Optimal Strategies in Pushdown Reachability Games

Arnaud Carayol[1]
Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, France
arnaud.carayol@u-pem.fr

Matthew Hague[2]
Royal Holloway, University of London, UK
matthew.hague@rhul.ac.uk
https://orcid.org/0000-0003-4913-3800

Abstract

An algorithm for computing optimal strategies in pushdown reachability games was given by Cachat. We show that the information tracked by this algorithm is too coarse and the strategies constructed are not necessarily optimal. We then show that the algorithm can be refined to recover optimality. Through a further non-trivial argument the refined algorithm can be run in 2EXPTIME by bounding the play-lengths tracked to those that are at most doubly exponential. This is optimal in the sense that there exists a game for which the optimal strategy requires a doubly exponential number of moves to reach a target configuration.

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

Keywords and phrases Pushdown Systems, Reachability Games, Optimal Strategies, Formal Methods, Context Free

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.42

Acknowledgements We would like to thank the anonymous reviewers for their comments.

1 Introduction

Pushdown systems are popular models for program verification. They are equipped with an unbounded stack that can model the call stack of a procedural program. That is, the control flow of first-order recursive programs (such as C and Java programs) can be accurately modelled [10].

In a pushdown game, configurations of a pushdown system belong to one of two players (Elvis and the Anarchist). The player who owns a configuration chooses which configuration the game moves to next. In a reachability game, Elvis wins if he is able to force the play into a target configuration, while the Anarchist must prevent this from happening.

One may consider a reachability game to be a competition between a program (Elvis) and its environment (the Anarchist). The program is required to reach a good terminating configuration under all conditions presented by the (uncontrollable) environment. In this situation it is interesting to be able to determine the configurations from which Elvis is able to always win the game, and, moreover, the strategy he should employ. That is, how should the program behave in order to respond to the given inputs to ensure a correct execution.

[1] Supported by the French National Research Agency (ANR), through the excellence program Bézout (ANR-10-LABX-58)
[2] Supported by EPSRC [EP/K009907/1].

The problem of constructing a winning strategy for Elvis corresponds to synthesising a complete, correct program from a given program skeleton.

It is known that the players have *positional* winning strategies in a pushdown reachability game. That is a winning strategy needs only to have access to the current state of the game (as opposed to the entire history of play) [16]. A variety of methods are known for constructing winning strategies in pushdown reachability games [15, 4, 14, 12, 11, 8, 5]. One such method introduced by Cachat computes the *optimal* strategy [3]. That is, the strategy that will reach a target configuration in the fewest number of steps. Such a strategy has obvious applications in the synthesis of efficient programs.

Cachat's algorithm is based on the saturation technique. Saturation is a technique that, beginning from a finite automaton representing a set of configurations, repeatedly adds new transitions to the automaton. The goal is to expand the representation to include all configurations either reachable from, or that can reach, the initial set.

It was shown by Büchi that the set of configurations reachable from the initial configuration of a pushdown system form a regular language and hence can be represented by a finite state automaton [2]. While Büchi's procedure is exponential, Caucal showed that this problem can be solved in polynomial time [6]. The improved algorithm is a saturation process where transitions are incrementally added to a finite automaton. This technique was simplified and adapted to the model-checking setting by Bouajjani *et al.* in [1] and independently by Finkel *et al.* in [7]. In particular, it was shown that the set of predecessors of a regular set of target configurations is also regular. In the same work, the saturation method was shown to work for pushdown reachability games, though the complexity increases to EXPTIME, for which the problem is complete. Cachat builds on this algorithm by annotating each transition added to the finite automaton with a corresponding move in the pushdown game, as well as a weight indicating its "distance" from the target set.

Unfortunately, Cachat's algorithm contains a non-trivial error. In short, by keeping only a single weight per transition, Cachat loses important information about the cost of the different paths of execution through which the Anarchist may force play. This leads to the algorithm computing non-optimal weights for some choices of Elvis, meaning the recommended moves may no longer be optimal. In this work we present the following contributions:

- A counter-example showing how Cachat's algorithm may compute non-optimal strategies.
- A corrected saturation-based algorithm using weights that are fine-grained enough to compute optimal strategies precisely.

  Termination of this algorithm relies on well-quasi orders and we do not have an elementary bound on its runtime.
- A non-trivial proof that the above algorithm can be restricted to only include weights that are doubly exponential in size (whilst still computing optimal strategies). With such a restriction optimal strategies can be computed in 2EXPTIME.
- A matching lower bound giving a game in which Elvis's optimal strategy requires a doubly exponential number of moves to reach a target configuration.

We give preliminaries in Section 2 and the basic saturation algorithm in Section 3. Cachat's algorithm for optimal strategies appears in Section 4 along with our counter-example. We correct the algorithm in Section 5 with the complexity results in Section 6.

2 Preliminaries

2.1 Alternating Finite Automata

To analyse two player games we will make use of alternating finite automata to represent sets of configurations of a pushdown system. For convenience, we will often refer to alternating finite automata simply as automata.

▶ **Definition 2.1** (Alternating Automata). An *alternating automaton* is a tuple $\mathcal{A} = (\mathbb{S}, \Sigma, \mathcal{F}, \delta)$ where $\mathbb{S}$ is a finite set of states, Σ is a finite alphabet, $\mathcal{F} \subseteq \mathbb{S}$ is the set of accepting states, and $\delta \subseteq \mathbb{S} \times \Sigma \times 2^{\mathbb{S}}$ is a transition relation.

We write $s \xrightarrow{A} S$ for a transition (s, A, S) in δ. To simplify the presentation, we will assume that S is always non-empty. A run of an alternating automaton over $w = A_1 \dots A_\ell \in \Gamma^*$ is an unordered unranked tree of depth ℓ with nodes labelled by states in $\mathbb{S}$ and edges labelled by transitions in δ such that for each node η at depth $0 \leq i \leq \ell - 1$ labelled s there is a transition $\tau = s \xrightarrow{A_{i+1}} \{s_1, \dots, s_m\}$ and η has children $\eta_1, \dots, \eta_m$ labelled $s_1, \dots, s_m$ respectively and each edge (η, η_j) for all $1 \leq j \leq m$ is labelled by τ.

If the root of the run is labelled by s and the set of states appearing at the leaves of the run is S, we say that the run starts with s and ends in S. The run is accepting if $S \subseteq \mathcal{F}$. For a state s, the set $\mathcal{L}_s(\mathcal{A})$ contains all words over which there is an accepting run of $\mathcal{A}$ from s.

A branch in a run is a sequence of nodes $\eta_0 \cdots \eta_\ell$ with η_0 the root of the run, η_ℓ a leaf and for each $1 \leq i \leq \ell$, we have η_i is a child of η_{i-1}.

For a word $w \in \Sigma^*$, a state s and a set of states S, we write $\left[s \overset{w}{\underset{\mathcal{A}}{\Rightarrow}} S\right]$ for the set of all runs of $\mathcal{A}$ starting from s and ending precisely with S and $s \overset{w}{\underset{\mathcal{A}}{\Rightarrow}} S$ to denote the existence of a run of $\mathcal{A}$ over w starting from s and ending precisely in S.

2.2 Pushdown Reachability Games

A pushdown reachability game is played between two players on the configuration graph of a pushdown system. The owner of a configuration is indicated by its state and the set of target configurations is accepted by an alternating finite automaton.

▶ **Definition 2.2** (Pushdown Reachability Games). A two-player pushdown reachability game is a tuple $G = (\mathcal{Q}, \Gamma, \Delta, \mathcal{A})$ such that $\mathcal{Q}$ is a finite set of control states partitioned $\mathcal{Q} = \mathcal{Q}_E \uplus \mathcal{Q}_A$ into Elvis and the Anarchist states respectively, Γ is the finite stack alphabet, $\Delta \subseteq (\mathcal{Q} \times \Gamma) \times \left(\mathcal{Q} \times \Gamma^{\leq 2}\right)$ is the set of transitions, and $\mathcal{A}$ is an alternating finite automaton $(\mathbb{S}, \Gamma, \mathcal{F}, \delta)$ with $\mathcal{Q} \subseteq \mathbb{S}$.

We write $(q, A) \to (p, w)$ for the transition $((q, A), (p, w))$ in Δ. A configuration is a tuple (q, w) where q is a state in $\mathcal{Q}$ and w is a stack content in Γ^*. Let $\mathcal{C}_G$ be the set of configurations of G. In the configuration (q, Aw), it is possible to apply a transition $(q, A) \to (p, u)$ to go to the configuration (p, uw). A configuration (q, w) is final if the stack content w is accepted by $\mathcal{A}$ from the state q (*i.e.* $w \in \mathcal{L}_q(\mathcal{A})$).

A play of a pushdown game is a (possibly infinite) sequence $(q_0, w_0), (q_1, w_1), \dots$ where (q_0, w_0) is some starting configuration and (q_{i+1}, w_{i+1}) (if defined) is obtained from (q_i, w_i) via some transition $(q_i, A) \to (q_{i+1}, w) \in \Delta$. In the case where $q_i \in \mathcal{Q}_E$, it is Elvis who chooses the transition to apply, otherwise the Anarchist chooses the transition.

Elvis wins the game if there is some i such that (q_i, w_i) is final or if q_i belongs to the Anarchist and (q_i, w_i) does not have any successors. Otherwise, the Anarchist wins the play.

A strategy for Elvis is a partial function $\sigma : \mathcal{C}_G^* \to \Delta$ which may assign to each play $(q_0, w_0), \ldots, (q_\ell, w_\ell)$ with $q_\ell \in \mathcal{Q}_E$ a transition in Δ applicable to the configuration (q_ℓ, w_ℓ). A given play $(q_0, w_0), (q_1, w_1), \ldots$ is played according to σ if for all i such that $q_i \in \mathcal{Q}_E$ and (q_i, w_i) has a successor in the play, we have $\sigma((q_0, w_0), \ldots, (q_i, w_i)) = r$ and $(q_i, w_i) \to (q_{i+1}, w_{i+1})$ via transition r. A strategy is winning for Elvis from a configuration (q_0, w_0) if all maximal plays starting from (q_0, w_0) and according to σ are winning for Elvis. A strategy is called positional if its value only depends on the last configuration of the play. Hence a positional strategy is fully described by a mapping from the set of configurations belonging to Elvis to Δ. Winning strategies for the Anarchist are defined analogously.

The winning region W of a pushdown reachability game is the set of all configurations from which Elvis has a winning strategy.

It is well-known (see [9]) that $W = \mathrm{Pre}_G^*(\mathcal{A})$ defined as $\mathrm{Pre}_G^*(\mathcal{A}) = \bigcup\limits_{i<\omega} \mathrm{Pre}_G^i(\mathcal{A})$ where

$$\begin{array}{rcl} \mathrm{Pre}_G^0(\mathcal{A}) & = & \{(q, w) \mid w \in \mathcal{L}_q(\mathcal{A})\} \\ \mathrm{Pre}_G^{i+1}(\mathcal{A}) & = & \mathrm{Pre}_G^i(\mathcal{A}) \cup \\ & & \left\{(q, w) \mid q \in \mathcal{Q}_E \wedge \exists (q, w) \to (q', w') \,.\, (q', w') \in \mathrm{Pre}_G^i(\mathcal{A})\right\} \cup \\ & & \left\{(q, w) \mid q \in \mathcal{Q}_A \wedge \forall (q, w) \to (q', w') \,.\, (q', w') \in \mathrm{Pre}_G^i(\mathcal{A})\right\} . \end{array}$$

The rank $\mathrm{rank}(p, w)$ of a configuration (p, w) in W is the smallest i such that $(p, w) \in \mathrm{Pre}_G^i(\mathcal{A})$. Intuitively the rank captures the distance for Elvis to a final configuration.

▶ **Definition 2.3** (Optimal Strategies). A strategy for Elvis is optimal if, for plays ending in (p, w) in the winning region of Elvis, with $p \in \mathcal{Q}_E$, it prescribes a move to (p', w') that minimises $\mathrm{rank}(p', w')$ amongst all possible moves.

Optimal strategies are positional and winning from all configurations in W.

3 The Saturation Algorithm

In [1], Bouajjani et al. present an algorithm that given a pushdown reachability game $(\mathcal{Q}, \Gamma, \Delta, \mathcal{A})$ with an automaton $\mathcal{A} = (\mathbb{S}, \Gamma, \mathcal{F}, \delta)$, constructs a new automaton $\mathcal{B}$ accepting $\mathrm{Pre}_G^*(\mathcal{A})$. The requirements[3] on $\mathcal{A}$ are that no transition in δ goes back to a state in $\mathcal{Q}$. This is required to ensure that the invariants maintained by the algorithm hold initially.

The algorithm proceeds by adding transitions to $\mathcal{A}$ until no new transition can be added. The resulting automaton $\mathcal{B}$ accepts $\mathrm{Pre}_G^*(\mathcal{A})$. That is $w \in \mathcal{L}_q(\mathcal{B})$ iff $(q, w) \in \mathrm{Pre}_G^*(\mathcal{A})$.

The intuition of the algorithm can be seen as follows. Suppose q is a state of Elvis, there is a rule $(q, A) \to (p, w) \in \Delta$, and the configuration (p, ww') is accepted by the automaton by a run beginning with $p \overset{w}{\Rightarrow} S$. The configuration (q, Aw') should also be accepted since an application of the rule reaches (p, ww') from which a target configuration can be reached. Thus, a transition $q \xrightarrow{A} S$ is added, meaning (q, Aw') can now be accepted using the run we know exists over w' from S. For a move of the Anarchist we use alternation to gather together runs from all possible next configurations.

The algorithm constructs a finite sequence $(\mathcal{A}_i)_{i \in [0,N]}$ of automata. The automaton $\mathcal{A}_0$ is $\mathcal{A}$. Each $\mathcal{A}_i$ is of the form $(\mathbb{S}, \mathcal{F}, \delta_i)$, meaning that they only differ by their set of transitions. The construction guarantees that for all $i \in [0, N-1]$ we have $\delta_i \subseteq \delta_{i+1}$. It terminates when $\delta_{i+1} = \delta_i$. This occurs since the set of possible transitions is finite.

The set δ_{i+1} is obtained from δ_i as the smallest set of transitions such that

[3] This requirement is easily met by adding a copy of each state in $\mathcal{Q}$ if necessary.

1. $\delta_i \subseteq \delta_{i+1}$, and
2. for each $q \in \mathcal{Q}_E$, if $(q, A) \to (p, w) \in \Delta$ and $p \overset{w}{\underset{\mathcal{A}_i}{\Longrightarrow}} S$, then $q \xrightarrow{A} S \in \delta_{i+1}$, and
3. for each $q \in \mathcal{Q}_A$ and $A \in \Gamma$ let

   $$(q, A) \to (p_1, u_1), \ldots, (q, A) \to (p_n, u_n)$$

   be all rules from (q, A) in Δ. For all sets $S_1, \ldots, S_n$ of states such that:

   $$p_1 \overset{u_1}{\underset{\mathcal{A}_i}{\Longrightarrow}} S_1, \ldots, p_n \overset{u_n}{\underset{\mathcal{A}_i}{\Longrightarrow}} S_n \quad \text{we have} \quad q \xrightarrow{A} \bigcup_j S_j \in \delta_{i+1} .$$

One can prove that $(p, w) \in \mathrm{Pre}^*_G(\mathcal{A})$ iff $w \in \mathcal{L}_p(\mathcal{B})$ to obtain regularity of the winning region. Since an alternating automaton has at most exponentially many transitions in the number of states (and we do not add any new states), we have that $\mathcal{B}$ is constructible in EXPTIME.

4 Cachat's Algorithm

In Section 4.1, we describe Cachat's min-rank algorithm [3, 4]. This algorithm constructs a weighted alternating automaton which is used to associate to every accepted configuration a weight. We will see in Section 4.2 that this weight is an upper-bound on the rank of the configuration. In Section 4.3, we will show that contrary to Cachat's claim, it is not equal to the rank of the configuration and hence the associated strategy is not an optimal strategy.

4.1 Saturation Algorithm with Weights

Cachat's algorithm proceeds by annotating new transitions of the saturated automaton with two pieces of information: a weight in $\mathbb{N}$ and a rule of the pushdown system. Intuitively if a transition $q \xrightarrow{A} S$ is introduced by the saturation algorithm this means that, for every configuration (q, Aw), Elvis has a strategy to win without ever popping the A or to ensure that the A is popped and that the resulting state belongs to S. The weight of the transition is meant to capture the length of the longest play under an optimal strategy for Elvis. We will see that it is only an upper-bound on this length. The rule in the annotation is the one responsible for the introduction of the transition in the saturation algorithm.

Before presenting the algorithm, we need to define the weight of a run and of a set of runs when the transitions of the automaton are given weights by a function $\mathcal{W}$. The weight of a run ρ, denoted by $\mathcal{W}^*(\rho)$ is the maximum weight of a branch in the run where the weight of a branch is simply the sum of the weights of the transitions appearing in this branch. By convention, a run of depth 0 has weight 0.

For a word $w \in \Sigma^*$, a state s and a set of states S such that $s \overset{w}{\underset{\mathcal{A}}{\Longrightarrow}} S$, we take:

$$\mathcal{W}^*(s \overset{w}{\underset{\mathcal{A}}{\Longrightarrow}} S) = \min\left\{\mathcal{W}^*(\rho) \mid \rho \in \left[s \overset{w}{\underset{\mathcal{A}}{\Longrightarrow}} S\right]\right\}$$

The saturation function is updated to assign weights to new transitions based on the maximum weights of the runs it is based on. Formally a function α is defined which associates to each transition of the saturated automaton a tuple consisting of a weight and a rule of the pushdown system. For convenience, we use # to indicate the absence of an associated rule (for transitions of the initial automaton and moves of the Anarchist). Moreover, we denote by $\mathcal{W}$ the projection of α on the weight component.

Given some initial reachability target set represented by an automaton $\mathcal{A}_0 = \mathcal{A} = (\mathbb{S}, \Gamma, \delta_0, \mathcal{F})$, we initially define $\alpha(\tau) = (0, \#)$ for all $\tau \in \delta_0$.

The annotation function α is updated as new transitions are added. At each iteration, we define $\mathcal{A}_{i+1} = (\mathbb{S}, \Gamma, \delta_{i+1}, \mathcal{F})$ where δ_{i+1} is the smallest set of transitions such that

1. $\delta_i \subseteq \delta_{i+1}$, and
2. for each $q \in \mathcal{Q}_E$, if $r = (q, A) \to (p, w) \in \Delta$ and $p \overset{w}{\underset{\mathcal{A}_i}{\Longrightarrow}} S$ then $\tau = q \xrightarrow{A} S \in \delta_{i+1}$ and furthermore, we assign
$$\alpha(\tau) = \left(1 + \mathcal{W}^*\left(p \overset{w}{\underset{\mathcal{A}_i}{\Longrightarrow}} S\right), r\right) \ .$$
3. for each $q \in \mathcal{Q}_A$ and $A \in \Gamma$ let $(q, A) \to (p_1, u_1), \ldots, (q, A) \to (p_m, u_m)$ be all rules from (q, A) in Δ. For each set of runs $p_1 \overset{u_1}{\underset{\mathcal{A}_i}{\Longrightarrow}} S_1, \ldots, p_n \overset{u_m}{\underset{\mathcal{A}_i}{\Longrightarrow}} S_m$ we have $\tau = q \xrightarrow{A} \bigcup_j S_j \in \delta_{i+1}$ and furthermore, we assign
$$\alpha(\tau) = \left(1 + \max_{1 \leq j \leq m} \left(\mathcal{W}^*\left(p_j \overset{u_j}{\underset{\mathcal{A}_i}{\Longrightarrow}} S_j\right)\right), \#\right)$$

Cachat writes that the algorithm terminates when "no new transitions can be added". The formulation of the algorithm seems to indicate that the weights are final and that the algorithm terminates when all transitions have been added. It is possible to construct an example in which the weight of a transition would decrease from its initial value. It would be easy to adapt the algorithm to allow the weight of the transitions to decrease after their initial introduction but this would not fix the deeper problem pointed out in Section 4.3. In the case where there is only one player (*i.e.* all states belong to Elvis), it is possible to ensure the weight of transitions are final by adding transitions one by one: at each round the transition with the smallest possible weight is created [13, p. 63]. In the following, we will consider that the algorithms stops when the transition structure is stable (*i.e.* $\delta_{i+1} = \delta_i$) and that a transition τ is added to δ_{i+1} only if does not belong to δ_i.

4.2 Min-Rank Strategy

We now assume that $\mathcal{B}$ is the saturated automaton produced by the previous algorithm and that α is the corresponding annotation function. Recall $\mathcal{W}$ is the projection of α on the weight component.

First we remark that as the saturated automaton $\mathcal{B}$ is identical to the one obtained in the original saturation algorithm, $\mathcal{B}$ accepts the winning region of Elvis. Hence the weight of a configuration (q, w) in the winning region can be defined as the minimal weight for an accepting run for this configuration:

$$\mathcal{W}^*(q, w) = \min\left(\mathcal{W}^*\left(q \overset{w}{\underset{\mathcal{B}}{\Rightarrow}} S\right) \mid S \subseteq \mathcal{F} \text{ and } q \overset{w}{\underset{\mathcal{B}}{\Rightarrow}} S\right) \ .$$

Cachat defines what he calls the min-rank positional strategy for Elvis. In this strategy Elvis plays, at a configuration $(q, w) \in W$ (which he owns), the move corresponding to the rule annotating the first transition in any accepting run of $\mathcal{B}$ on w starting with p of minimal weight $\mathcal{W}^*(q, w)$. If the rule annotating the top-most transition is undefined then the configuration is final and no moves needs to be played.

As stated by Cachat, the move outputted by the strategy can be computed in time linear in the length of the input configuration and exponential in the size of the pushdown game (with an exponential precomputation in the size of the game). The algorithm consists in reading w from right to left while maintaining for each state q the weight of the minimal run accepting the stack content from q. This information can be updated in $O(|\mathcal{B}|)$ upon reading a new stack symbol.

From the definition of the algorithm, it can be shown that for all configurations $(q, w) \in W$:

- if (q, w) is owned by the Anarchist, then for all configurations (q', w') with $(q, w) \to (q', w')$, we have $\mathcal{W}^*(q, w) > \mathcal{W}^*(q', w')$.
- if (q, w) is owned by Elvis, then for a configuration (q', w') prescribed by the min-rank positional strategy (if it exists) then $\mathcal{W}^*(q, w) > \mathcal{W}^*(q', w')$.

These properties allow the following properties of the min-rank strategy to be proved.

▶ **Theorem 4.1.** *[3] The min-rank strategy is positional and winning from all configurations in Elvis's winning region.*

In [3], Cachat in addition claims that the min-rank strategy is optimal, which is not the case. The mistake lies in [3, Proposition 6] where Cachat's claims in that the weight of a configuration $(q, w) \in W$ is equal to the rank of this configuration. However, this turns out not to be true as we will see in the next section. Cachat's proof [4, p. 34] in fact shows that $\mathcal{W}^*(p, w)$ is an upper bound on $\mathrm{rank}(p, w)$, but not the converse inequality.

▶ Proposition 4.2 ([3]). For any configuration $(q, w) \in \mathrm{Pre}^*_G(\mathcal{A})$ we have $\mathrm{rank}(q, w) \leq \mathcal{W}^*(q, w)$.

From this, we can obtain the following corollary which we will need in the sequel.

▶ **Corollary 4.3** (Upper Bound). *Take a pushdown game $G = (\mathcal{Q}, \Gamma, \Delta, \mathcal{A})$ with an alternating automaton $\mathcal{A} = (\mathbb{S}, \Gamma, \mathcal{F}, \delta)$ and let $\mathcal{B}$ be the result of Cachat's saturation algorithm. For $C = 2^{|\mathcal{Q}|\cdot|\Gamma|\cdot 2^{|\mathbb{S}|}}$, we have:*

- *for all transitions $q \xrightarrow{A} S$ in $\mathcal{B}$, its weight is bounded by C, and*
- *for any configuration $(p, w) \in \mathrm{Pre}^*_G(\mathcal{A})$ we have $\mathrm{rank}(p, w) \leq C \cdot |w|$.*

Proof. At each iteration of the saturation, the weight of a new transition is at most $1 + 2k$ where k is the maximum weight appearing on a transition in the previous iteration. A direct induction shows that for all $i \geq 0$, the maximum weight of a transition in δ_i is at most $2^i - 1$. As there are at most as many iterations as there are possible transitions of the saturated automaton, after at most $|\mathcal{Q}| \cdot |\Gamma| \cdot 2^{|\mathbb{S}|}$ iterations, no new transitions will be added. It follows that the weight of a transition in the saturated automaton is at most $2^{|\mathcal{Q}|\cdot|\Gamma|\cdot 2^{|\mathbb{S}|}}$ which is the announced constant.

Now consider a configuration $(q, w) \in \mathrm{Pre}^*_G(\mathcal{A})$. From Proposition 4.2, we know that $\mathrm{rank}(q, w)$ is bounded by the weight of any accepting run of $\mathcal{B}$ on w starting from q. The cost of such a run is a most $C \cdot |w|$ which concludes. ◀

4.3 Non-optimality of the Min-Rank Strategy

We give a counter-example in which the weight of configurations are strictly greater than their rank. Then we adapt this counter-example into a game where the min-rank strategy is not optimal. The goal of our counter example is to introduce a transition $q \xrightarrow{A} \{r, s\}$ corresponding to a situation in the game where

- the cost of a play to r is low, but the play from r to a target configuration is long, and
- the cost of a play to s is high, but the play from s to a target configuration is short.

Our game has one control state p belonging to the Anarchist and all other states q, q', r, r', s, and f belong to Elvis. The goal is to reach a configuration $(f, \bot)$. We give the moves below, and show in Figure 1 the game graph from $(p, AA\bot)$. No matter how the Anarchist plays, it will take four steps to reach $(f, \bot)$.

$$\begin{tikzcd}
 & (r, A\bot) \arrow[r] & (r, AA\bot) \arrow[r] & (r', A\bot) \arrow[dr] & \\
(p, AA\bot) \arrow[ur] \arrow[dr] & & & & (f, \bot) \\
 & (q, AAA\bot) \arrow[r] & (q', AA\bot) \arrow[r] & (s, A\bot) \arrow[ur] &
\end{tikzcd}$$

Figure 1 A game showing a counter example to the algorithm of Cachat.

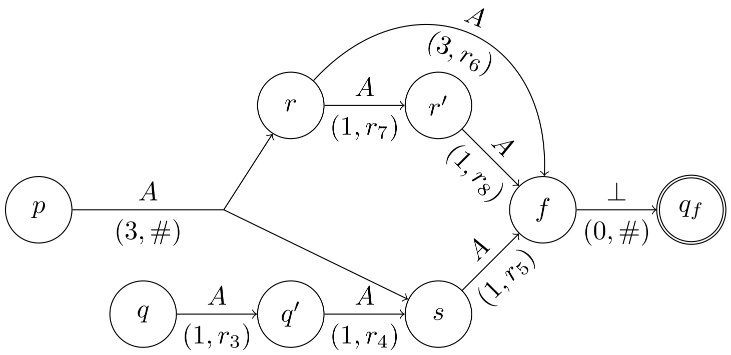

Figure 2 The saturated automaton.

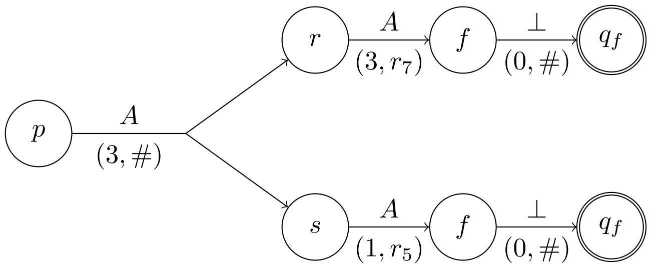

Figure 3 The accepting run of $(p, aa\bot)$.

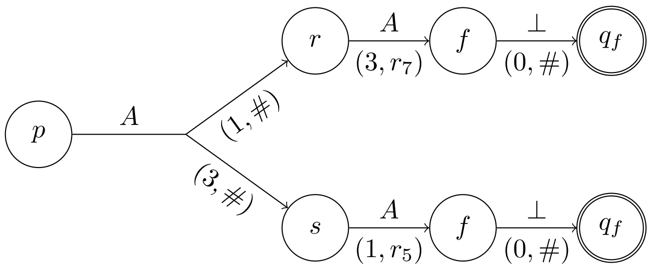

Figure 4 A run of $(p, aa\bot)$ with fine-grained weights.

The moves available to the Anarchist are

$$r_1 = (p, A) \to (r, \varepsilon) \qquad \text{and} \qquad r_2 = (p, A) \to (q, AA)$$

and the remaining moves are available to Elvis and consist of

$$\begin{array}{lll} r_3 = (q, A) \to (q', \varepsilon) & r_4 = (q', A) \to (s, \varepsilon) & r_5 = (s, A) \to (f, \varepsilon) \\ r_6 = (r, A) \to (r, AA) & r_7 = (r, A) \to (r', \varepsilon) & r_8 = (r', A) \to (f, \varepsilon) \ . \end{array}$$

We start with $\mathcal{A}_0$ containing only the transition $f \xrightarrow[(0,\#)]{\perp} q_f$ where q_f is the only accepting state, and $(0, \#)$ is the annotation.

We then begin saturation. The pop rules r_3, r_4, r_5, r_7, and r_8 from states belonging to Elvis immediately lead to the introduction of new transitions. These can be seen in Figure 2 where the annotations show which rule lead to each new transition. The transitions from p and r are described below.

First, we can deal with the push at control state r using r_6. This leads to the introduction of a transition $r \xrightarrow[(3,r_6)]{A} f$ because of the run $r \xrightarrow[(1,r_7)]{A} r' \xrightarrow[(1,r_8)]{A} f$. Next, both rules from p need to be considered simultaneously. That is we introduce a transition $p \xrightarrow[(3,\#)]{A} \{r, s\}$ because of the rule r_1 which is a pop rule from p to r and r_2 and the run $q \xrightarrow[(1,r_3)]{A} q' \xrightarrow[(1,r_4)]{A} s$. No more transitions can be added. The result is shown in Figure 2. The alternating transition is shown with a split arrow.

Given this automaton, we consider the accepting run of $(p, AA\perp)$ shown in Figure 3. Note that the weight of this run is 6, but the longest run that the Anarchist can enforce is 4.

One can extend this to a full counter example to the optimality of Cachat's algorithm as follows. From an initial configuration $(p_0, AAA\perp)$ we give Elvis the choice of moving to $(p, AA\perp)$ via a pop, or to another configuration $(p_1, AA\perp)$ from which 5 steps are required to reach $(f, \perp)$. Since the rank of $(p, AA\perp)$ is estimated to be 6 rather than 4, the strategy will choose to move to $(p_1, AA\perp)$ rather than $(p, AA\perp)$, leading to a play that is not optimal.

To be more explicit, in the full counter example, the game has one control state p belonging to the Anarchist and all other states q, q', r, r', s, and f as well as p_0, p_1, p_2, p_3, p_4, and p_5 belong to Elvis. The goal is to reach a configuration $(f, \perp)$. The moves available to the Anarchist are as before

$$r_1 = (p, A) \to (r, \varepsilon) \qquad \text{and} \qquad r_2 = (p, A) \to (q, AA)$$

and the remaining moves are available to Elvis and consist of the previous rules

$$\begin{array}{lll} r_3 = (q, A) \to (q', \varepsilon) & r_4 = (q', A) \to (s, \varepsilon) & r_5 = (s, A) \to (f, \varepsilon) \\ r_6 = (r, A) \to (r, AA) & r_7 = (r, A) \to (r', \varepsilon) & r_8 = (r', A) \to (f, \varepsilon) \ . \end{array}$$

as well as

$$\begin{array}{lll} r_9 = (p_0, A) \to (p, \varepsilon) & r_{10} = (p_0, A) \to (p_1, \varepsilon) & r_{11} = (p_1, A) \to (p_2, A) \\ r_{12} = (p_2, A) \to (p_3, A) & r_{13} = (p_3, A) \to (p_4, A) & r_{14} = (p_4, A) \to (p_5, \varepsilon) \\ r_{15} = (p_5, A) \to (f, \varepsilon) \ . & & \end{array}$$

At $(p_0, AAA\perp)$ Elvis has two possible moves. The first is to $(p, AA\perp)$ which, as shown above, has a calculated rank of 6. The second is to $(p_1, AA\perp)$ which one can easily verify has a rank of 5. Thus, Cachat's strategy is to move to $(p_1, AA\perp)$ which is not optimal.

This overapproximation occurs because the information stored on the transition from p is too coarse. The weight of 3 comes from the weight of the run to s. The weight of the run to r is 1. Thus, if we were to store a weight for each control state we would obtain two weights on the transition from p. This would allow us to identify that the true cost of the run is 4. This can be seen in Figure 4 and is the basis of our corrected algorithm.

5 Computing Optimal Strategies

We refine Cachat's algorithm to compute optimal strategies in pushdown reachability games. The key idea is to replace simple annotations $\alpha\left(s \xrightarrow{A} S\right) = (d, r)$ with a more fine-grained version, which assigns a separate weight to each state in S. For this, we include annotations in the definition of a transition. That is, a transition is a tuple (s, A, S, D, r) where s is a state in $\mathbb{S}$, A is a character in Γ, $S \subseteq \mathbb{S}$, $D : S \mapsto \mathbb{N}$ is a weight function, and r is a rule of G. We write such transitions as $s \xrightarrow[D,r]{A} S$. As before, we calculate the weights of runs.

5.1 Profile of a run

In an automaton run ρ with annotated transitions, we define the weight of a branch inductively

$$\mathcal{W}^*(\eta_0) = 0 \quad \text{and} \quad \mathcal{W}^*(\eta_0 \cdots \eta_\ell) = D(s_1) + \mathcal{W}^*(\eta_1 \cdots \eta_\ell)$$

where D is the weight function the transition labeling (η_0, η_1) and s_1 is the state labeling η_1. We define a run profile.

▶ **Definition 5.1** (Profiles). Given a run ρ over a word w, the *profile* of ρ is given by $\mathbb{P}(\rho) = (S, D)$ where S is the set of states in $\mathbb{S}$ labelling leaves of ρ and $D : S \to \mathbb{N}$ is the function such that, for all $s \in S$, we have that $D(s)$ is the maximum weight of a branch from the root node to a leaf labelled s. Moreover, we define

$$(1 + D)(s) = 1 + D(s) \quad \text{and} \quad \max(D_1, \ldots, D_m)(s) = \max_{1 \le j \le m} (D_j(s)) \ .$$

Finally, given (S, D) and (S, D'), we write $D \le D'$ when for all $s \in S$ we have $D(s) \le D(s')$. By Dickson's Lemma, $\le$ is a well-quasi-ordering on the weights.

5.2 Saturation

We use the saturation algorithm with run profiles rather than Cachat's annotations. At each iteration, we set $\mathcal{A}_{i+1} = (\mathbb{S}, \Gamma, \delta_{i+1}, \mathcal{F})$ where δ_{i+1} is the smallest set of transitions with

1. $\delta_i \subseteq \delta_{i+1}$, and
2. for each $q \in \mathcal{Q}_E$, if $r = (q, A) \to (p, w) \in \Delta$ and ρ is a run of $\mathcal{A}_i$ over w from p with profile $\mathbb{P}(\rho) = (S, D)$ then

$$\tau = q \xrightarrow[1+D,r]{A} S \in \delta_{i+1} \ .$$

3. for each $q \in \mathcal{Q}_A$ and $A \in \Gamma$ let $(q, A) \to (p_1, u_1), \ldots, (q, A) \to (p_m, u_m)$ be all rules from (q, A) in Δ. For each set $\rho_1, \ldots, \rho_m$ of runs of $\mathcal{A}_i$ such that for each $1 \le j \le m$ the run ρ_j is a run over u_j with root note labelled s_j and profile (S_j, D_j), we have

$$\tau = q \xrightarrow[D',\#]{A} \bigcup_j S_j \in \delta_{i+1}$$

where $D' = 1 + \max(D_1, \ldots, D_m)$.

The algorithm terminates at the first i such that for all $s \xrightarrow[D,r]{A} S \in \delta_{i+1}$ there exists $s \xrightarrow[D',r']{A} S \in \delta_i$ with $D' \leq D$. Equivalently, the algorithm terminates when the set of transitions with a minimal weights stabilizes.

▶ **Lemma 5.2.** *The saturation algorithm terminates and computes* $\mathrm{Pre}^*_G(\mathcal{A})$.

Proof. We have $\mathrm{Pre}^*_G(\mathcal{A})$ from saturation without weights. We terminate as the set of possible transitions (without weight) is finite and $\leq$ is a well-quasi-ordering on the weights. ◀

5.3 Defined Strategy

We define our optimal strategy σ_O as follows. Let $\mathcal{A}'$ be the saturated automaton and let $\mathrm{AccRuns}(q, w)$ be the set of accepting runs of $\mathcal{A}'$ over w from q. We define

$$\mathcal{W}^*(q, w) = \min_{\rho \in \mathrm{AccRuns}(q,w)} \left(\max_{\substack{\mathbb{P}(\rho)=(S,D), \\ s \in S}} (D(s)) \right) .$$

From each configuration (q, w) in the winning region of Elvis, if it is the move of Elvis, he plays the move which leads to the configuration (q', w') that minimises the value of $\mathcal{W}^*(q', w')$. In particular, let $q \xrightarrow[D,r]{A} S'$ be the first transition on a run with weight $\mathcal{W}^*(q, w)$. If it is Elvis's move, he should play the rule r. If $r = \#$, then either Elvis has already reached a target configuration or it is the Anarchist's move. Note, this strategy is non-deterministic since there may be multiple choices of minimal run. In order to define a strategy that is a function, we can fix an ordering on the moves of the game, and always choose the smallest. Note, moreover, that σ_O is positional.

▶ **Lemma 5.3.** *The strategy σ_O is an optimal winning strategy.*

6 Optimal Computation of Optimal Strategies

We show how to reduce the complexity to 2EXPTIME as well as give a lower bound example showing that a doubly exponential number of moves is optimal. For this we will first show that we can bound the value of the weights appearing in *minimal* transitions by a constant K which is doubly exponential in the size of the pushdown game. Then we will restrict the saturation to only consider transitions with weights at most K. Finally we will give a lower bound showing that the constant K needs to be doubly exponential.

6.1 Bounding Play Lengths

We show that there exists a constant K such that any point-wise minimal transitions has all its weights below K. For this we need a stronger result below.

▶ **Lemma 6.1.** *Let $K = 2^{|\mathbb{Q}|\cdot|\mathbb{S}|\cdot|\Gamma|\cdot 2^{|\mathbb{S}|}+|\mathbb{S}|}$. For all transitions $q \xrightarrow[D,r]{A} S$ such that for some $s \in S$ we have $D(s) > K$ then there exists a transition $q \xrightarrow[D',r']{A} S$ such that $D' \leq D$ and $D'(s) \leq K$ for all $s \in S$.*

The proof of Lemma 6.1 is non-trivial. To give the idea of the proof, we consider the case of a pushdown game whose target is the empty stack. Intuitively we proceed as follows.

Let C be the bound obtained in Corollary 4.3. Fix a transition $q \xrightarrow[D,r]{A} S$ of the saturated automaton. We consider the *reduced game*, which broadly corresponds to the game starting with (q, A) and where Elvis aims to empty the stack whilst reaching one of the control states in S. Let σ be a strategy for Elvis associated with the transition in the sense that it ensures that any play following σ and ending in $s \in S$ has length at most $D(s)$.

The simplest case is when $D(s) \geq C$ for all $s \in S$. Then we can find a transition by Cachat that improves on all points. Otherwise, there is at least one $D(s) < C$. We make a new strategy which plays according to σ for $D(s)$ moves. If we have not reached s in this time, we know that playing by σ will never reach s. In particular, there exists a strategy to reach $S \setminus \{s\}$. Moreover, we have increased the stack height by at most C. Thus, we know from Cachat that we can empty the stack and reach any state in $S \setminus \{s\}$ in $C \cdot C$ moves (that is at most C moves per stack character that needs to be removed). We repeat the above argument but this time remove some state s' with $D(s') < C^2$. We play until we are sure not to reach s', increasing the stack height by at most C^2. This means we can reach $S \setminus \{s, s'\}$ in $C \cdot (C + C^2)$ moves and so on. The existence of the strategy in turns implies the existence of a transition in the saturated automaton. In this way, we obtain the bound $K = 2^{|\mathbb{S}|} \cdot C^{|\mathbb{S}|}$.

▶ Remark. We know from Corollary 4.3 (Upper Bound) that there is a doubly exponential bound C on the weight of individual transitions. It is therefore tempting to consider this bound as a proof of the sufficiency of the saturation algorithm with shortcuts described in Section 6.2. However, the bound obtained from Cachat does not guarantee *a priori* that for every transition outside of the bound, there is a transition within the bound that is a pointwise improvement. For example, it is conceivable that Elvis may have two strategies for reaching either the state q_1 or q_2 from q whilst removing A from the stack. The first, corresponding to Cachat's bound, may give a play length of 2 whether the Anarchist forces play to q_1 or q_2. The second, which may violate Cachat's bound, may give a play length of 1 if the Anarchist forces play to q_1, but an extremely large play length (violating Cachat's bound) if the Anarchist forces play to q_2. Let's say this large play length is 100.

Now, consider a configuration (q, Aw). Suppose (q_1, w) requires 100 steps to reach a target configuration, and (q_2, w) is a target configuration. The strategy corresponding to the within-bounds transition has rank 102, whilst the strategy corresponding to the out-of-bounds transition has rank 101. Thus, we have not dismissed the need for out-of-bounds transitions.

6.2 Shortcutting Saturation

We adjust the saturation algorithm by insisting that a transition $s \xrightarrow[D,r]{A} S$ only appears in δ_{i+1} if for all $s \in S$ we have $D(s) \leq K$. Lemma 6.1 guarantees that the set of point-wise minimal transitions is not affected by this restriction.

With this restriction, the worst case running time of saturation becomes doubly exponential. The number of possible weight functions is $K^{|\mathbb{S}|}$ and hence the number of possible transitions is doubly exponential. Since at least one transition must be added during each step of the saturation, the algorithm must terminate in 2EXPTIME.

Like Cachat using a backward algorithm, the move for this strategy on (q, w) can be computed in time linear in the length of w and doubly exponential in the size of the pushdown game (assuming that the saturated automaton has been computed). We read w from right to left maintaining for each state s the value $\mathcal{W}^*(s, u)$ (if it exists) where u is the word read so far. This information can be updated in $\mathcal{O}(|\mathcal{B}|)$ upon reading a new letter and allows the first transition of an accepting run of minimal weight to be found.

6.3 Lower Bound

We give an example game where the shortest run to the target set is doubly exponential, hence showing that the bound K needs to be doubly exponential. The intuition is simple. First, suppose we wanted to force an exponential-length run. In this case we could store a binary number of n digits on the stack, with the least significant bit at the top. E.g. the number 3 would be encoded in a configuration $(q_0, 11000)$ when $n = 5$. To increment the number, we pop 1 characters from the stack until we find the first 0. We record in the control state how many pops are needed. In this case we need two pops and reach $(q_2, 000)$. Then, we replace the topmost 0 with a 1 and push 0s onto the stack until the height is n again. In our example, we reach $(q_0, 00100)$. The goal is to reach a stack with only 1 character, from a stack with only 0. This requires 2^n steps and can be done even in a single-player game.

To generate a doubly exponential run, we follow the same outline, but require the binary encoding to be exponentially long. We cannot use only the control states to enforce this length since it would require an exponential number of them. However, when rebuilding the stack during the increment, we can build a game which forces Elvis to construct a stack of at least exponential height. To do this, after changing the first 0 to 1 Elvis must push any number of 0 characters. Once he is done the Anarchist may accept that the stack is large enough, or challenge the height. To challenge the height we use the fact that the least common divisor of the first n prime numbers is exponential in n. Hence, the Anarchist can pick any of the first n primes, say p, and start a subgame with control states $q_0, \ldots, q_{p-1}$. From each of these control states q_i the only move is a pop to $q_{(i+1 \bmod p)}$. This sub-game is won only if q_0 is reached when the bottom of the stack is reached. Consequently, Elvis must have built the stack up to a multiple of the least common divisor of the first n primes. Note, Elvis may build a stack that is taller than the least common divisor, but this only makes reaching the target state harder.

7 Conclusion

We have studied optimal strategy construction for pushdown reachability games. Initial results due to Cachat [3] unfortunately were too coarse in their analysis and the claimed optimality is in fact an over-approximation. We showed that a refinement of Cachat's algorithm can be made to compute the optimal strategy accurately; however, the additional information required makes it difficult to obtain good complexity results. We gave a non-trivial argument that the algorithm can be refined further to obtain a 2EXPTIME algorithm. Moreover, the doubly exponential weights computed by the algorithm are optimal as demonstrated by a game where the winning strategy requires a doubly exponential number of moves to reach a target configuration.

References

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

2 R. J Büchi. Regular canonical systems. *Archive for Mathematical Logic*, 6(3):91–111, 1964. `doi:10.1007/BF01969548`.

3 T. Cachat. Symbolic strategy synthesis for games on pushdown graphs. In *ICALP*, pages 704–715, 2002. `doi:10.1007/3-540-45465-9\_60`.

4 T. Cachat. *Games on Pushdown Graphs and Extensions*. PhD thesis, RWTH Aachen, 2003. URL: `http://darwin.bth.rwth-aachen.de/opus3/volltexte/2004/957/pdf/Cachat_Thierry.pdf`.

5 Arnaud Carayol and Matthew Hague. Regular strategies in pushdown reachability games. In *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, pages 58–71, 2014. URL: `https://doi.org/10.1007/978-3-319-11439-2_5`, `doi:10.1007/978-3-319-11439-2_5`.

6 D. Caucal. Récritures suffixes de mots. Research Report RR-0871, INRIA, 1988.

7 A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. In *INFINITY*, volume 9, pages 27–37, 1997.

8 Wladimir Fridman. Formats of winning strategies for six types of pushdown games. In *Proceedings First Symposium on Games, Automata, Logic, and Formal Verification, GANDALF 2010, Minori (Amalfi Coast), Italy, 17-18th June 2010.*, pages 132–145, 2010. URL: `https://doi.org/10.4204/EPTCS.25.14`, `doi:10.4204/EPTCS.25.14`.

9 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. URL: `https://doi.org/10.1007/3-540-36387-4`, `doi:10.1007/3-540-36387-4`.

10 N. D. Jones and S. S. Muchnick. Even simple programs are hard to analyze. *J. ACM*, 24:338–350, April 1977. URL: `http://doi.acm.org/10.1145/322003.322016`, `doi:http://doi.acm.org/10.1145/322003.322016`.

11 O. Kupferman, N. Piterman, and M. Y. Vardi. An automata-theoretic approach to infinite-state systems. In *Essays in Memory of Amir Pnueli*, pages 202–259, 2010.

12 N. Piterman and M. Y. Vardi. Global model-checking of infinite-state systems. In *CAV*, pages 387–400, 2004.

13 S. Schwoon. *Model-checking Pushdown Systems*. PhD thesis, Technical University of Munich, 2002.

14 O. Serre. *Contribution à l'étude des jeux sur des graphes de processus à pile*. PhD thesis, Université Paris 7 – Denis Diderot, UFR d'informatique, 2004. URL: `http://tel.archives-ouvertes.fr/tel-00011326`.

15 I. Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001. `doi:10.1006/inco.2000.2894`.

16 W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.