Model checking CTL Properties of Pushdown Systems

Igor Walukiewicz *

Institute of Informatics, Warsaw University,
Banacha 2, 02-097 Warsaw, POLAND,
e-mail: igw@mimuw.edu.pl

Abstract: A *pushdown system* is a graph $G(P)$ of configurations of a pushdown automaton P. The *model checking problem* for a logic L is: given a pushdown automaton P and a formula $\alpha \in L$ decide if α holds in the vertex of $G(P)$ which is the initial configuration of P. Computation Tree Logic (CTL) and its fragment EF are considered. The model checking problems for CTL and EF are shown to be EXPTIME-complete and PSPACE-complete, respectively.

1 Introduction

A *pushdown system* is a graph $G(P)$ of configurations of a pushdown automaton P. The edges in this graph correspond to single steps of computation of the automaton. The *pushdown model checking problem* (PMC problem) for a logic L is: given a pushdown automaton P and a formula $\alpha \in L$ decide if α holds in the vertex of $G(P)$ which is the initial configuration of P. This problem is a strict generalization of a more standard model checking problem where only finite graphs are considered.

In this paper we consider PMC problem for two logics: CTL and EF. CTL is the standard Computation Tree Logic [4, 5]. EF is a fragment of CTL containing only operators: exists a successor ($\exists \circ \alpha$), and exists a reachable state ($\exists F \alpha$). Moreover, EF is closed under conjunction and negation. We prove the following:

- The PMC problem for EF logic is PSPACE-complete.
- The PMC problem for CTL is EXPTIME-complete.

The research on the PMC problem continues for some time. The decidability of this problem for monadic second order logic (MSOL) follows from [8] (for a simpler argument see [2]). This implies decidability of the problem for all those logics which have effective translations to MSOL. Among them are the μ-calculus, CTL* as well as the logics considered here. This general result however gives only nonelementary upper bound on the complexity of PMC. In [9] an EXPTIME-completeness of PMC for the μ-calculus was proved. This result was slightly encouraging because the complexity is not that much bigger than the complexities of known algorithms for the model checking problem over finite

* The author was supported by Polish KBN grant No. 8 T11C 027 16.

graphs. In [1] it was shown that the PMC problems for LTL and linear time μ-calculus are EXPTIME-complete.

The PMC problem for EF was considered already in [1]. It was shown there that the problem is PSPACE-hard. Moreover it was argued the the general method of the paper gives a PSPACE algorithm for the problem. Later, a closer analysis showed that there is no obvious way of implementing the method in polynomially bounded space [6]. The algorithm presented here follows the idea used in [9] for the μ-calculus.

The EXPTIME hardness results for the alternation free μ-calculus and LTL show two different reasons for the hardness of the PMC problem. One is unbounded alternation, the other is the ability to compare two consecutive blocks of states on a path. The reachability problem for pushdown systems is of course solvable in polynomial time (see [7] for a recent paper on this problem). Over finite graphs the model checking problem for CTL reduces to a sequence of reachability tests. This suggested that PMC problem for CTL may be PSPACE-complete. In this light EXPTIME-hardness result is slightly surprising. The argument combines ideas from the hardness results for the μ-calculus and LTL. It essentially shows that $\exists G\alpha$ operator (there is a path on which α always holds) is enough to obtain EXPTIME-hardness. This result makes EF logic more interesting as it is a fragment of CTL that disallows $\exists G\alpha$ but still allows $\forall G\alpha$ (for all paths α always holds).

Next section gives definitions concerning logics and pushdown systems. Section 3 presents an assumption semantics of EF. This semantics allows to formulate the induction argument in the correctness proof of the model checking algorithm. The proof is described in Section 4. The final section presents EXPTIME-hardness result of the PMC problem for CTL.

2 Preliminaries

In this section we present CTL and EF logics. We define pushdown systems and the model checking problem.

CTL and EF logics Let *Prop* be a set of propositional letters; let $p, p', \dots$ range over *Prop*.

The set of formulas of EF logic, $Form(EF)$, is given by the grammar: $\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \beta \mid \exists{\circ}\alpha \mid \exists F\alpha$. For CTL the grammar is extended with the clauses: $\exists(\alpha_1 U \alpha_2) \mid \exists\neg(\alpha_1 U \alpha_2)$.

The models for the logic are labelled graphs $\langle V, E, \rho\rangle$; where V is the set of vertices, E is the edge relation and $\rho : Prop \to \mathcal{P}(V)$ is a labelling function assigning to each vertex a set of propositional letters. Such labelled graphs are called *transition systems* here. In this context vertices are also called states.

Let $M = \langle V, E, \rho\rangle$ be a transition system. The meaning of a formula α in a state v is defined by induction. Tha clauses for propositional letters, negation and conjunction are standard. For the other constructs we have:

- $M, v \vDash \exists{\circ}\alpha$ if there is a successor v' of v such that $M, v' \vDash \alpha$.

– $M, v \vDash \exists F\alpha$ if there is a path from v to v', s.t. $M, v' \vDash \alpha$.
– $M, v \vDash \exists(\alpha U \beta)$ if there is a path from v to v', s.t. $M, v' \vDash \beta$ and for all the verticies v'' on the path other than v' we have $M, v'' \vDash \alpha$.
– $M, v \vDash \exists\neg(\alpha U \beta)$ if there is a maximal (i.e., infinte or finite ending in a vertex without successors) path π from v s.t. for every vertex v' on π with $M, v' \vDash \beta$ there is an earlier vertex v'' on π with not $M, v'' \vDash \alpha$.

We will freelly use abbreviations:

$$\alpha \vee \beta = \neg(\neg\alpha \wedge \neg\beta) \qquad \forall\circ\alpha = \neg\exists\circ\neg\alpha \qquad \forall G\alpha = \neg\exists F\neg\alpha$$

Using these one can convert every formula of EF logic to an equivalent *positive formula* where all the negations occur only before propositional letters.

Pushdown systems A *pushdown system* is a tuple $P = \langle Q, \Gamma, \Delta, q_0, \bot\rangle$ where Q is a finite set of *states*, Γ is a finite *stack alphabet* and $\Delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma^*)$ is the set of *transition rules*. State $q_0 \in Q$ is the *initial state* and symbol $\bot \in \Gamma$ is the *initial stack symbol*.

We will use q, z, w to range over Q, Γ and Γ^* respectively. We will write $qz \rightarrowtail_\Delta q'w$ instead of $((q, z), (q', w)) \in \Delta$. We will omit subscript Δ if it is clear from the context.

In this paper we will restrict ourselves to pushdown systems with transition rules of the form $qz \rightarrowtail q'$ and $qz \rightarrowtail qz'z$. Operations pushing more elements on the stack can be simulated with only polynomial increase of the size of a pushdown system. We will also assume that $\bot$ is never taken from the stack, i.e., that there is no rule of the form $q\bot \rightarrowtail q'$ for some q, q'.

Let us now give the semantics of a pushdown system $P = \langle Q, \Gamma, \Delta, q_0, \bot\rangle$. A *configuration* of P is a word $qw \in Q \times \Gamma^*$. The configuration $q_0\bot$ is the *initial configuration*. A pushdown system P defines an infinite graph $G(P)$ which nodes are configurations and which edges are: $(qzw, q'w) \in E$ if $qz \rightarrowtail_\Delta q'$, and $(qzw, q'z'zw) \in E$ if $qz \rightarrowtail_\Delta q'z'z$; for arbitrary $w \in \Gamma^*$.

Given a valuation $\rho : Q \to \mathbb{P}(Prop)$ we can extend it to $Q \times \Gamma^*$ by putting $\rho(qw) = \rho(q)$. This way a pushdown system P and a finite valuation ρ define a, potentially infinite, transition system $M(P, \rho)$ which graph is $G(P)$ and which valuation is given by ρ as described above.

The *model checking problem* is:

given P, ρ and φ decide if $M(P, \rho), q_0\bot \vDash \varphi$

Please observe that the meaning of φ in the initial configuration $q_0\bot$ depends only on the part of $M(P, \rho)$ that is reachable from $q_0\bot$.

3 Assumption semantics

For this section let us fix a pushdown system P and a valuation ρ. Let us abbreviate $M(P, \rho)$ by M.

We are going to present a modification of the semantics of EF-logic. This modified semantics is used as an induction assumption in the algorithm we are going to present later. From the definition of a transition system M it follows that there are no edges from vertices q, i.e., configurations with the empty stack. We will look at such vertices not as dead ends but as places where some parts of the structure where cut out. We will take a function $S : Q \to \mathbb{P}(Form(EF))$ and interpret $S(q)$ as an assumption that in the vertex q formulas $S(q)$ hold. This view leads to the following definition.

Definition 1. *Let $S : Q \to \mathbb{P}(Form(EF))$ be a function. For a vertex v of $G(P)$ and a formula α we define the relation $M, v \vDash_S \alpha$ as the least relation satisfying the following conditions:*

- *$M, q \vDash_S \alpha$ for every $\alpha \in S(q)$.*
- *$M, v \vDash_S p$ if $p \in \rho(v)$.*
- *$M, v \vDash_S \alpha \wedge \beta$ if $M, v \vDash_S \alpha$ and $M, v \vDash_S \beta$.*
- *$M, v \vDash_S \alpha \vee \beta$ if $M, v \vDash_S \alpha$ or $M, v \vDash_s \beta$.*
- *$M, v \vDash_S \exists\circ\alpha$ for $v \notin Q$ if there is a successor v' of v such that $M, v' \vDash_S \alpha$.*
- *$M, v \vDash_S \forall\circ\alpha$ for $v \notin Q$ if for every successor v' of v we have that $M, v' \vDash_S \alpha$.*
- *$M, v \vDash_S \exists F\alpha$ if there is a path from v to v', s.t. $M, v' \vDash_S \alpha$ or $v' = q$ for some state $q \in Q$ and $\exists F\alpha \in S(q)$.*
- *$M, v \vDash_S \forall G\alpha$ iff for every path π from v which is either infinite or finite ending in a vertex from Q we have that $M, v' \vDash_S \alpha$ for every vertex v' of π and moreover if π is finite and ends in a vertex $q \in Q$ then $\forall G\alpha \in S(q)$.*

Of course taking arbitrary S in the above semantics makes little sense. We need some consistency conditions as defined below.

Definition 2. *A set of formulas B is* saturated *if*

- *for every formula α either $\alpha \in B$ or $\neg\alpha \in B$ but not both;*
- *if $\alpha \in B$ and $\beta \in B$ then $\alpha \wedge \beta \in B$;*
- *if $\alpha \in B$ then $\alpha \vee \beta \in B$ and $\beta \vee \alpha \in B$ for arbitrary β;*
- *if $\alpha \in B$ then $\exists F\alpha \in B$.*

Definition 3 (Assumption function). *A function $S : Q \to \mathbb{P}(Form(EF))$ is* saturated *if $S(q)$ is saturated for every $q \in Q$. A function S is* consistent with *ρ if $S(q) \cap Prop = \rho(q)$ for all $q \in Q$. We will not mention ρ if it is clear from the context. We say that S is an* assumption function *(for ρ) if it is saturated and consistent.*

Lemma 1. *For every assumption function S and every vertex v of M: $M, v \vDash_S \alpha$ iff not $M, v \vDash_S \neg\alpha$.*

The next lemma says that the truth of α depends only on assumptions about subformulas of α.

Definition 4. *For a formula α, let* $\mathrm{cl}(\alpha)$ *be the set of subformulas of α and their negations.*

Lemma 2. *Let α be a formula. Let S, S' be two assumption functions such that $S(q) \cap \mathrm{cl}(\alpha) = S'(q) \cap \mathrm{cl}(\alpha)$ for all $q \in Q$. For every v we have that: $M, v \vDash_S \alpha$ iff $M, v \vDash_{S'} \alpha$.*

We have asumed that the initial stack symbol $\perp$ cannot be taken from the stack. Hence no state q is reachable from configuration $q_0\perp$. In this case our semantics is equivalent to the usual one:

Lemma 3. *For arbitrary S and α we have $M, q_0\perp \vDash_S \alpha$ iff $M, q_0\perp \vDash \alpha$.*

We finish this section with a composition lemma which is the main property of our semantics. We will use it in induction arguments.

Definition 5. *For a stack symbol z and an assumption function S we define the function $S\uparrow_z$ by: $S\uparrow_z (q') = \{\beta : M, q'z \vDash_S \beta\}$, for all $q' \in Q$.*

Lemma 4 (Composition Lemma). *Let α be a formula, z a stack symbol and S an assumption function. Then $S\uparrow_z$ is an assumption function and for every configuration qwz' reachable from qz' we have:*

$$M, qwz'z \vDash_S \alpha \quad \textit{iff} \quad M, qwz' \vDash_{S\uparrow_z} \alpha$$

4 Model checking EF

As in the previous section let us fix a pushdown system P and a valuation ρ. Let us write M instead of $M(P, \rho)$ for the transition system defined by P and ρ.

Instead of the model checking problem we will solve a more general problem of deciding if $M, qz \vDash_S \beta$ holds for given q, z, S and β. A small difficulty here is that S is an infinite object. Fortunately, by Lemma 2 to decide if $M, qz \vDash_S \beta$ holds it is enough to work with S restricted to subformulas of β, namely with $S|_\beta$ defined by $S|_\beta(q') = S(q') \cap \mathrm{cl}(\beta)$ for all $q' \in Q$. In this case we will also say that S is *extending* $S|_\beta$.

Definition 6. *Let α be a formula, q a state, z a stack symbol, and $\overline{S} : Q \to \mathbb{P}(Form(EF))$ a function assigning to each state a subset of $\mathrm{cl}(\alpha)$. We will say that a tuple $(\alpha, q, z, \overline{S})$ is* good *if there is an assumption function S such that $S|_\alpha = \overline{S}$ and $M, qz \vDash_S \alpha$.*

Below we describe a procedure which checks if a tuple $(\alpha, q, z, \overline{S})$ is good. It uses an auxiliary procedure Search(q, z, q') which checks whether there is a path from the configuration qz to the configuration q'.

- Check$(p, q, z, \overline{S}) = 1$ if $p \in \rho(q)$;
- Check$(\alpha \wedge \beta, q, z, \overline{S}) = 1$ if Check$(\alpha, q, z, \overline{S}) = 1$ and Check$(\beta, q, z, \overline{S}) = 1$;
- Check$(\neg\alpha, q, z, \overline{S}) = 1$ if Check$(\alpha, q, z, \overline{S}) = 0$;
- Check$(\exists\circ\alpha, q, z, \overline{S}) = 1$ if either
  - there is $qz \rightarrowtail q'$ and $\alpha \in \overline{S}(q')$; or

  - there is $qz \rightarrowtail q'z'z$ and $\text{Check}(\alpha, q', z', \overline{S}') = 1$, where $\overline{S}'$ is defined by: $\overline{S}'(q'') = \{\beta \in \text{cl}(\alpha) : \text{Check}(\beta, q'', z, \overline{S}) = 1\}$, for all $q'' \in Q$.
- $\text{Check}(\exists F\alpha, q, z, \overline{S}) = 1$ if either
  - $\text{Check}(\alpha, q, z, \overline{S}) = 1$; or
  - there is $qz \rightarrowtail q'$ and $\exists F\alpha \in \overline{S}(q')$; or
  - there is $qz \rightarrowtail q'z'z$ and $q'' \in Q$ for which $\text{Search}(q', z', q'') = 1$ and $\text{Check}(\exists F\alpha, q'', z, \overline{S}) = 1$; or
  - there is $qz \rightarrowtail q'z'z$ with $\text{Check}(\exists F\alpha, q', z', \overline{S}') = 1$ for $\overline{S}'$ defined by: $\overline{S}'(q'') = \{\exists F\alpha : \text{Check}(\alpha, q'', z, \overline{S}) = 1\} \cup \{\neg\exists F\alpha : \text{Check}(\alpha, q'', z, \overline{S}) = 0\} \cup \{\beta \in \text{cl}(\alpha) : \text{Check}(\beta, q'', z, \overline{S}) = 1\}$, for all $q'' \in Q$.
- In other cases $\text{Check}(\alpha, q, z, \overline{S}) = 0$.
- $\text{Search}(q_1, z, q_2) = 1$ if either
  - there is $q_1 z \rightarrowtail q_2$; or
  - there is $q_1 z \rightarrowtail q_1' z' z$ and $q_2' \in Q$ for which $\text{Search}(q_1', z', q_2') = 1$ and $\text{Search}(q_2', z, q_2) = 1$.

Lemma 5. *We have* $\text{Search}(q_1, z, q_2) = 1$ *iff there is a path from the configuration* $q_1 z$ *to the configuration* q_2*. The procedure can be implemented on a Turing machine working in* $\mathcal{O}(|Q|^2|\Gamma|)$ *time and space.*

Proof
The proof of the correctness of the procedure is easy. The procedure can be implemented using dynamic programming. The implementation can construct a table of all good values (q_1, z, q_2). □

Lemma 6. *Procedure* $\text{Check}(\alpha, q, z, \overline{S})$ *can be implemented on a Turing machine working in* $Sp(|\alpha|) = \mathcal{O}((|\alpha|\log(|Q|)|Q||\Gamma|)^2)$ *space.*

Proof
The proof is by induction on the size of α. All the cases except for $\alpha = \exists F\beta$ are straightforward.

For $\alpha = \exists F\beta$ consider the graph of exponential size which nodes are of the form $\text{Check}(\exists F\beta, q, z, \overline{S})$ for arbitrary $q, z, \overline{S}$. The edges are given by the rules:

- $\text{Check}(\exists F\beta, q_1, z, \overline{S}) \rightarrow \text{Check}(\exists F\beta, q_2, z, \overline{S})$ whenever $q_1 z \rightarrowtail q_1' z' z$ and $\text{Search}(q_1', z', q_2) = 1$;
- $\text{Check}(\exists F\beta, q_1, z_1, \overline{S}_1) \rightarrow \text{Check}(\exists F\beta, q_2, z_2, \overline{S}_2)$ if $q_1 z_1 \rightarrowtail q_2 z_2 z_1$ and $\overline{S}_2$ is defined by $\overline{S}_2(q'') = \{\beta \in \text{cl}(\alpha) : \text{Check}(\beta, q'', z_1, \overline{S}_1) = 1\} \cup \{\neg\exists F\alpha : \text{Check}(\alpha, q'', z_1, \overline{S}_1) = 0\} \cup \{\exists F\alpha : \text{Check}(\alpha, q'', z_1, \overline{S}_1) = 1\}$

Observe that by induction assumption we can calculate whether there is an edge between two nodes using space $Sp(|\beta|)$. A node $\text{Check}(\exists F\beta, q, z, \overline{S})$ is *successful* if either $\text{Check}(\beta, q, z, \overline{S}) = 1$ or there is $qz \rightarrowtail q'$ with $\exists F\beta \in \overline{S}(q')$.

It is easy to see that $\text{Check}(\exists F\beta, q, z, \overline{S}) = 1$ iff in the graph described above there is a path from the node $\text{Check}(\exists F\beta, q, z, \overline{S})$ to a successful node.

We need $\mathcal{O}(\log(|Q|)|\Gamma||Q||\beta|)$ space to store a node of the graph. So we need $\mathcal{O}((\log(|Q|)|\Gamma||Q||\beta|)^2)$ space to implement Savitch algorithm performing deterministic reachability test in this graph. We also need $S(|\beta|)$ space for an oracle to calculate edges and $\mathcal{O}(|Q|^2|\Gamma|)$ space for Search procedure. All this fits into $Sp(|\exists F\beta|)$ space. $\square$

Remark: It does not seem that this lemma follows from the fact that alternating machines with bounded alternation can be simulated by deterministic ones with small space overhead (c.f. the theorem attributed in [3] to a personal communication from A. Borodin).

Lemma 7. *A tuple* $(\alpha, q, z, \overline{S})$ *is good iff* $\mathrm{Check}(\alpha, q, z, \overline{S}) = 1$

Proof

The proof is by induction on the size of α. The case when α is a propositional letter is obvious. The case when $\alpha = \neg\beta$ follows from Lemma 1. The case for conjunction is easy using Lemma 2. We omit the case for $\alpha = \exists\circ\beta$ because the arguments is simpler than in the case of F operator.

Case $\alpha = \exists F\beta$**.** Suppose that $(\alpha, q, z, \overline{S})$ is good. This means that there is an assumption function S such that $S|_\alpha = \overline{S}$ and $M, qz \vDash_S \alpha$. By the definition of the semantic, there is a vertex v reachable from qz such that $M, v \vDash_S \beta$ or $v = q'$ and $\exists F\beta \in S(q')$. Suppose that v is such a vertex at the smallest distance from qz. We show that $\mathrm{Check}(\alpha, q, z, \overline{S}) = 1$ by induction on the distance to v.

If $v = qz$ then, as β is a subformula of α, we have by the main induction hypothesis that $\mathrm{Check}(\beta, q, z, \overline{S}) = 1$. So $\mathrm{Check}(\alpha, q, z, \overline{S}) = 1$. If $qz \rightarrowtail q'$ and $\exists F\beta \in S(q')$ then we also get $\mathrm{Check}(\alpha, q, z, \overline{S}) = 1$. Otherwise we have $qz \rightarrowtail q'z'z$ and $q'z'z$ is the first vertex on the shortest path to v.

Suppose that on the path to v there is a configuration of the form $q''z$ for some q''. Assume moreover that it is the first configuration of this form on the path. We have that $\mathrm{Search}(q', z', q'') = 1$ and $M, q''z \vDash \exists F\beta$. As the distance to v from $q''z$ is smaller than from qz, we get $\mathrm{Check}(\exists F\beta, q'', z, \overline{S}) = 1$ by the induction hypothesis. Hence $\mathrm{Check}(\alpha, q, z, \overline{S}) = 1$.

Otherwise, i.e., when there is no configuration of the form $q''z$ on the path to v, we know that $v = q''wz'z$ for some $q'' \in Q$ and $w \in \Gamma^*$. Moreover we know that $q''wz'$ is reachable from $q'z'$. By Composition Lemma we have that $M, q''wz' \vDash_{S\uparrow_z} \beta$. Let $\overline{S}_1$ be a function defined by $\overline{S}_1(q_1) = (S\uparrow_z)|_\beta(q_1) \cup \{\neg\exists F\beta : \beta \notin S\uparrow_z (q_1)\} \cup \{\exists F\beta : \beta \in S\uparrow_z (q_1)\}$. It can be checked that $\overline{S}_1$ can be extended to an assumption function S_1. By Lemma 2 we have $M, q''wz' \vDash_{S_1} \beta$. Hence $M, q'z' \vDash_{S_1} \exists F\beta$. We have $\mathrm{Check}(\exists F\beta, q', z', \overline{S}_1) = 1$ from induction hypothesis. By definition of S_1 and the induction hypothesis we have that $S_1(q_1) = \{\gamma \in \mathrm{cl}(\beta) : \mathrm{Check}(\gamma, q_1, z, S) = 1\} \cup \{\neg\exists F\beta : \mathrm{Check}(\beta, q_1, z, S) = 0\} \cup \{\exists F\beta : \mathrm{Check}(\beta, q_1, z, S) = 1\}$. Which gives $\mathrm{Check}(\alpha, q, z, S) = 1$.

For the final case suppose that $\alpha = \exists F\beta$ and that $\mathrm{Check}(\alpha, q, z, \overline{S}) = 1$. We want to show that $(\alpha, q, z, \overline{S})$ is good using additional induction on the length of the computation of $\mathrm{Check}(\alpha, q, z, \overline{S})$. Let S be an assumption function such that $S|_\alpha = \overline{S}$.

Skiping a couple of easy cases suppose that there is $qz \rightarrowtail q'z'z$ and that we have $\mathrm{Check}(\exists F\beta, q', z', \overline{S}') = 1$ for $\overline{S}'$ defined by $\overline{S}'(q'') = \{\gamma \in \mathrm{cl}(\beta) : \mathrm{Check}(\gamma, q'', z, S) = 1\} \cup \{\neg\exists F\beta\}$ or $\overline{S}'(q'') = \{\gamma \in \mathrm{cl}(\beta) : \mathrm{Check}(\gamma, q'', z, S) =$

$1\} \cup \{\exists F\beta\}$ depending on whether Check$(\beta, q'', z, S) = 0$ or not. By the induction hypothesis, $M, q'z' \vDash_{S'} \exists F\beta$ for an assumption function S' such that $S'|_\alpha = \overline{S}'$

Consider $S \uparrow_z$. We have that $S \uparrow_z |_\beta = S'|_\beta$ by the induction hypothesis. It is also the case that for every $q'' \in Q$, whenever $\exists F\beta \in S'(q'')$ then $\exists F\beta \in S \uparrow_z (q'')$. Hence, by Lemma 2 and the definition of our semantics, we have that $M, q'z' \vDash_{S\uparrow_z} \exists F\beta$. By Composition Lemma we have $M, q'z'z \vDash_S \exists F\beta$. Which gives $M, qz \vDash_S \exists F\beta$. So $(\alpha, q, z, \overline{S})$ is good. $\square$

5 Model checking CTL

In this section we show that the model checking problem for pushdown systems and CTL is EXPTIME hard. The problem can be solved in EXPTIME as there is a linear translation of CTL to the μ-calculus and the model checking for the later logic can be done in EXPTIME [9].

Let M be an alternating Turing machine using n tape cells on input of size n. For a given configuration c we will construct a pushdown system P_M^c, valuation ρ_M, and a CTL formula α_M such that: $M(P_M^c, \rho_M), q_0\bot \vDash \alpha_M$ iff M has an accepting computation from c. As P_M^c and α_M will be polynomial in the size of c this will show EXPTIME hardness of the model checking problem.

We will do the construction in two steps. First, we will code the acceptance problem into the reachability problem for a pushdown system extended with some test operations. Then, we will show how to simulate these tests in the model checking problem.

We assume that the nondeterminism of M is limited so that from every configuration M has at most two possible moves. A move is a pair $m = (a, d)$ where a is a letter to put and d is a direction for moving the head. We use $c \vdash_m c'$ to mean that c' is obtained from c by doing the move m. The transition function of M assigns to each pair (state,letter) a pair of moves of M. A computation of M can be represented as a tree of configurations. If the machine is in a universal state then the configuration has two sons corresponding to the two moves in the pair given by the transition function. If the machine is in an existential state then there is only one son for one of the moves from the pair.

An *extended pushdown system* is obtained by adding two kinds of test transitions. Formally each of the kinds of transitions depends on a parameter n which is a natural number. To make notation clearer we fix this number in advance. Transition $q \rightarrowtail^A q'$ checks whether the first n letters from the top of the stack form an accepting configuration of M. Transition $q \rightarrowtail^M q'$ checks, roughly, whether the first $2n$ letters from the top of the stack form two configurations such that the first is the successor of the second. A formal definition of these transitions is given below when we define a particular extended pushdown system.

Let us fix n as the size of input to our Turing machine. We define an extended pushdown system EP_M simulating computations of M on inputs of size n. The set of states of the system is $Q = \{q, q_M, q_A\}$. The stack alphabet is $\Gamma = \Gamma_M \cup Q_M \cup \text{Moves}_M \times \text{Moves}_M \cup \{E, L, R\}$; where Γ_M is the tape alphabet of M, Q_M

is the set of states of M; Moves_M is the set of moves of M; and E, L, R are new special letters which stand for arbitrary, left and right element of a pair respectively. Before defining transitions of EP_M let us formalize the definition of $\rightarrowtail^A$ and $\rightarrowtail^M$ transitions. These transitions add the following edges in the graph of configurations of the system:

- For a transition $q \rightarrowtail^A q'$ and for an arbitrary $w \in \Gamma^*$ we have the edge $qcw \to q'cw$ if c is an accepting configuration of M.
- For a transition $q \rightarrowtail^M q'$, for an arbitrary $w \in \Gamma^*$ and a letter $? \in \{E, L, R\}$ we have the edge $qc'?(m_1, m_2)cw \to q'c'?(m_1, m_2)cw$ if (m_1, m_2) is the move form a configuration c and $c \vdash_m c'$ where $m = m_1$ if $? = L$; $m = m_2$ if $? = R$; and $m \in \{m_1, m_2\}$ if $? = E$.

Finally, we present the transition rules of EP_M. Below, a' stands for any letter other than E, L or R. We use c, c' to stand for a configuration of M, i.e., a string of length $n + 1$.

$$
\begin{array}{rlrl}
q & \rightarrowtail q_A & q_A c & \rightarrowtail^A q \\
qa' & \rightarrowtail q_M c' L(m_1, m_2) a' & q_M & \rightarrowtail^M q \\
qa' & \rightarrowtail q_M c' E(m) a' & qL(m_1, m_2) & \rightarrowtail q_M c' R(m_1, m_2) \\
qR(m_1, m_2)c & \rightarrowtail q & qE(m)c & \to q
\end{array}
$$

It is easy to see that the transitions putting or taking a whole configuration from the stack can be simulated by a sequence of simple transitions working with one letter at the time. In the above, transition $q_A c \rightarrowtail^A q$ (which removes a configuration and at the same time checks whether it is accepting) is not exactly in the format we allow. Still it can be simulated by two transitions in our format. We use $G(EP_M)$ to denote the graph of configurations of EP_M, i.e., the graph which vertices are configurations and which edges correspond to one application of the transition rules.

The idea behind the construction of EP_M is described by the following lemma.

Lemma 8. *For every configuration c of M we have that: M accepts from c iff in the graph $G(EP_M)$ of configurations of EP_M configuration q is reachable from configuration qc.*

Proof
We present only a part of the argument for the left to right direction. The proof proceeds by induction on the height of the tree representing an accepting computation of M on c.

If c is an accepting configuration then we have a path $qc \to q_A c \to q$ in $G(EP_M)$.

Suppose now that the first move of M in its computation is (m_1, m_2) and it is an existential move. Then we have a path:

$$qc \to q_M c' E(m_1, m_2)c \to qc' E(m_1, m_2)c \to \cdots \to qE(m_1, m_2)c \to q$$

where the existence of a path $qc'E(m_1,m_2)c \to \cdots \to qE(m_1,m_2)c$ follows from the induction hypothesis.

Suppose now that the first move of M in an accepting computation from c is (m_1,m_2) and it is a universal move. We have a path:

$$\begin{aligned} qc \to q_M c' L(m_1,m_2)c \to qc'L(m_1,m_2)c \to \cdots \to qL(m_1,m_2)c \to \\ q_M c'' R(m_1,m_2)c \to qc''R(m_1,m_2)c \to \cdots \to qR(m_1,m_2)c \to q. \end{aligned}$$

Once again the existence of dotted out parts of the path follows from the induction hypothesis.

This completes the proof from the left to right direction. The opposite direction is analogous. $\square$

The next step in our proof is to code the above reachability problem into the model checking problem for a normal pushdown system. First, we change extended pushdown system EP_M into a normal pushdown system P_M. We add new states q_{TA}, q_{TM}, q_F and q_R^a for every letter a of the stack alphabet. The role of q_{TA} and q_{TM} is to initiate test performed originally by $\rightarrowtail^A$ and $\rightarrowtail^M$ transitions, respectively. State q_F is a terminal state signalling success. States q_R^a are used in the test. They take out all the letters from the stack and give information about what letters are taken out. In the rules below c, c' range over configurations; a, b over single letters; and a' over letters other than E, L or R.

$$\begin{array}{rlrl} qa' & \rightarrowtail q_A a' & q_A & \rightarrowtail q, q_{TA} \\ qa' & \rightarrowtail q_M c' L(m_1,m_2)a' & q_M & \rightarrowtail q, q_{TM} \\ qa' & \rightarrowtail q_M c' E(m_1,m_2)a' & qL(m_1,m_2) & \rightarrowtail q_M c' R(m_1,m_2) \\ qR(m_1,m_2)c & \rightarrowtail q & qE(m_1,m_2)c & \to q \\ q_{TA}a & \rightarrowtail q_R^a & q_{TM}a & \to q_R^a \\ q_R^a b & \rightarrowtail q_R^b & q\bot & \to q_F\bot \end{array}$$

Recall that $\bot$ is the initial stack symbol of a pushdown automaton. As before we use $G(P_M)$ to denote the graph of configurations of P_M.

To simplify matters we will use states also as names of propositions and take valuation ρ_M such that in a state q' exactly proposition q' holds, i.e., $\rho_M(q') = \{q'\}$.

First we take two EF formulas Accept and Move such that:

- $M(P_M,\rho_M), q_{TA}w \vDash$ Accept iff w starts with an accepting configuration of M.
- $M(P_M,\rho_M), q_{TM}w \vDash$ Move iff w is of the form $c'?(m_1,m_2)cw'$, (m_1,m_2) is the move of M, and $c \vdash_m c'$ where $m = m_1$ if $? = L$; $m = m_2$ if $? = R$; and $m \in \{m_1,m_2\}$ if $? = E$.

From states q_{TA} and q_{TM} the behaviour of P_M is deterministic. It only takes letters from the stack one by one. The formula Accept is $\bigvee_{i=1,\dots,n+1} \exists\circ^i q_R^F$ where q_R^F signals an accepting state of M. The formula Move is slightly more complicated as it needs to code the behaviour of M. Still its construction is standard.

The formula we are interested in is:

$$\alpha = \exists\big[(q \vee q_A \vee q_M) \wedge (q_A \Rightarrow \exists\circ(q_{TA} \wedge \text{Accept}))\wedge \\ (q_M \Rightarrow \exists\circ(q_{TM} \wedge \text{Move}))\big]\; U\, q_F$$

It says that there is a path going only through states q, q_A or q_M and ending in a state q_F. Moreover, whenever there is a state q_A on the path then there is a turn to a configuration with a state q_{TA} from which Accept formula holds. Similarly for q_M.

Lemma 9. *For every word w over the stack alphabet: q is reachable from qw in $G(EP_M)$ iff $M(P_M, \rho_M), qw\bot \vDash \alpha$.*

Proof

The proof in both directions is by induction on the length of the path. We will only present a part of the proof for the direction from left to right.

If in $G(EP_M)$ the path is $qw \to q_A w \to q$ then in $G(P_M)$ we have:

$$\begin{tikzcd} qw\bot \arrow[r] & q_a w\bot \arrow[r] \arrow[dr] & q\bot \\ & & q_{TA} w\bot \end{tikzcd}$$

The edge $q_A w \to q$ exists in $G(EP_M)$ only if w is an accepting configuration. Hence, we have that $M(P_M, \rho_M), q_{TA} w \vDash$ Accept and consequently we have the thesis of the lemma.

If the path is $qw \to q_M c'?(m_1, m_2)w \to qc'?(m_1, m_2)w \to \cdots$ then in $G(P_M)$ we have:

$$\begin{tikzcd} qw\bot \arrow[r] & q_M c'?(m_1,m_2)w\bot \arrow[r] \arrow[dr] & qc'?(m_1,m_2)w\bot \arrow[r] & \ldots \\ & & q_{TM} c'?(m_1,m_2)w\bot & \end{tikzcd}$$

The edge $q_M c'?(m_1, m_2)w \to qc'?(m_1, m_2)w$ exists in $G(EP_M)$ only when the stack content $c'?(m_1, m_2)w$ satisfies the conditions of $\rightarrowtail^M$ transition. This means that $M(P_M, \rho_M), q_{TM} c'?(m_1, m_2)w\bot \vDash$ Move. From the induction assumption we have $M(P_M, \rho_M), qc'?(m_1, m_2)w \vDash \alpha$. Hence $M(P_M, \rho_M), qw \vDash \alpha$. $\square$

Theorem 1. *The model checking problem for pushdown systems and CTL is EXPTIME-complete*

Proof

The problem can be solved in EXPTIME as there is a linear translation of CTL to the μ-calculus and the model checking for the later logic can be done in EXPTIME [9].

To show hardness part let M be an alternating Turing machine as considered in this section. For an input word v of length n we construct in polynomial time a pushdown system P_M^v, valuation ρ_M and a formula α_M such that: v is accepted by M iff $M(P_M^v, \rho_M), q_0\bot \vDash \exists\circ^{n+1}\alpha$. Let c_0^v be the initial configuration of M on v. It has the length $n+1$.

Valuation ρ_M and formula α_M are ρ and α as described before Lemma 9. The system P_M^v is such that started in $q_0\bot$ it first puts the initial configuration c_0^v on the stack and then behaves as the system P_M.

By Lemma 8 we have that M has an accepting computation from c_0^v iff there is a path from qc_0^v to q in $G(EP_M)$. By Lemma 9 this is equivalent to the fact that $M(P_M, \rho_M), qc_0^v\bot \vDash \alpha_M$. By the construction of P_M^v this the same as saying that $M(P_M^v, \rho_M), q_0\bot \vDash \exists\circ^{n+1}\alpha_M$. $\square$

References

1. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Applications to model checking. In *CONCUR'97*, volume 1243 of *LNCS*, pages 135–150, 1997.
2. D. Caucal. On infinite transition graphs a having decidable monadic theory. In *ICALP'96*, LNCS, 1996.
3. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.
4. E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logics of Programs*, volume 131 of *LNCS*, pages 52–71. Springer-Verlag, 1981.
5. E. A. Emerson. Temporal and modal logic. In J. Leeuven, editor, *Handbook of Theoretical Computer Science Vol.B*, pages 995–1072. Elsevier, 1990.
6. J. Esparza. Private communication.
7. J. Esparza, D. Hansel, and P. Rossmanith. Efficient algorithms for model checking pushdown systems. In *CAV '00*, LNCS, 2000. to appear.
8. D. Muller and P. Schupp. The theory of ends, pushdown automata and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.
9. I. Walukiewicz. Pushdown processes: Games and model checking. In *CAV'96*, volume 1102 of *LNCS*, pages 62–74, 1996. To appear in Information and Computation.