Symbolic Strategy Synthesis for Games on Pushdown Graphs

Thierry Cachat

Lehrstuhl für Informatik VII, RWTH, D-52056 Aachen
Fax: (49) 241-80-22215, Email: cachat@informatik.rwth-aachen.de

Abstract We consider infinite two-player games on pushdown graphs, the reachability game where the first player must reach a given set of vertices to win, and the Büchi game where he must reach this set infinitely often. We provide an automata theoretic approach to compute uniformly the winning region of a player and corresponding winning strategies, if the goal set is regular. Two kinds of strategies are computed: positional ones which however require linear execution time in each step, and strategies with pushdown memory where a step can be executed in constant time.

1 Introduction

Games are an important model of reactive computation and a versatile tool for the analysis of logics like the μ-calculus [5,6]. In recent years, games over infinite graphs have attracted attention as a framework for the verification and synthesis of infinite-state systems [8]. In the present paper we consider pushdown graphs (transition graphs of pushdown automata). It was shown in [12] that in two-player parity games played on pushdown graphs, winning strategies can be realized also by pushdown automata. The drawback of these results [8,12] are a dependency of the analysis on a given initial game position, and a lack of algorithmic description of the (computation of) winning strategies. Such an algorithmic (or "symbolic") solution must transform the finite presentation of the pushdown game into a finite description of the winning regions of the two players as well as of their strategies.

In this paper we develop such an algorithmic approach, also leading to uniform complexity bounds. The nodes of the game graph, *i.e.*, the game positions, are *unbounded* finite objects: stack contents of a given pushdown automaton. Our "symbolic approach" uses finite automata as defining devices for sets of game positions and for the description of strategies. This lifts the results of [1] and [2] from CTL and LTL model checking over pushdown systems to the level of program synthesis.

In this paper we only consider reachability games and Büchi games (where a winning play should reach a given regular set of nodes, respectively should pass infinitely often through this set). This restriction is justified by two aspects: First, reachability and Büchi games are the typical cases in the analysis of safety and liveness conditions. Secondly, as our construction shows, these cases can be handled with set-oriented computations (in determining fixed-points) which fits well to the symbolic approach. So far, it seems open whether in the more general case of parity games the treatment of individual game positions can be avoided in order to have a symbolic computation.

Our paper is structured as follows: we first exhibit a computation of the winning region of a reachability game. In third section, we derive from it two kinds of winning strategies. Finally in fourth section this material is used to solve Büchi games on pushdown graphs. The proofs can be found on the Web at `http://www-i7.informatik.rwth-aachen.de/~cachat/` .

2 Reachability Game, Computing the Attractor

2.1 Technical Preliminaries

A Pushdown Game System (PDS) $\mathscr{P}$ is a triple (P, Γ, Δ), where Γ is the finite stack alphabet, $P = P_0 \uplus P_1$ the partitioned finite set of control locations, where P_i indicates the game positions of Player i, and $\Delta \subseteq P \times \Gamma \times P \times \Gamma^*$ the finite set of (unlabelled) transition rules.

Each macro-state or *configuration* is a pair pw of a control location p and a stack content w. The set of nodes of the pushdown game graph $\mathscr{G} = (V, \hookrightarrow)$ is the set of *all* configurations: $V = P\Gamma^*$, and the arcs are exactly the pairs

$$p\gamma v \hookrightarrow qwv, \text{ for } (p, \gamma, q, w) \in \Delta, \text{ where } \gamma \in \Gamma \text{ and } v \in \Gamma^* .$$

Referring to the case $v = \epsilon$ we also write rules of Δ in the form $p\gamma \hookrightarrow qw$. In the following γ is always a single letter from Γ. If one needs a bottom stack symbol ($\perp$) one has to declare it explicitly in Γ and Δ. The set of nodes of Player 0 is $V_0 = P_0\Gamma^*$, that of Player 1 is $V_1 = P_1\Gamma^*$. Starting in a given configuration $\pi_0 \in V$, a play in $\mathscr{G}$ proceeds as follows: if $\pi_0 \in V_0$, Player 0 picks the first transition (move) to π_1, else Player 1 does, and so on from the new configuration π_1. A play is a (possibly infinite) maximal sequence $\pi_0\pi_1 \cdots$

We describe sets of configurations (and thus also winning conditions in pushdown games) by finite automata. We are thus interested in regular sets of configurations. We define them from alternating $\mathscr{P}$*-automata*. They are alternating word automata with a special convention about initial states. A $\mathscr{P}$-automaton $\mathscr{A}$ is a tuple $(\Gamma, Q, \longrightarrow, P, F)$, where Q is a finite set of states, $\longrightarrow \subseteq Q \times \Gamma \times 2^Q$ a set of transitions, $P \subseteq Q$ a set of initial states (which are taken here as the control locations of $\mathscr{P}$), and $F \subseteq Q$ a set of final states. For each $p \in P$ and $w \in \Gamma^*$, the automaton $\mathscr{A}$ accepts a configuration pw iff there exists a successful $\mathscr{A}$-run on w *from* p. Formally a transition has the form $r \xrightarrow{\gamma} \beta$, where β is a positive boolean formula over Q in Disjunctive Normal Form. To simplify the exposition we allow AND-transitions $r \xrightarrow{\gamma} r_1 \wedge \cdots \wedge r_n$, written as $r \xrightarrow{\gamma} \{r_1, \cdots, r_n\}$, and we capture disjunction by nondeterminism. So a transition like $r \xrightarrow{\gamma} (r_1 \wedge r_2) \vee (r_3 \wedge r_4)$ is represented here by *two* transitions $r \xrightarrow{\gamma} \{r_1, r_2\}$ and $r \xrightarrow{\gamma} \{r_3, r_4\}$.

We define the global transition relation of $\mathscr{A}$, the reflexive and transitive closure of $\longrightarrow$, denoted $\longrightarrow_* \subseteq Q \times \Gamma^* \times 2^Q$, as follows:

– $r \xrightarrow{\epsilon}_* \{r\}$, ($\epsilon$ is the empty word),
– $r \xrightarrow{\gamma} \{r_1, \cdots, r_n\} \;\wedge\; \forall i,\; r_i \xrightarrow{w}_* S_i \quad\Rightarrow\quad r \xrightarrow{\gamma w}_* \bigcup_i S_i$.

The automaton $\mathscr{A}$ accepts the word pw iff there *exists* a run $p \xrightarrow{w}_* S$ with $S \subseteq F$, *i.e.*, all finally reached states are final.

In section 3 we will need the description of a run. The run trees of an alternating automaton $\mathscr{A}$ (where the branching captures the AND-transitions) can be transformed to "run DAGs" (Directed Acyclic Graphs, see [9,10]). In such a run DAG, the states occurring on each level of the tree are collected in a set, and a transition $r \longrightarrow \{r_1, \cdots, r_k\}$ connects state r of level i with states $\{r_1, \cdots, r_k\}$ of level $i+1$. Note that every transition of level i is labelled by the same i-th letter of the input word. Let Φ be the set of partial functions from Q to the transition relation $\longrightarrow$ of $\mathscr{A}$. A run DAG from state p labelled by $w = \gamma_0 \cdots \gamma_n$ is described by a sequence $\sigma_0, \cdots, \sigma_n$ of elements of Φ and a sequence $Q_0, Q_1, \cdots, Q_n$ of subsets of Q, such that $Q_i = Dom(\sigma_i)$, and from each $q \in Q_i$ the transition $\sigma_i(q)$ is used from q:

$$Q_0 = \{p\} \xrightarrow[\sigma_0]{\gamma_0} Q_1 \xrightarrow[\sigma_1]{\gamma_1} \cdots \; Q_n \xrightarrow[\sigma_n]{\gamma_n} S \;.$$

So σ_i describes the step $Q_i \xrightarrow{\gamma_i} Q_{i+1}$ by the transitions used. We write shortly $\{p\} \xrightarrow[\sigma]{w}_* S$, assuming $\sigma = \sigma_0, \cdots, \sigma_n$, or just $\{p\} \xrightarrow{w}_* S$ to denote the run.

2.2 Reachability

We consider a regular *goal set* $R \subseteq P\Gamma^*$, defined by a $\mathscr{P}$-automaton $\mathscr{A}_R$. Player 0 wins a play iff it reaches a configuration of R. Our goal is to compute the winning region W_0 of this game: the set of nodes from which Player 0 can force the play to reach the set R or a deadlock for Player 1. The set W_0 is clearly the "0-attractor of R" (see [11]), denoted $Attr_0(R)$, and defined inductively by

$$\begin{aligned} Attr_0^0(R) &= R \;, \\ Attr_0^{i+1}(R) &= Attr_0^i(R) \cup \big\{u \in V_0 \mid \exists v, u \hookrightarrow v,\; v \in Attr_0^i(R)\big\} \\ &\quad \cup \big\{u \in V_1 \mid \forall v, u \hookrightarrow v \Rightarrow v \in Attr_0^i(R)\big\} \;, \\ Attr_0(R) &= \textstyle\bigcup_{i \in \mathbb{N}} Attr_0^i(R) \;. \end{aligned}$$

As the degree of the game graph is finite, an induction on ω is sufficient. According to this definition, we adopt the convention that if the play is in a deadlock (before reaching R), the Player who should play has lost.

Our task is to transform a given automaton $\mathscr{A}_R$ recognizing R into an automaton $\mathscr{A}_{Att(R)}$ recognizing $Attr_0(R)$. Without loss of generality, we can assume that there is no transition in $\mathscr{A}_R$ leading to an initial state (a state of P).

Algorithm 1 (saturation procedure)
Input: *a PDS $\mathscr{P}$, a $\mathscr{P}$-automaton $\mathscr{A}_R$ that recognizes the goal set R, without transition to the initial states.*
Output: *a $\mathscr{P}$-automaton $\mathscr{A}_{Att(R)}$ that recognizes $Attr_0(R)$.*

Let $\mathscr{A}_{Att(R)} := \mathscr{A}_R$. *Transitions are added to $\mathscr{A}_{Att(R)}$ according to the following saturation procedure.*
repeat

(Player 0) *if* $p \in P_0$, $p\gamma \hookrightarrow qv$ *and* $q \xrightarrow{v}_* S$ *in* $\mathscr{A}_{Att(R)}$, *then add a new transition* $p \xrightarrow{\gamma} S$.

(Player 1) *if* $p \in P_1$, $\begin{cases} p\gamma \hookrightarrow q_1 v_1 \\ \vdots \qquad \vdots \\ p\gamma \hookrightarrow q_n v_n \end{cases}$ *are* all *the moves (rules) starting from* $p\gamma$ *and* $\begin{cases} q_1 \xrightarrow{v_1}_* S_1 \\ \vdots \qquad \vdots \\ q_n \xrightarrow{v_n}_* S_n \end{cases}$ *in* $\mathscr{A}_{Att(R)}$, *then add a new transition* $p \xrightarrow{\gamma} \bigcup_i S_i$.

until *no new transition can be added.*

Note that $\mathscr{A}_{Att(R)}$ has exactly the same state space as $\mathscr{A}_R$. The algorithm eventually stops because there are only finitely many possible new transitions, and the "saturation" consists in adding as many transitions as possible. The idea of adding a new transition $p \xrightarrow{\gamma} S$ for $p \in P_0$ is that, if $qv \in Attr_0(R)$, and $p\gamma \hookrightarrow qv$, then $p\gamma \in Attr_0(R)$ too, and then $p\gamma$ *should* have the same behavior as qv in the automaton. For $p \in P_1$, $Attr_0(R)$ is defined by a conjunction, expressed in $\mathscr{A}_{Att(R)}$ by the AND-transition. The algorithm and the proof is a generalization of [2,7] from nondeterministic automata (for simple reachability) to alternating automata (for game reachability). In [2], one deals with the case $P = P_0$, and the "winning region" is the set of "predecessors" of R, denoted $pre^*(R)$. In [1], alternating (pushdown) automata were already considered, but they were not used to solve a game, and winning strategies were not treated.

Theorem 2 *The automaton $\mathscr{A}_{Att(R)}$ constructed by Algorithm 1 recognizes the set $Attr_0(R)$, if $\mathscr{A}_R$ recognizes R.*

The algorithm runs in time $\mathcal{O}(|\Delta|\ 2^{c|Q|^2})$, where $|\Delta|$ is the sum of the lengths of the rules in Δ. An implementation of Algorithm 1 was developed in [4].

We have chosen a *regular* goal set R, and proved that $Attr_0(R)$ is also regular. If we consider a context free goal set R, the situation diverges for the cases of simple reachability and game reachability:

Proposition 3 *If the goal set R is a context free language, then $pre^*(R)$ is also context free, but $Attr_0(R)$ is not necessarily context free.*

The first part can be deduced from [3]. For the proof of the second part, we just remark that the intersection of two context free languages may not be context free. If R_1 and R_2 are two context free languages over $\{a, b, c\}$ and the first move of Player 1 goes from pu to q_1u or q_2u, $u \in \{a, b, c\}^*$, and if the goal set is $q_1R_1 \cup q_2R_2$, then the winning region is $p(R_1 \cap R_2) \cup q_1R_1 \cup q_2R_2$.

2.3 Determining Membership in the Attractor

In [8] and [12], for a given initial position of the game, an EXPTIME procedure determines if it is in the winning region W_0 of Player 0. In contrast our solution is uniform: after a single EXPTIME procedure, we can determine in linear time if any given configuration is in the winning region W_0.

To determine whether a given configuration belongs to $Attr_0(R)$, we can use a polynomial time algorithm, that searches backwards all the accepting runs of the automaton $\mathscr{A}_{Att(R)}$ (from now on, we skip corresponding claims for Player 1). We repeat here the classical algorithm because variants of it will be used in the next section. The correctness proof is easy and omitted here.

Algorithm 4 (Membership)
Input: *an alternating* $\mathscr{P}$*-automaton* $\mathscr{B} = (\Gamma, Q, \longrightarrow, P, F)$ *recognizing* $Attr_0(R) = L(\mathscr{B})$*, a configuration* $pw \in P\Gamma^*$, $w = a_1 \ldots a_n$.
Output: *Answer whether* $pw \in L(\mathscr{B})$ *or* $pw \notin L(\mathscr{B})$.

Let $S := F$;
for $i := n$ down to 1 **do** $S := \{s \in Q \mid \exists\ (s \xrightarrow{a_i} X) \text{ in } \mathscr{B}, X \subseteq S\}$ **end for**
If $p \in S$, *answer "*$pw \in L(\mathscr{B})$*"* else *answer "*$pw \notin L(\mathscr{B})$*"*

The space complexity of Algorithm 4 is $\mathcal{O}(|Q|)$, the time complexity is $\mathcal{O}(nm|Q|)$ where m is the number of transitions of $\mathscr{B}$, and n is the length of the input configuration.

3 Winning Strategy for Player 0

A *strategy* for Player 0 is a function which associates to a prefix $\pi_0\pi_1 \cdots \pi_n \in V^*V_0$ of a play a "next move" π_{n+1} such that $\pi_n \hookrightarrow \pi_{n+1}$.

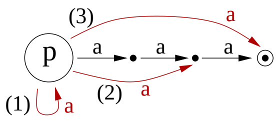
3.1 Preparation

A move of Player 0 consists in a choice of a PDS-Rule. Given a configuration $pw \in Attr_0(R)$, our aim is to extract such a choice from an accepting run of $\mathscr{A}_{Att(R)}$ on pw. In Algorithm 1, a new transition $p \xrightarrow{\gamma} S$ of $\mathscr{A}_{Att(R)}$ is generated by a (unique) rule $p\gamma \hookrightarrow qv$ of the PDS under consideration, if $p \in P_0$. We extend now the algorithm so that it computes the partial function $Rule$ from $\longrightarrow$ to Δ. This function remembers the link between a new transition of the finite automaton for $Attr_0(R)$ and the rule of the Pushdown Graph that was used to construct it. We shall write in the algorithm $Rule(p \xrightarrow{\gamma} S) := p\gamma \hookrightarrow qv$. For transitions $p \xrightarrow{\gamma} S$ of the original automaton $\mathscr{A}$, $Rule(p \xrightarrow{\gamma} S)$ is undefined.

Now, given a configuration $p\gamma w \in V_0$ accepted by $\mathscr{A}_{Att(R)}$, with a run $\{p\} \xrightarrow{\gamma} S \xrightarrow{w}_* T$ (and if $p\gamma w \notin R$), a first idea would be to choose the move $Rule(p \xrightarrow{\gamma} S) = p\gamma \hookrightarrow qv$, *hoping* to get closer to R. Unfortunately this does not, in general, define a winning strategy. Still it ensures that we remain in the winning region. The following example illustrates this situation.

Example 5 *Let* $\Gamma = \{a\}$, $P = P_0 = \{p\}$, $P_1 = \emptyset$ *and* $\Delta = \{pa \hookrightarrow p, pa \hookrightarrow paa\}$.

It is clear that Player 0 can add and remove as many a's as he wants. Let $R = \{pa^3\}$ (R is regular), then the winning region is $Attr_0(R) = pa^+$, as shown



$Rule((1)) = pa \hookrightarrow p$
$Rule((2)) = pa \hookrightarrow paa$
$Rule((3)) = pa \hookrightarrow paa$ $\quad p \hookleftarrow pa \rightleftarrows paa \rightleftarrows [paaa] \cdots$

Figure 1. Automaton from Algorithm 1 and Example 5, game graph

by the automaton in Figure 1, from Algorithm 1. There are two different runs of $\mathscr{A}_{Att(R)}$ that accept paa: through transitions (1)(3), or through (2). The strategy associated to (1)(3) plays to pa, and therefore is not successful. To define a winning strategy using finite automaton $\mathscr{A}_{Att(R)}$, we need to select the most suitable run on a given configuration, or to remember information about an accepting run, to play coherently the following moves. We give two solutions: the first one, a positional strategy, associates a cost to each transition added while constructing $\mathscr{A}_{Att(R)}$, in order to compute the distance to R. The second one, a pushdown strategy, uses a stack to remember how $\mathscr{A}_{Att(R)}$ accepts the current configuration.

3.2 Positional Min-rank Strategy

The *rank* of a configuration pw is the smallest i such that $pw \in Attr_0^i(R)$ (it is ∞ if $pw \notin Attr_0(R) = W_0$). It is the "distance" of the configuration pw to R. In the following we consider only configurations in W_0. Then Player 0 will be able, from a configuration in $Attr_0^i(R)$, to move to $Attr_0^{i-1}(R)$, and Player 1 does this with each possible move. In order to implement this, during the construction of $\mathscr{A}_{Att(R)}$ we will attribute to each $\mathscr{A}_{Att(R)}$-transition τ a cost $Cost(\tau)$. Initially, each transition of $\mathscr{A}_R$ has the cost 0 (with these transitions $\mathscr{A}_{Att(R)}$ recognizes configurations that are already in R).

The function $Cost$ from the transition relation $\longrightarrow$ of $\mathscr{A}_{Att(R)}$ to $\mathbb{N}$ is extended to a function $Cost^*$ from the run DAGs to $\mathbb{N}$. Given a fixed run $\{q\} \xrightarrow{w}_* S$ of the automaton $\mathscr{A}_i$ (obtained at step i in the construction of $\mathscr{A}_{Att(R)}$), its cost $Cost^*(\{q\} \xrightarrow{w}_* S)$ is the maximal sum of the costs of the transitions along a single path (branch) of the run DAG $\{q\} \xrightarrow{w}_* S$. Inductively $Cost^*$ is defined by the following clauses:
$Cost^*(\{q\} \xrightarrow{\epsilon}_* \{q\}) = 0$
$Cost^*(\{q\} \xrightarrow{\gamma} \{q_1, \cdots, q_n\} \xrightarrow{u}_* \bigcup_i S_i) =$

$$Cost(q \xrightarrow{\gamma} \{q_1, \cdots, q_n\}) + \max_{1 \leqslant i \leqslant n} (Cost^*(\{q_i\} \xrightarrow{u}_* S_i)) .$$

When adding a new transition $p \xrightarrow{\gamma} S$ to $\mathscr{A}_i$, to obtain $\mathscr{A}_{i+1}$, its cost is computed by an extension of Algorithm 1, using the costs of the existing transitions. In the main loop of Algorithm 1, we add the following assignments:
- if $p \in P_0$ ..., let $Cost(p \xrightarrow{\gamma} S) := 1 + Cost^*(\{q\} \xrightarrow{v}_* S)$,
- if $p \in P_1$ ..., let $Cost(p \xrightarrow{\gamma} \bigcup_i S_i) := 1 + \max_j(Cost^*(\{q_j\} \xrightarrow{v_j}_* S_j))$.

The significance of $Cost^*$ follows clearly from next proposition:

Proposition 6 *For any configuration $pw \in Attr_0(R)$,*

$$rank(pw) = \min\{Cost^*(\{p\} \xrightarrow{w}_* S) \mid \{p\} \xrightarrow{w}_* S \subseteq F \text{ in } \mathscr{A}_{Att(R)}\} \ .$$

In the Example 5, one gets $Cost((1)) = 1$, $Cost((2)) = 1$, $Cost((3)) = 2$. So using the transitions (1) and (3) is not the best way to accept paa, and transition (2) is taken. We are now able to define the desired strategy.

> **Min-rank Strategy** for Player 0
> **Input**: alternating automaton $\mathscr{A}_{Att(R)}$ for $Attr_0(R)$, functions $Rule$ and $Cost$ (as computed from Algorithm 1 from PDS $\mathscr{P}$), configuration $pw \in Attr_0(R)$, $p \in P_0$.
> **Output**: "next move" from configuration pw.
>
> Find an accepting run $\{p\} \xrightarrow{w}_* S \subseteq F$ of $\mathscr{A}_{Att(R)}$ with minimal cost $Cost(\{p\} \xrightarrow{w}_* S)$.
> If the cost is 0, $pw \in R$ and the play is won, else decompose this run: $w = \gamma w'$, $\{p\} \xrightarrow{\gamma} T \xrightarrow{w'}_* S$, and choose the rule $Rule(p \xrightarrow{\gamma} T)$.

Theorem 7 *The min-rank strategy is positional, winning from all configurations of the winning region W_0 of Player 0. It can be computed in time $\mathcal{O}(n)$ in the length n of the input configuration.*

Algorithm 4 can be easily extended to compute the distance to R and the strategy. By Proposition 6, the min-rank strategy is optimal in the sense that it finds a shortest path to R. It reevaluates its choices at each step of the game (particularly if Player 1 goes much "closer" to R than needed). We will present in the next subsection a strategy that is not necessarily optimal but easier to compute.

3.3 Pushdown Strategy

A *pushdown strategy*, as defined in [12] is a deterministic pushdown automaton with input and output. It "reads" the moves of Player 1 and outputs the moves (choices) of Player 0, like a pushdown transducer. For simplicity, we will restrict our presentation to the following form of pushdown strategy:

Definition 8 *Given a PDS (P, Γ, Δ), $P = P_0 \uplus P_1$, where Δ_i is the set of transition rules in Δ departing from Player i configurations, a* pushdown strategy *for Player 0 in this game is a* deterministic *pushdown automaton $\mathscr{S} = (P, A, \Pi)$, where $A = \Gamma \times \Sigma$, Σ is any alphabet, $\Pi \subseteq ((P_1 \times A \times \Delta_1) \times (P \times A^*)) \cup ((P_0 \times A) \times (P \times A^* \times \Delta_0))$ is a finite set of transition rules.*

A transition of $\mathscr{S}$ either reads a move of Player 1 or outputs a move for Player 0, in both cases updating its stack. We will now define a pushdown strategy, starting from the automaton $\mathscr{A}_{Att(R)}$. Given a configuration $pw \in Attr_0(R)$, there is an accepting run $\{p\} \xrightarrow{w}_* S$ of $\mathscr{A}_{Att(R)}$:

$$Q_0 = \{p\} \xrightarrow[\sigma_0]{\gamma_0} Q_1 \xrightarrow[\sigma_1]{\gamma_1} \cdots \ Q_n \xrightarrow[\sigma_n]{\gamma_n} S, \qquad w = \gamma_0\gamma_1 \cdots \gamma_n \ .$$

Our aim is to store in the stack of the strategy the description of this run. The corresponding configuration of $\mathscr{S}$ is $p(\gamma_0, \sigma_0) \cdots (\gamma_n, \sigma_n)$. We fix for the alphabet Σ the set Φ (see Section 2.1).

At the beginning of the play, if the initial configuration pw is in $Attr_0(R)$, we have to initialize the stack of $\mathscr{S}$ with the description of an accepting run of $\mathscr{A}_{Att(R)}$ (not necessarily the cheapest according to the costs defined above). Algorithm 4 can initialize the stack at the same time when searching an accepting run (in linear time). We define now the unique transition rule of Π from $(p, (\gamma_0, \sigma_0))$ or $(p, (\gamma_0, \sigma_0), \delta_1)$ (with $\delta_1 \in \Delta_1$). By construction $\sigma_0(p)$ is the "good" transition $\tau = p \xrightarrow{\gamma_0} Q_1$ used in the run of $\mathscr{A}_{Att(R)}$.

- If $p \in P_0$ then output the move $Rule(\sigma_0(p)) = p\gamma_0 \hookrightarrow qv$ that corresponds to τ. Remove the first letter of the stack. Push on the stack the description of the run $\{q\} \xrightarrow{v}_* Q_1$ used in Algorithm 1) to generate τ. Go to control state q.
- If $p \in P_1$ and Player 1 chooses the transition $\delta_1 = p\gamma_0 \hookrightarrow qv$ in Δ_1, by construction of the automaton $\mathscr{A}_{Att(R)}$, $q \xrightarrow{v}_* S$, and S is a subset Q_1. Go to control state q, remove the first letter of the stack, push the description of the run $\{q\} \xrightarrow{v}_* S$ used in Algorithm 1) to generate τ.

For Example 5 (Figure 1), we can see that the pushdown strategy is winning even if the initialization is not optimal. The configuration paa is in the winning region and an accepting run is coded on the stack of the strategy:

$$p(a, \{(p,1)\})(a, \{(p,3)\}) \ .$$

According to the strategy, the following play is generated (the symbol "$-$" denotes a value that is not relevant):

$$\begin{array}{rll} (\text{by } Rule((1)) = pa \hookrightarrow p) & \text{proceed to} & p(a, \{(p,3)\}) \\ (\text{by } Rule((3)) = pa \hookrightarrow paa) & \text{proceed to} & p(a, \{(p,2)\})(a, -) \\ (\text{by } Rule((2)) = pa \hookrightarrow paa) & \text{proceed to} & p(a, -)(a, -)(a, -) \ . \end{array}$$

Theorem 9 *One can construct effectively a pushdown strategy that is winning from each node of the winning region of a pushdown reachability game. Its transition function is defined uniformly for the whole winning region. The initialization of the stack is possible in linear time in the length of the initial game position, and the computation of the "next move" is in constant time (for fixed Δ).*

Although there is no need to compute costs to define this strategy, it is useful to refer to the costs of the previous subsection for the correctness proof. The strategy for Player 1 in this "safety game" is much easier to define and compute: he just has to stay in $V \setminus Attr_0(R)$.

3.4 Discussion

The stack of the pushdown strategy needs to be initialized at the beginning of a play (in linear time in the length of the configuration); then the computation of the "next move" is done in constant time (execution of one transition of the strategy). In contrast, the min-rank strategy needs for each move a computation in linear time in the length of the configuration. So we can say that in the case of pushdown graphs a positional strategy can be more expensive than a strategy with memory. This effect does not appear over finite-state game graphs.

4 Büchi Condition

Given $\mathscr{P}$ and R as in the preceding sections, the (Büchi) winning condition is now the following:

Player 0 wins a play iff it meets infinitely often the goal set R, or ends in a deadlock for Player 1.

To determine the winning region of this game one defines $Attr_0^+$ inductively, similarly to $Attr_0$: for $T \subseteq V$, let

$$\begin{aligned} X_0(T) &= \emptyset\ , \\ X_{i+1}(T) &= X_i(T) \cup \{u \in V_0 \mid \exists v, u \hookrightarrow v,\ v \in T \cup X_i(T)\} \\ &\quad \cup \{u \in V_1 \mid \forall v, u \hookrightarrow v \Rightarrow v \in T \cup X_i(T)\}\ , \\ Attr_0^+(T) &= \textstyle\bigcup_{i \geqslant 0} X_i(T) \qquad \text{(the degree of the game graph is bounded).} \end{aligned}$$

Following the approach as in [11] (where the definition of the X_i's needs adjustment) the following claims are easy:

Proposition 10 *$Attr_0^+(T)$ is the set of nodes from which Player 0 can join T in* at least *one move, whatever Player 1 does.*

Proposition 11 *Let $Y_0 = V$, and $\forall i \geqslant 0, Y_{i+1} = Attr_0^+(Y_i \cap R)$, then the fixed point $Y^\infty = \bigcap_{i \geqslant 0} Y_i$ is the winning region of the Büchi game with goal set R.*

In the case of finite graphs, this induction can be effectively carried out: the sequence $(Y_i)_{i>0}$ is strictly decreasing until it reaches a fixed point. For pushdown graphs, the regular languages Y_i are also smaller and smaller, but the question of convergence is nontrivial. Following the symbolic approach, we will construct finite automata that are strictly "decreasing" until they reach a fixed point.

In the following we will proceed in two steps: firstly Algorithm 12 computes an automaton that recognizes Y_i for a given i, secondly Algorithm 14 computes directly an automaton for Y^∞. The first algorithm helps to understand the second, but is not a pre-computation.

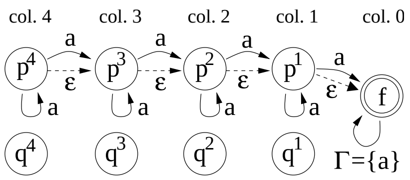
Remark 1. In this section we will consider a simple goal set of the form $R = F\Gamma^*$ for some $F \subseteq P$ (it only depends on the control state). This is not an essential restriction: given a PDS $\mathscr{P} = (P, \Gamma, \Delta)$ that defines the original game and a regular set R of configurations, we can reduce to the case of simple goal sets by proceeding to a new PDS $\mathscr{P} \times \mathcal{D}$, where $\mathcal{D}$ is a finite automaton recognizing R.

So from now on we consider a simple goal set $R = F\Gamma^*$ for some $F \subseteq P$. Of course it is regular. Algorithm 1 is modified (by new steps involving ϵ-transitions) to compute $Attr_0^+(R)$. The intersection with R is easy to compute, and Algorithm 12 determines successively $Y_1, Y_2, Y_3 \dots$

Algorithm 12 (computation of Y_j)
Input: *PDS* $\mathscr{P} = (P, \Gamma, \Delta)$, $j \geqslant 1$ *and* $F \subseteq P$ *that defines the goal set* $R = F\Gamma^*$.
Output: *a* $\mathscr{P}$*-automaton* $\mathscr{B}_j$ *that recognizes* Y_j.

Initialization: *the state space of* $\mathscr{B}_j$ *is* $\{f\} \cup (P \times [1, j])$, *and* f *is the unique final state (we write* (p, i) *as* p^i*). For all* $\gamma \in \Gamma$, $f \xrightarrow{\gamma} f$. *For all* $p \in P$, p^0 *is set to be* f.

for $i := 1$ *to* j ***do***
(compute generation i, *consider now the* p^i*'s as initial states.)*
Add an ϵ*-transition from* p^i *to* p^{i-1} *for each* $p \in F$ *(only). (If* $i = 1$, p^0 *is* f*.)*
Add new transitions to $\mathscr{B}_j$ *according to Algorithm 1:*
repeat
(Player 0) if $p \in P_0$, $p\gamma \hookrightarrow qw$ *and* $q^i \xrightarrow{w}_* S$ *in the current automaton,* then *add a new transition* $p^i \xrightarrow{\gamma} S$.
(Player 1) if $p \in P_1$, $\{p\gamma \hookrightarrow q_1 w_1, \cdots, p\gamma \hookrightarrow q_n w_n\}$ *are* all *the moves (rules) starting from* $p\gamma$ *and* $\forall k$, $q_k^i \xrightarrow{w_k}_* S_k$ *in the current automaton,* then *add a new transition* $p^i \xrightarrow{\gamma} \bigcup_k S_k$.
until *no new transition can be added*
remove the ϵ*-transitions. (generation number* i *is done)*
end for



Example of execution where $j = 4$. We consider $\Delta = \{pa \hookrightarrow p\}$, $R = p\Gamma^*$ $(F = \{p\})$, $P = P_0 = \{p, q\}$, $P_1 = \emptyset$. The i-th generation is given by the states of column i. The a-edges from p^i are added in the construction of generation i, since $pa \hookrightarrow p$ and $p^i \xrightarrow{\epsilon}_* p^i$, $p^i \xrightarrow{\epsilon}_* p^{i-1}$.

Figure 2. Automaton from Algorithm 12

Proposition 13 *Algorithm 12 constructs a* $\mathscr{P}$*-automaton* $\mathscr{B}_j$ *that recognizes exactly* Y_j *(using the states* p^j *as initial states).*

The sequence (Y_i) might be strictly decreasing, so that the previous algorithm can not reach a fixed point: for the example above one gets $Y_i = pa^i\Gamma^*$, $\forall i > 0$. But on the symbolic level we can modify Algorithm 12 to obtain an automaton for Y^∞ directly. As a preparation one defines a function ϕ which is the translation one column to the right of the elements of S in the figure above, with the convention that f stays f. For all finite sets $S \subseteq P \times \mathbb{N}$ let

$$\phi(S) = \begin{cases} \{q^i \mid q^{i+1} \in S\} \cup \{f\} & \text{if } f \in S \text{ or } \exists q^1 \in S \\ \{q^i \mid q^{i+1} \in S\} & \text{else} \end{cases}$$

We also use the projection $\pi^i(S)$ of the states in $P \times [1,i]$ to the i-th column (except for f). For all $i > 0$ and sets $S \subseteq P \times [1,i] \cup \{f\}$, let

$$\pi^i(S) = \{q^i \mid \exists i \geqslant k > 0, q^k \in S\} \cup \{f \mid f \in S\} \ .$$

Algorithm 14 (computation of Y^∞)
Input: *PDS $\mathscr{P}$, $F \subseteq P$ that defines the goal set $R = F\Gamma^*$*
Output: *a $\mathscr{P}$-automaton $\mathscr{C}$ that recognizes Y^∞*

Initialization: *the state space of $\mathscr{C}$ is a subset of $(P \times \mathbb{N}) \cup \{f\}$, where (p,i) is denoted by p^i and f is the unique final state. For all $\gamma \in \Gamma, f \xrightarrow{\gamma} f$ in $\mathscr{C}$. By convention p^0 is f for all $p \in F$.*
$i := 0$.

repeat
$i := i + 1$ *(consider now the p^i's as initial states.)*
Add an ϵ-transition from p^i to p^{i-1} for each $p \in F$ (only)
Add new transitions to $\mathscr{C}$ according to Algorithm 1 (see inner loop of Algorithm 12). Remove the ϵ-transitions.
Replace each transition $p^i \xrightarrow{\gamma} S$ by $p^i \xrightarrow{\gamma} \pi^i(S)$
until *$i > 1$ and the outgoing transitions from the p^i's are "the same" as from the p^{i-1}'s:*

$$p^i \xrightarrow{\gamma} S \Leftrightarrow p^{i-1} \xrightarrow{\gamma} \phi(S) \ .$$

Applying the algorithm to the simple example above, we see that the ϵ-transitions and the a-transitions from p^2 to p^1 and from p^3 to p^2 are deleted (by the projection), and that only three generations are created.

Theorem 15 *The automaton constructed by Algorithm 14 recognizes Y^∞, the winning region of the Büchi game given by the PDS $\mathscr{P}$ and goal set R.*

The theorem implies that Y^∞ is regular. Similarly to Section 2 each execution of the inner loop (saturation procedure) is done in time $|\Delta|\ 2^{\mathcal{O}(|Q|^2)}$, where $|Q| = 2|P| + 1$. At each step of the outer loop, we "lose" at least one transition. Since there are at most $|\Gamma|\ |P|\ 2^{|Q|}$ of them, the global time complexity of the algorithm is $\mathcal{O}(|\Gamma|\ |\Delta|\ 2^{c|P|^2})$. With an extension of the arguments of the previous section we obtain corresponding winning strategies:

Theorem 16 *For a Büchi game given by a PDS $\mathscr{P}$ and goal set R, one can compute from the automaton constructed by Algorithm 14 a min-rank (positional) winning strategy and a pushdown winning strategy, uniformly for the winning region of Player 0.*

5 Conclusion

We developed a symbolic approach to solve pushdown games with reachability and Büchi winning conditions. It allows to handle uniformly the whole winning region (of a given player) and to define uniformly two types of winning strategies, a positional one and a pushdown strategy. As an extension to the results presented here, one could consider parity games on pushdown graphs, or more general winning conditions. In this context it would be interesting to connect the present symbolic approach in a tighter way with the work of Walukiewicz [12]. Another generalization is to study games on prefix recognizable graphs.

Acknowledgement

Great thanks to Wolfgang Thomas, and also to Christophe Morvan, Christof Löding and Tanguy Urvoy for their helpful advice, to Olivier Corolleur for implementing the algorithm of Section 2, and to the referees for their comments.

References

1. A. Bouajjani, J. Esparza, and O. Maler, *Reachability analysis of pushdown automata: Application to model-checking*, CONCUR '97, LNCS 1243, pp 135–150, 1997.
2. A. Bouajjani, J. Esparza, A. Finkel, O. Maler, P. Rossmanith, B. Willems, and P. Wolper, *An efficient automata approach to some problems on context-free grammars*, Information Processing Letters, Vol 74, 2000.
3. D. Caucal, *On the regular structure of prefix rewriting*, CAAP '90, LNCS 431, pp. 87–102, 1990.
4. O. Corolleur, *Etude de jeux sur les graphes de transitions des automates à pile*, Rapport de stage d'option informatique, Ecole Polytechnique, 2001.
5. E. A. Emerson and C. S. Jutla, *Tree automata, mu-calculus and determinacy*, FoCS '91, IEEE Computer Society Press, pp. 368–377, 1991.
6. E. A. Emerson, C. S. Jutla, and A. P. Sistla, *On model-checking for fragments of μ-calculus*, CAV '93, LNCS 697, pp. 385–396, 1993.
7. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon, *Efficient Algorithm for Model Checking Pushdown Systems*, Technische Universität München, 2000.
8. O. Kupferman and M. Y. Vardi, *An Automata-Theoretic Approach to Reasoning about Infinite-State Systems*, CAV 2000, LNCS 1855, 2000.
9. O. Kupferman and M. Y. Vardi, *Weak Alternating Automata Are Not That Weak*, ISTCS'97, IEEE Computer Society Press, 1997.
10. C. Löding and W. Thomas, *Alternating Automata and Logics over Infinite Words*, IFIP TCS '00, LNCS 1872, pp. 521–535, 2000.
11. W. Thomas, *On the synthesis of strategies in infinite games*, STACS '95, LNCS 900, pp. 1–13, 1995.
12. I. Walukiewicz, *Pushdown processes: games and model checking*, CAV '96, LNCS 1102, pp 62–74, 1996. Full version in Information and Computation 164, pp. 234–263, 2001.

Appendix

We present here the proofs of the previous theorems and propositions.

Proof: Theorem 2
We consider the step-by-step construction of $\mathscr{A}_{Att(R)}$:

$$\mathscr{A}_R = \mathscr{A}_0,\ \mathscr{A}_1,\ \mathscr{A}_2,\ \cdots,\ \mathscr{A}_m = \mathscr{A}_{Att(R)}\ ,$$

where $\forall i < m$, "$\mathscr{A}_{i+1} = \mathscr{A}_i \cup \{p \xrightarrow{\gamma} S\}$", that is to say, exactly one transition is added.
The set $Attr_0(R)$ was defined by induction:
$Attr_0^0 = R$,
$Attr_0^{i+1} = Attr_0^i \cup \{pw \mid p \in P_0,\ \exists\ pw \hookrightarrow qv,\ qv \in Attr_0^i\}$
$\cup \{pw \mid p \in P_1,\ \forall\ pw \hookrightarrow qv,\ qv \in Attr_0^i\}$,
$Attr_0(R) = \bigcup_{i\in\mathbb{N}} Attr_0^i$.
We note $L(\mathscr{A}_m)$ the language recognized by $\mathscr{A}_m$.

First part: $L(\mathscr{A}_m) \supseteq Attr_0(R)$.
We use an induction on i to show that $\forall i \geqslant 0,\ L(\mathscr{A}_m) \supseteq Attr_0^i$.

- for $i = 0$, $R = Attr_0^0 = L(\mathscr{A}_0) \subseteq L(\mathscr{A}_m)$, because the transitions of $(\mathscr{A}_0)$ are still present in $\mathscr{A}_m$,
- induction hypothesis: $L(\mathscr{A}_m) \supseteq Attr_0^i$ for some i,
- then consider $pw \in Attr_0^{i+1} \backslash Attr_0^i$.

- First case: $p \in P_0$.
By the definition of $Attr_0^{i+1}$,

$$\exists\ pw \hookrightarrow qv,\ qv \in Attr_0^i \subseteq L(\mathscr{A}_m)\ .$$

Thus there is a path $q \xrightarrow{v}_* S,\ S \subseteq F$. We decompose the transition $\hookrightarrow$: $\exists \gamma \in \Gamma,\ w = \gamma u,\ v = v'u,\ p\gamma \hookrightarrow qv'$; and the path $q \xrightarrow{v}_* S$:

$$q \xrightarrow{v'}_* S' \xrightarrow{u}_* S\ .$$

By definition of Algorithm 1,

$$q \xrightarrow{v'}_* S'\ \wedge\ p\gamma \hookrightarrow qv' \quad \Rightarrow \quad \exists \text{ transition } p \xrightarrow{\gamma} S' \text{ in } \mathscr{A}_m\ .$$

As a consequence there is a path $p \xrightarrow{\gamma}_* S' \xrightarrow{u}_* S$ in $\mathscr{A}_m$, and so $p \xrightarrow{w}_* S \subseteq F$, $pw \in L(\mathscr{A}_m)$.

- Second case: $p \in P_1$.
By the definition of $Attr_0^{i+1}$, $\forall\ pw \hookrightarrow qv, qv \in Attr_0^i$. More precisely,
$\begin{cases} p\gamma u \hookrightarrow q_1 v_1 u \\ \vdots \qquad\quad \vdots \\ p\gamma u \hookrightarrow q_n v_n u \end{cases}$ are *all* the arcs in $\mathscr{G}$ starting from $p\gamma u$ and $\forall j,\ q_j v_j u \in Attr_0^i \subseteq$

$L(\mathscr{A}_m)$.
Thus there is paths $q_j \xrightarrow{v_j u}_* S_j \subseteq F$, that we decompose into

$$q_j \xrightarrow{v_j}_* S'_j \xrightarrow{u}_* S_j \ .$$

In the construction of $\mathscr{A}_m$, a new transition $p \xrightarrow{\gamma} \bigcup_j S'_j$ was added, and so

$$p \xrightarrow{\gamma} \bigcup_j S'_j \xrightarrow{u}_* \bigcup_j S_j \subseteq F \quad \Rightarrow \quad p\gamma u \in L(\mathscr{A}_m) \ .$$

From the induction we can conclude that $L(\mathscr{A}_m) \supseteq Attr_0(R)$.

Second part: $L(\mathscr{A}_m) \subseteq Attr_0(R)$.
This part of the proof is contained in those of Lemma 17, just forgetting all about the notions of cost ($Cost$) and number of moves.
We repeat here only the sketch of the proof.

We can prove by induction on m that $\forall p \in P,\ w \in \Gamma^*,$ if $p \xrightarrow{w}_* S$ in $\mathscr{A}_m$, then starting from pw, Player 0 can reach a configuration in the set

$$\left\{ p'w' \in P\Gamma^* \ \middle|\ \exists\ S' \subseteq S,\ p' \xrightarrow[\mathscr{A}_0]{w'}_* S' \right\}$$

whatever Player 1 does (in between). That is to say the play starting from pw will reach after some steps the given set *if Player 0 wants to*, but Player 0 can not choose which element of this set will be reached (this is more or less the choice of Player 1). We denote $\xrightarrow[\mathscr{A}_0]{w'}_*$ the global transition relation of $\mathscr{A}_0$.
In particular if $p \xrightarrow{w}_* S \ \subseteq F$ in $\mathscr{A}_m$, then from pw Player 0 can reach

$$\left\{ p'w' \mid p' \xrightarrow[\mathscr{A}_0]{w'}_* S' \subseteq S \subseteq F \right\} \ \subseteq \ R \ ,$$

which proves that he can win. ■

For the proofs of **Theorem 7** and **Proposition 6** we need the following lemma:

Lemma 17 *If a node pw is accepted by $\mathscr{A}_{Att(R)}$ with a run $\{p\} \xrightarrow{w}_* S \subseteq F$, then from this node Player 0 can join R in at most $Cost^*(\{p\} \xrightarrow{w}_* S)$ steps.*

Proof:
This is also the second part of the proof of Theorem 2, if we forget all about the notions of cost ($Cost$) and number of moves. We will prove by induction on m that $\forall p \in P,\ w \in \Gamma^*,$ if there is a run $\{p\} \xrightarrow{w}_* S$ in $\mathscr{A}_m$, then starting from pw, Player 0 can reach a configuration in the set

$\left\{ p'w' \in P\Gamma^* \ \middle|\ \exists\ S' \subseteq S,\ p' \xrightarrow[\mathscr{A}_0]{w'}_* S' \right\}$ after no more than $Cost(p \xrightarrow{w}_* S)$ moves,

whatever Player 1 does. We are counting the moves of both players.
In particular if there is a run $\{p\} \xrightarrow{w}_* S \ \subseteq F$ in $\mathscr{A}_m$, then from pw Player 0 can

win after no more than $Cost^*(\{p\} \xrightarrow{w}_* S)$ moves.

Recall first that we have supposed that in $\mathscr{A}_0$ no transition is leading to an initial state (a state of $P \subseteq Q$), and so

$$p' \xrightarrow[\mathscr{A}_0]{w'}_* S',\ S' \cap P \neq \emptyset \implies w' = \epsilon,\ p' \in S' \ . \tag{1}$$

We denote

$$\xrightarrow[m]{w}_* \text{ the global transition relation of } \mathscr{A}_m,$$
$$\xrightarrow[0]{w}_* \text{ the global transition relation of } \mathscr{A}_0 = \mathscr{A}_R.$$

We write $p \xrightarrow{w}_* S$ for the *existence* of a run from p to S labelled by w, whereas $\{p\} \xrightarrow[m]{w}_* S$ denotes a fixed run.
Induction on m:

- For $m = 0$, $p \xrightarrow[m]{w}_* S \iff p \xrightarrow[0]{w}_* S$, and from pw Player 0 can guarantee that pw is reached with 0 move. In fact $Cost^*\left(\{p\} \xrightarrow[0]{w}_* S\right) = 0 \geqslant 0$.
- Assume it is true for some $m \geqslant 0$,
- then we are considering $\mathscr{A}_{m+1}$:

$$\xrightarrow[m+1]{} = \xrightarrow[m]{} \cup \{t\},\ t = p_0 \xrightarrow{\gamma} S_0\ ,$$

  $Cost(t)$ is defined by the construction.

Let $p \xrightarrow[m+1]{w}_* S$.
We are now using an induction on j, the number of times that t is used in the run $\{p\} \xrightarrow[m+1]{w}_* S$.
- If $j = 0$, then $p \xrightarrow[m]{w}_* S$. From the induction hypothesis on m, we get the result.
- Suppose it is true for some $j \geqslant 0$.
- Consider that t is used $j + 1$ times in $\{p\} \xrightarrow[m+1]{w}_* S$. Decompose $w = u\gamma v$, such that:

$$\begin{array}{ccccccc} \{p\} & \xrightarrow[m]{u}_* & T_1 & \xrightarrow[m+1]{\gamma} & T_2 & \xrightarrow[m+1]{v}_* & S \\ & & p_0 & \xrightarrow{\gamma} & S_0 & & \end{array} \tag{2}$$

with $p_0 \in T_1,\ S_0 \subseteq T_2$, and $t = (p_0 \xrightarrow{\gamma} S_0)$ "is used" in $T_1 \xrightarrow[m+1]{\gamma} T_2$, for the "first time" in the run $\{p\} \xrightarrow[m+1]{w}_* S$.
From the induction on m, $\{p\} \xrightarrow[m]{u}_* T_1 \implies$

$$\begin{array}{c} pu \ \text{ guarantees } \ \left\{p_1 v_1 \mid p_1 \xrightarrow[0]{v_1}_* T_1' \subseteq T_1\right\} \\ \text{with no more moves than } Cost^*\left(\{p\} \xrightarrow[m]{u}_* T_1\right). \end{array} \tag{3}$$

From (2) we have also

$$\begin{array}{ccccc} T_1 \backslash \{p_0\} & \xrightarrow[m]{\gamma} & T_2' & \xrightarrow[m+1]{v}_* & T_3 \subseteq S \\ & & \mid\!\cap & & \\ & & T_2 & & \end{array}$$

The new transition t is used in the last formula (see $\xrightarrow[m+1]{v}{}^*$) less often than in (2) (less than $j+1$ times), so from the induction hypothesis on j, we obtain: from each configuration $t_1\gamma v$ in $(T_1\backslash\{p_0\})\gamma v$ Player 0 can reach the set

$$\left\{p_3 w_3 \mid p_3 \xrightarrow[0]{w_3}{}^* T'_3 \subseteq T_3\right\}$$
$$\text{with no more moves than } Cost^*\left(\{t_1\} \xrightarrow[m]{\gamma} T''_2 \xrightarrow[m+1]{v}{}^* T'_3\right). \tag{4}$$

For the rest of the proof, we will distinguish two cases:

case 0: $p_0 \in P_0$. By definition of $\mathscr{A}_{m+1}$,

$$p_0\gamma \hookrightarrow q_0 w_0, \tag{5}$$
$$q_0 \xrightarrow[m]{w_0}{}^* S_0$$
$$\text{and } Cost(t) = 1 + Cost^*\left(\{q_0\} \xrightarrow[m]{w_0}{}^* S_0\right) . \tag{6}$$

Together with (2) we get

$$\begin{array}{ccccc} \{q_0\} \xrightarrow[m]{w_0}{}^* & S_0 & \xrightarrow[m+1]{v}{}^* U_0 \subseteq S \\ & \rotatebox{90}{$\subseteq$} & \\ & T_2 & \end{array}$$

The new transition t is used in the last formula (see $\xrightarrow[m+1]{v}{}^*$) less often than in (2), so from the induction hypothesis on j, we obtain:

$$q_0 w_0 v \ \text{ guarantees } \ \left\{t_0 x_0 \mid t_0 \xrightarrow[0]{x_0}{}^* D_0 \subseteq U_0\right\}$$
$$\text{with no more moves than } Cost^*\left(\{q_0\} \xrightarrow[m]{w_0}{}^* S_0 \xrightarrow[m+1]{v}{}^* U_0\right). \tag{7}$$

case 1: $p_0 \in P_1$. By definition of $\mathscr{A}_{m+1}$,

$$\begin{cases} p_0\gamma \hookrightarrow q_1 w_1 \\ \vdots \qquad\quad \vdots \\ p_0\gamma \hookrightarrow q_n w_n \end{cases} \text{are all the moves from } p_0\gamma \text{ and } \forall i \ \ q_i \xrightarrow[m]{w_i}{}^* S_i, \tag{8}$$

$$S_0 = \textstyle\bigcup_i S_i,$$
$$Cost(t) = 1 + \max_i\left(Cost^*\left(\{q_i\} \xrightarrow[m]{w_i}{}^* S_i\right)\right) \tag{9}$$

Together with (2) we get

$$\begin{array}{ccccc} \forall i, \quad \{q_i\} \xrightarrow[m]{w_i}{}^* & S_i & \xrightarrow[m+1]{v}{}^* U_i \subseteq S \\ & \rotatebox{90}{$\subseteq$} & \\ & S_0 & \\ & \rotatebox{90}{$\subseteq$} & \\ & T_2 & \end{array}$$

The new transition t is used less often than in (2), so we have (induction on j):

$$\forall i, \ q_i w_i v \ \text{ guarantees } \ \left\{t_i x_i \mid t_i \xrightarrow[0]{x_i}{}^* D_i \subseteq U_i\right\}$$
$$\text{with no more moves than } Cost^*\left(\{q_i\} \xrightarrow[m]{w_i}{}^* S_i \xrightarrow[m+1]{v}{}^* U_i\right). \tag{10}$$

Now we can put all together.
From (3), we have in particular $pu\gamma v$ guarantees a configuration $p_1 v_1 \gamma v$ such that $p_1 \xrightarrow[0]{v_1}{}^{*} T_1' \subseteq T_1$, with no more than $Cost^*\left(\{p\} \xrightarrow[m]{u}{}^{*} T_1\right)$ moves. We distinguish the two following cases

either the path $\{p_1\} \xrightarrow[0]{v_1}{}^{*} T_1' \subseteq T_1$ is not leading to any initial state of $\mathscr{A}_0$, *i.e.*, $T_1' \cap P = \emptyset$, then with (2) the path

$$\begin{array}{ccccccc} \{p_1\} \xrightarrow[0]{v_1}{}^{*} & T_1' & \xrightarrow[m+1]{\gamma} & T_2'' & \xrightarrow[m+1]{v}{}^{*} S' & \subseteq S \\ & \rotatebox{90}{$\supseteq$} & & \rotatebox{90}{$\supseteq$} & & \\ & T_1 & & T_2 & & \end{array}$$

uses only transitions that were already in $\mathscr{A}_0$ (the new transitions are always starting from an initial state, of $P \subseteq Q$). It follows that

$$p_1 \xrightarrow[0]{v_1 \gamma v}{}^{*} S' \subseteq S \ ,$$

Note that $Cost^*\left(\{p_1\} \xrightarrow[0]{v_1\gamma v}{}^{*} S'\right) = 0 = Cost^*\left(\{p_1\} \xrightarrow[m+1]{v_1\gamma v}{}^{*} S'\right)$.

or the path $\{p_1\} \xrightarrow[0]{v_1}{}^{*} T_1' \subseteq T_1$ is actually leading to an initial state of $\mathscr{A}_0$ (in $T_1' \cap P$), then $v_1 = \epsilon$, $p_1 \in T_1'$ (see (1)), and $p_1 v_1 \gamma v = p_1 \gamma v$.
We consider two sub-cases (α) and (β).
(α) If $p_1 \in T_1 \backslash \{p_0\}$, then by (4),

$$p_1 \gamma v \text{ guarantees } \left\{p_3 w_3 \mid p_3 \xrightarrow[0]{w_3}{}^{*} T_3' \subseteq T_3 \subseteq S\right\}$$

$$\text{with no more moves than } Cost^*\left(\{t_1\} \xrightarrow[m]{\gamma} T_2'' \xrightarrow[m+1]{v}{}^{*} T_3'\right) \ .$$

(β) If $p_1 = p_0$ ($\in T_1 \cap P$)

case 0: $p_0 \in P_0$, Player 0 can choose, from $p_0 \gamma v$, to go to $q_0 w_0 v$ (see (5)), *in one move* and with (7),

$$q_0 w_0 v \text{ guarantees } \left\{t_0 x_0 \mid t_0 \xrightarrow[0]{x_0}{}^{*} D_0 \subseteq U_0 \subseteq S\right\}$$

$$\text{with no more moves than } Cost^*\left(\{q_0\} \xrightarrow[m]{w_0}{}^{*} S_0 \xrightarrow[m+1]{v}{}^{*} U_0\right) \ .$$

case 1: $p_0 \in P_1$, Player 0 just can wait, but from $p_0 \gamma v$ Player 1 can only choose *with one move* one of the states $q_i w_i v$ (see (8)), and with (10),

$$q_i w_i v \text{ guarantees } \left\{t_i x_i \mid t_i \xrightarrow[0]{x_i}{}^{*} D_i \subseteq U_i \subseteq S\right\}$$

$$\text{with no more moves than } Cost^*\left(\{q_i\} \xrightarrow[m]{w_i}{}^{*} S_i \xrightarrow[m+1]{v}{}^{*} U_i\right) \ .$$

In every case $pu\gamma v$ guarantees $p_1 v_1 \gamma v$, that guarantees (or is already in)

$$\left\{p' w' \mid p' \xrightarrow[0]{w'}{}^{*} S' \subseteq S\right\} \ .$$

By transitivity of "guarantees", we have

$$\text{from } pu\gamma v \text{ Player 0 can reach } \left\{p' w' \mid p' \xrightarrow[\mathscr{A}_0]{w'}{}^{*} S' \subseteq S\right\} \ ,$$

with no more than ("either")

$$Cost^*\left(\{p\} \xrightarrow[m]{u}_* T_1\right)$$

respectively ("or", α)

$$Cost^*\left(\{p\} \xrightarrow[m]{u}_* T_1\right) + Cost^*\left(\{t_1\} \xrightarrow[m]{\gamma} T_2'' \xrightarrow[m+1]{v}_* T_3'\right)$$

respectively ("or", β, case 0), see (6)

$$Cost^*\left(\{p\} \xrightarrow[m]{u}_* T_1\right) + 1 + Cost^*\left(\{q_0\} \xrightarrow[m]{w_0}_* S_0 \xrightarrow[m+1]{v}_* U_0\right)$$

$$= Cost^*\left(\{p\} \xrightarrow[m]{u}_* T_1\right) + Cost(t) + Cost^*\left(S_0 \xrightarrow[m+1]{v}_* U_0\right)\right)$$

respectively ("or", β, case 1), see (9)

$$Cost^*\left(\{p\} \xrightarrow[m]{u}_* T_1\right) + 1 + Cost^*\left(\{q_i\} \xrightarrow[m]{w_i}_* S_i \xrightarrow[m+1]{v}_* U_i\right)$$

$$= Cost^*\left(\{p\} \xrightarrow[m]{u}_* T_1\right) + Cost(t) + Cost^*\left(S_i \xrightarrow[m+1]{v}_* U_i\right)\right)$$

moves.
That is in every case no more than $Cost^*(\{p\} \xrightarrow[m+1]{w}_* S)$ moves, by the definition of the cost and by (2).
The property is proved for $\mathscr{A}_{m+1}$. From the induction we can conclude that it is proved for each $m \geqslant 0$.

(One can simplify a little beat the end of the proof in the case where $\mathscr{A}_0$ is a finite automaton.) ■

Proof: Theorem 7 According to the strategy, if $p \in P_0$, Player 0 has to find the cheapest run on $\mathscr{A}_m$. Decompose this run: $w = \gamma w'$,

$$\{p\} \xrightarrow{\gamma} T \xrightarrow{w'}_* S$$

choose the transition $Rule(p \xrightarrow{\gamma} T) = p\gamma \hookrightarrow qv$ that were used to construct the transition $p \xrightarrow{\gamma} T$ in $\mathscr{A}_{Att(R)}$. After that the play is in the state qvw'. We can remark that

$$Cost^*\left(\{p\} \xrightarrow{\gamma} T \xrightarrow{w'}_* S\right) = 1 + Cost^*\left(\{q\} \xrightarrow{v}_* T \xrightarrow{w'}_* S\right) .$$

So the "distance" to R has decreased. It is the same if $p \in P_1$ (for each possible move of Player 1). The maximal number of steps needed to reach R is decreasing, so the play will eventually reach R, see Lemma 17. (and it is possible, if Player 1 plays "badly", that the number of steps decreases faster) ■

The Theorem 7 has been proved without using the Proposition 6, with the previous lemma, but the proposition states that the strategy is optimal.

Proof: Proposition 6 From the previous facts it follows that

$$rank(pw) \leqslant \min\{Cost^*(\{p\} \xrightarrow{w}_* S) \mid \{p\} \xrightarrow{w}_* S \subseteq F\} .$$

For the converse inequality, it is easy to show by induction that for all $i \geqslant 0$

$$rank(pw) = i \ \Rightarrow\ \min\{Cost^*(\{p\} \xrightarrow{w}_* S) \mid \{p\} \xrightarrow{w}_* S \subseteq F\} \geqslant i \ .$$

■

Proof: Theorem 9 We consider a play according to the strategy: a path in the transition graph of the strategy. This is a sequence $\pi_0 \cdots \pi_n$ of configurations of the strategy such that the stack of π_0 describes an accepting run of $\mathscr{A}_{Att(R)}$. By definition of Algorithm 1 it is easy to show by induction on n that:

- the stack π_n describes also an accepting run of $\mathscr{A}_{Att(R)}$,
- the cost of this run is strictly smaller than the cost of the run described in π_{n-1} (each move of the strategy replaces the first segment of the run by a "cheaper" run).

■

Proof: Remark 1
There is a deterministic (complete) finite automaton $\mathcal{D} = (\Gamma, Q, \delta, q_0, E)$ that recognizes R *backwards*: reading words from right to left. We can simulate $\mathcal{D}$ in a new PDS $\mathscr{P} \times \mathcal{D} = (P \times Q, \Gamma \times Q, \Delta')$ where a configuration

$$p\gamma_0 \cdots \gamma_n$$

of $\mathscr{P}$ is associated, via the unique run of $\mathcal{D}$

$$q_{n+2} \xleftarrow{p} q_{n+1} \xleftarrow{\gamma_0} q_n \cdots \xleftarrow{\gamma_n} q_0$$

to the configuration of $\mathscr{P} \times \mathcal{D}$

$$(p, q_{n+1})(\gamma_0, q_n) \cdots (\gamma_n, q_0) \ .$$

Then we know that $p\gamma_0 \cdots \gamma_n \in R \ \Leftrightarrow \ \delta(q_{n+1}, p) \in E$. Let $F = \{(p,q) \in P \times Q \mid \delta(q,p) \in E\}$ and define the transitions of $\mathscr{P} \times \mathcal{D}$ to be

$$\langle (p,q)(\gamma, q_1) \hookrightarrow (p',q')(\gamma_n, q_n) \cdots (\gamma_1, q_1) \rangle \in \Delta' \Leftrightarrow$$

$$(p\gamma \hookrightarrow p'\gamma_n \cdots \gamma_1) \in \Delta, q_{i+1} = \delta(q_i, \gamma_i), q' = \delta(q_n, \gamma_n) \ .$$

(if $n = 0$, $q' = q_1$) If the initialization is done correctly, the stack will contain a correct run at every step, and the connected component of the graph of $\mathscr{P}$ is isomorphic to the corresponding connected component of $\mathscr{P} \times \mathcal{D}$, with the property that R corresponds to $R' = F\Gamma^*$. ■

Proof: Proposition 13
By induction on j:
We note $\mathscr{B}_j$ the automaton obtained after the j-th generation, with initial states p^j, $p \in P$.
- using the convention that p^0 is f for all $p \in P$, the automaton $\mathscr{B}_0$ recognizes

$P\Gamma^* = V = Y_0$
- induction hypothesis: $\mathscr{B}_j$ recognizes Y_j
- then $Y_j \cap R$ is recognized on $\mathscr{B}_j$ from the states $p^j, p \in F$, because of the simple structure of R (*i.e.*, $pw \in Y_j \cap R$ if pw is accepted by $\mathscr{B}_j$ *and* $p \in F$).
By the help of the ϵ-transitions, the automaton $\mathscr{B}_{j+1}$ recognizes exactly $Y_j \cap R$ before the saturation procedure starts. The construction of $\mathscr{B}_{j+1}$ from $\mathscr{B}_j$ was proved in Section 2 to compute the attractor: $\mathscr{B}_{j+1}$ recognizes then $Attr_0(Y_j \cap R)$. After we remove the ϵ-transitions, it defines $Attr_0^+(Y_j \cap R)$: the configurations of $Attr_0(Y_j \cap R)$ such that after one move we are in $Attr_0(Y_j \cap R)$. In other words if a configuration pw can be accepted by $\mathscr{B}_{j+1}$ *only* by using an ϵ-transition, then it is in $Y_j \cap R$, but not in $Attr_0^+(Y_j \cap R)$. ■

The next proposition states that there are less and less transitions in the higher generations (we already know that $Y_{i+1} \subseteq Y_i$ by monotonicity).

Proposition 18 *In Algorithm 12, for all $u \in \Gamma^*, p \in P, i \geqslant 1$,*

$$p^{i+1} \xrightarrow{u}_* S \ \Rightarrow\ p^i \xrightarrow{u}_* \phi(S) \ .$$

Proof: Proposition 18
We proceed by induction on i.
It is a direct consequence of the same property over the transitions (by induction on the length of the word).

- For $i = 1$, we use another induction on the number of transitions starting from p^2 added by the saturation procedure.
  - At the beginning of the second iteration, one has no other transitions from the q^2's than the $q^2 \xrightarrow{\epsilon} \{q^2\}$, and for all $q \in F$, $q^2 \xrightarrow{\epsilon} \{q^1\}$. And from the q^1's there may be paths $q^1 \xrightarrow{w}_* S$ such that $S \subseteq P \times \{1\} \cup \{f\}$, hence $\phi(S) = \{f\}$.
  Respectively, at the beginning of the first iteration, before the saturation procedure starts, one *had* $q^1 \xrightarrow{\epsilon} \{q^1\}$, if $q \in F$, then $q^1 \xrightarrow{\epsilon} \{f\}$, and $\forall \gamma, f \xrightarrow{\gamma} \{f\}$.
  - as a new transition $p^2 \xrightarrow{\gamma} S$ is added, it is through an existing path $q^2 \xrightarrow{w}_* S$, by induction hypothesis one has also $q^1 \xrightarrow{w}_* \phi(S)$ during the saturation procedure of the iteration one, so the transition $p^1 \xrightarrow{\gamma} \phi(S)$ was already added.
- induction hypothesis: $\forall S,\ p^i \xrightarrow{a} S \Rightarrow p^{i-1} \xrightarrow{a} \phi(S)$
- the proof for $i+1$ is similar to the case $i = 1$

■
Note that in Algorithm 14 we can erase the p^{i-1}'s and their transitions as soon as the generation i is done. Similarly to the first algorithm, we have the following property.

Proposition 19 *In Algorithm 14, for all $u \in \Gamma^*, p \in P, i \geqslant 1$,*

$$p^{i+1} \xrightarrow{u}_* S \Rightarrow p^i \xrightarrow{u}_* \phi(S) \ .$$

Remark that because of the projection π, the transitions $p^i \xrightarrow{u}_* S$ respect $S \subseteq \{f\} \cup P \times \{i\}$.

Proof: Proposition 19
The proof is similar to that of the first algorithm.
We proceed by induction on i.
It is a direct consequence of the same property over the transitions (by induction on the length of the word).

- For $i = 1$, we use another induction on the number of transitions starting from p^2 added by the saturation procedure. Which means in this case, if a transition $p^2 \xrightarrow{\gamma} S$ is added, then its projection respect the property, that is
$$p^1 \xrightarrow{\gamma} \phi(\pi^2(S)) \ ,$$
because at the end of generation 2, we will have $p^2 \xrightarrow{\gamma} \pi^2(S)$.
- At the beginning of the second iteration, one has no other transitions from the q^2's than the $q^2 \xrightarrow{\epsilon} \{q^2\}$, and for all $q \in F$, $q^2 \xrightarrow{\epsilon} \{q^1\}$. We have to check that $q^1 \xrightarrow{\epsilon} \phi(\pi^2(q^1)) = \phi(q^2) = q^1$, which is clear.
And from the q^1's there may be paths $q^1 \xrightarrow{w}_* S$ such that $S \subseteq P \times \{1\} \cup \{f\}$.
Hence from q^2 (only if $q \in F$), we have
$q^2 \xrightarrow{\epsilon} q^1 \xrightarrow{w}_* S \subseteq P \times \{1\} \cup \{f\}$ with
$\pi^2(S) \subseteq P \times \{2\} \cup \{f\}$, $\phi(\pi^2(S)) \subseteq P \times \{1\} \cup \{f\}$, and $\phi(\pi^2(S)) = S$.
So we have $q^1 \xrightarrow{w}_* \phi(\pi^2(S))$.
- *during* the saturation procedure, as a new transition $p^2 \xrightarrow{\gamma} S$ is added, it is through an existing path $q^2 \xrightarrow{w}_* S$, by induction hypothesis one has also $q^2 \xrightarrow{w}_* \pi^2(S) \Rightarrow q^1 \xrightarrow{w}_* \phi(\pi^2(S))$ so the transition $p^1 \xrightarrow{\gamma} \phi(\pi^2(S))$ exists since the projection of the first generation.
- induction hypothesis: $\forall S,\ p^i \xrightarrow{a} S \Rightarrow p^{i-1} \xrightarrow{a} \phi(S)$,
- the proof for $i+1$ is here exactly the same as the case $i = 1$.

■

Proof: Theorem 15
Proof of termination
Thanks to the projections π^i's, the number of possible transitions from each row of p^i's is bounded. And thanks to Proposition 19, there is less and less transitions until the algorithm reaches a fixed point.

Proof of correctness
We note Z_i the language recognized by $\mathscr{C}$ from the initial states p^i's. We denote $n+1$ the last generation of the algorithm, which is such that $Z_n = Z_{n+1}$, but we still consider $Z_i = Z_n$ for all $i \geqslant n$. One has to show that $Z_n = Y^\infty$.

First part: $Z_n \subseteq Y^\infty$.
We prove by induction on i that for all i, $Z_i \subseteq Y_i$.

- By construction, $Z_1 = Y_1$.

- Induction hypothesis: $Z_i \subseteq Y_i$.
- The algorithm first determines the attractor$^+$ of the language. By monotonicity,

$$\tilde{Z}_{i+1} = Attr_0^+(Z_i \cap R) \subseteq Attr_0^+(Y_i \cap R) = Y_{i+1} \ .$$

  After the projection of the transitions, the obtained language Z_{i+1} is contained in $\tilde{Z}_{i+1}$: Proposition 19 shows that an accepting path from a state p_{i+1} was possible before the projection (through the q^i's). So $Z_{i+1} \subseteq \tilde{Z}_{i+1} \subseteq Y_{i+1}$.

We conclude that $\forall i, Z_n = Z_{n+1} \subseteq Y_i$, and so $Z_n \subseteq Y^\infty$.

Second part: $Y^\infty \subseteq Z_n$.
We prove by induction on i that for all i, $Y^\infty \subseteq Z_i$.

- By construction, $Y^\infty \subseteq Y_1 = Z_1$.
- Induction hypothesis: $Y^\infty \subseteq Z_i$.
- Before the projection we have

$$Y^\infty = Attr_0^+(Y^\infty \cap R) \subseteq Attr_0^+(Z_i \cap R) = \tilde{Z}_{i+1} \ .$$

  We proceed by induction on the number of transitions that are changed by the projection, *i.e.*, we consider the successive automata $\mathscr{A}_0, \cdots, \mathscr{A}_m$, where $L(\mathscr{A}_0) = \tilde{Z}_{i+1}, L(\mathscr{A}_m) = Z_{i+1}$, and $\mathscr{A}_{j+1}$ is obtained from $\mathscr{A}_j$ by "projecting" one transition. We have to prove by induction on m that $Y^\infty \subseteq L(\mathscr{A}_m)$.
  - If $m = 0$, $Y^\infty \subseteq L(\mathscr{A}_0) = \tilde{Z}_{i+1} = Z_{i+1}$.
  - Induction hypothesis: $Y^\infty \subseteq L(\mathscr{A}_m)$.
  - We suppose by absurd that there is a configuration $pw \in L(\mathscr{A}_m) \backslash L(\mathscr{A}_{m+1})$ such that $pw \in Y^\infty$. We choose that of minimal length $|pw|$. For each accepting path in $\mathscr{A}_m$ for pw, there is a decomposition $p^{i+1} \xrightarrow[m]{u}_* S \xrightarrow[m]{v}_* \{f\}$ such that $w = uv$, with $q^i \in S$, and in $\mathscr{A}_{m+1}$: $\neg(q^{i+1} \xrightarrow[m+1]{v}_* \{f\})$ (the transition that is projected was "leading" to q^i in $\mathscr{A}_m$ and is now "leading" to q^{i+1}).
  This means that $qv \in Z_i$ and $qv \notin L(\mathscr{A}_{m+1})$.
  If $qv \notin L(\mathscr{A}_m)$, then $qv \notin Y^\infty$ (Induction hypothesis).
  If $qv \in L(\mathscr{A}_m) \backslash L(\mathscr{A}_{m+1})$, then qv cannot be in Y^∞ by hypotheses: $u \neq \epsilon$ and $|qv| < |pw|$.
  In both cases, pw should not be in Y^∞ (see Section 2), hence the contradiction.
  We conclude that $Y^\infty \subseteq L(\mathscr{A}_{m+1})$.

■

Proof: Theorem 16
By definition of Y^∞ and by Algorithm 14, $Attr_0^+(Y^\infty \cap R) = Y^\infty$, and Player 0 can use the strategy associated to that attractor. ■