Electronic Notes in Theoretical Computer Science 18 (1998)
URL: http://www.elsevier.nl/locate/entcs/volume18.html 12 pages

Strict Lower Bounds for Model Checking BPA

Richard Mayr

Institut für Informatik
Technische Universität München
Arcisstr. 21, D-80290 München, Germany
e-mail: mayrri@informatik.tu-muenchen.de
Web: http://www7.informatik.tu-muenchen.de/~mayrri

Abstract

We show strict lower bounds for the complexity of several model checking problems for BPA (Basic Process Algebra). Model checking BPA with Hennessy-Milner Logic is *PSPACE*-hard, while model checking BPA with the (alternation-free) modal μ-calculus is *EXPTIME*-hard. Model checking BPA with LTL is also *EXPTIME*-hard. By combining these results with already established upper bounds, it follows that the model checking problems are *PSPACE*-complete and *EXPTIME*-complete, respectively.

1 Introduction

Basic Process Algebra (BPA) processes were defined by Bergstra and Klop in [1]. They are transition systems associated with Greibach normal form (GNF) context-free grammars in which only left-most derivations are permitted. BPA-processes are also called *context-free processes*. They are a subclass of pushdown processes, where the finite control of the pushdown automaton has only one state.

It has been known for some time that model checking pushdown processes with the modal μ-calculus is *EXPTIME*-complete [14,15,5]. Furthermore, the problem is even *EXPTIME*-hard for a fixed formula in the alternation-free modal μ-calculus. For the much simpler logic EF, the model checking problem for pushdown processes is *PSPACE*-complete [2]. Again the hardness result even holds for a fixed EF-formula. For CTL the complexity is only known to be between *PSPACE* and *EXPTIME*. For LTL (linear-time temporal logic) model checking pushdown processes is *EXPTIME*-complete [2], but polynomial in the size of the process for every fixed formula.

There is an important difference between BPA and pushdown processes in the complexity of model checking. Burkart and Steffen [4] showed that for every fixed formula in the alternation-free modal μ-calculus the model checking problem is polynomial in the size of the BPA-process. This result was later generalized to the full modal μ-calculus [14,15,5]. The model checking algorithms for BPA are only exponential in the size of the formula. So far, there were no hardness results for model checking BPA, not even for the full modal μ-calculus. On the other hand, model checking finite-state systems with the alternation-free modal μ-calculus is linear [11,12], and model checking finite-state systems with the full modal μ-calculus is in $\mathcal{NP} \cap$ co-$\mathcal{NP}$ [7,12,9] (and so might be polynomial as well). It has thus been conjectured that at least some model checking problems for BPA might be polynomial. Here we show that this is not the case. Even for the simple Hennessy-Milner Logic, model checking BPA is *PSPACE*-hard. For the modal μ-calculus model checking is *EXPTIME*-hard. In fact, this hardness result even holds for the alternation-free modal μ-calculus. We also show that the *EXPTIME*-hardness result for pushdown processes and LTL [2,3] carries over to BPA. Since the upper bound trivially carries over, model checking BPA with LTL is *EXPTIME*-complete.

The rest of the paper is structured as follows. In Section 2 we define BPA and the temporal logics that are used here. In Sections 3, 4 and 5 we show the hardness results for Hennessy-Milner Logic, the alternation-free modal μ-calculus and LTL. In Section 6 we present the general picture of the complexity of model checking BPA.

2 Preliminaries

The states of a BPA-process are represented by sequences of symbols from a set *Const*. The dynamics is described by finite sets Δ of rewrite rules of the form $X \stackrel{a}{\to} \alpha$, where $X \in Const$ is a single symbol, $a \in Act$ is an atomic action and $\alpha \in Const^*$ is a sequence of symbols. The rewriting formalism is prefix-rewriting, which means that the rules are only applied at the leftmost position in the term. For example, a rule $A \stackrel{a}{\to} BC$ can be applied to a term $ADEF$ and yields the transition $ADEF \stackrel{a}{\to} BCDEF$.

Now we repeat the definition of some branching-time logics that are used to describe properties of BPA-processes. The formulae of Hennessy-Milner Logic have the following syntax:

$$\Phi ::= true \;\mid\; \neg\Phi \;\mid\; \Phi_1 \wedge \Phi_2 \;\mid\; \langle a \rangle \Phi$$

The denotation $[\![\Phi]\!]$ of a formula Φ is a subset of the set of states Ω that is defined inductively as follows:

$$\begin{aligned} [\![true]\!] &:= \Omega \\ [\![\neg\Phi]\!] &:= \Omega - [\![\Phi]\!] \\ [\![\Phi_1 \wedge \Phi_2]\!] &:= [\![\Phi_1]\!] \cap [\![\Phi_2]\!] \\ [\![\langle a \rangle \Phi]\!] &:= \{s \in \Omega \;\mid\; \exists s' \in \Omega.\ s \stackrel{a}{\to} s' \;\wedge\; s' \in [\![\Phi]\!]\} \end{aligned}$$

The one-step next operator $\langle a \rangle$ is sometimes also denoted by $\bigcirc_a$. We write $\bigcirc$ if the action does not play a role.

The modal μ-calculus [8] is the extension of Hennessy-Milner Logic by variables and fixpoint operators. The semantics of formulae is defined w.r.t. a valuation $\mathcal{V} : Var \mapsto 2^{\Omega}$ that assigns every variable X in the logic a set of states which satisfy it.

$$[\![X]\!]_{\mathcal{V}} := \mathcal{V}(X)$$

The syntax and semantics of the minimal fixpoint operator is defined by

$$[\![\mu X.\Phi]\!]_{\mathcal{V}} := \bigcap \{S \subseteq \Omega \mid [\![\Phi]\!]_{\mathcal{V}[X:=S]} \subseteq S\}$$

where

$$\mathcal{V}[X := S](X') := \begin{cases} \mathcal{V}(X'), & \text{if } X \neq X' \\ S, & \text{if } X = X' \end{cases}$$

In model checking we use only *closed formulae*. These are the formulae where every variable is bound by a fixpoint operator. Also there is the restriction that every variable occurs within the scope of an even number of negations. The property that $s \in [\![\Phi]\!]$ is also denoted by $s \models \Phi$.
It is easy to see that Computation Tree Logic (CTL) [6] is a fragment of the alternation-free modal μ-calculus.

$$E[\Phi_1 \,\mathcal{U}\, \Phi_2] = \mu X.(\Phi_2 \vee (\Phi_1 \wedge \bigcirc X))$$
$$E[\Phi_1 \, w\mathcal{U}\, \Phi_2] = \mu X.(\Phi_2 \vee (\Phi_1 \wedge \bigcirc X)) \;\vee\; \nu X.(\Phi_1 \wedge (\bigcirc X \vee \neg \bigcirc \mathit{true}))$$

An interesting fragment of CTL is the logic EF, which uses the same operators as Hennessy-Milner Logic and the operator *EF* which can be expressed by

$$EF\ \Phi = E[\mathit{true}\,\mathcal{U}\,\Phi]$$

Linear-Time Logic (LTL) [10] is defined on runs of processes. It uses boolean operators, the one-step next operator and the "until"-operators.

$$[\![\Phi_1 \,\mathcal{U}\, \Phi_2]\!] := \left\{\pi \mid \exists i.\ \pi^i \in [\![\Phi_2]\!] \;\wedge\; \forall j < i.\ \pi^j \in [\![\Phi_1]\!]\right\}$$

Intuitively, this means that the run π has a suffix π^i that satisfies Φ_2 and all prefixes satisfy Φ_1.

$$[\![\Phi_1 \, w\mathcal{U}\, \Phi_2]\!] := \left\{\pi \mid (\exists i.\ \pi^i \in [\![\Phi_2]\!] \;\wedge\; \forall j < i.\ \pi^j \in [\![\Phi_1]\!]) \;\vee\; \forall i.\ \pi^i \in [\![\Phi_1]\!]\right\}$$

Here it is also possible that the run π always satisfies Φ_1, but never Φ_2.
A state s satisfies an LTL-formula Φ iff all runs that start at s satisfy Φ.

$$s \models \Phi \;:\Longleftrightarrow\; \mathit{runs}(s) \subseteq [\![\Phi]\!]$$

The model checking problem is defined as follows:

Model checking

Instance: BPA-process Δ with initial state s, formula Φ
Question: $s \models \Phi$?

Since in practice the process can often be large, while the formula is often small, one considers two questions.

(i) The complexity of the general model checking problem in the size of the process and the formula.

(ii) The complexity of the model checking problem in the size of the process for a fixed formula. However, one assumes the worst case for this fixed formula.

3 Hardness of Hennessy-Milner Logic

In this section we show that model checking BPA with Hennessy-Milner Logic is *PSPACE*-hard. We do this by reducing the problem of quantified boolean formulae (QBF) to the model checking problem.

Let $n \in \mathbb{N}$ and let $x_1, \ldots, x_n$ be boolean variables. W.r. we assume that n is even. A literal is either a variable or the negation of a variable. A clause is a disjunction of literals. The quantified boolean formula Q is given by

$$Q := \exists x_1 \forall x_2 \exists x_3 \ldots \exists x_{n-1} \forall x_n (Q_1 \wedge \ldots \wedge Q_k)$$

where the Q_i are clauses. The problem is if Q is valid. We reduce this problem to the model checking problem.

The intuition is that first we nondeterministically choose values for the variables and then check if these choices satisfy Q. The existential or universal nature of these choices is handled by the Hennessy-Milner Logic formula. Now we define a BPA whose initial state is the symbol Z_1. The rules are as follows:

$$\begin{array}{lcl}
Z_1 & \xrightarrow{c_1} & Z_2.X_1 \\
Z_1 & \xrightarrow{c_1} & Z_2.\bar{X}_1 \\
Z_2 & \xrightarrow{c_2} & Z_3.X_2 \\
Z_2 & \xrightarrow{c_2} & Z_3.\bar{X}_2 \\
 & \vdots & \\
Z_{n-1} & \xrightarrow{c_{n-1}} & Z_n.X_{n-1} \\
Z_{n-1} & \xrightarrow{c_{n-1}} & Z_n.\bar{X}_{n-1} \\
Z_n & \xrightarrow{c_n} & X_n \\
Z_n & \xrightarrow{c_n} & \bar{X}_n
\end{array}$$

Furthermore we add rules $X_i \xrightarrow{q_j} X_i$, if the literal x_i is in the clause Q_j for any $1 \leq i \leq n$ and $1 \leq j \leq k$.

In the same way we add rules $\bar{X}_i \xrightarrow{q_j} \bar{X}_i$, if the literal $\neg x_i$ is in the clause Q_j for any $1 \leq i \leq n$ and $1 \leq j \leq k$.

Finally, we add rules $X_i \xrightarrow{d} \epsilon$ and $\bar{X}_i \xrightarrow{d} \epsilon$ for every $1 \leq i \leq n$.

Now we construct the Hennessy-Milner Logic formula. We use the abbreviation $\langle d \rangle^i$ for $\langle d \rangle \ldots \langle d \rangle$ (i times). The formula is

$$\Phi := \langle c_1 \rangle [c_2] \langle c_3 \rangle [c_4] \ldots \langle c_{n-1} \rangle [c_n] (\Phi_1 \wedge \ldots \wedge \Phi_k)$$

where

$$\Phi_j := \bigvee_{0 \le i \le n-1} \langle d \rangle^i \langle q_j \rangle \mathit{true}$$

It follows that Q is valid iff $Z_1 \models \Phi$. Note that the size of Φ is $\mathcal{O}(n^2 k)$. Thus Φ is not a fixed formula, but grows polynomially with the size of Q. Also the BPA process has size $\mathcal{O}(nk)$. Since QBF is *PSPACE*-complete we get the following lemma.

Lemma 3.1 *Model checking BPA with Hennessy-Milner Logic is PSPACE-hard.*

It was shown in [2] that even for the more general logic EF and pushdown processes, model checking can be done in polynomial space. Thus we get the following theorem.

Theorem 3.2 *Model checking BPA with EF or Hennessy-Milner Logic is PSPACE-complete.*

4 Hardness of the Modal μ-Calculus

Model checking pushdown processes with the modal μ-calculus is *EXPTIME*-complete [14,15,5]. There is even a fixed formula in the alternation-free modal μ-calculus, for which model checking pushdown processes is *EXPTIME*-hard. This hardness result does not carry over to BPA. In fact, for every fixed modal μ-calculus formula, model checking is polynomial in the size of the BPA-process.

Here we show that model checking BPA with general (non-fixed) formulae in the alternation-free modal μ-calculus is *EXPTIME*-hard. This is shown by a reduction from the acceptance problem for linearly space bounded alternating Turing-machines.

An alternating Turing machine (ATM) is described by a tuple $(Q, \Sigma, \delta, q_0, l)$, where Q are the states of the finite control, Σ the tape symbols, δ the transition relation, q_0 the initial state and l is a function that labels states as existential, universal, accepting or rejecting. The computation of an ATM is defined just like the computation of a normal Turing machine, but the acceptance condition is more complex. Since the machine is nondeterministic, the computation can be represented as a computation tree in which the branches represent different possible computations. The states of the finite control of the ATM are assigned labels by the function l as existential, universal, accepting or rejecting. Now the states in the computation tree are labeled as accepting or rejecting by the following rules:

(i) A leaf of the computation tree is labeled accepting (rejecting) if the finite control of the ATM in this state is accepting (rejecting).

(ii) An internal node where the finite control is labeled universal (existential) is accepting if and only if all (at least one) of its successor nodes are (is) accepting. Otherwise it is rejecting.

(iii) A node is labeled undefined if the label cannot be determined by the other rules. (This only happens if there are infinite branches.)

Without loss of generality let $|\delta(q,a)| = 2$ for every universal state q and symbol a. We choose an arbitrary order on the two elements of $\delta(q,a)$ and call them the first and second successor configuration of (q,a). An ATM M is called linearly bounded if there is a constant k, such that for every word w in the language of M, M has an accepting computation that uses at most $k \cdot |w|$ space. Let $n := k \cdot |w|$. We only consider linearly bounded ATMs and thus avoid the problem of infinite branches and undefined labels. The acceptance problem for linearly bounded alternating Turing-machines is *EXPTIME*-complete [13].

The idea is to guess a sequence of configurations of the ATM and to store this sequence in a BPA-process term. A formula in the alternation-free modal μ-calculus is used to check if this sequence represents an accepting computation of the ATM.

Let $M = (Q, \Sigma, \delta, q_0, l)$ be the ATM, w the input word and $n := k \cdot |w|$ the length of the tape. Let M's head be over the first cell of the tape. We construct in polynomial time a BPA Δ with initial state $\#q_0w\#$ and an alternation-free modal μ-calculus formula Φ s.t. M accepts w iff $\#q_0w\# \models \Phi$, w.r.t. Δ. The rules Δ for the BPA are as follows:

$$
\begin{array}{rll}
\# \xrightarrow{put} T_1.a & \text{for every } a \in \Sigma \\
T_i \xrightarrow{put} T_{i+1}.a & \text{for every } a \in \Sigma,\ 1 \le i \le n-1 \\
T_i \xrightarrow{put} T'_i.q & \text{for every } q \in Q,\ 1 \le i \le n \\
T'_i \xrightarrow{put} T'_{i+1}.a & \text{for every } a \in \Sigma,\ 1 \le i \le n-1 \\
T'_n \xrightarrow{put} \# & \\
\# \xrightarrow{\#} \epsilon & \\
a \xrightarrow{a} \epsilon & \text{for every } a \in \Sigma \\
q \xrightarrow{q} \epsilon & \text{for every } q \in Q \\
a \xrightarrow{drop} \epsilon & \text{for every } a \in \Sigma \\
q \xrightarrow{drop} \epsilon & \text{for every } q \in Q \\
\# \xrightarrow{drop} \epsilon &
\end{array}
$$

The size of this set of rules is $\mathcal{O}(n^2)$. Now in every state where the symbol $\#$ is at the top, the state has the form $\#c_1\#c_2\#\ldots\#c_m\#$ where the c_i are configurations of the ATM. Every c_i has the form $u.q.u'$ where $q \in Q$ is the state, u' is the contents of the tape under the head and to the right of it and

u is the contents of the tape to the left of the head. (u can be empty, but u' cannot.) Also we have that $\mathit{length}(u) + \mathit{length}(u') = n$.
Now we define some auxiliary formulae: Let $Q_{acc} \subseteq Q$ be the set of accepting states, $Q_{univ} \subseteq Q$ be the set of universal states and $Q_{ex} \subseteq Q$ be the set of existential states. The formula *accept* means that the top symbol is $\#$ and the state in the uppermost configuration c_1 is accepting.

$$\mathit{accept} := \langle \# \rangle \bigvee_{0 \le i \le n-1} \langle \mathit{drop} \rangle^i \bigvee_{q \in Q_{acc}} \langle q \rangle \mathit{true}$$

The formulae *univ* and *ex* are defined in the same way with Q_{univ} or Q_{ex}, respectively. These formulae have size $\mathcal{O}(n^2)$.
The formula *succ* encodes the property that the current state of the BPA-process has the form $\#c_1\#c_2\#\ldots\#c_m\#$ for some $m \ge 2$ and that the configuration c_1 is a successor configuration of c_2. The actual construction is cumbersome and depends on the relation δ of the ATM M. However, it is easy to see that it can be done with the help of the following formulae: Let $x \in \Sigma \cup Q$ and $i \in \{0, \ldots, n\}$. $\Psi^1_{x,i}$ means that the i-th symbol in c_1 is x.

$$\Psi^1_{x,i} := \langle \# \rangle \langle \mathit{drop} \rangle^i \langle x \rangle \mathit{true}$$

$\Psi^2_{x,i}$ means that the i-th symbol in c_2 is x.

$$\Psi^2_{x,i} := \langle \# \rangle \langle \mathit{drop} \rangle^{n+2+i} \langle x \rangle \mathit{true}$$

(Note that both c_1 and c_2 have length $n+1$, because the state q counts extra.)
In the same way formulae $\mathit{succ_1}$ ($\mathit{succ_2}$) can be constructed that mean that the configuration c_1 is the first successor (second successor) of c_2 if c_2 is a universal configuration. The formulae $\Psi^1_{x,i}$, $\Psi^2_{x,i}$ have size $\mathcal{O}(n)$ and the formulae *succ*, $\mathit{succ_1}$ and $\mathit{succ_2}$ have size $\mathcal{O}(n^2)$.
Now we are ready to construct the formula Φ.

$$\begin{aligned}
\mu X.\ & \mathit{accept} \\
& \vee \\
& \mathit{univ} \quad \wedge \langle \mathit{put} \rangle^{n+2} (\mathit{succ_1} \wedge X) \\
& \qquad\quad\ \wedge \langle \mathit{put} \rangle^{n+2} (\mathit{succ_2} \wedge X) \\
& \vee \\
& \mathit{ex} \qquad \wedge \langle \mathit{put} \rangle^{n+2} (\mathit{succ} \wedge X)
\end{aligned}$$

Note that Φ is a very simple formula, since it uses only one fixpoint operator. Thus it is a formula in the alternation-free modal μ-calculus. However, it is not a CTL-formula. The size of Φ is $\mathcal{O}(n^2)$. We have that $\#q_0w\# \models \Phi$ iff M accepts w.

Lemma 4.1 *Model checking BPA with the alternation-free modal μ-calculus is EXPTIME-hard.*

Containment in *EXPTIME* has been shown in [14,15,5]. Thus we get the following theorem.

Theorem 4.2 *Model checking BPA with the full modal μ-calculus and the alternation-free modal μ-calculus is EXPTIME-complete.*

5 Hardness of LTL

Now we consider model checking BPA-processes with linear-time logic. Model checking pushdown processes with LTL is *EXPTIME*-complete [2], but only polynomial in the size of the process for any fixed formula. The question was if *EXPTIME*-hardness also holds for BPA-processes.

We show that model checking with LTL is *EXPTIME*-hard even for BPA. We generalize the proof of *EXPTIME*-hardness for pushdown processes and LTL of [2]. (This proof for pushdown processes is in the appendix of [2] and can be found in [3].)

The proof of *EXPTIME*-hardness is done by a reduction of the acceptance problem for linearly bounded alternating Turing machines [13] (see also Section 4).

Lemma 5.1 *Model checking BPA with LTL is EXPTIME-hard.*

Proof. We reduce the acceptance problem of a linearly bounded ATM to the model checking problem. Let $M = (Q, \Sigma, \delta, q_0, l)$ be the ATM, w the input word and $n := k \cdot |w|$ the length of the tape. Let M's head be over the first cell of the tape. We construct in polynomial time a BPA Δ with initial state I and a LTL-formula Φ s.t. M accepts w iff $I \models \Phi$, w.r.t. Δ.

First we describe the intuition for the construction, then we formally define the BPA Δ and finally we construct the LTL-formula that characterizes exactly the runs of the system that are faithful simulations of M.

The intuition is as follows. A configuration of M is described by words of $\Sigma^* Q \Sigma^*$ of length n. In a configuration $\alpha q \beta$, α is the content of the tape to the left of the head, q is the state of the finite control and β is the content of the tape under the head and to the right of it. The configuration is accepting if q is an accepting state. The computation of the BPA is now defined as an attempt to guess a computation of M. This is a finite or infinite tree of configurations in which every node has at most two successors. The BPA attempts (by guessing nondeterministically) to simulate a traversal of this tree in infix order. In the sequence that describes its state it stores the sequence $\#c_1\#c_2\#\ldots\#c_k$ of configurations $c_1, \ldots, c_k$ describing the path in this tree from the root to the actual configuration. Of course, most of these guesses are wrong or not even meaningful. Later we'll use the LTL formula to enforce a faithful simulation of M.

Now we define the BPA Δ. As the tree is traversed in infix order the BPA always does one of two things:

(i) It outputs $\#fc\#$, where c is a configuration of M and f (meaning 'forward') is a special action and writes $\#c\#$ onto the stack. This symbolizes that we enter a node from the parent node in the computation tree of M.

(ii) It outputs $\#bc^r\#$, where c^r is the reverse of the configuration c and b (meaning 'backward') is a special action, and pops $\#c^r\#$ from the stack.

At the beginning of the execution the BPA outputs $\#fq_0w\#$ and writes $\#q_0w\#$ onto the stack. A computation is successful if it leads to a state where the stack is empty.
The rules describing this are as follows: (We use a shorthand notation where the rules can have strings as labels instead of single actions.)

$$
\begin{array}{lll}
I \xrightarrow{\#f} I'.\# & \\
I' \xrightarrow{q_0 w} T'_n.w^r.q_0 & \\
T'_n \xrightarrow{\#f} T_0.\# & \\
T_i \xrightarrow{a} T_{i+1}.a & \text{for } a \in \Sigma,\ 0 \leq i \leq n-2 \\
T_i \xrightarrow{q} T'_i.q & \text{for } q \in Q,\ 1 \leq i \leq n \\
T'_i \xrightarrow{a} T'_{i+1}.a & \text{for } a \in \Sigma,\ 1 \leq i \leq n-1 \\
T'_n \xrightarrow{\#b} \epsilon & \\
a \xrightarrow{a} \epsilon & \text{for } a \in \Sigma \\
q \xrightarrow{q} \epsilon & \text{for } q \in Q \\
\# \xrightarrow{\#b} \epsilon & \\
\# \xrightarrow{\#f} T_0.\# &
\end{array}
$$

It is easy to see that M accepts w iff the actions performed by the BPA-process form a string $\#d_0\#d_1\#\ldots\#d_m\#$ such that this string is a faithful simulation of the computation of M on w and is in a state with empty stack (a deadlock) afterwards.
Now we define when a simulation is faithful and construct the LTL formula that characterizes exactly the faithful simulations. The simulation is faithful iff the following properties hold for every $0 \leq i \leq m-1$

(i) If $d_i = fc$ and c is an existential configuration, then $d_{i+1} = fc'$ and c' is a successor configuration of c.

(ii) If $d_i = fc$ and c is a universal configuration, then $d_{i+1} = fc'$ and c' is the first successor configuration of c.

(iii) If $d_i = bc^r$ and $d_{i+1} = fc'$, then c is a universal configuration, and c' is the second successor configuration of c.

(iv) The configuration in d_i is not a rejecting configuration.

(v) If the configuration in d_i is an accepting configuration, then $d_{i+1} = bc^r$ for some c.

These properties can be encoded in LTL. For each symbol $a \in \Sigma \cup Q \cup \{f, b, \#\}$ we define a proposition $p_a := \bigcirc_a true$. We use the abbreviation $\bigcirc^i \Phi$ for $\bigcirc \ldots \bigcirc \Phi$ (i-times).
We define $G\,\Phi := \Phi\, w\mathcal{U}\, false$ and $F\,\Phi := true\, \mathcal{U}\, \Phi$.

A run of the BPA is a faithful simulation of M if it satisfies the following LTL formula:

$$\mathit{faithful} := G((p_{\#} \wedge \bigcirc^{n+3} p_{\#}) \Rightarrow (\Psi_1 \wedge \Psi_2 \wedge \Psi_3 \wedge \Psi_4 \wedge \Psi_5))$$

where $p_{\#} \wedge \bigcirc^{n+3} p_{\#}$ expresses that the current state is a #-position different from the last, and $\Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5$ encode parts (1)–(5) of the properties above. It is $n+3$, because n is the length of the configuration and the symbol #, the symbol f (or b) and the state q count extra.
M accepts w if there is a run of the BPA that is faithful (satisfies the LTL formula *faithful*) and leads to a state of deadlock. Let

$$\Phi := \mathit{faithful} \wedge F(\neg \bigcirc \mathit{true})$$

Therefore M accepts w iff there is a run of the BPA that satisfies Φ. This is true iff not all runs satisfy $\neg\Phi$. Thus M accepts w iff

$$\neg(I \models \neg\Phi)$$

We only show how to construct the formula Ψ_1, since Ψ_2 – Ψ_5 are similar.

- "$d_i = fc$" is encoded as $\bigcirc p_f$.
- "c is an existential configuration" is encoded as

$$\bigvee_{j=1}^{n+2} \bigcirc^j (\bigvee_{q \in Q_{ex}} p_q)$$

  where $Q_{ex} \subseteq Q$ is the set of existential states.
- "c is not rejecting" is encoded as

$$\neg \bigvee_{j=1}^{n+2} \bigcirc^j (\bigvee_{q \in Q_{rej}} p_q)$$

  where $Q_{rej} \subseteq Q$ is the set of rejecting states.
- "$d_{i+1} = fc'$" is encoded as $\bigcirc^{n+3} p_f$.
- "c' is a successor configuration of c" is encoded as a disjunction of formulae, one for each possible successor configuration of c. These formulae are in turn a conjunction of formulae of the form

$$(\bigcirc^{j-1} p_{a_1} \wedge \bigcirc^j p_q \wedge \bigcirc^{j+1} p_{a_2}) \Rightarrow \bigcirc^{j+(n+3)+k} p_x$$

  where $k \in \{-1, 0, 1\}$, $q \in Q$, $a \in \Sigma$ and $x \in Q \cup \Sigma$ and $3 \leq j \leq n+1$. They are determined only by the transition relation of the ATM.

  For example let $a_1 = 0, a_2 = 1$, $q \in Q$ and $(q', 0, L) \in \delta(q, 1)$. Then, for every j, there would be three such formulae with different right hand sides. These right hand sides are

$$\bigcirc^{j+(n+3)-1} p_{q'} \quad \bigcirc^{j+(n+3)}\, p_1 \quad \bigcirc^{j+(n+3)+1}\, p_0$$

□

By combining this hardness result with the already established *EXPTIME*-upper bound [2], we get the following theorem.

Theorem 5.2 *Model checking BPA with LTL is EXPTIME-complete.*

6 Conclusion

The results on the complexity of model checking BPA can be summarized as follows:

BPA	**general**	**fixed formula**
Hennessy-Milner Logic	*PSPACE*-complete	$\in \mathcal{P}$
EF	*PSPACE*-complete	$\in \mathcal{P}$
alt.-free modal μ-calc.	*EXPTIME*-complete	$\in \mathcal{P}$
modal μ-calc.	*EXPTIME*-complete	$\in \mathcal{P}$
LTL	*EXPTIME*-complete	$\in \mathcal{P}$

In comparison, model checking pushdown processes is often harder in the case of fixed formulae.

Pushdown	**general**	**fixed formula**
Hennessy-Milner Logic	*PSPACE*-complete	$\in \mathcal{P}$
EF	*PSPACE*-complete	*PSPACE*-complete
alt.-free modal μ-calc.	*EXPTIME*-complete	*EXPTIME*-complete
modal μ-calc.	*EXPTIME*-complete	*EXPTIME*-complete
LTL	*EXPTIME*-complete	$\in \mathcal{P}$

These results show the complexity of model checking BPA with the most common temporal logics, except for CTL. The *EXPTIME*-hardness proof in Section 4 does not carry over to CTL. For both BPA and pushdown processes, model checking with CTL is only known to be between *PSPACE* and *EXPTIME*.

References

[1] J. A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science (TCS)*, 37:77–121, 1985.

[2] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model checking. In *International Conference on Concurrency Theory (CONCUR'97)*, volume 1243 of *LNCS*. Springer Verlag, 1997.

[3] A. Bouajjani, J. Esparza, and O. Maler. Reachability Analysis of Pushdown Automata: Application to Model Checking. Technical report, VERIMAG, 1997. ftp://ftp.imag.fr/imag/SPECTRE/ODED/pda.ps.gz.

[4] O. Burkart and B. Steffen. Model checking for context-free processes. In *Proc. of CONCUR'92*, volume 630 of *LNCS*, pages 123–137, 1992.

[5] O. Burkart and B. Steffen. Model checking the full modal mu-calculus for infinite sequential processes. In *Proceedings of ICALP'97*, volume 1256 of *LNCS*. Springer Verlag, 1997.

[6] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. volume 131 of *LNCS*, pages 52–71, 1981.

[7] E. Emerson, C.S. Jutla, and A. Sistla. On model checking for fragments of μ-calculus. In *Proceedings of CAV'93*, volume 697 of *LNCS*, pages 385–396. Springer Verlag, 1993.

[8] D. Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, 1983.

[9] A. Mader. *Verification of Modal Properties Using Boolean Equation Systems.* PhD thesis, TU-München, 1997.

[10] A. Pnueli. The temporal logic of programs. In *FOCS'77*. IEEE, 1977.

[11] B. Steffen and R. Cleaveland. A linear–time model–checking algorithm for the alternation–free modal mu–calculus. *International Journal on Formal Methods in System Design*, 1, 1993.

[12] C. Stirling and D. Walker. Local model checking in the modal μ-calculus. *Theoretical Computer Science*, 89:161–177, 1991.

[13] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science: Volume A, Algorithms and Complexity*. Elsevier, 1990.

[14] I. Walukiewicz. Pushdown processes: games and model checking. In *International Conference on Computer Aided Verification (CAV'96)*, volume 1102 of *LNCS*. Springer Verlag, 1996.

[15] I. Walukiewicz. Pushdown processes: games and model checking. Technical Report RS-96-54, BRICS, Aarhus, Denmark, 1996. Longer version of a CAV'96 paper.