(q,c) \overset{\delta}{\mapsto} (q',c')$. We show that $\sigma_\Box$ is either winning for $\Box$ or visits $((\text{state},q'),$ $q'x'\#qx\#\gamma,\varepsilon)$ where $x' = \text{Ind}_{k-1}(c')$.

Let there be a play compliant with $\sigma_\Box$. By strategy $\sigma_\Box$, the first visited position is $((\text{transition},\delta),qx\#\gamma,\varepsilon)$.

Case 1 ($\Box$ = Eve)*:* By $\sigma_{\text{Eve}} = \sigma_\Box$, the play visits $((\text{pushedConf},\delta),x'\#qx\#\gamma,\varepsilon)$ where $x' = \text{Ind}_{k-1}(c')$. This exists, since we assumed that $(q,c) \overset{\delta}{\mapsto} (q',c')$. Then, if the play continues to $(\text{checkIndexing},x'\#qx\#\gamma,\varepsilon)$, $\Box$ possesses a winning strategy from there by Theorem 49, since x' is a k-indexing. If the play continues to $(\text{checkConfiguration},x'\#qx\#\gamma,\varepsilon)$, $\Box$ possesses a winning strategy from there by Theorem 51, since c' is a configuration word. If the play continues to $((\text{checkSuccessor},$ $\delta),x'\#qx\#\gamma,\varepsilon)$, $\Box$ possesses a winning strategy from there by Theorem 51, since $(q,c) \overset{\delta}{\mapsto} (q',c')$. Otherwise, the play continues to $((\text{state},q'),q'x'\#qx\#\gamma,\varepsilon)$.

Case 2 ($\Box$ = Ana)*:* By construction, the play either continues to $((\text{checkEnabled},$ $\delta),x\#\gamma,\varepsilon)$ from where Ana possesses a winning strategy by Theorem 51, because the transisiton is enabled by $(q,c) \overset{\delta}{\mapsto} (q',c')$. Otherwise, the play continues to $((\text{pushedConf},\delta),q'x'\#qx\#\gamma,\varepsilon)$ where $x' \in (\Gamma\Sigma^*_{\leq k})^*$. If x' is not a k-indexing, then $\sigma_\Box = \sigma_{\text{Ana}}$ continues to $(\text{checkIndexing},x'\#qx\#\gamma,\varepsilon)$. From there, Ana possesses a winning strategy by Theorem 49, since x' is not a $(k-1)$-indexing. If $x' = \text{Ind}_{k-1}(c')$ but c' is not configuration word, the play continues to $(\text{checkConfiguration},$ $x'\#qx\#\gamma,\varepsilon)$. From there, Ana possesses a winning strategy by Theorem 51, since c' is not a configuration word. If $x' = \text{Ind}_{k-1}(c')$ and c' is a configuration word, but not $(q,c) \overset{\delta}{\mapsto} (q',c')$, the play continues to $((\text{checkSuccessor},\delta),x'\#qx\#\gamma,\varepsilon)$. Ana possesses a winning strategy from there by Theorem 51, since x' does not index the successing configuration word for δ, else $(q,c) \overset{\delta}{\mapsto} (q',c')$ would hold. Otherwise, the play continues to $((\text{state},q'),q'x'\#qx\#\gamma,\varepsilon)$, with $x' = \text{Ind}_{k-1}(c')$ and $(q,c) \overset{\delta}{\mapsto} (q',c')$.

Let there be a play compliant with strategy $\sigma_\blacksquare$. By contstruction, it first visits $((\text{transition},\delta),qx\#\gamma,\varepsilon)$ for some $\delta \in \Delta$. From there, the argumentation is analogue to the above. In particular, $\sigma_\blacksquare$ is winning, if there is no $\delta \in \Delta$ enabled in (q,c). Moreover, it visits a position in state $\blacksquare$win in that case. $\Box$

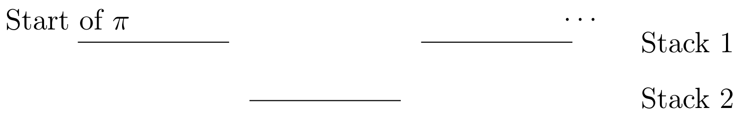
5 Conclusion

We have presented a lower bound of k-EXPTIME on the complexity of $k+2$ bounded context switching and k bounded phase multi-pushdown reachability games. We further showed how to convert a $k+2$ bounded context switching multi-pushdown parity game into a finite state game of size k-EXP. It is to be noted that similar to Walukiewicz's construction for single pushdown games [16], the finite state game can only decide the winner for starting positions with empty stack contents.

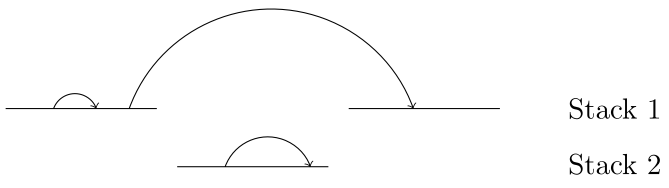
It is noteworthy that there is a mismatch between the complexity for bounded phase and bounded context switching of exactly two exponents. Initially, it was surprising that the complexity of bounded context switching is quite close to the complexity of bounded phase. By intuition, bounded phase allows for arbitrary long words to be moved between stacks, which seems like a very powerful operation when comparing this to bounded context switching. One insight lies in the fact that moving a word from one stack to another is not a very powerful operation on its own. It needs to be equipped with a mechanism to reference positions within the word in order to obtain computability.

During the work, a property was found that gives a suggestion on the complexity of the multi-pushdown game model and also answers the question about the complexity mismatch between the two models treated. It is derived by analyzing the computations possible within the model across all games under this model.

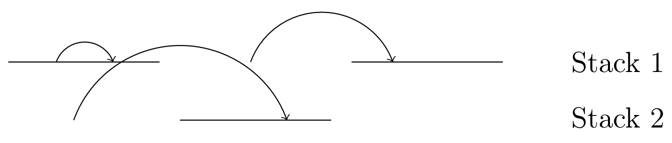
Let us first give an intuition about the property on an example of bounded phase and bounded context switching multi-pushdown games. Imagine a computation π of an mpds with contexts or phases bounded to 3. Let us highlight the contexts by splitting the computation.



Now we add push-pop-pairs to the computation for bounded phase and bounded context switching. A push-pop-pair is marked by an arrow.
Bounded context switching:



Bounded phase:



Note that the push-pop-pairs in bounded context switching are well nested, while in bounded phase, the push-pop-pairs overlap. The push-pop-pairs of the bounded phase example create an overlapping chain: the first push-pop-pair overlaps with another push-pop-pair which itself overlaps with the third push-pop-pair. Identify that as a push-pop-chain of depth 3. Actually, with only 3 contexts, no overlaps can happen in any computation and all computations only allow push-pop-chains of depth 1. More general, $k+2$ bounded context switching allows for push-pop-chains with depth k, while already k bounded phase allow for a depth of k.

It seems that the possible length of push-pop-chains across all computations of the model are connected to the complexity of the model in the sense that push-pop-chains of depth k create the complexity of k-EXPTIME to solve a game on the model. This seems very reasonable with respect to the gadgets created in the lower bound construction. Plays that can grow push-pop-chains up depth k seem to be able to handle a $k-1$ layered indexing system, which gives the connection about the maximal length of words such that reasonable computation on them is still possible.

However, the plays of those gadgets can produce multiple push-pop-chains of depth k so further criteria might be neccessary to conclude an actual rule.

This property is also an indication that games on ordered multi-pushdown systems have complexity of k-EXPTIME and it might a criterium for fixed schedule computations.

Bibliography

[1] Mohamed Faouzi Atig, Ahmed Bouajjani, K. Narayan Kumar, and Prakash Saivasan. Model checking branching-time properties of multi-pushdown systems is hard. *CoRR*, abs/1205.6928, 2012.

[2] Mohamed Faouzi Atig, Ahmed Bouajjani, K. Narayan Kumar, and Prakash Saivasan. Parity games on bounded phase multi-pushdown systems. In Amr El Abbadi and Benoît Garbinato, editors, *Networked Systems*, pages 272–287, Cham, 2017. Springer International Publishing.

[3] Kshitij Bansal and Stéphane Demri. Model-checking bounded multi-pushdown systems. In Andrei A. Bulatov and Arseny M. Shur, editors, *Computer Science – Theory and Applications*, pages 405–417, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[4] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In Antoni Mazurkiewicz and Józef Winkowski, editors, *CONCUR '97: Concurrency Theory*, pages 135–150, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[5] Thierry Cachat. Uniform solution of parity games on prefix-recognizable graphs. *Electronic Notes in Theoretical Computer Science*, 68(6):71 – 84, 2003. Infinity 2002, 4th International Workshop on Verification of Infinite-State Systems (CONCUR 2002 Satellite Workshop).

[6] Thierry Cachat and Igor Walukiewicz. The complexity of games on higher order pushdown automata. *CoRR*, abs/0705.0262, 2007.

[7] E.W. Dijkstra. The humble programmer. *Commun*, 15(10):859–866, 1972.

[8] Robert W Floyd. Assigning meanings to programs. *Mathematical Aspects of Computer Science. Proceedings of Symposium on Applied Mathematics.*, 19:19–32, 1967.

[9] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[10] J. McCarthy. A basis for a mathematical theory of computation. *Computer Programming and Formal Systems. Studies in Logic and the Foundations of Mathematics*, 35:33–70, 1963.

[11] Shaz Qadeer and Jakob Rehof. Context-bounded model checking of concurrent software. In Nicolas Halbwachs and Lenore D. Zuck, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 93–107, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[12] H.G. Rice. Classes of recursively enumerable sets and their decision problems. *Trans*, 74:258–366, 1937.

[13] Anil Seth. Games on multi-stack pushdown systems. In *Proceedings of the 2009 International Symposium on Logical Foundations of Computer Science*, LFCS '09, pages 395–408, Berlin, Heidelberg, 2009. Springer-Verlag.

[14] Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. A robust class of context-sensitive languages. In *LICS (12/07/07)*, pages 161–170, 2007. Event Dates: 10-12 July 2007.

[15] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1937.

[16] Igor Walukiewicz. Pushdown processes: Games and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification*, pages 62–74, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

Aufgabe der Masterarbeit von S. van der Wall

Das Forschungsgebiet der Programmsynthese entwickelt Algorithmen, die Programmskizzen automatisiert in vollständige Programme überführen. Der Un- terschied zwischen Programmskizzen und vollständigen Programmen besteht im Abstraktionsgrad. Während Programme gezwungen sind, das Verhalten des Rechners exakt zu steuern, formulieren Skizzen allein die Logik der Berechnung. Dieses höhere Maß an Abstraktion erleichtert die Programmierung, mit dem un- mittelbaren Ziel, die Produktivität der Software-Entwicklung zu erhöhen, und dem langfristigen Ziel, Programmierung auch Laien zugänglich zu machen.

Synthesealgorithmen unterscheiden sich anhand der Klasse an Skizzen, die als Eingabe erwartet werden. Schon Skizzen rekursiver Berechnungen führen zu einem unendlichen Zustandsraum, und die Programmsynthese gelingt nur durch Algorithmen, die eine endliche Faktorisierung vornehmen. Für rekursi- ve Berechnungen, die in parallelen Threads ausgeführt werden sollen, ist be- kannt, dass Synthesealgorithmen nicht existieren. Präziser formuliert existieren keine Synthesealgorithmen, die exakt sind insofern, als sie möglichst effiziente Programme erzeugen. Wohl aber existieren approximative Synthesealgorithmen, die Programme mit eingeschränktem Verhalten liefern. In der Literatur haben zwei Approximationen besondere Beachtung erfahren. Die Bounded-Context- Einschränkung fordert, dass die Threads den Prozessor nur k-Mal wechseln. Die Bounded-Phase-Einschränkung ist liberaler und erlaubt k-Synchronisationen zwischen den Threads, bis schließlich ein Masterthread die Berechnung leitet.

Die Komplexität der soeben definierten approximativen Syntheseprobleme ist ein ungelöstes Problem. Atig et al. haben gezeigt, dass der Einfluss des Pa- rameters k auf die Effizienz der Synthese dramatisch ist. Seth hat einen Syn- thesealgorithmus vorgelegt, der für jedes k aus Skizzen Programme erzeugt. Die genaue Abhängigkeit der Komplexität vom Parameter k ist unbekannt. Diesem Problem soll sich die Masterarbeit von Sören van der Wall widmen.

Das erste Ziel der Masterarbeit ist es, abhängig von k obere Schranken für die Effizienz des (hochgradig nicht-trivialen) Verfahrens von Seth nachzuweisen. Ist dieses Ziel zu hoch gesteckt, sind Einschränkungen auf die Gestalt der Skizzen zu formulieren, unter denen sich das Resultat zeigen lässt. Zu beachten ist, dass kein Algorithmenentwurf stattfinden soll.

vDas zweite Ziel der Masterarbeit ist, die von Atig et al. angegebenen unteren Schranken für festes k zu präzisieren — idealerweise so, dass sie den erzielten oberen Schranken entsprechen.