Theoretical Computer Science 549 (2014) 127–145

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Efficient CTL model-checking for pushdown systems

Fu Song [a,*], Tayssir Touili [b]

[a] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, P.R. China
[b] LIAFA, CNRS and Univ. Paris Diderot, France

ARTICLE INFO

Article history:
Received 10 September 2012
Received in revised form 25 April 2014
Accepted 1 July 2014
Available online 9 July 2014
Communicated by P. Aziz Abdulla

Keywords:
Pushdown system
Model-checking
CTL
CTL with regular valuations

ABSTRACT

Pushdown systems (PDS) are well adapted to model sequential programs with (possibly recursive) procedure calls. Therefore, it is important to have efficient model checking algorithms for PDSs. We consider in this paper CTL model checking for PDSs. We consider the "standard" CTL model checking problem where whether a configuration of a PDS satisfies an atomic proposition or not depends only on the control state of the configuration. We consider also CTL model checking with regular valuations, where the set of configurations in which an atomic proposition holds is a regular language. We reduce these problems to the emptiness problem in Alternating Büchi Pushdown Systems, and we give an algorithm to solve this emptiness problem. Our algorithms are more efficient than the other existing algorithms for CTL model checking for PDSs in the literature. We implemented our techniques in a tool, and we applied it to different case studies. Our results are encouraging. In particular, we were able to confirm the existence of known bugs in Linux source code.

1. Introduction

Pushdown Systems (PDS for short) are an adequate formalism to model sequential, possibly recursive, programs [1,2]. It is then important to have verification algorithms for pushdown systems. This problem has been intensively studied by the verification community. Several model-checking algorithms have been proposed for both linear-time logics [3,2,4–6], and branching-time logics [3,7–11,5,6]. While model-checking for PDSs against pushdown specifications is undecidable [12].

In this paper, we first study the model-checking problem for PDSs against the "standard" branching-time temporal logic CTL [13]. In this setting, whether a configuration satisfies an atomic proposition or not depends only on the control state of the configuration, not on its stack content. This problem is known to be EXPTIME-complete [14] and the set of PDS configurations satisfying a CTL formula is known to be regular [3]. CTL corresponds to a fragment of μ-calculus and of CTL*. Existing algorithms for model-checking these logics for PDSs could then be applied for CTL model-checking. However, these algorithms either allow only to decide whether a given configuration satisfies the formula i.e., they cannot symbolically compute the regular set of PDS configurations where the formula holds [15,8–10], or have a high complexity [11,7,3,16,17,5,6,18,19].

In this work, we propose a new efficient algorithm for CTL model-checking for PDSs. Our algorithm allows to symbolically compute the regular set of PDS configurations that satisfy a given CTL formula. Our procedure is more efficient than the existing model-checking algorithms for μ-calculus and CTL* that are able to symbolically compute the regular set

* Corresponding author.
E-mail addresses: fsong@sei.ecnu.edu.cn (F. Song), touili@liafa.univ-paris-diderot.fr (T. Touili).

http://dx.doi.org/10.1016/j.tcs.2014.07.001
0304-3975/© 2014 Elsevier B.V. All rights reserved.

of configurations where a given property holds [11,7,3,16,17,5,6,18,19]. Our technique reduces CTL model-checking to the problem of computing the set of configurations from which an Alternating Büchi Pushdown System (ABPDS for short) has an accepting run. We show that this set can be effectively represented using an alternating finite word automaton.

Then, we consider CTL model checking with regular valuations. In this setting, the set of configurations where an atomic proposition holds is given by a finite state automaton. Indeed, since a configuration of a PDS has a control state and a stack content, it is natural that the validity of an atomic proposition in a configuration depends on both the control state *and the stack*. For example, in one of the case studies we considered, we needed to check that whenever a function *call_hpsb_send_phy_config* is invoked, there is a path where *call_hpsb_send_packet* is called before *call_hpsb_send_phy_config* returns. We need propositions about the stack to express this property. "Standard" CTL is not sufficient. We provide an efficient algorithm that solves CTL model checking with regular valuations for PDSs. Our procedure reduces the model-checking problem to the problem of computing the set of configurations from which an ABPDS has an accepting run.

We implemented our techniques in a tool for CTL model-checking for pushdown systems. Our tool deals with both "standard" model-checking, and model-checking with regular valuations. As far as we know, this is the *first* tool for CTL model-checking for PDSs. Indeed, existing model-checking tools for PDSs like Moped [20] consider only reachability and LTL model-checking, they don't consider CTL. The only other tool that can check branching time properties for PDSs is PDSolver [18]. We run several experiments on our tool. We obtained encouraging results. In particular, we were able to confirm the existence of known bugs in source files of the Linux system, in a watchdog driver of Linux, and in an IEEE 1394 driver of Linux. We needed regular valuations to express the properties of some of these examples. Moreover, we showed in [21] that our tool is much more efficient than PDSolver for the benchmark considered in [21].

Outline. The rest of the paper is structured as follows. Section 2 gives the basic definitions used in the paper. In Section 3, we present an algorithm for computing an alternating finite word automaton recognizing all the configurations from which an ABPDS has an accepting run. Sections 4 and 5 describe the reductions from "standard" CTL model-checking for PDSs and CTL model-checking for PDSs with regular valuations, to the emptiness problem in ABPDS. The experiments are provided in Section 6. Section 7 describes the related work.

2. Preliminaries

2.1. The temporal logic CTL

We consider the standard branching-time temporal logic CTL [13]. For technical reasons, we use the *release* operator R as a dual of the until operator U for which the stop condition is not required to occur; and we suppose w.l.o.g. that formulas are given in positive normal form, i.e., negations are applied only to atomic propositions. Indeed, each CTL formula can be written in positive normal form by pushing the negations inside.

Definition 1. Let $AP = \{a, b, c, \ldots\}$ be a finite set of atomic propositions. The set of CTL formulas is given by (where $a \in AP$) [13]:

$$\begin{aligned}\varphi ::= {} & a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid AX\varphi \mid EX\varphi \mid \\ & A[\varphi U \varphi] \mid E[\varphi U \varphi] \mid A[\varphi R \varphi] \mid E[\varphi R \varphi].\end{aligned}$$

The closure $cl(\varphi)$ of a CTL formula φ is the set of all the subformulas of φ, including φ. Let $AP^+(\varphi) = \{a \in AP \mid a \in cl(\varphi)\}$ and $AP^-(\varphi) = \{a \in AP \mid \neg a \in cl(\varphi)\}$. The size $|\varphi|$ of φ is the number of elements in $cl(\varphi)$. Let $T = (S, \longrightarrow, c_0)$ be a transition system where S is a set of states, $\longrightarrow \subseteq S \times S$ is a set of transitions, and c_0 is the initial state. Let $s, s' \in S$. s' is a successor of s iff $s \longrightarrow s'$. A path is an infinite sequence of states $s_0, s_1, \ldots$ such that for every $i \geq 0$, $s_i \longrightarrow s_{i+1}$. Let $\lambda : AP \to 2^S$ be a labeling function that assigns to each atomic proposition a set of states in S. The validity of a formula φ in a state s w.r.t. the labeling function λ, denoted $s \models_\lambda \varphi$, is defined inductively in Fig. 1. $T \models_\lambda \varphi$ iff $c_0 \models_\lambda \varphi$. Note that a path π satisfies $\psi_1 R \psi_2$ iff either ψ_2 holds everywhere in π, or the first occurrence in the path where ψ_2 does not hold must be preceded by a position where ψ_1 holds.

2.2. Pushdown systems

Definition 2. A Pushdown System *(PDS for short)* $\mathcal{P}$ is a tuple $(P, \Gamma, \Delta, \sharp)$, where P is a finite set of control locations, Γ is a finite stack alphabet, $\Delta \subseteq (P \times \Gamma) \times (P \times \Gamma^*)$ is a finite set of transition rules and $\sharp \in \Gamma$ is a bottom stack symbol.

A configuration of $\mathcal{P}$ is an element $\langle p, \omega \rangle$ of $P \times \Gamma^*$. We write $\langle p, \gamma \rangle \hookrightarrow \langle q, \omega \rangle$ instead of $((p, \gamma), (q, \omega)) \in \Delta$. For technical reasons, we consider the bottom stack symbol $\sharp$, and we assume w.l.o.g. that it is never popped from the stack, i.e., there is no transition rule of the form $\langle p, \sharp \rangle \hookrightarrow \langle q, \omega \rangle \in \Delta$. The successor relation $\rightsquigarrow_{\mathcal{P}} \subseteq (P \times \Gamma^*) \times (P \times \Gamma^*)$ is defined as follows: if $\langle p, \gamma \rangle \hookrightarrow \langle q, \omega \rangle$, then $\langle p, \gamma\omega' \rangle \rightsquigarrow_{\mathcal{P}} \langle q, \omega\omega' \rangle$ for every $\omega' \in \Gamma^*$. A run of $\mathcal{P}$ is an infinite sequence of configurations $c_0, c_1, \ldots$ such that for every $i \geq 0$: $c_i \rightsquigarrow_{\mathcal{P}} c_{i+1}$.

$s \models_\lambda a$	$\Longleftrightarrow$	$s \in \lambda(a)$.
$s \models_\lambda \neg a$	$\Longleftrightarrow$	$s \notin \lambda(a)$.
$s \models_\lambda \psi_1 \wedge \psi_2$	$\Longleftrightarrow$	$s \models_\lambda \psi_1$ and $s \models_\lambda \psi_2$.
$s \models_\lambda \psi_1 \vee \psi_2$	$\Longleftrightarrow$	$s \models_\lambda \psi_1$ or $s \models_\lambda \psi_2$.
$s \models_\lambda AX\psi$	$\Longleftrightarrow$	$s' \models_\lambda \psi$ for every successor s' of s.
$s \models_\lambda EX\psi$	$\Longleftrightarrow$	There exists a successor s' of s s.t. $s' \models_\lambda \psi$.
$s \models_\lambda A[\psi_1 U \psi_2]$	$\Longleftrightarrow$	For every path of $T, \pi = s_0, s_1, \ldots,$ with $s_0 = s, \exists i \geq 0$ s.t. $s_i \models_\lambda \psi_2$ and $\forall 0 \leq j < i, s_j \models_\lambda \psi_1$.
$s \models_\lambda E[\psi_1 U \psi_2]$	$\Longleftrightarrow$	There exists a path of $T, \pi = s_0, s_1, \ldots,$ with $s_0 = s$, s.t. $\exists i \geq 0, s_i \models_\lambda \psi_2$ and $\forall 0 \leq j < i, s_j \models_\lambda \psi_1$.
$s \models_\lambda A[\psi_1 R \psi_2]$	$\Longleftrightarrow$	For every path of $T, \pi = s_0, s_1, \ldots,$ with $s_0 = s, \forall i \geq 0$ s.t. $s_i \not\models_\lambda \psi_2, \exists 0 \leq j < i,$ s.t. $s_j \models_\lambda \psi_1$.
$s \models_\lambda E[\psi_1 R \psi_2]$	$\Longleftrightarrow$	There exists a path of $T, \pi = s_0, s_1, \ldots,$ with $s_0 = s,$ s.t. $\forall i \geq 0$ s.t. $s_i \not\models_\lambda \psi_2, \exists 0 \leq j < i$ s.t. $s_j \models_\lambda \psi_1$.

Fig. 1. Semantics of CTL.

The reachability relation $\Longrightarrow_{\mathcal{P}} \subseteq (P \times \Gamma^*) \times (P \times \Gamma^*)$ is the reflexive and transitive closure of the successor relation. Formally $\Longrightarrow_{\mathcal{P}}$ is defined as follows: (1) $c \Longrightarrow_{\mathcal{P}} c$ for every $c \in P \times \Gamma^*$, (2) if $c \Longrightarrow_{\mathcal{P}} c''$ and $c'' \rightsquigarrow_{\mathcal{P}} c'$, then $c \Longrightarrow_{\mathcal{P}} c'$.

Let c be a given initial configuration of $\mathcal{P}$. Starting from c, $\mathcal{P}$ induces the transition system $T^c_{\mathcal{P}} = (P \times \Gamma^*, \rightsquigarrow_{\mathcal{P}}, c)$. Let AP be a set of atomic propositions, φ be a CTL formula on AP, and $\lambda : AP \to 2^{P \times \Gamma^*}$ be a labeling function. We say that $(\mathcal{P}, c) \models_\lambda \varphi$ iff $T^c_{\mathcal{P}} \models_\lambda \varphi$.

2.3. Alternating Büchi pushdown systems

Definition 3. An Alternating Büchi Pushdown System *(ABPDS for short)* is a tuple $\mathcal{BP} = (P, \Gamma, \Delta, F)$, where P is a finite set of control locations, Γ is the stack alphabet, $F \subseteq P$ is a finite set of accepting control locations and Δ is a function that assigns to each element of $P \times \Gamma$ a *positive boolean formula* over $P \times \Gamma^*$.

A configuration of an ABPDS is a pair $\langle p, \omega \rangle$, where $p \in P$ is a control location and $\omega \in \Gamma^*$ is a stack content. We assume w.l.o.g. that the boolean formulas are in disjunctive normal form. This allows to consider Δ as a subset of $(P \times \Gamma) \times 2^{P \times \Gamma^*}$. Thus, rules of Δ of the form[1] $\langle p, \gamma \rangle \hookrightarrow \bigvee_{j=1}^{n} \bigwedge_{i=1}^{m_j} \langle p_i^j, \omega_i^j \rangle$ can be denoted by the union of n rules of the form $\langle p, \gamma \rangle \hookrightarrow \{\langle p_1^j, \omega_1^j \rangle, \ldots, \langle p_{m_j}^j, \omega_{m_j}^j \rangle\}$, where $1 \leq j \leq n$. Let $t = \langle p, \gamma \rangle \hookrightarrow \{\langle p_1, \omega_1 \rangle, \ldots, \langle p_n, \omega_n \rangle\}$ be a rule of Δ. For every $\omega \in \Gamma^*$, the configuration $\langle p, \gamma\omega \rangle$ (resp. $\{\langle p_1, \omega_1\omega \rangle, \ldots, \langle p_n, \omega_n\omega \rangle\}$) is an immediate predecessor (resp. successor) of $\{\langle p_1, \omega_1\omega \rangle, \ldots, \langle p_n, \omega_n\omega \rangle\}$ (resp. $\langle p, \gamma\omega \rangle$).

A run ρ of $\mathcal{BP}$ from an initial configuration $\langle p_0, \omega_0 \rangle$ is a tree in which the root is labeled by $\langle p_0, \omega_0 \rangle$, and the other nodes are labeled by elements of $P \times \Gamma^*$. If a node of ρ is labeled by $\langle p, \omega \rangle$ and has n children labeled by $\langle p_1, \omega_1 \rangle, \ldots, \langle p_n, \omega_n \rangle$, respectively, then necessarily, $\{\langle p_1, \omega_1 \rangle, \ldots, \langle p_n, \omega_n \rangle\}$ is an immediate successor of $\langle p, \omega \rangle$ in $\mathcal{BP}$. A path $c_0 c_1 \ldots$ of a run ρ is an *infinite* sequence of configurations such that c_0 is the root of ρ and for every $i \geq 0$, c_{i+1} is one of the children of the node c_i in ρ. The path is accepting from the initial configuration c_0 if and only if it visits infinitely often configurations with control locations in F. A run ρ is accepting if and only if all its paths are accepting. Note that an accepting run has only *infinite* paths; it does not involve finite paths. A configuration c is accepted (or recognized) by $\mathcal{BP}$ iff $\mathcal{BP}$ has an accepting run starting from c. The language of $\mathcal{BP}$, $\mathcal{L}(\mathcal{BP})$ is the set of configurations accepted by $\mathcal{BP}$.

The reachability relation $\Longrightarrow_{\mathcal{BP}} \subseteq (P \times \Gamma^*) \times 2^{P \times \Gamma^*}$ is the reflexive and transitive closure of the immediate successor relation. Formally $\Longrightarrow_{\mathcal{BP}}$ is defined as follows: (1) $c \Longrightarrow_{\mathcal{BP}} \{c\}$ for every $c \in P \times \Gamma^*$, (2) $c \Longrightarrow_{\mathcal{BP}} C$ if C is an immediate successor of c, (3) if $c \Longrightarrow_{\mathcal{BP}} \{c_1, \ldots, c_n\}$ and $c_i \Longrightarrow_{\mathcal{BP}} C_i$ for every $i : 1 \leq i \leq n$, then $c \Longrightarrow_{\mathcal{BP}} \bigcup_{i=1}^{n} C_i$. We will write $\{c_1, \ldots, c_n\} \Longrightarrow_{\mathcal{BP}} \bigcup_{i=1}^{n} C_i$ if for every $i : 1 \leq i \leq n$: $c_i \Longrightarrow_{\mathcal{BP}} C_i$.

The functions $Pre_{\mathcal{BP}} : 2^{P \times \Gamma^*} \longrightarrow 2^{P \times \Gamma^*}$, $Pre^*_{\mathcal{BP}} : 2^{P \times \Gamma^*} \longrightarrow 2^{P \times \Gamma^*}$ and $Pre^+_{\mathcal{BP}} : 2^{P \times \Gamma^*} \longrightarrow 2^{P \times \Gamma^*}$ are defined as follows: $Pre_{\mathcal{BP}}(C) = \{c \in P \times \Gamma^* \mid \exists C' \subseteq C$ s.t. C' is an immediate successor of $c\}$. $Pre^*_{\mathcal{BP}}(C) = \{c \in P \times \Gamma^* \mid \exists C' \subseteq C$ s.t. $c \Longrightarrow_{\mathcal{BP}} C'\}$. $Pre^+_{\mathcal{BP}}(C) = Pre_{\mathcal{BP}} \circ Pre^*_{\mathcal{BP}}(C)$.

To represent (infinite) sets of configurations of ABPDSs, we use Alternating Multi-Automata:

Definition 4. (See [3].) Let $\mathcal{BP} = (P, \Gamma, \Delta, F)$ be an ABPDS. An Alternating Multi-Automaton *(AMA for short)* is a tuple $\mathcal{A} = (Q, \Gamma, \delta, I, Q_f)$, where Q is a finite set of states that contains P, Γ is the input alphabet, $\delta \subseteq (Q \times \Gamma) \times 2^Q$ is a finite set of transition rules, $I \subseteq P$ is a finite set of initial states, $Q_f \subseteq Q$ is a finite set of final states.

A Multi-Automaton *(MA for short)* is an AMA such that $\delta \subseteq (Q \times \Gamma) \times Q$.

We define the reflexive and transitive transition relation $\longrightarrow_\delta \subseteq (Q \times \Gamma^*) \times 2^Q$ as follows: (1) $q \xrightarrow{\epsilon}_\delta \{q\}$ for every $q \in Q$, where ϵ is the empty word, (2) $q \xrightarrow{\gamma}_\delta Q'$, if $q \xrightarrow{\gamma} Q' \in \delta$, (3) if $q \xrightarrow{\omega}_\delta \{q_1, \ldots, q_n\}$ and $q_i \xrightarrow{\gamma}_\delta Q_i$ for every $i : 1 \leq i \leq n$, then

[1] This rule represents $\Delta(p, \gamma) = \bigvee_{j=1}^{n} \bigwedge_{i=1}^{m_j} (p_i^j, \omega_i^j)$.

$q \xrightarrow{\omega\gamma}_\delta \bigcup_{i=1}^n Q_i$. By abuse of notation, we sometimes write $\{q_1, ..., q_n\} \xrightarrow{\omega}_\delta \bigcup_{i=1}^n Q_i$ if for every $i : 1 \leq i \leq n$, $q_i \xrightarrow{\omega}_\delta Q_i$. The automaton $\mathcal{A}$ recognizes a configuration $\langle p, \omega\rangle$ iff there exists $Q' \subseteq Q_f$ such that $p \xrightarrow{\omega}_\delta Q'$ and $p \in I$. The language of $\mathcal{A}$, $L(\mathcal{A})$, is the set of configurations recognized by $\mathcal{A}$. A set of configurations is regular if it can be recognized by an AMA. It is easy to show that AMAs are closed under boolean operations and that they are equivalent to MAs. Given an AMA, one can compute an equivalent MA by performing a kind of powerset construction as done for the determinization procedure. Similarly, MAs can also be used to recognize (infinite) regular sets of configurations for PDSs.

Proposition 1. *(See [22].) Let $\mathcal{A} = (Q, \Gamma, \delta, I, Q_f)$ be an AMA. Deciding whether a configuration $\langle p, \omega\rangle$ is accepted by $\mathcal{A}$ can be done in $O(|Q| \cdot |\delta| \cdot |\omega|)$ time.*

3. Computing the language of an ABPDS

Our goal in this section is to compute the set of accepting configurations of an Alternating Büchi PushDown System $\mathcal{BP} = (P, \Gamma, \Delta, F)$. We show that it is regular and that it can effectively be represented by an AMA. Determining whether $\mathcal{BP}$ has an accepting run is a non-trivial problem because a run of $\mathcal{BP}$ is an *infinite* tree with an infinite number of paths labeled by PDS configurations, which are control states and stack contents. All the paths of an accepting run are infinite and should all go through final control locations infinitely often. The difficulty comes from the fact that we cannot reason about the different paths of an ABPDS independently, we need to reason about *runs labeled with PDS configurations*. We proceed as follows: first, we characterize the set of configurations from which $\mathcal{BP}$ has an accepting run. Then, based on this characterization, we compute an AMA representing this set.

3.1. Characterizing $\mathcal{L}(\mathcal{BP})$

We give in this section a characterization of $\mathcal{L}(\mathcal{BP})$, i.e., the set of configurations from which $\mathcal{BP}$ has an accepting run. Let $(X_i)_{i\geq 0}$ be the sequence defined as follows: $X_0 = P \times \Gamma^*$ and $X_{i+1} = Pre^+(X_i \cap F \times \Gamma^*)$ for every $i \geq 0$. Let $Y_{\mathcal{BP}} = \bigcap_{i\geq 0} X_i$. We can show that $\mathcal{L}(\mathcal{BP}) = Y_{\mathcal{BP}}$. To prove this result, we first show that for every configuration $c \in P \times \Gamma^*$, $\mathcal{BP}$ has a run from c such that each path of the run visits some accepting control locations at least k times iff $c \in X_k$.

Lemma 1. *$\mathcal{BP}$ has a run ρ from a configuration $\langle p, \omega\rangle$ such that each path of ρ visits configurations with control locations in F at least k times iff $\langle p, \omega\rangle \in X_k$.*

Proof. ($\Longrightarrow$) The proof proceeds by induction on k. We directly obtain that $\langle p, \omega\rangle \in X_0 = P \times \Gamma^*$. We only need to show that $\langle p, \omega\rangle \in X_k$ when $k \geq 1$.

Let $\langle p_1, \omega_1\rangle, .., \langle p_n, \omega_n\rangle$ be the first nodes of ρ that are visited in each path of ρ such that $p_i \in F$. Then we get that (a) $\langle p, \omega\rangle \Longrightarrow_{\mathcal{BP}} \{\langle p_1, \omega_1\rangle, .., \langle p_n, \omega_n\rangle\}$, (b) for every $i : 1 \leq i \leq n$, $p_i \in F$, (c) for every $i : 1 \leq i \leq n$, $\mathcal{BP}$ has a run ρ_i from the configuration $\langle p_i, \omega_i\rangle$ such that all the paths of ρ_i can visit some configurations with control locations in F at least $k - 1$ times.

By applying the induction hypothesis to (c), we obtain that $\langle p_i, \omega_i\rangle \in X_{k-1}$ for every $i : 1 \leq i \leq n$. Since $p_i \in F$ for every $i : 1 \leq i \leq n$, we get that $\langle p_i, \omega_i\rangle \in X_{k-1} \cap F \times \Gamma^*$ for every $i : 1 \leq i \leq n$. Since $X_k = Pre^+(X_{k-1} \cap F \times \Gamma^*)$ and the result from (a), we get that $\langle p, \omega\rangle \in X_k$.

($\Longleftarrow$) Let's apply the induction on k, it is straightforward when $k = 0$. We only need to show that $\mathcal{BP}$ has a run ρ from the configuration $\langle p, \omega\rangle$ such that each path of ρ can visit some configurations with control locations in F at least k times when $k \geq 1$.

Since $\langle p, \omega\rangle \in X_k$ where $X_k = Pre^+(X_{k-1} \cap F \times \Gamma^*)$, we obtain that $\langle p, \omega\rangle \Longrightarrow_{\mathcal{BP}} \{\langle p_1, \omega_1\rangle, .., \langle p_n, \omega_n\rangle\}$, and $\langle p_i, \omega_i\rangle \in X_{k-1} \cap F \times \Gamma^*$ for every $i : 1 \leq i \leq n$. By applying the induction hypothesis to $\langle p_i, \omega_i\rangle \in X_{k-1}$ for every $i : 1 \leq i \leq n$, we get that $\mathcal{BP}$ has a run ρ_i from the configuration $\langle p_i, \omega_i\rangle$ such that each path of ρ_i can visit some configurations with control locations in F at least $k - 1$ times. Since $\langle p, \omega\rangle \Longrightarrow_{\mathcal{BP}} \{\langle p_1, \omega_1\rangle, .., \langle p_n, \omega_n\rangle\}$, and $\langle p_i, \omega_i\rangle \in X_{k-1} \cap F \times \Gamma^*$ for each $i : 1 \leq i \leq n$, we obtain that $\mathcal{BP}$ has a run ρ from the configuration $\langle p, \omega\rangle$ such that each path of ρ can visit some configurations with control locations in F at least k times. □

Now, let us prove that $\mathcal{L}(\mathcal{BP}) = Y_{\mathcal{BP}}$.

Theorem 1. *$\mathcal{BP}$ has an accepting run from a configuration $\langle p, \omega\rangle$ iff $\langle p, \omega\rangle \in Y_{\mathcal{BP}}$.*

Proof. ($\Longrightarrow$) First we show that if $\mathcal{BP}$ has an accepting run from the configuration $\langle p, \omega\rangle$, then the configuration $\langle p, \omega\rangle$ must be in $Y_{\mathcal{BP}}$. We prove that if the configuration $\langle p, \omega\rangle$ is not in $Y_{\mathcal{BP}}$, then $\mathcal{BP}$ has no accepting run from $\langle p, \omega\rangle$. Since $\langle p, \omega\rangle \notin Y_{\mathcal{BP}}$ and $Y_{\mathcal{BP}} = \bigcap_{i\geq 0} X_i$, there exists $k \geq 0$ such that $\langle p, \omega\rangle \notin X_k$. By Lemma 1, all the runs from the configuration $\langle p, \omega\rangle$ can visit configurations with control locations in F at most $k - 1$ times, otherwise $\langle p, \omega\rangle \in X_k$, which contradicts the fact that $\langle p, \omega\rangle \notin X_k$. Thus, $\mathcal{BP}$ has no accepting run from the configuration $\langle p, \omega\rangle$.

Algorithm 1: Computation of $Y_{\mathcal{BP}}$.

```
   Input  : An ABPDS $\mathcal{BP} = (P, \Gamma, \Delta, F)$;
   Output: An AMA $\mathcal{A} = (\mathcal{Q}, \Gamma, \delta, I, \mathcal{Q}_f)$ that recognizes $Y_{\mathcal{BP}}$;
1  Let $i = 0, \delta = \{(q_f, \gamma, \{q_f\}) \mid$ for every $\gamma \in \Gamma, p \in P\}$ and $p^0 = q_f$ for every $p \in P$;
2  repeat we call this loop $loop_1$
3      $i := i + 1$;
4      Add in $\delta$ a new transition rule $p^i \xrightarrow{\epsilon} p^{i-1}$, for every $p \in F$;
5      repeat we call this loop $loop_2$
6          For every $\langle p, \gamma \rangle \hookrightarrow \{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\}$ in $\Delta$,
7          and every case where $p_k^i \xrightarrow{\omega_k}_\delta Q_k$, for every $k : 1 \le k \le n$;
8              Add a new rule $p^i \xrightarrow{\gamma} \bigcup_{k=1}^{n} Q_k$ in $\delta$;
9      until No new transition rule can be added;
10     Remove from $\delta$ the transition rules $p^i \xrightarrow{\epsilon} p^{i-1}$, for every $p \in F$;
11     Replace in $\delta$ every transition rule $p^i \xrightarrow{\gamma} R$ by $p^i \xrightarrow{\gamma} \pi^i(R)$, for every $p \in P, \gamma \in \Gamma, R \subseteq \mathcal{Q}$
12 until $i > 1$ and for every $p \in P, \gamma \in \Gamma, R \subseteq P \times \{i\} \cup \{q_f\}$, $p^i \xrightarrow{\gamma} R \in \delta \iff p^{i-1} \xrightarrow{\gamma} \pi^{-1}(R) \in \delta$;
```

($\Longleftarrow$) Now we prove the other direction, i.e., we prove that if the configuration $\langle p, \omega \rangle$ is in $Y_{\mathcal{BP}}$, then $\mathcal{BP}$ has an accepting run from $\langle p, \omega \rangle$. Since the function $f(X) = Pre^+(X \cap F \times \Gamma^*)$ is monotone, by the Tarski theorem [23], $Y_{\mathcal{BP}}$ is the greatest fixpoint of f, we get that $Y_{\mathcal{BP}} = Pre^+(Y_{\mathcal{BP}} \cap F \times \Gamma^*)$. Since $\langle p, \omega \rangle \in Y_{\mathcal{BP}}$, we get that $\langle p, \omega \rangle \in Pre^+(Y_{\mathcal{BP}} \cap F \times \Gamma^*)$. By the definition of Pre^+, there exists a set of configurations $\{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\} \subseteq Y_{\mathcal{BP}} \cap F \times \Gamma^*$ such that $\langle p, \omega \rangle \Longrightarrow_{\mathcal{BP}} \{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\}$. Since $\{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\} \subseteq Y_{\mathcal{BP}} \cap F \times \Gamma^*$, we obtain that $\langle p_i, \omega_i \rangle \in Y_{\mathcal{BP}}$ and $p_i \in F$ for every $i : 1 \le i \le n$. Let's construct a finite tree ρ with root $\langle p, \omega \rangle$, the leaves of ρ are $\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle$, the inner nodes of ρ are the successors during the derivation of $\langle p, \omega \rangle \Longrightarrow_{\mathcal{BP}} \{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\}$. Each path of ρ can visit some configurations with control locations in F at least once. Since $\langle p_i, \omega_i \rangle \in Y_{\mathcal{BP}}$ for every $i : 1 \le i \le n$, we can repeatedly construct a finite tree ρ_i for the configuration $\langle p_i, \omega_i \rangle$ such that ρ_i has the same properties as ρ. Let's replace each leaf $\langle p_i, \omega_i \rangle$ in ρ by the tree ρ_i and obtain a new tree ρ such that each path of the new tree ρ can visit some configurations with control locations in F at least twice. Now we infinitely repeat this procedure to the leaves of the latest tree ρ. Finally, each path of the infinite limit tree ρ can infinitely often visit some configurations with control locations in F. We obtain that ρ is an accepting run. □

3.2. Computing $\mathcal{L}(\mathcal{BP})$

Our goal is to compute $Y_{\mathcal{BP}} = \bigcap_{i \ge 0} X_i$, where $X_0 = P \times \Gamma^*$ and for every $i \ge 0$, $X_{i+1} = Pre^+(X_i \cap F \times \Gamma^*)$. We provide a saturation procedure that computes the set $Y_{\mathcal{BP}}$. Our procedure is inspired from the algorithm given in [22] to compute the winning region of a Büchi game on a pushdown graph.

We show that $Y_{\mathcal{BP}}$ is regular and it can be represented by an AMA $\mathcal{A} = (\mathcal{Q}, \Gamma, \delta, I, \mathcal{Q}_f)$ whose set of states $\mathcal{Q}$ is a subset of $P \times \mathbb{N} \cup \{q_f\}$, where q_f is a special state denoting the final state ($\mathcal{Q}_f = \{q_f\}$). From now on, for every $p \in P$ and $i \in \mathbb{N}$, we write p^i to denote (p, i).

Intuitively, to compute $Y_{\mathcal{BP}}$, we will compute iteratively the different X_i's by applying the saturation procedure of [3]. The iterative procedure computes different automata. The automaton computed during the iteration i uses states of the form p^i having i as index. To force termination, we use an acceleration criterion. For this, we need to define two projection functions π^{-1} and π^i defined as follows: for every $S \subseteq P \times \mathbb{N} \cup \{q_f\}$,

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

$$\pi^i(S) = \left\{q^i \,\middle|\, \exists 1 \le j \le i \text{ s.t. } q^j \in S\right\} \cup \{q_f \mid q_f \in S\}.$$

The AMA $\mathcal{A}$ is computed iteratively using Algorithm 1. Let us explain the intuition behind the different lines of this algorithm. Let A_i be the automaton obtained at step i (a step starts at Line 3). For every $p \in P$, the state p^i is meant to represent state p at step i, i.e., A_i recognizes a configuration $\langle p, \omega \rangle$ iff $p^i \xrightarrow{\omega}_\delta q_f$. Let A_0 be the automaton obtained after the initialization step (Line 1). It is clear that A_0 recognizes $X_0 = P \times \Gamma^*$. Suppose now that the algorithm is at the beginning of the ith iteration ($loop_1$). Line 4 adds the ϵ-transition $p^i \xrightarrow{\epsilon} p^{i-1}$ for every control state $p \in F$. After this step, we obtain $L(A_{i-1}) \cap F \times \Gamma^*$. $loop_2$ at Lines 5–9 is the saturation procedure of [3]. It computes the Pre^* of $L(A_{i-1}) \cap F \times \Gamma^*$. Line 10 removes the ϵ-transition added by Line 4. After this step, the automaton recognizes $Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$. Let us call **Algorithm B** the above algorithm without Line 11. It follows from the explanation above that if **Algorithm B** terminates, it will produce $Y_{\mathcal{BP}}$. However, this procedure will never terminate if the sequence (X_i) is strictly decreasing. Consider for example the ABPDS $\mathcal{BP} = (\{q\}, \{\gamma\}, \Delta, \{q\})$, where $\Delta = \{\langle q, \gamma \rangle \hookrightarrow \langle q, \epsilon \rangle\}$. Then, for every $i \ge 0$, $X_i = \{\langle q, \gamma^i \omega \rangle \mid \omega \in \gamma^*\}$. It is clear that **Algorithm B** will never terminate on this example.

The substitution at Line 11 is the acceleration used to force the termination of the algorithm, tested at Line 12. We can show that thanks to Line 11 and to the test of Line 12 (note that there is no acceleration when $i = 1$), our algorithm always terminates and produces $Y_{\mathcal{BP}}$. To prove this, we need to prove some auxiliary results. First, we show that for every transition rule at the iteration $i+1$, there exists a similar transition rule at the iteration i.

Proposition 2. *In Algorithm 1, for every $\gamma \in \Gamma$, $\omega \in \Gamma^*$, $p \in P$, $S \subseteq \mathcal{Q}$; at each step $i \geq 1$, the following holds:*

(a) *if $p^i \xrightarrow{\gamma} S \in \delta$, then $p^{i-1} \xrightarrow{\gamma} \pi^{-1}(\pi^i(S)) \in \delta$;*
(b) *if $p^i \xrightarrow{\omega}_\delta S$, then $p^{i-1} \xrightarrow{\omega}_\delta \pi^{-1}(\pi^i(S))$.*

Proof. As the transition rule could be added by either the saturation procedure (Lines 5–9) or the substitution (Line 11), both situations should be considered. We proceed by induction on i.

Basis. $i=1$. In this case, whenever a transition rule $p^1 \xrightarrow{\gamma} S$ is added into δ by either the saturation procedure or the substitution, we can get that $\pi^{-1}(\pi^1(S)) = \{q_f\}$. Since $p^0 = q_f$ and $q_f \xrightarrow{\gamma} \{q_f\}$ in δ, we obtain that $p^0 \xrightarrow{\gamma} \{q_f\}$. Hence, $p^0 \xrightarrow{\gamma} \pi^{-1}(\pi^1(S))$. Therefore, the statement (a) holds.

Now let us suppose that $p^1 \xrightarrow{\omega}_\delta S$, we show that $p^0 \xrightarrow{\omega}_\delta \pi^{-1}(\pi^i(S))$. Since $p^0 \xrightarrow{\gamma} \{q_f\}$ and $q_f \xrightarrow{\gamma'} \{q_f\}$ for every $\gamma, \gamma' \in \Gamma$ and since $\pi^{-1}(\pi^1(S)) = \{q_f\}$ for every $S \subseteq P \times \{1,0\} \cup \{q_f\}$, we obtain that $p^0 \xrightarrow{\omega}_\delta \pi^{-1}(\pi^i(S))$, for every $p \in P, \omega \in \Gamma^+, S \subseteq P \times \{1,0\} \cup \{q_f\}$. The case where $\omega = \epsilon$ is trivial, as $p^0 \xrightarrow{\epsilon}_\delta \{p^0\}$ and either $p^1 \xrightarrow{\epsilon}_\delta \{p^0\}$ if $p \in F$ or $p^1 \xrightarrow{\epsilon}_\delta \{p^1\}$. Hence, the statement (b) holds.

Step. $i \geq 2$. Let n be the number of transition rules added at the step i. We proceed by induction on n.

- **Basis.** $n=0$. Then, there is no transition rule in the form of $p^i \xrightarrow{\gamma} S$ in δ which implies that the statement (a) holds. For every $p^i \xrightarrow{\omega}_\delta S$, we get that either $p^i \xrightarrow{\epsilon}_\delta p^{i-1} \xrightarrow{\omega}_\delta S$ with $S \subseteq P \times \{i-1\} \cup \{q_f\}$ if $p \in F$ or $p^i \xrightarrow{\epsilon}_\delta S$ with $S = \{p^i\}$ and $\omega = \epsilon$. Since $\pi^{-1}(\pi^i(S)) = S$ and $p^{i-1} \xrightarrow{\epsilon}_\delta \{p^{i-1}\}$, we get that the statement (b) holds.
- **Step.** $n \geq 1$.

  For statement (a). W.l.o.g., let $t = p^i \xrightarrow{\gamma} S$ be the nth transition rule added by the saturation procedure (the case where t is added by the substitution will be discussed later). Then there exists a transition rule in $\mathcal{BP}$

$$\langle p, \gamma\rangle \longrightarrow \big\{\langle p_1, \omega_1\rangle, \ldots, \langle p_m, \omega_m\rangle\big\} \in \Delta \tag{1}$$

  such that $p_j^i \xrightarrow{\omega_j}_\delta S_j$ for every $j : 1 \leq j \leq m$ and $S = \bigcup_{j=1}^m S_j$. By applying the induction hypothesis on i to $p_j^i \xrightarrow{\omega_j}_\delta S_j$ for every $j : 1 \leq j \leq m$, we get that $p_j^{i-1} \xrightarrow{\omega_j}_\delta \pi^{-1}(\pi^i(S_j))$ for every $j : 1 \leq j \leq m$. Thus, to show that $p^{i-1} \xrightarrow{\gamma} \pi^{-1}(\pi^i(S))$ (i.e., the statement (a) holds), it is sufficient to show that for every $j : 1 \leq j \leq m$, there exists R_j such that $\pi^{i-1}(R_j) = \pi^{-1}(\pi^i(S_j))$ and $p_j^{i-1} \xrightarrow{\omega_j}_\delta R_j$ exists during the saturation procedure at the $(i-1)$th iteration.

  If the derivation of $p_j^{i-1} \xrightarrow{\omega_j}_\delta \pi^{-1}(\pi^1(S_j))$ does not use any transition rule that is added by the substitution at Line 11, then, $p_j^{i-1} \xrightarrow{\omega_j}_\delta \pi^{-1}(\pi^1(S_j))$ exists during the saturation procedure at the $(i-1)$th iteration. Otherwise, there is a transition rule $q^{i-1} \xrightarrow{\gamma'}_\delta R$ which is used in the derivation of $p_j^{i-1} \xrightarrow{\omega_j}_\delta \pi^{-1}(\pi^1(S_j))$ and is obtained by replacing $q^{i-1} \xrightarrow{\gamma'}_\delta R'$ at Line 11, where $R = \pi^{i-1}(R')$. Let us decompose $p_j^{i-1} \xrightarrow{\omega_j}_\delta \pi^{-1}(\pi^1(S_j))$ as follows:
  – $\omega_j = u\gamma' v$ with $u, v \in \Gamma^*$,
  – $p_j^{i-1} \xrightarrow{u}_\delta G \cup \{q^{i-1}\}$ with $G \subseteq P \times \{i-1\} \cup \{q_f\}$,
  – $G \xrightarrow{\gamma' v}_\delta G'$,
  – $R \xrightarrow{v}_\delta G''$,
  – $\pi^{-1}(\pi^1(S_j)) = G' \cup G''$.

  By applying the induction hypothesis on i to $R \xrightarrow{v}_\delta G''$, there exists G''' such that $R' \xrightarrow{v}_\delta G'''$ is obtained by applying the saturation procedure at the $(i-1)$th iteration and $G'' = \pi^{i-1}(G''')$. Thus, there must exist R_j such that $\pi^{i-1}(R_j) = \pi^{-1}(\pi^i(S_j))$ and the derivation of $p_j^{i-1} \xrightarrow{\omega_j}_\delta R_j$ uses transition rules added by the substitution at Line 11 less often than the derivation of $p_j^{i-1} \xrightarrow{\omega_j}_\delta S_j$. We can apply the same reasoning to $p_j^{i-1} \xrightarrow{\omega_j}_\delta R_j$ to show that there exists R_j' such that $p_j^{i-1} \xrightarrow{\omega_j}_\delta R_j'$ holds during the saturation procedure at the $(i-1)$th iteration. Hence the statement (a) holds.

  If a transition rule $p^i \xrightarrow{\gamma} \pi^i(S)$ is added by the substitution at Line 11 due to the transition t, then (a) still hold.

For statement (b). Let us consider the statement (b) where we show that if $p^i \xrightarrow{\omega}_\delta S$, then $p^{i-1} \xrightarrow{\omega}_\delta \pi^{-1}(\pi^i(S))$. Suppose $t = p_0^i \xrightarrow{\gamma} \{q_1^i, \ldots, q_m^i, q_{m+1}^{i-1}, \ldots, q_{m+m'}^{i-1}\}$ is the nth transition rule added by either the saturation procedure or the substitution. Let ℓ be the number of times that t is used in the derivation of $p^i \xrightarrow{\omega}_\delta S$. We proceed by induction on ℓ. In the basic case when $\ell = 0$, statement (b) holds by applying the induction hypothesis on n.
Let us consider the case where $\ell \geq 1$. Then, there exist $u, v \in \Gamma^*$ such that
- $\omega = v\gamma u$,
- $p^i \xrightarrow{v}_\delta G \cup \{p_0^i\}$ for some $G \subseteq \mathcal{Q}$ and t is not used in the derivation of $p^i \xrightarrow{v}_\delta G \cup \{p_0^i\}$,
- $G \xrightarrow{\gamma u}_\delta G'$,
- $q_j^i \xrightarrow{u}_\delta S_j$ for every $j : 1 \leq j \leq m$,
- $q_j^{i-1} \xrightarrow{u}_\delta S_j$ for every $j : m+1 \leq j \leq m+m'$,
- $S = G' \cup \bigcup_{j=1}^{m+m'} S_j$.

By applying the induction hypothesis on n to $p^i \xrightarrow{v}_\delta G \cup \{p_0^i\}$ we obtain that $p^{i-1} \xrightarrow{v}_\delta \pi^{-1}(\pi^i(G)) \cup \{p_0^{i-1}\}$. By applying the induction hypothesis on ℓ to $G \xrightarrow{\gamma u}_\delta G'$ and $q_j^i \xrightarrow{u}_\delta S_j$ for every $j : 1 \leq j \leq m$, we obtain that $\pi^{-1}(\pi^i(G)) \xrightarrow{\gamma u}_\delta \pi^{-1}(\pi^i(G'))$ and $q_j^{i-1} \xrightarrow{u}_\delta \pi^{-1}(\pi^i(S_j))$ for every $j : 1 \leq j \leq m$. By applying the statement (a) to $t = p_0^i \xrightarrow{\gamma} \{q_1^i, \ldots, q_m^i, q_{m+1}^{i-1}, \ldots, q_{m+m'}^{i-1}\}$, we obtain that $p_0^{i-1} \xrightarrow{\gamma} \{q_1^{i-1}, \ldots, q_{m+m'}^{i-1}\}$. Since $\pi^{-1}(\pi^i(S)) = \pi^{-1}(\pi^i(G')) \cup \bigcup_{j=1}^{m} \pi^{-1}(\pi^i(S_j)) \cup \bigcup_{j=m+1}^{m+m'} S_j$, we get that $p^{i-1} \xrightarrow{\omega}_\delta \pi^{-1}(\pi^i(S))$. □

To show the termination of Algorithm 1, we will show that there exists a fixpoint such that the termination condition of *loop*$_1$ is true. To show this, let **Algorithm C** be Algorithm 1 without Line 12, i.e., without the termination condition of *loop*$_1$. We show that there exists a fixpoint n such that $L(A_n) = L(A_{i+1})$ for every $i \geq n$.

Lemma 2. *Let $n \geq 1$ be the first number in* **Algorithm C** *such that for every $p \in P, \gamma \in \Gamma$, $S \subseteq P \times \{n+1\} \cup \{q_f\}$, $p^{n+1} \xrightarrow{\gamma} S \in \delta \iff p^n \xrightarrow{\gamma} \pi^{-1}(S) \in \delta$. For every $i \geq n$, $L(A_{i+1}) = L(A_n)$.*

Proof. Since Line 11 of **Algorithm C** will replace $p^{i+1} \xrightarrow{\gamma} S$ by $p^{i+1} \xrightarrow{\gamma} \pi^{i+1}(S)$, then each path $p^{i+1} \xrightarrow{\omega}_\delta \{q_f\}$ only uses states of $P \times \{i+1\} \cup \{q_f\}$. In order to prove that $L(A_{i+1}) = L(A_n)$ for every $i \geq n$, it is sufficient to prove that for every $p \in P$, $\gamma \in \Gamma$, $p^{i+1} \xrightarrow{\gamma} \{q_1^{i+1}, \ldots, q_m^{i+1}\} \in \delta \iff p^n \xrightarrow{\gamma} \{q_1^n, \ldots, q_m^n\} \in \delta$ by induction on i.

- **Basis.** $i = n$. We get directly from the condition of n that

$$p^{n+1} \xrightarrow{\gamma} \{q_1^{n+1}, \ldots, q_m^{n+1}\} \in \delta \quad \iff \quad p^n \xrightarrow{\gamma} \{q_1^n, \ldots, q_m^n\} \in \delta \tag{0}$$

- **Step.** $i > n$. By applying the induction hypothesis (induction on i), we obtain that for every $p \in P, \gamma \in \Gamma$,

$$p^i \xrightarrow{\gamma} \{q_1^i, \ldots, q_m^i\} \in \delta \quad \iff \quad p^n \xrightarrow{\gamma} \{q_1^n, \ldots, q_m^n\} \in \delta \tag{1}$$

Since the result (1), $p^{i+1} \xrightarrow{\gamma} \{q_1^{i+1}, \ldots, q_m^{i+1}\}$ is added based on A_i, for every $p \in P$, $\gamma \in \Gamma$ and $p^{n+1} \xrightarrow{\gamma} \{q_1^{n+1}, \ldots, q_m^{n+1}\}$ is added based on A_n, for every $p \in P$, $\gamma \in \Gamma$, we obtain that:

$$p^{i+1} \xrightarrow{\gamma} \{q_1^{i+1}, \ldots, q_m^{i+1}\} \in \delta \quad \iff \quad p^{n+1} \xrightarrow{\gamma} \{q_1^{n+1}, \ldots, q_m^{n+1}\} \in \delta$$

From (0), we get that $p^{i+1} \xrightarrow{\gamma} \{q_1^{i+1}, \ldots, q_m^{i+1}\} \in \delta \iff p^n \xrightarrow{\gamma} \{q_1^n, \ldots, q_m^n\} \in \delta$. □

To show the correctness of Algorithm 1, we first show the following lemmas.

Lemma 3. *In Algorithm 1, for every $i \geq 1$, just after Line 10, A_i accepts $Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$.*

Proof. Before Line 4, $L(A_i) = \emptyset$. By adding transition rules $p^i \xrightarrow{\epsilon} p^{i-1}$ for every $p \in F$ at Line 4, we get that for every $p \in P, \omega \in \Gamma^*$, $p^i \xrightarrow{\omega}_\delta \{q_f\}$ iff $p^i \xrightarrow{\epsilon}_\delta \{p^{i-1}\} \xrightarrow{\omega}_\delta \{q_f\}$ and $p \in F$. Thus, we get that $L(A_i) = L(A_{i-1}) \cap F \times \Gamma^*$.
It is shown in [3], by applying the saturation procedure Lines 5–9, that A_i accepts $Pre^*(L(A_{i-1}) \cap F \times \Gamma^*)$.
To show that A_i accepts exactly $Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$ after Line 10, we first show that each configuration accepted by A_i after Line 10 is in $Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$, then we show that each configuration $c \in Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$ is accepted by A_i after Line 10.

- Suppose $\langle p, \omega\rangle \in P \times \Gamma^*$ is accepted by A_i after Line 10, we show that $\langle p, \omega\rangle \in Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$. Since there is no path of the form $p^i \xrightarrow{\epsilon}_\delta \{q_f\}$ after Line 10, we get that $|\omega| \geq 1$. Then, there exist $\gamma \in \Gamma, u \in \Gamma^+, Q \subseteq \mathcal{Q}$ such that $\omega = \gamma u$, $t = p^i \xrightarrow{\gamma} Q$ is in δ and $Q \xrightarrow{u}_\delta \{q_f\}$.

  Let $\langle p, \gamma\rangle \hookrightarrow \{\langle p_1, \omega_1\rangle, ..., \langle p_m, \omega_m\rangle\}$ be the transition used by the saturation procedure to add t such that $p^i_j \xrightarrow{\omega_j} Q_j$ for every $j : 1 \leq j \leq m$ and $Q = \bigcup_{j=1}^m Q_j$. This implies that $\{\langle p_1, \omega_1 u\rangle, ..., \langle p_m, \omega_m u\rangle\} \subseteq Pre^*(L(A_{i-1}) \cap F \times \Gamma^*)$. Thus, we get that $\langle p, \gamma u\rangle \in Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$.
- Suppose $\langle p, \omega\rangle \in Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$, we show that A_i after Line 10 accepts $\langle p, \omega\rangle$. Since $Pre^+(L(A_{i-1}) \cap F \times \Gamma^*) = Pre^*(Pre(L(A_{i-1}) \cap F \times \Gamma^*))$, we obtain that $Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$ is the limit of the infinite sequence $\{C_i\}_{i\geq 0}$ given by $C_0 = Pre(L(A_{i-1}) \cap F \times \Gamma^*)$ and $C_{j+1} = C_j \cup Pre(C_j)$ for every $j \geq 0$. Note that for every $j \geq 0$, $C_j \subseteq C_{j+1}$. It is sufficient to show that for every $j \geq 0$, every configuration $\langle p, \omega\rangle \in C_j$, A_i has a path $p^i \xrightarrow{\omega}_\delta \{q_f\}$ whose derivation does not use any transition rule in the form of $q^i \xrightarrow{\epsilon} \{q^{i-1}\}$ (note that such path allows A_i to recognize $\langle p, \omega\rangle$ after Line 10). We proceed by induction on j.
  - **Basis** $j = 0$. By applying the saturation procedure (Lines 6–8), A_i accepts C_0 if only the outcoming states p^i of the added transition rules are regarded as initial states. These transition rules starting from these new initial states are in the form of $p^i \xrightarrow{\gamma} Q$ where $Q \subseteq P \times \{i-1\} \cup \{q_f\}$. For every configuration $\langle p, \omega\rangle \in C_0$, A_i has a path of the form $p^i \xrightarrow{\omega}_\delta \{q_f\}$ whose derivation does not use any transition rule in the form of $q^i \xrightarrow{\epsilon} \{q^{i-1}\}$.
  - **Step** $j \geq 1$. For every configuration $\langle p, \omega\rangle \in C_j$, then, we obtain that either $\langle p, \omega\rangle \in C_{j-1}$ or $\langle p, \omega\rangle \in Pre(C_{j-1})$. The result follows from the induction hypothesis if $\langle p, \omega\rangle \in C_{j-1}$. If $\langle p, \omega\rangle \in Pre(C_{j-1})$, then there exist a transition rule $\langle p, \gamma\rangle \hookrightarrow \{\langle p_1, \omega_1\rangle, ..., \langle p_m, \omega_m\rangle\}$ and $u \in \Gamma^*$ such that $\omega = \gamma u$ and $\langle p_k, \omega_k u\rangle \in C_{j-1}$ for every $k : 1 \leq k \leq m$. By applying the induction hypothesis: we get that for every $k : 1 \leq k \leq m$, A_i has a path $p^i_k \xrightarrow{\omega_k}_\delta Q_k \xrightarrow{u}_\delta \{q_f\}$ whose derivation does not use any transition of the form $q^i \xrightarrow{\epsilon} \{q^{i-1}\}$. The saturation procedure will add a transition rule $p^i \xrightarrow{\gamma} \bigcup_{k=1}^m Q_k$. This implies that A_i has a path $p^i \xrightarrow{\gamma} \bigcup_{k=1}^m Q_k \xrightarrow{u}_\delta \{q_f\}$ whose derivation does not use any transition of the form $q^i \xrightarrow{\epsilon} \{q^{i-1}\}$. □

Lemma 4. *In **Algorithm C**, for every $i \geq 0$,*

(a) *for every accepting run ρ of $\mathcal{BP}$ from $\langle p, \omega\rangle$ with any $\langle p, \omega\rangle \in P \times \Gamma^*$, there exists a path $p^i \xrightarrow{\omega}_\delta \{q_f\}$ in A_i and for every decomposition $p^i \xrightarrow{u}_\delta Q \xrightarrow{v}_\delta \{q_f\}$ of the path $p^i \xrightarrow{\omega}_\delta \{q_f\}$, if $Q \neq \{q_f\}$, then for all q^i or q^{i-1} in $Q \setminus \{q_f\}$, some path of the run ρ will reach the configuration $\langle q, v\rangle$;*
(b) *$Y_{\mathcal{BP}} \subseteq L(A_i)$, after the substitution at Line 11.*

Proof. Let us apply induction on i.

Basis: $i = 0$. The statement (a) directly follows from the fact that for every configuration $\langle p, \omega\rangle \in P \times \Gamma^*$, there exists a path $p^0 \xrightarrow{\omega}_\delta \{q_f\}$ in A_0 and for every decomposition $p^0 \xrightarrow{u}_\delta Q \xrightarrow{v}_\delta \{q_f\}$ of the path $p^0 \xrightarrow{\omega}_\delta \{q_f\}$, $Q = \{q_f\}$. The statement (b) follows from the fact that $Y_{\mathcal{BP}} \subseteq P \times \Gamma^* = L(A_0)$.

Step: $i \geq 1$. For the statement (a). Let $H(\rho)$ be the maximum over the numbers of steps required by the paths of ρ (from the root) to reach some configuration in $F \times \Gamma^*$. Note that since ρ is an accepting run of $\mathcal{BP}$, $H(\rho)$ is well-defined and finite. We apply a nested induction on $H(\rho)$.

- **Basis**: $H(\rho) = 0$. Since the root of ρ is $\langle p, \omega\rangle$, we obtain that $\langle p, \omega\rangle \in F \times \Gamma^*$. By the transition rules added at Line 4 during the ith iteration, we get that $p^i \xrightarrow{\epsilon} \{p^{i-1}\}$ is a transition rule of A_i. Thus, by applying the induction hypothesis on i, the result immediately follows.
- **Step**: $H(\rho) \geq 1$. Let $\rho^1, ..., \rho^m$ be the subtrees of ρ rooted by the children of the root $\langle p, \omega\rangle$, respectively. Then, there exists a transition rule $\langle p, \gamma\rangle \hookrightarrow \{\langle p_1, \omega_1\rangle, ..., \langle p_m, \omega_m\rangle\}$ such that the roots of $\rho^1, ..., \rho^m$ are $\langle p, \omega_1\omega'\rangle, ..., \langle p, \omega_m\omega'\rangle$ and $\omega = \gamma\omega'$. Now, $\rho^1, ..., \rho^m$ are accepting runs of $\mathcal{BP}$ from the configurations $\langle p, \omega_1\omega'\rangle, ..., \langle p, \omega_m\omega'\rangle$. Since $H(\rho) \geq 1$, we get that $\langle p, \omega\rangle \notin F \times \Gamma^*$. Hence, $H(\rho^j) < H(\rho)$ for every $j : 1 \leq j \leq m$. Thus, we can apply the nested induction hypothesis on $H(\rho^j)$ for every $j : 1 \leq j \leq m$ for statement (a), we get that there exists a path $p^i_j \xrightarrow{\omega_j\omega'}_\delta \{q_f\}$ in A_i and for every decomposition $p^i_j \xrightarrow{u}_\delta Q \xrightarrow{v}_\delta \{q_f\}$ of the path $p^i_j \xrightarrow{\omega_j\omega'}_\delta \{q_f\}$, if $Q \neq \{q_f\}$, then for all $q^k \in Q \setminus \{q_f\}$ with $k \in \{i, i-1\}$, some path of the run ρ will reach the configuration $\langle q, v\rangle$.

  In particular, for every $j : 1 \leq j \leq m$, there exists $p^i_j \xrightarrow{\omega_j}_\delta Q_j$ in A_i (which corresponds to a prefix of $p^i_j \xrightarrow{\omega_j\omega'}_\delta \{q_f\}$). By applying the saturation procedure, we get that there exists a transition $p^i \xrightarrow{\gamma} \bigcup_{j=1}^m Q_j$ in A_i. Hence, there exists a path $p^i \xrightarrow{\gamma} \bigcup_{j=1}^m Q_j \xrightarrow{\omega'}_\delta \{q_f\}$ in A_i and for every decomposition $p^i \xrightarrow{u}_\delta Q \xrightarrow{v}_\delta \{q_f\}$ of the path $p^i \xrightarrow{\gamma\omega'}_\delta \{q_f\}$, if $Q \neq \{q_f\}$, then for all $q^k \in Q \setminus \{q_f\}$ with $k \in \{i, i-1\}$, some path of the run ρ will reach the configuration $\langle q, v\rangle$.

For the statement (b). Since $Y_{\mathcal{BP}} = Pre^+(Y_{\mathcal{BP}} \cap F \times \Gamma^*)$ and $Y_{\mathcal{BP}} \subseteq L(A_{i-1})$, we get that $Y_{\mathcal{BP}} \subseteq Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$. By Lemma 3, we get that just before the substitution at Line 10, A_i accepts $Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$. Thus, it is sufficient to show that for every $\langle p, \omega\rangle \in Y_{\mathcal{BP}}$, A_i accepts $\langle p, \omega\rangle$ after the substitution at Line 11. Let n be the number of transition rules substituted at Line 11. For all $m \le n$, let A_i^m be the automaton obtained by substituting m transition rules. We show that $Y_{\mathcal{BP}} \subseteq L(A_i^m)$ by induction on m.

- **Basis.** $m = 0$. We directly get that $Y_{\mathcal{BP}} \subseteq L(A_i^0)$.
- **Step.** $m \ge 1$. By applying the induction hypothesis: we get that $Y_{\mathcal{BP}} \subseteq L(A_i^{m-1})$. If $L(A_i^{m-1}) \subseteq L(A_i^m)$, the result follows from the fact that $Y_{\mathcal{BP}} \subseteq L(A_i^{m-1})$. Otherwise, if $L(A_i^{m-1}) \setminus L(A_i^m) \neq \emptyset$, let $\langle p, \omega\rangle \in L(A_i^{m-1}) \setminus L(A_i^m)$ be some configuration such that $|\omega|$ is the minimum of $\{|\omega'| \mid \langle p', \omega'\rangle \in L(A_i^{m-1}) \setminus L(A_i^m)\}$ s.t. $\langle p, \omega\rangle \in Y_{\mathcal{BP}}$. We prove by contradiction that $\langle p, \omega\rangle$ should not be in $Y_{\mathcal{BP}}$.
  For every path of the form $p^i \xrightarrow{\omega}_\delta \{q_f\}$ in A_i^{m-1}, there exist $u \in \Gamma^+, v \in \Gamma^*$ and $q \in P$ such that $\omega = uv$ and $p^i \xrightarrow{u}_\delta Q \cup \{q^{i-1}\} \xrightarrow{v}_\delta \{q_f\}$ in A_i^{m-1} and A_i^m does not have $\{q^i\} \xrightarrow{v}_\delta \{q_f\}$ (otherwise $\langle p, \omega\rangle \in L(A_i^m)$). By the statement (a), for each accepting run ρ of $\mathcal{BP}$ starting from $\langle p, \omega\rangle$, one path of this run ρ will reach such a configuration $\langle q, v\rangle$. It is sufficient to show that $\langle q, v\rangle \notin Y_{\mathcal{BP}}$.
  Now, let us show that $\langle q, v\rangle \notin Y_{\mathcal{BP}}$. If $\langle q, v\rangle \notin L(A_i^{m-1})$, by applying the induction hypothesis on m, we get that $\langle q, v\rangle \notin Y_{\mathcal{BP}}$.
  If $\langle q, v\rangle \in L(A_i^{m-1})$, then $\langle q, v\rangle \in L(A_i^{m-1}) \setminus L(A_i^m)$. If $\langle q, v\rangle \in Y_{\mathcal{BP}}$, then $|v| < |\omega|$ which contradicts the fact that $|\omega|$ is the minimum of $\{|\omega'| \mid \langle p', \omega'\rangle \in L(A_i^{m-1}) \setminus L(A_i^m)\}$ such that $\langle p, \omega\rangle \in Y_{\mathcal{BP}}$. Thus, we obtain that $\langle q, v\rangle \notin Y_{\mathcal{BP}}$. □

Now, we are ready to prove that Algorithm 1 always terminates and outputs $Y_{\mathcal{BP}}$.

Theorem 2. *Algorithm 1 always terminates and produces $Y_{\mathcal{BP}}$.*

Proof. We prove termination and correctness.

Termination: There are two loops in Algorithm 1, we need to prove that both loops terminate.

Loop$_2$: Suppose *loop*$_2$ is in the ith iteration of *loop*$_1$. Since only states of the form $p^i \in P \times \{i\}$ can be added into $\mathcal{A}$ at the ith iteration, we get that *Loop*$_2$ only add a finite number of transition rules at the ith iteration. This implies that for every $i \ge 1$, *Loop*$_2$ always terminates at the ith iteration.

loop$_1$: Now we consider the termination of *loop*$_1$. For every $i \ge 1$, Line 11 ensures that at the end of the ith iteration, each transition rule of $\mathcal{A}$ is of the form $p^j \xrightarrow{\gamma} S$ where $j \le i$ and $S \subseteq P \times \{j\} \cup \{q_f\}$. Hence, by Proposition 2(a), at the $(i+1)$th iteration with $i \ge 1$, either the termination condition at Line 12 of Algorithm 1 is satisfied or the number of transition rules is strictly smaller than in the ith iteration. Thus Algorithm 1 will always terminate.

Correctness: Let $n > 1$ be the fixpoint of Algorithm 1 such that for every $p \in P, \gamma \in \Gamma$, $S \subseteq P \times \{n+1\} \cup \{q_f\}$, $p^{n+1} \xrightarrow{\gamma} S \in \delta \Longleftrightarrow p^n \xrightarrow{\gamma} \pi^{-1}(S) \in \delta$. Then $L(A_n) = L(A_{n+1})$. We will prove that $L(A_n) = Y_{\mathcal{BP}}$.

If we use **Algorithm C**, by Lemma 2, Proposition 2(b) and the fact that $L(A_0) = P \times \Gamma^*$, we have that for all $i \ge n$

$$L(A_i) = L(A_{i-1}) = \ldots = L(A_n) \subseteq L(A_{n-1}) \subseteq \ldots \subseteq L(A_0) \tag{1}$$

($\subseteq$) We show that $L(A_n) \subseteq Y_{\mathcal{BP}}$. From (1), since $Y_{\mathcal{BP}} = \bigcap_{i \ge 0} X_i$ and $X_{i+1} = Pre^+(X_i \cap F \times \Gamma^*)$, it is sufficient to prove that $L(A_i) \subseteq X_i$ for every $i \ge 0$. We proceed by induction on i.

- **Basis.** $i = 0$. We directly get that $L(A_0) \subseteq X_0$.
- **Step.** $i \ge 1$. We will show that $L(A_i) \subseteq X_i$. By applying the induction hypothesis (induction on i), we get that $L(A_{i-1}) \subseteq X_{i-1}$. By the definition of $X_i = Pre^+(X_{i-1} \cap F \times \Gamma^*)$, we obtain that

$$Pre^+\big(L(A_{i-1}) \cap F \times \Gamma^*\big) \subseteq X_i \tag{2}$$

  By Lemma 3, A_i accepts $Pre^+(L(A_{i-1}) \cap F \times \Gamma^*)$ before Line 11 of the algorithm. By Proposition 2(b), Line 11 can only remove configurations from A_i, we obtain that

$$L(A_i) \subseteq Pre^+\big(L(A_{i-1}) \cap F \times \Gamma^*\big). \tag{3}$$

  From (2) and (3), we get that $L(A_i) \subseteq X_i$.

($\supseteq$) The converse inclusion directly follows from Lemma 4(b). □

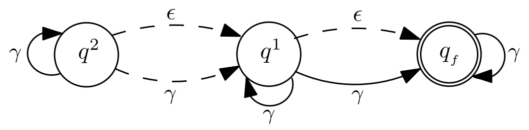

Fig. 2. The result automaton.

Thus, since $\mathcal{L}(\mathcal{BP}) = Y_{\mathcal{BP}}$, we get that:

Theorem 3. *Given an ABPDS $\mathcal{BP} = (P, \Gamma, \Delta, F)$, we can effectively compute an AMA $\mathcal{A}$ with $O(|P|)$ states and $O(|P| \cdot |\Gamma| \cdot 2^{|P|})$ transition rules that recognizes $\mathcal{L}(\mathcal{BP})$. This AMA can be computed in time $O(|P|^2 \cdot |\Delta| \cdot |\Gamma| \cdot 2^{5|P|})$.*

Proof. The correctness follows from Theorem 1 and Theorem 2.

Complexity: Given an ABPDS $\mathcal{P}$ with n control locations and transition rules Δ, an AMA A with m non-initial states, [24] provides a procedure that can implement the saturation procedure $loop_2$ to compute the Pre^* of $L(A)$ in $O((n+m) \cdot |\Delta| \cdot 2^{2n+2m})$ time. We integrated this efficient algorithm into our saturation procedure ($loop_2$). Let us compute the number n and m in our algorithm.

Thanks to Line 11 of Algorithm 1, in each iteration ith of $loop_1$ we only need to keep the states $(P \times \{i, i-1\}) \cup \{q_f\}$, in which $P \times \{i\}$ are initial states of the AMA A and $(P \times \{i-1\}) \cup \{q_f\}$ are non-initial states, we obtain that $n+m$ is at most $2|P|+1$. Thus, $loop_2$ can be done in $O(|P| \cdot |\Delta| \cdot 2^{4|P|})$ time. In Line 4 and Line 10, adding or removing ϵ-transition rules can be done in $|F|$ times. Since the number of transition rules of $\mathcal{A}$ is at most $|\Gamma| \cdot |P| \cdot 2^{2|P|+1}$ after Line 10 and at most $|\Gamma| \cdot |P| \cdot 2^{|P|+1}$ after Line 11, the number of times of substitution (Line 11) is at most $|\Gamma| \cdot |P| \cdot 2^{2|P|+1}$. The termination condition can be done in time $O(|\Gamma| \cdot |P| \cdot 2^{|P|+1})$. At each iteration of $loop_1$, the number of transition rules of $\mathcal{A}$ will be smaller and smaller until reaching a fixpoint. Thus, $loop_1$ can be done in $O(|P| \cdot |\Gamma| \cdot 2^{|P|+1})$ time.

Putting all these estimations together, the algorithm runs in $O(|P|^2 \cdot |\Delta| \cdot |\Gamma| \cdot 2^{5|P|})$ time. $\square$

Example. Let us illustrate our algorithm by an example. Consider an ABPDS $\mathcal{BP} = (\{q\}, \{\gamma\}, \Delta, \{q\})$, where $\Delta = \{\langle q, \gamma\rangle \hookrightarrow \langle q, \epsilon\rangle\}$. The automaton produced by Algorithm 1 is shown in Fig. 2. The dashed lines denote the transitions removed by Lines 10 and 11. In the first iteration, $t_1 = q^1 \xrightarrow{\epsilon} q_f$ is added by Line 4, the saturation procedure (Lines 5–9) adds two transitions $q^1 \xrightarrow{\gamma} q_f$ and $q^1 \xrightarrow{\gamma} q^1$. Then the transition t_1 is removed by Line 10. In the second iteration, $t_2 = q^2 \xrightarrow{\epsilon} q^1$ is added by Line 4. The saturation procedure adds the transitions $t_3 = q^2 \xrightarrow{\gamma} q^1$ and $q^2 \xrightarrow{\gamma} q^2$. Finally, t_2 is removed by Line 10 and t_3 is replaced by $q^2 \xrightarrow{\gamma} q^2$ (this transition already exists in the automaton). Now the termination condition is satisfied and the algorithm terminates. In this case, $\mathcal{BP}$ has no accepting run.

3.2.1. Efficient implementation of Algorithm 1

We show that we can improve the complexity of Algorithm 1 as follows:

Improvement 1. For every $q \in \mathcal{Q}$ and $\gamma \in \Gamma$, if $t_1 = q \xrightarrow{\gamma} Q_1$ and $t_2 = q \xrightarrow{\gamma} Q_2$ are two transitions in δ such that $Q_1 \subseteq Q_2$, then we can remove t_2. This means that if $\mathcal{A}$ contains two transitions $t_1 = p \xrightarrow{\gamma} \{q_1, q_2, q_3\}$ and $t_2 = p \xrightarrow{\gamma} \{q_1, q_2\}$, then we can remove t_1 without changing the language of $\mathcal{A}$. Indeed, if a path $q \xrightarrow{\omega}_\delta q_f$ uses the transition rule t_1, then there must be necessarily a path $q \xrightarrow{\omega}_\delta q_f$ that uses the transition rule t_2 instead of t_1.

Improvement 2. Each transition $q^i \xrightarrow{\gamma} R$ added by the saturation procedure will be substituted by $q^i \xrightarrow{\gamma} \pi^i(R)$ in Line 11. Transitions of the form $q^i \xrightarrow{\gamma} \{q_1^i, q_1^{i-1}\} \cup R$ and $q^i \xrightarrow{\gamma} \{q_1^{i-1}\} \cup R$ have the same substitution $q^i \xrightarrow{\gamma} \{q_1^i\} \cup \pi^i(R)$. We show that each transition $q^i \xrightarrow{\gamma} \{q_1^i, q_1^{i-1}\} \cup R$ can be replaced by $q^i \xrightarrow{\gamma} \{q_1^{i-1}\} \cup R$ in the saturation procedure (i.e., during $loop_2$). Moreover, we show that if both $t_1 = q^i \xrightarrow{\gamma} \{q_1^{i-1}, \ldots, q_n^{i-1}\} \cup R$ and $t_2 = q^i \xrightarrow{\gamma} \{q_1^i, \ldots, q_n^i\} \cup R$ exist during $loop_2$, then t_2 can be removed. This is due to the fact that they both have the same substitution rule.

More precisely, we can show that these two improvements are sound and complete.

Lemma 5. *Given an AMA $A = (\mathcal{Q}, \Gamma, \delta, I, \mathcal{Q}_f)$, for every $q \in \mathcal{Q}$, $\gamma \in \Gamma$, suppose δ contains two transition rules $t_1 = q \xrightarrow{\gamma} Q_1$ and $t_2 = q \xrightarrow{\gamma} Q_2$ such that $Q_1 \subseteq Q_2$. Let A' be the automaton obtained from A by removing t_2. Then $L(A') = L(A)$.*

Proof. W.l.o.g., we assume that $\mathcal{Q}_f = \{q_f\}$.

($\subseteq$) Since A' is obtained by eliminating the transition rule t_2, A' preserves the other transition rules of A. It is easy to see that $L(A') \subseteq L(A)$.

($\supseteq$) Let $\langle p, \omega \rangle \in L(A)$, we will show that $\langle p, \omega \rangle \in L(A')$. Since $\langle p, \omega \rangle \in L(A)$, if there is a path $p \xrightarrow{\omega}_\delta q_f$ which does not use the transition rule t_2, $\langle p, \omega \rangle \in L(A')$ holds.

If $p \xrightarrow{\omega}_\delta q_f$ uses the transition rule t_2, then there exist $u, v \in \Gamma^*$, $R_1 \subseteq \mathcal{Q}$ such that (1) $\omega = u\gamma v$, (2) $p \xrightarrow{u}_\delta R_1 \cup \{q\}$, (3) $R_1 \xrightarrow{\gamma v}_\delta q_f$, (4) $Q_2 \xrightarrow{v}_\delta q_f$. Since $Q_1 \subseteq Q_2$ and $Q_2 \xrightarrow{v}_\delta q_f$, we get that $Q_1 \xrightarrow{v}_\delta q_f$. Since $p \xrightarrow{u}_\delta R_1 \cup \{q\}$, $q \xrightarrow{\gamma} Q_1$, $Q_1 \xrightarrow{v}_\delta q_f$ and $R_1 \xrightarrow{\gamma v}_\delta q_f$, we obtain that $p \xrightarrow{u}_\delta R_1 \cup \{q\} \xrightarrow{\gamma v}_\delta q_f$ in A'. Since $\omega = u\gamma v$, we obtain that $\langle p, \omega \rangle \in L(A')$. □

Lemma 6. *During the ith iteration of loop$_1$ of Algorithm 1, (a) and (b) hold for every $i \geq 1$.*

(a) *For every $q, q_1 \in \mathcal{Q}$, $\gamma \in \Gamma$, $R \subseteq \mathcal{Q}$, a transition rule of the form $t_1 = q^i \xrightarrow{\gamma} \{q_1^i, q_1^{i-1}\} \cup R$ can be replaced by the transition rule $t_2 = q^i \xrightarrow{\gamma} \{q_1^i\} \cup R$ in the saturation procedure without changing the language of the automaton A_i.*

(b) *For every $q \in \mathcal{Q}$, $\gamma \in \Gamma$, $R \subseteq \mathcal{Q}$, if both transition rules $t_1 = q^i \xrightarrow{\gamma} \{q_1^{i-1}, ..., q_n^{i-1}\} \cup R$ and $t_2 = q^i \xrightarrow{\gamma} \{q_1^i, ..., q_n^i\} \cup R$ exist during the saturation procedure, we can remove t_2 without changing the language of A_i.*

Proof. (a) First we show that A_i does not lose any configuration by removing t_1, then we show that we do not introduce any new configuration by adding t_2.

- Suppose A_i accepts a configuration $\langle q, \gamma u \rangle$ by $q^i \xrightarrow{\gamma} \{q_1^i, q_1^{i-1}\} \cup R \xrightarrow{u}_\delta q_f$ for some $u \in \Gamma^*$, it is easy to see that $q^i \xrightarrow{\gamma} \{q_1^i\} \cup R \xrightarrow{u}_\delta q_f$. This implies that $\langle q, \gamma u \rangle$ is accepted by A_i after removing t_1.
- Suppose A_i accepts a configuration $\langle q, \gamma u \rangle$ by $q^i \xrightarrow{\gamma} \{q_1^i\} \cup R \xrightarrow{u}_\delta q_f$. To prove that A_i accepts the configuration $\langle q, \gamma u \rangle$ before adding t_2, it is sufficient to show that $q^i \xrightarrow{\gamma} \{q_1^i, q_1^{i-1}\} \cup R \xrightarrow{u}_\delta q_f$. The proof depends on whether $i = 1$ or not. If $i = 1$, then $q_1^{i-1} = q_f$. This implies that $q^i \xrightarrow{\gamma} \{q_1^i, q_1^{i-1}\} \cup R \xrightarrow{u}_\delta q_f$. If $i \neq 1$, then $i \geq 2$. Since $q_1^i \xrightarrow{u}_\delta q_f$, by Proposition 2, we obtain that $q_1^{i-1} \xrightarrow{u}_\delta q_f$. Thus, $q^i \xrightarrow{\gamma} \{q_1^i, q_1^{i-1}\} \cup R \xrightarrow{u}_\delta q_f$.

(b) Suppose A_i accepts a configuration $\langle q, \gamma u \rangle$ by $q^i \xrightarrow{\gamma} \{q_1^i, ..., q_n^i\} \cup R \xrightarrow{u}_\delta q_f$, we show that $q^i \xrightarrow{\gamma} \{q_1^{i-1}, ..., q_n^{i-1}\} \cup R \xrightarrow{u}_\delta q_f$. The proof depends on whether $i = 1$ or not. If $i = 1$, then $q_k^{i-1} = q_f$ for every $k : 1 \leq k \leq n$. This implies that $q^i \xrightarrow{\gamma} \{q_1^{i-1}, ..., q_n^{i-1}\} \cup R \xrightarrow{u}_\delta q_f$. If $i \neq 1$, then $i \geq 2$. Since $q_k^i \xrightarrow{u}_\delta q_f$ for every $k : 1 \leq k \leq n$, by Proposition 2, we obtain that $q_k^{i-1} \xrightarrow{u}_\delta q_f$ for every $k : 1 \leq k \leq n$. Thus, $q^i \xrightarrow{\gamma} \{q_1^{i-1}, ..., q_n^{i-1}\} \cup R \xrightarrow{u}_\delta q_f$. □

4. CTL model-checking for pushdown systems

We consider in this section "standard" CTL model checking for pushdown systems as considered in the literature, i.e., the case where whether an atomic proposition holds for a given configuration c or not depends only on the control state of c, not on its stack. Let $\mathcal{P} = (P, \Gamma, \Delta, \sharp)$ be a pushdown system, c_0 its initial configuration, AP a set of atomic propositions, φ a CTL formula, $f : AP \to 2^P$ a function that associates atomic propositions to sets of control states, and $\lambda_f : AP \to 2^{P \times \Gamma^*}$ a labeling function such that for every $a \in AP$, $\lambda_f(a) = \{\langle p, \omega \rangle \mid p \in f(a), \omega \in \Gamma^*\}$. We provide in this section an algorithm to determine whether $(\mathcal{P}, c_0) \models_{\lambda_f} \varphi$. We proceed as follows: Roughly speaking, we compute an Alternating Büchi PushDown System $\mathcal{BP}$ that recognizes the set of configurations c such that $(\mathcal{P}, c) \models_{\lambda_f} \varphi$. Then $(\mathcal{P}, c_0) \models_{\lambda_f} \varphi$ holds iff $c_0 \in \mathcal{L}(\mathcal{BP})$. This can be effectively checked due to Theorem 3 and Proposition 1.

Let $\mathcal{BP}_\varphi = (P', \Gamma, \Delta', F)$ be the ABPDS defined as follows: $P' = P \times cl(\varphi)$; $F = \{[p, a] \mid a \in cl(\varphi) \cap AP$ and $p \in f(a)\} \cup \{[p, \neg a] \mid \neg a \in cl(\varphi), a \in AP$ and $p \notin f(a)\} \cup P \times cl_R(\varphi)$, where $cl_R(\varphi)$ is the set of formulas of $cl(\varphi)$ of the form $E[\varphi_1 R \varphi_2]$ or $A[\varphi_1 R \varphi_2]$; and Δ' is the smallest set of transition rules such that for every control location $p \in P$, every subformula $\psi \in cl(\varphi)$, and every $\gamma \in \Gamma$, we have:

1. if $\psi = a$, $a \in AP$ and $p \in f(a)$; $\langle [p, \psi], \gamma \rangle \hookrightarrow \langle [p, \psi], \gamma \rangle \in \Delta'$,
2. if $\psi = \neg a$, $a \in AP$ and $p \notin f(a)$; $\langle [p, \psi], \gamma \rangle \hookrightarrow \langle [p, \psi], \gamma \rangle \in \Delta'$,
3. if $\psi = \psi_1 \wedge \psi_2$; $\langle [p, \psi], \gamma \rangle \hookrightarrow \langle [p, \psi_1], \gamma \rangle \wedge \langle [p, \psi_2], \gamma \rangle \in \Delta'$,
4. if $\psi = \psi_1 \vee \psi_2$; $\langle [p, \psi], \gamma \rangle \hookrightarrow \langle [p, \psi_1], \gamma \rangle \vee \langle [p, \psi_2], \gamma \rangle \in \Delta'$,
5. if $\psi = EX\psi_1$; $\langle [p, \psi], \gamma \rangle \hookrightarrow \bigvee_{\langle p, \gamma \rangle \hookrightarrow \langle p', \omega \rangle \in \Delta} \langle [p', \psi_1], \omega \rangle \in \Delta'$,
6. if $\psi = AX\psi_1$; $\langle [p, \psi], \gamma \rangle \hookrightarrow \bigwedge_{\langle p, \gamma \rangle \hookrightarrow \langle p', \omega \rangle \in \Delta} \langle [p', \psi_1], \omega \rangle \in \Delta'$,
7. if $\psi = E[\psi_1 U \psi_2]$; $\langle [p, \psi], \gamma \rangle \hookrightarrow \langle [p, \psi_2], \gamma \rangle \vee \bigvee_{\langle p, \gamma \rangle \hookrightarrow \langle p', \omega \rangle \in \Delta} (\langle [p, \psi_1], \gamma \rangle \wedge \langle [p', \psi], \omega \rangle) \in \Delta'$,
8. if $\psi = A[\psi_1 U \psi_2]$; $\langle [p, \psi], \gamma \rangle \hookrightarrow \langle [p, \psi_2], \gamma \rangle \vee \bigwedge_{\langle p, \gamma \rangle \hookrightarrow \langle p', \omega \rangle \in \Delta} (\langle [p, \psi_1], \gamma \rangle \wedge \langle [p', \psi], \omega \rangle) \in \Delta'$,
9. if $\psi = E[\psi_1 R \psi_2]$; $\langle [p, \psi], \gamma \rangle \hookrightarrow (\langle [p, \psi_1], \gamma \rangle \vee \bigvee_{\langle p, \gamma \rangle \hookrightarrow \langle p', \omega \rangle \in \Delta} \langle [p', \psi], \omega \rangle) \wedge \langle [p, \psi_2], \gamma \rangle \in \Delta'$,
10. if $\psi = A[\psi_1 R \psi_2]$; $\langle [p, \psi], \gamma \rangle \hookrightarrow (\langle [p, \psi_1], \gamma \rangle \vee \bigwedge_{\langle p, \gamma \rangle \hookrightarrow \langle p', \omega \rangle \in \Delta} \langle [p', \psi], \omega \rangle) \wedge \langle [p, \psi_2], \gamma \rangle \in \Delta'$.

The ABPDS $\mathcal{BP}_\varphi$ above can be seen as the "product" of $\mathcal{P}$ with the formula φ. Intuitively, $\mathcal{BP}_\varphi$ has an accepting run from $\langle [p,\psi],\omega\rangle$ if and only if the configuration $\langle p,\omega\rangle$ satisfies ψ. Let us explain the intuition behind the different items defining Δ'.

Let $\psi = a \in AP$. If $p \in f(a)$ then for every $\omega \in \Gamma^*$, $\langle p,\omega\rangle$ satisfies ψ. Thus, $\mathcal{BP}_\varphi$ should accept $\langle [p,a],\omega\rangle$, i.e., have an accepting run from $\langle [p,a],\omega\rangle$. This is ensured by Item 1 that adds a loop in $\langle [p,a],\omega\rangle$, and the fact that $[p,a] \in F$.

Let $\psi = \neg a$, where $a \in AP$. If $p \notin f(a)$ then for every $\omega \in \Gamma^*$, $\langle p,\omega\rangle$ satisfies ψ. Thus, $\mathcal{BP}_\varphi$ should accept $\langle [p,\neg a],\omega\rangle$, i.e., have an accepting run from $\langle [p,\neg a],\omega\rangle$. This is ensured by Item 2 and the fact that $[p,\neg a] \in F$.

Item 3 expresses that if $\psi = \psi_1 \wedge \psi_2$, then for every $\omega \in \Gamma^*$, $\mathcal{BP}_\varphi$ has an accepting run from $\langle [p,\psi_1 \wedge \psi_2],\omega\rangle$ iff $\mathcal{BP}_\varphi$ has an accepting run from $\langle [p,\psi_1],\omega\rangle$ and $\langle [p,\psi_2],\omega\rangle$; meaning that $\langle p,\omega\rangle$ satisfies ψ iff $\langle p,\omega\rangle$ satisfies ψ_1 and ψ_2. Item 4 is similar to Item 3.

Item 5 means that if $\psi = EX\psi_1$, then for every $\omega \in \Gamma^*$, $\langle p,\omega\rangle$ satisfies ψ iff there exists an immediate successor $\langle p',\omega'\rangle$ of $\langle p,\omega\rangle$ such that $\langle p',\omega'\rangle$ satisfies ψ_1. Thus, $\mathcal{BP}_\varphi$ should have an accepting run from $\langle [p,\psi],\omega\rangle$ iff it has an accepting run from $\langle [p',\psi_1],\omega'\rangle$. Similarly, item 6 states that if $\psi = AX\psi_1$, then for every $\omega \in \Gamma^*$, $\langle p,\omega\rangle$ satisfies ψ iff $\langle p',\omega'\rangle$ satisfies ψ_1 for every immediate successor $\langle p',\omega'\rangle$ of $\langle p,\omega\rangle$.

Item 7 expresses that if $\psi = E[\psi_1 U \psi_2]$, then for every $\omega \in \Gamma^*$, $\langle p,\omega\rangle$ satisfies ψ iff either it satisfies ψ_2, or it satisfies ψ_1 and there exists an immediate successor $\langle p',\omega'\rangle$ of $\langle p,\omega\rangle$ such that $\langle p',\omega'\rangle$ satisfies ψ. Item 8 is similar to Item 7.

Item 9 expresses that if $\psi = E[\psi_1 R \psi_2]$, then for every $\omega \in \Gamma^*$, $\langle p,\omega\rangle$ satisfies ψ iff it satisfies ψ_2, and either it satisfies also ψ_1, or it has a successor that satisfies ψ. This guarantees that ψ_2 holds either always, or until both ψ_1 and ψ_2 hold. The fact that the state $[p,\psi]$ is in F ensures that paths where ψ_2 always hold are accepting. The intuition behind Item 10 is analogous.

Formally, we can show that:

Theorem 4. *Let $\mathcal{P} = (P,\Gamma,\Delta,\sharp)$ be a PDS, $f : AP \longrightarrow 2^P$ a labeling function, φ a CTL formula, and $\langle p,\omega\rangle$ a configuration of $\mathcal{P}$. Let $\mathcal{BP}_\varphi$ be the ABPDS computed above. Then, $(\mathcal{P},\langle p,\omega\rangle) \models_{\lambda_f} \varphi$ iff $\mathcal{BP}_\varphi$ has an accepting run from the configuration $\langle [p,\varphi],\omega\rangle$.*

Proof. Theorem 4 is a special case of Theorem 5. We refer the reader to the proof of Theorem 5. $\square$

It follows from Theorems 3 and 4 that:

Corollary 1. *Given a PDS $\mathcal{P} = (P,\Gamma,\Delta,\sharp)$, a labeling function $f : P \longrightarrow 2^{AP}$, and a CTL formula φ, we can construct an AMA $\mathcal{A}$ in time $O(|P|^2 \cdot |\varphi|^3 \cdot (|P| \cdot |\Gamma| + |\Delta|) \cdot |\Gamma| \cdot 2^{5|P||\varphi|})$ such that for every configuration $\langle p,\omega\rangle$ of $\mathcal{P}$, $(\mathcal{P},\langle p,\omega\rangle) \models_{\lambda_f} \varphi$ iff the AMA $\mathcal{A}$ recognizes the configuration $\langle [p,\varphi],\omega\rangle$.*

The complexity follows from the complexity of Algorithm 1 and the fact that $\mathcal{BP}_\varphi$ has $O(|P||\varphi|)$ states and $O((|P||\Gamma| + |\Delta|)|\varphi|)$ transitions.

5. CTL model-checking for pushdown systems with regular valuations

So far, we considered the "standard" model-checking problem for CTL, where the validity of an atomic proposition in a configuration c depends only on the control state of c, not on the stack. In this section, we go further and consider an extension where the set of configurations in which an atomic proposition holds is a regular set of configurations.

Let $\mathcal{P} = (P,\Gamma,\Delta,\sharp)$ be a pushdown system, c_0 its initial configuration, AP a set of atomic propositions, φ a CTL formula, and $\lambda : AP \to 2^{P\times\Gamma^*}$ a labeling function such that for every $a \in AP$, $\lambda(a)$ is a regular set of configurations. We say that λ is a regular labeling. We give in this section an algorithm that checks whether $(\mathcal{P},c_0) \models_\lambda \varphi$. We proceed as in Section 4. Roughly speaking, we compute an ABPDS $\mathcal{BP}'_\varphi$ such that $\mathcal{BP}'_\varphi$ recognizes a configuration c iff $(\mathcal{P},c) \models_\lambda \varphi$. Then $(\mathcal{P},c_0)$ satisfies φ iff c_0 is accepted by $\mathcal{BP}'_\varphi$. As in Section 4, this can be checked using Theorem 3 and Proposition 1.

For every $a \in AP$, since $\lambda(a)$ is a regular set of configurations, let $M_a = (Q_a,\Gamma,\delta_a,I_a,F_a)$ be a multi-automaton such that $L(M_a) = \lambda(a)$, and $M_{\neg a} = (Q_{\neg a},\Gamma,\delta_{\neg a},I_{\neg a},F_{\neg a})$ such that $L(M_{\neg a}) = P \times \Gamma^* \setminus \lambda(a)$ be a multi-automaton that recognizes the complement of $\lambda(a)$, i.e., the set of configurations where a does not hold. Since for every $a \in AP$ and every control state $p \in P$, p may be an initial state in both Q_a and $Q_{\neg a}$, to distinguish between all these initial states, for every $a \in AP$, we will denote in the following the initial state corresponding to p in Q_a (resp. in $Q_{\neg a}$) by p_a (resp. $p_{\neg a}$).

Let $\mathcal{BP}'_\varphi = (P'',\Gamma,\Delta'',F')$ be the ABPDS defined as follows[2]: $P'' = P \times cl(\varphi) \cup \bigcup_{a\in AP^+(\varphi)} Q_a \cup \bigcup_{a\in AP^-(\varphi)} Q_{\neg a}$; $F' = P \times cl_R(\varphi) \cup \bigcup_{a\in AP^+(\varphi)} F_a \cup \bigcup_{a\in AP^-(\varphi)} F_{\neg a}$; and Δ'' is the smallest set of transition rules such that for every control location $p \in P$, every subformula $\psi \in cl(\varphi)$, and every $\gamma \in \Gamma$, we have:

1. if $\psi = a$, $a \in AP$; $\langle [p,\psi],\gamma\rangle \hookrightarrow \langle p_a,\gamma\rangle \in \Delta''$,
2. if $\psi = \neg a$, $a \in AP$; $\langle [p,\psi],\gamma\rangle \hookrightarrow \langle p_{\neg a},\gamma\rangle \in \Delta''$,

[2] $AP^+(\varphi)$ and $AP^-(\varphi)$ are as defined in Section 2.1.

3. if $\psi = \psi_1 \wedge \psi_2$; $\langle [p,\psi],\gamma\rangle \hookrightarrow \langle [p,\psi_1],\gamma\rangle \wedge \langle [p,\psi_2],\gamma\rangle \in \Delta''$,
4. if $\psi = \psi_1 \vee \psi_2$; $\langle [p,\psi],\gamma\rangle \hookrightarrow \langle [p,\psi_1],\gamma\rangle \vee \langle [p,\psi_2],\gamma\rangle \in \Delta''$,
5. if $\psi = EX\psi_1$; $\langle [p,\psi],\gamma\rangle \hookrightarrow \bigvee_{\langle p,\gamma\rangle \hookrightarrow \langle p',\omega\rangle \in \Delta} \langle [p',\psi_1],\omega\rangle \in \Delta''$,
6. if $\psi = AX\psi_1$; $\langle [p,\psi],\gamma\rangle \hookrightarrow \bigwedge_{\langle p,\gamma\rangle \hookrightarrow \langle p',\omega\rangle \in \Delta} \langle [p',\psi_1],\omega\rangle \in \Delta''$,
7. if $\psi = E[\psi_1 U \psi_2]$; $\langle [p,\psi],\gamma\rangle \hookrightarrow \langle [p,\psi_2],\gamma\rangle \vee \bigvee_{\langle p,\gamma\rangle \hookrightarrow \langle p',\omega\rangle \in \Delta} (\langle [p,\psi_1],\gamma\rangle \wedge \langle [p',\psi],\omega\rangle) \in \Delta''$,
8. if $\psi = A[\psi_1 U \psi_2]$; $\langle [p,\psi],\gamma\rangle \hookrightarrow \langle [p,\psi_2],\gamma\rangle \vee \bigwedge_{\langle p,\gamma\rangle \hookrightarrow \langle p',\omega\rangle \in \Delta} (\langle [p,\psi_1],\gamma\rangle \wedge \langle [p',\psi],\omega\rangle) \in \Delta''$,
9. if $\psi = E[\psi_1 R \psi_2]$; $\langle [p,\psi],\gamma\rangle \hookrightarrow (\langle [p,\psi_1],\gamma\rangle \vee \bigvee_{\langle p,\gamma\rangle \hookrightarrow \langle p',\omega\rangle \in \Delta} \langle [p',\psi],\omega\rangle) \wedge \langle [p,\psi_2],\gamma\rangle \in \Delta''$,
10. if $\psi = A[\psi_1 R \psi_2]$; $\langle [p,\psi],\gamma\rangle \hookrightarrow (\langle [p,\psi_1],\gamma\rangle \vee \bigwedge_{\langle p,\gamma\rangle \hookrightarrow \langle p',\omega\rangle \in \Delta} \langle [p',\psi],\omega\rangle) \wedge \langle [p,\psi_2],\gamma\rangle \in \Delta''$.

Moreover:

11. for every transition $q_1 \xrightarrow{\gamma} q_2$ in $(\bigcup_{a\in AP^+(\varphi)} \delta_a) \cup (\bigcup_{a\in AP^-(\varphi)} \delta_{\neg a})$; $\langle q_1,\gamma\rangle \hookrightarrow \langle q_2,\epsilon\rangle \in \Delta''$,
12. for every $q \in (\bigcup_{a\in AP^+(\varphi)} F_a) \cup (\bigcup_{a\in AP^-(\varphi)} F_{\neg a})$; $\langle q,\sharp\rangle \hookrightarrow \langle q,\sharp\rangle \in \Delta''$.

The ABPDS $\mathcal{BP}'_\varphi$ has an accepting run from $\langle [p,\psi],\omega\rangle$ if and only if the configuration $\langle p,\omega\rangle$ satisfies ψ according to the regular labellings M_a's. Let us explain the intuition behind the rules above. Let $p \in P$, $\psi = a \in AP$, and $\omega \in \Gamma^*$. The ABPDS $\mathcal{BP}'_\varphi$ should accept $\langle [p,a],\omega\rangle$, iff $\langle p,\omega\rangle \in L(M_a)$. To check this, $\mathcal{BP}'_\varphi$ goes to state p_a, the initial state corresponding to p in M_a (Item 1); and then, from this state, it checks whether ω is accepted by M_a. This is ensured by Items 11 and 12. Item 11 allows $\mathcal{BP}'_\varphi$ to mimic a run of M_a on ω: if $\mathcal{BP}'_\varphi$ is in state q_1 with γ on top of its stack, and if $q_1 \xrightarrow{\gamma} q_2$ is a rule in δ_a, then $\mathcal{BP}'_\varphi$ moves to state q_2 while popping γ from the stack. Popping γ allows to check the rest of the word. The configuration is accepted if the run (with label ω) in M_a reaches a final state, i.e., if $\mathcal{BP}'_\varphi$ reaches a state $q \in F_a$ with an empty stack, i.e., a stack containing only the bottom stack symbol $\sharp$. Thus, F_a is in F'. Since all the accepting runs of $\mathcal{BP}'_\varphi$ are infinite, we add a loop on every configuration in control state $q \in F_a$ and having $\sharp$ as content of the stack (Item 12).

The intuition behind Item 2 is similar. This item applies to ψ of the from $\neg a$. Items 3–10 are similar to Items 3–10 in the construction underlying Theorem 4. We get that:

Theorem 5. *$(\mathcal{P}, \langle p,\omega\rangle) \models_\lambda \varphi$ iff $\mathcal{BP}'_\varphi$ has an accepting run from the configuration $\langle [p,\varphi],\omega\rangle$.*

Proof. ($\Longrightarrow$) Suppose $(\mathcal{P}, \langle p,\omega\rangle) \models_\lambda \psi$, we show that $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle [p,\psi],\omega\rangle$ by induction on the structure of ψ.

Case $\psi = a$: Since $(\mathcal{P}, \langle p,\omega\rangle) \models_\lambda \psi$, then $\langle p,\omega\rangle \in \lambda(a)$. By the definition of M_a, M_a has an accepting run from the initial state, $p_a \xrightarrow{\omega}_{\delta_a} f$ where $f \in F_a$. We will prove that $\mathcal{BP}'_\psi$ has an accepting run from $\langle p_a,\omega\rangle$ by induction on $m = |\omega|$ (the length of ω) which is greater than 0. Note that the bottom of the stack is $\sharp$.

- **Basis.** $m = 1$ (note that $\sharp$ will never be popped), hence $\omega = \sharp$. Then $p_a \xrightarrow{\sharp} f$. We get that $\langle p_a,\sharp\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle f,\sharp\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle f,\sharp\rangle$. Since f is an accepting control location, $\mathcal{BP}'_\psi$ has an accepting run from $\langle p_a,\sharp\rangle$.
- **Step.** $m \geq 2$. Then there exist $\gamma \in \Gamma, u \in \Gamma^*, q \in Q_a$ such that $\omega = \gamma u$ and $p_a \xrightarrow{\gamma} q \xrightarrow{u}_{\delta_a} f$ in M_a. By applying the induction hypothesis (induction on m) to $q \xrightarrow{u}_{\delta_a} f$, $\mathcal{BP}'_\psi$ has an accepting run from $\langle q,u\rangle$. Since $\langle p_a,\gamma u\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle q,u\rangle$, $\mathcal{BP}'_\psi$ has an accepting run from $\langle p_a,\omega\rangle$.

Since $\langle [p,a],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle p_a,\omega\rangle$, we get that $\mathcal{BP}'_\psi$ has an accepting run from $\langle [p,a],\omega\rangle$.

Case $\psi = \neg a$: Since $(\mathcal{P}, \langle p,\omega\rangle) \models_\lambda \psi$, then $\langle p,\omega\rangle \notin \lambda(a)$. By the definition of $M_{\neg a}$, $M_{\neg a}$ has an accepting path $p_{\neg a} \xrightarrow{\omega}_{\delta_{\neg a}} f$ where $f \in F_{\neg a}$. We will prove that $\mathcal{BP}'_\psi$ has an accepting run from $\langle p_{\neg a},\omega\rangle$ by induction on $m = |\omega|$ (the length of ω).

- **Basis.** $m = 1$. Then $p_{\neg a} \xrightarrow{\sharp}_{\delta_{\neg a}} f$. Since we have $\langle p_{\neg a},\sharp\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle f,\sharp\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle f,\sharp\rangle$, $\mathcal{BP}'_\psi$ has an accepting run from $\langle f,\sharp\rangle$.
- **Step.** $m \geq 2$. Then there exist $\gamma \in \Gamma, u \in \Gamma^*, q \in Q_{\neg a}$ such that $\omega = \gamma u$ and $p_{\neg a} \xrightarrow{\gamma} q \xrightarrow{u}_{\delta_{\neg a}} f$ in $M_{\neg a}$. By applying the induction hypothesis (induction on m), $\mathcal{BP}'_\psi$ has an accepting run from $\langle q,u\rangle$. Since $\langle p_{\neg a},\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle q,u\rangle$, we obtain that $\mathcal{BP}'_\psi$ has an accepting run from $\langle p_{\neg a},\omega\rangle$.

Since $\langle [p,\psi],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle p_{\neg a},\omega\rangle$, $\mathcal{BP}'_\psi$ has an accepting run from $\langle [p,\psi],\omega\rangle$.

Case $\psi = \psi_1 \wedge \psi_2$: Since $(\mathcal{P}, \langle p,\omega\rangle) \models_\lambda \psi$, we get that $(\mathcal{P}, \langle p,\omega\rangle) \models_\lambda \psi_1$ and $(\mathcal{P}, \langle p,\omega\rangle) \models_\lambda \psi_2$. By applying the induction hypothesis, $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle [p,\psi_1],\omega\rangle$, and $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p,\psi_2],\omega\rangle$. Since $\langle[p,\psi],\gamma\rangle \hookrightarrow \langle[p,\psi_1],\gamma\rangle \wedge \langle[p,\psi_2],\gamma\rangle$, we get that $\langle[p,\psi],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle[p,\psi_1],\omega\rangle, \langle[p,\psi_2],\omega\rangle\}$. So $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p,\psi],\omega\rangle$.

Case $\psi = \psi_1 \vee \psi_2$: Since $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$, we get that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi_1$ or $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi_2$. By applying the induction hypothesis, $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p,\psi_1],\omega\rangle$ or $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p,\psi_2],\omega\rangle$. Since $\langle[p,\psi],\gamma\rangle \hookrightarrow \langle[p,\psi_1],\gamma\rangle \vee \langle[p,\psi_2],\gamma\rangle$, we get that $\langle[p,\psi],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle[p,\psi_1],\omega\rangle\}$ and $\langle[p,\psi],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle[p,\psi_2],\omega\rangle\}$. So $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p,\psi],\omega\rangle$.

Case $\psi = EX\psi_1$: Since $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$, then there exists an immediate successor $\langle p',\omega'\rangle$ of $\langle p,\omega\rangle$, such that $(\mathcal{P},\langle p',\omega'\rangle) \models_\lambda \psi_1$. By applying the induction hypothesis, $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p',\psi_1],\omega'\rangle$.

Since $\langle p',\omega'\rangle$ is an immediate successor of $\langle p,\omega\rangle$, we obtain that $\langle[p,\psi],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle[p',\psi_1],\omega'\rangle$. Hence $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p,\psi],\omega\rangle$.

Case $\psi = AX\psi_1$ is similar to the case $\psi = EX\psi_1$.

Case $\psi = E[\psi_1 U \psi_2]$: Since $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda E[\psi_1 U\psi_2]$, then there exists a path $\langle p_0,\omega_0\rangle, \langle p_1,\omega_1\rangle, \langle p_2,\omega_2\rangle\ldots$ from $\langle p,\omega\rangle$ such that there exists $i \geq 0$, $(\mathcal{P},\langle p_i,\omega_i\rangle) \models_\lambda \psi_2$ and for every $0 \leq j < i$, $(\mathcal{P},\langle p_j,\omega_j\rangle) \models_\lambda \psi_1$. Thus, by applying the induction hypothesis, we obtain that $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p_i,\psi_2],\omega_i\rangle$ and for every $j: 0 \leq j < i$, $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p_j,\psi_1],\omega_j\rangle$. Since $\langle[p_i,\psi],\gamma\rangle \hookrightarrow \langle[p_i,\psi_2],\gamma\rangle \vee \bigvee_{\langle p_i,r\rangle\hookrightarrow\langle p',\omega\rangle}(\langle[p_i,\psi_1],\gamma\rangle \wedge \langle[p',\psi],\omega\rangle)$, we get that $\langle[p_i,\psi],\omega_i\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle[p_i,\psi_2],\omega_i\rangle$, so $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p_i,\psi],\omega_i\rangle$. If $i=0$, then $\langle[p,\psi],\omega\rangle = \langle[p_i,\psi],\omega_i\rangle$, and $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p,\psi],\omega\rangle$. Otherwise if $i>0$, we show that $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p_j,\psi],\omega_j\rangle$ by induction on $l = i-j$. (Note that $\langle[p_0,\psi],\omega_0\rangle = \langle[p,\psi],\omega\rangle$.)

- **Basis.** $l=1$. $\langle p_i,\omega_i\rangle$ is an immediate successor of $\langle p_j,\omega_j\rangle$. Since $\langle[p_j,\psi],\omega_j\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle[p_j,\psi_1],\omega_j\rangle, \langle[p_i,\psi],\omega_i\rangle\}$, $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p_j,\psi],\omega_j\rangle$.
- **Step.** $l>1$. $\langle p_{j+1},\omega_{j+1}\rangle$ is an immediate successor of $\langle p_j,\omega_j\rangle$, then $\langle[p_j,\psi],\omega_j\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle[p_j,\psi_1],\omega_j\rangle, \langle[p_{j+1},\psi],\omega_{j+1}\rangle\}$. By applying the induction hypothesis, $\mathcal{BP}'_\psi$ has an accepting run both from $\langle[p_{j+1},\psi],\omega_{j+1}\rangle$ and $\langle[p_j,\psi_1],\omega_j\rangle$. Thus, $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p_j,\psi],\omega_j\rangle$.

Case $\psi = A[\psi_1 U\psi_2]$ is similar to the case $\psi = E[\psi_1 U\psi_2]$.

Case $\psi = E[\psi_1 R\psi_2]$: Since $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda E[\psi_1 R\psi_2]$, by the semantics of CTL, $\mathcal{P}$ has a path $\langle p_0,\omega_0\rangle, \langle p_1\omega_1\rangle, \langle p_2,\omega_2\rangle\ldots$ from $\langle p,\omega\rangle$ such that:

1. either for every $i \geq 0$ $(\mathcal{P},\langle p_i,\omega_i\rangle) \models_\lambda \psi_2$,
2. or there exists $i \geq 0$ such that $(\mathcal{P},\langle p_i,\omega_i\rangle) \models_\lambda \psi_1$ and for every $j: 0 \leq j \leq i$ $(\mathcal{P},\langle p_i,\omega_i\rangle) \models_\lambda \psi_2$

First let us consider Item 2, it can be proved that $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p,\psi],\omega\rangle$ by applying the induction on $i-j$ similar to the case $\psi = E[\psi_1 U\psi_2]$.

Let's consider Item 1, we will show that $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p,\psi],\omega\rangle$. Let us construct an accepting run ρ of $\mathcal{BP}'_\psi$ from $\langle[p,\psi],\omega\rangle$ as follows. Note that $\langle[p,\psi],\omega\rangle = \langle[p_0,\psi],\omega_0\rangle$.

Let $\langle[p_0,\psi],\omega_0\rangle$ be the root of ρ. For every $k \geq 0$, $\langle[p_k,\psi],\omega_k\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle[p_k,\psi_2],\omega_k\rangle, \langle[p_{k+1},\psi],\omega_{k+1}\rangle\}$, we can let $\langle[p_{k+1},\psi],\omega_{k+1}\rangle$ and $\langle[p_k,\psi_2],\omega_k\rangle$ be the children of $\langle[p_k,\psi],\omega_k\rangle$. By applying the induction hypothesis to $(\mathcal{P},\langle p_k,\omega_k\rangle) \models_\lambda \psi_2$, we obtain that $\mathcal{BP}'_\psi$ has an accepting run ρ_k from $\langle[p_k,\psi_2],\omega_k\rangle$. We replace the child $\langle[p_k,\psi_2],\omega_k\rangle$ of $\langle[p_k,\psi],\omega_k\rangle$ in ρ by the run ρ_k. By the above construction, we obtain an infinite run ρ of $\mathcal{BP}'_\psi$ such that ρ has an infinite path $\langle[p_0,\psi],\omega_0\rangle, \langle[p_1,\psi],\omega_1\rangle, \ldots$ and all the other paths infinitely often visit some accepting control locations. Since for every $k \geq 0$, $[p_k,\psi] \in F'$, we obtain that each path of ρ infinitely often visits some accepting control locations, i.e., $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p,\psi],\omega\rangle$.

Case $\psi = A[\psi_1 R\psi_2]$: it can be proved as for the case $\psi = E[\psi_1 R\psi_2]$.

($\Longleftarrow$) Suppose $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle[p,\psi],\omega\rangle$, we show that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$ by induction on the structure of ψ.

Case $\psi = a$: then, $\langle[p,\psi],\gamma\rangle \hookrightarrow \langle p_a,\gamma\rangle$ for every $\gamma \in \Gamma$, $\langle q_1,\gamma\rangle \hookrightarrow \langle q_2,\epsilon\rangle$ for every $q_1 \xrightarrow{\gamma} q_2$ in δ_a and $\langle f,\sharp\rangle \hookrightarrow \langle f,\sharp\rangle$ for every $f \in F_a$. Since $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p,\psi],\omega\rangle$, there exists a state $f \in F_a$ such that $\langle[p,a],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle p_a,\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle f,\sharp\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle f,\sharp\rangle$. Thus M_a has a corresponding path $p_a \xrightarrow{\omega}_{\delta_a} f$. This implies that $\langle p,\omega\rangle \in L(M_a)$. Thus, $\langle p,\omega\rangle \in \lambda(a)$. We obtain that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$.

Case $\psi = \neg a$: then $\langle[p,\neg a],\gamma\rangle \hookrightarrow \langle p_{\neg a},\gamma\rangle$ for every $\gamma \in \Gamma$, $\langle q_1,\gamma\rangle \hookrightarrow \langle q_2,\epsilon\rangle$, for every $q_1 \xrightarrow{\gamma} q_2$ in $\delta_{\neg a}$ and $\langle f,\sharp\rangle \hookrightarrow \langle f,\sharp\rangle$, for every $f \in F_{\neg a}$. Since $\mathcal{BP}'_\psi$ has an accepting run from $\langle[p,\neg a],\omega\rangle$, there exists a state $f \in F_{\neg a}$ such that

$$\big\langle[p,\neg a],\omega\big\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle p_{\neg a},\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle f,\sharp\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \langle f,\sharp\rangle.$$

Then $M_{\neg a}$ has a corresponding path, $p_{\neg a} \xrightarrow{\omega}_{\delta_{\neg a}} f$, which implies that $\langle p,\omega\rangle \in L(M_{\neg a})$. Thus, $\langle p,\omega\rangle \notin \lambda(a)$. We obtain that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$.

Case $\psi = \psi_1 \wedge \psi_2$: then $\langle [p,\psi],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle [p,\psi_1],\omega\rangle, \langle [p,\psi_2],\omega\rangle\}$. So $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle [p,\psi_1],\omega\rangle$ and $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle [p,\psi_2],\omega\rangle$. By applying the induction hypothesis, we get that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi_1$ and $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi_2$. Thus, we get that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$.

Case $\psi = \psi_1 \vee \psi_2$: then we get that either $\langle [p,\psi],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle [p,\psi_1],\omega\rangle\}$ or $\langle [p,\psi],\omega\rangle \Longrightarrow_{\mathcal{BP}'_\psi} \{\langle [p,\psi_2],\omega\rangle\}$, so $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle [p,\psi_1],\omega\rangle$ or $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle [p,\psi_2],\omega\rangle$. By applying the induction hypothesis, we get that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi_1$ or $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi_2$. This implies that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$.

Case $\psi = EX\psi_1$ is similar to the case $\psi = AX\psi_1$.

Case $\psi = AX\psi_1$: Suppose $\{\langle [p_1,\psi_1],\omega_1\rangle, ..., \langle [p_n,\psi_1],\omega_n\rangle\}$ is the immediate successor of $\langle [p,\psi],\omega\rangle$ in the accepting run. Then $\mathcal{BP}'_\psi$ has an accepting run from $\langle [p_i,\psi_1],\omega_i\rangle$, for each $i : 1 \le i \le n$. By applying the induction hypothesis, we get that $(\mathcal{P},\langle p_i,\omega_i\rangle) \models_\lambda \psi_1$, for each $i : 1 \le i \le n$. By the construction, the immediate successors in $\mathcal{P}$ of $\langle p,\omega\rangle$ are $\langle p_1,\omega_1\rangle, ..., \langle p_n,\omega_n\rangle$. Thus, we obtain that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$.

Case $\psi = E[\psi_1 U \psi_2]$: Let ρ be the accepting run from $\langle [p,\psi],\omega\rangle$. By the construction, each configuration $\langle [p_i,\psi],\omega_i\rangle$ in ρ has either two children $\langle [p_i,\psi_1],\omega_i\rangle$ and $\langle [p_{i+1},\psi],\omega_{i+1}\rangle$ or has only one child $\langle [p_i,\psi_2],\omega_i\rangle$. Since ρ is an accepting run, there exists a configuration $\langle [p_n,\psi],\omega_n\rangle$ in ρ such that $\langle [p_n,\psi],\omega_n\rangle$ has only one child $\langle [p_n,\psi_2],\omega_n\rangle$. In particular, there is a path of ρ of the form $\langle [p_0,\psi],\omega_0\rangle, ..., \langle [p_n,\psi],\omega_n\rangle, ...$ with $\langle [p_0,\psi],\omega_0\rangle = \langle [p,\psi],\omega\rangle$, then $\mathcal{BP}'_\psi$ has an accepting run from $\langle [p_i,\psi_1],\omega_i\rangle$ for every $i : 0 \le i < n$, and $\mathcal{BP}'_\psi$ has an accepting run from $\langle [p_n,\psi_2],\omega_n\rangle$. By applying the induction hypothesis, we get that $(\mathcal{P},\langle p_n,\omega_n\rangle) \models_\lambda \psi_2$ and $(\mathcal{P},\langle p_i,\omega_i\rangle) \models_\lambda \psi_1$, for every $i : 0 \le i < n$. Since $\langle p,\omega\rangle, ...\langle p_n,\omega_n\rangle...$ is a run of $\mathcal{P}$, we get that $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$.

Case $\psi = A[\psi_1 U \psi_2]$: This case is similar to the case $\psi = E[\psi_1 U \psi_2]$.

Case $\psi = E[\psi_1 R \psi_2]$: Let ρ be an accepting run from $\langle [p,\psi],\omega\rangle$, then each configuration $\langle [p_i,\psi],\omega_i\rangle$ in ρ has two children (1) $\langle [p_i,\psi_2],\omega_i\rangle$ and $\langle [p_{i+1},\psi],\omega_{i+1}\rangle$, or (2) $\langle [p_i,\psi_1],\omega_i\rangle$ and $\langle [p_i,\psi_2],\omega_i\rangle$. Thus, there are two cases.

- First we consider case 1 where each configuration $\langle [p_i,\psi],\omega_i\rangle$ in ρ has two children $\langle [p_i,\psi_2],\omega_i\rangle$ and $\langle [p_{i+1},\psi],\omega_{i+1}\rangle$. Hence, there is an infinite path of ρ of the form $\langle [p_0,\psi],\omega_0\rangle, ..., \langle [p_{i+1},\psi],\omega_{i+1}\rangle, ...$, with $\langle [p_0,\psi],\omega_0\rangle = \langle [p,\psi],\omega\rangle$, and $\mathcal{BP}'_\psi$ has an accepting run from the configuration $\langle [p_i,\psi_2],\omega_i\rangle$ for every $i \ge 0$. By applying the induction hypothesis, we get that $(\mathcal{P},\langle p_i,\omega_i\rangle) \models_\lambda \psi_2$ for every $i \ge 0$. Thus, we get that $\langle p_0,\omega_0\rangle, ..., \langle p_n,\omega_n\rangle, ...$ is a run of $\mathcal{P}$ with $\langle p_0,\omega_0\rangle = \langle p,\omega\rangle$ and $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$.
- Let's consider case 2 where there exists a configuration $\langle [p_n,\psi],\omega_n\rangle$ in ρ whose children are $\langle [p_n,\psi_1],\omega_n\rangle$ and $\langle [p_n,\psi_2],\omega_n\rangle$. Then $\mathcal{BP}'_\psi$ has an infinite path $\langle [p_0,\psi],\omega_0\rangle, ..., \langle [p_n,\psi],\omega_n\rangle, \langle [p_n,\psi_1],\omega_n\rangle...$, where $\langle [p_0,\psi],\omega_0\rangle = \langle [p,\psi],\omega\rangle$. Each configuration $\langle [p_i,\psi],\omega_i\rangle$ in this path has children $\langle [p_i,\psi_2],\omega_i\rangle$ and $\langle [p_{i+1},\psi],\omega_{i+1}\rangle$. Thus $\mathcal{BP}'_\psi$ has an accepting run from $\langle [p_n,\psi_1],\omega_n\rangle$ and $\mathcal{BP}'_\psi$ has an accepting run from $\langle [p_i,\psi_2],\omega_i\rangle$, for $1 \le i \le n$. By applying the induction hypothesis, $(\mathcal{P},\langle p_n,\omega_n\rangle) \models_\lambda \psi_1$ and $(\mathcal{P},\langle p_i,\omega_i\rangle) \models_\lambda \psi_2$, for each $i : 1 \le i \le n$. Thus, $\langle p_0,\omega_0\rangle, ..., \langle p_n,\omega_n\rangle, ...$ is a run of $\mathcal{P}$ with $\langle p_0,\omega_0\rangle = \langle p,\omega\rangle$ and $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \psi$.

Case $\psi = A[\psi_1 R \psi_2]$: This case is similar to the case $\psi = E[\psi_1 R \psi_2]$. □

From this theorem and Theorem 3, it follows that:

Corollary 2. *Given a PDS $\mathcal{P} = (P, \Gamma, \Delta, \sharp)$, a regular labeling function λ, and a CTL formula φ, we can construct an AMA $\mathcal{A}$ such that for every configuration $\langle p,\omega\rangle$ of $\mathcal{P}$, $(\mathcal{P},\langle p,\omega\rangle) \models_\lambda \varphi$ iff the AMA $\mathcal{A}$ recognizes the configuration $\langle [p,\varphi],\omega\rangle$. This AMA can be computed in time $O(|P|^3 \cdot |\Gamma|^2 \cdot |\varphi|^3 \cdot k^2 \cdot |\Delta| \cdot d \cdot 2^{5(|P||\varphi|+k)})$, where $k = \sum_{a\in AP^+(\varphi)} |Q_a| + \sum_{a\in AP^-(\varphi)} |Q_{\neg a}|$ and $d = \sum_{a\in AP^+(\varphi)} |\delta_a| + \sum_{a\in AP^-(\varphi)} |\delta_{\neg a}|$.*

The complexity follows from the complexity of Algorithm 1 and the fact that $\mathcal{BP}'_\varphi$ has $O(|P||\varphi| + k)$ states and $O((|P||\Gamma| + |\Delta|)|\varphi| + d)$ transitions.

Remark 1. Note that to improve the complexity, we represent the regular valuations M_a's using AMAs instead of MAs. The construction of $\mathcal{BP}'_\varphi$ can be easily generalized to the class of AMA to represent regular valuations. This prevents the exponential blow-up when complementing these automata to compute $M_{\neg a}$.

6. Experiments

We implemented all the algorithms presented in the previous sections in a tool. As far as we know, this is the first tool for CTL model-checking for PDSs. We applied our tool to the verification of sequential programs. Indeed, PDSs are well adapted to model sequential (possibly recursive) programs [1,2]. We carried out several experiments. We obtained interesting results. In particular, we were able to confirm the existence of known bugs in Linux drivers. Our results are reported in Fig. 3. **Column** *formula size* gives the size of the formula. **Columns** *time (s)* and *mem (kb)* give the time (in seconds) and memory (in kb). **Column** *Recu.* gives the number of iterations of *loop*$_1$. The last **Column** *result* gives the result

Examples		$\|P\|+\|\Gamma\|$ $+\|\Delta\|$	Formula size	Recu	Time(s)	Mem(kb)	Result
Algorithm 1	1	3+3+4	-	3	0	22.34	Y
	2	17+5+24	-	4	0	33.23	N
	3	73+5+73	-	4	0.02	128.28	Y
	4	75+6+75	-	5	0.02	81.36	N
	5	3+4+4	-	4	0	22.36	N
	6	3+4+5	-	3	0	21.54	Y
	7	3+4+4	-	3	0	20.11	Y
	8	3+4+4	-	4	0	27.40	Y
	9	74+6+76	-	5	0.02	87.54	Y
	10	17+5+24	-	3	0	28.46	Y
	11	18+5+28	-	3	0	26.15	Y
Standard	Plotter.1	1+19+24	2	3	0.02	41.56	Y
	Plotter.2	1+19+24	2	3	0	43.52	N
	Plotter.3	1+19+24	14	9	0.03	241.32	Y
	ATM.1	2+18+45	8	6	0.03	169.64	Y
	ATM.2	2+18+45	10	6	0.03	192.53	Y
	Lock.1	6+37+82	7	11	0.11	387.15	Y
	Lock.2	6+37+82	7	11	0.11	379.46	N
	Lock-err	6+37+82	3	9	0.00	186.52	N
	M-WO.1	1+7+12	6	2	0	40.20	Y
	M-WO.2	1+7+12	6	7	0	37.28	N
	File.1	1+5+9	2	3	0	34.77	Y
	File.2	1+5+9	2	4	0.02	32.51	N
	W.G.C.	16+1+40	23	2	0.05	202.01	Y
	btrfs/file.c	2+14+20	3	10	0	64.32	N
	btrfs/file.c-fixed	2+15+22	3	9	0.02	82.52	Y
	bluetooth	32+12+294	5	8	0.12	821.03	N
	w83627ehf	1+20+20	5	9	0.02	132.76	N
	w83627ehf-fixed	1+21+22	5	4	0.03	121.69	Y
	w83697ehf	1+56+57	6	11	0.35	394.61	Y
	advantech	2+16+31	7	6	0.05	120.41	Y
	at91rm9200	4+15+64	7	5	0.06	234.42	N
	at91rm9200-fixed	4+16+67	7	6	0.12	255.62	Y
	at32ap700x	4+25+105	7	8	0.15	356.04	N
	at32ap700x-fixed	4+25+109	7	9	0.22	334.42	Y
	pcf857x	1+98+106	10	18	0.23	541.35	Y
Regular Valuation	ATM.3	2+18+45	8	6	0.20	352.47	Y
	File.3	1+5+9	5	5	0	33.21	Y
	RSM1	1+8+11	25	4	0.06	438.23	Y
	RSM2	1+8+12	30	4	0.48	1231.45	Y
	RSM3	1+11+17	45	4	12.11	6206.73	Y
	RSM4	1+11+18	45	4	0.72	1269.26	Y
	RSM5	1+11+16	35	4	12.14	6212.2	Y
	ieee1394_core_1	1+104+108	12	14	0.20	413.69	Y
	ieee1394_core_2	1+104+108	13	14	0.19	422.17	Y
	ieee1394_core_3	1+104+108	14	17	0.19	438.42	N
	ieee1394_core_4	1+104+109	14	14	0.19	414.27	Y

Fig. 3. The performance of our tool.

whether the formula is satisfied or not (Y is satisfied, otherwise N). The first eleven lines of the table describe experiments done to evaluate Algorithm 1 that computes the set of configurations from which an ABPDS has an accepting run. The second part of the table describes experiments for "standard" CTL model-checking in which most of the specifications cannot be expressed in LTL. The last part considers CTL model-checking with regular valuations.

Plotter controls a plotter that creates random bar graphs [20]. We checked three CTL properties for this example (**Plotter.1**, **Plotter.2** and **Plotter.3**). In Plotter.1, we checked whether the program can eventually terminates or not. In Plotter.2, we checked whether the program always terminate or not. In Plotter.3, we checked a desirable correctness property that an upward movement should never be immediately followed by a downward movement and vice versa. This property is specified as the CTL formula $(AG(up \rightarrow A[\neg down\ U\ up \vee right])) \wedge AG(down \rightarrow A[\neg up\ U\ down \vee right])$. This formula states that whenever *up* (resp. *down*) holds, then *down* (resp. *up*) cannot hold until *up* (resp. *down*) or *right* holds. **ATM** is an automatic teller machine controller. We checked that if the pincode is correct, then the ATM will provide money (**ATM.1**), and otherwise, it will set an alarm (**ATM.2**). **ATM.3** checks that the ATM gives the money only if the pincode is correct, and if it is accessed from the main session. This property is expressed in CTL with regular valuations as the formula: $AG((input\_pincode \wedge EX\ pincode\_correct) \rightarrow \Gamma\ main\_session)$, where $\Gamma\ main\_session$ is a regular predicate stating that the return address *main_session* is on the stack, i.e., the pincode input session is accessed from the main session (note that when a function call is made, its return address is pushed onto the stack). This formula expresses that whenever *pincode_correct* can be true from the *input_pincode* state, the return address *main_session* is on the stack. Regular valuations are needed to express this property. **Lock** is a lock-unlock program. In Lock.1, we checked that an acquired lock cannot be acquired again until it is released. In Lock.2, we checked that a released lock cannot be released again until it is acquired. **Lock-err** is a buggy version of **Lock** which acquires an acquired lock without releasing. **M-WO** is a Micro-Wave Oven controller taken from [13]. We checked that the oven will stop once it is hot, and that it cannot continue heating forever. **File** is a file management program. In File.1 (resp. File.2), we checked that a file could be closed eventually (resp. immediately). In File.3, we checked that a file should be opened before reading or writing. This property is expressed in CTL with regular valuations as $AG((read \vee write) \rightarrow \Gamma\ open)$, where $\Gamma\ open$ is a regular predicate stating that the file is opened, as we push *open* onto the stack when the file is opened. **W.G.C.** checks the Wolf, Goat and Cabbage problem, where the CTL formula expresses that Wolf, Goat and Cabbage can eventually cross the river. **btrfs/file.c** models the source file *file.c* from the Linux btrfs file system. We confirm the existence of the known error in this file which was reported by Xin Zhong.[3] In this program, a lock is continually acquired twice without releasing. **btrfs/file.c-fixed** is the bug fixed version of **btrfs/file.c**. **Bluetooth** is a simplified model of a Bluetooth driver [25]. We also confirm the existence of the data race found by [25]. **w83627ehf**, **w83697ehf** and **advantech** are watchdog Linux drivers. **w83627ehf** had an error that the watchdog is not enabled after timeout.[4] Our tool confirmed the existence of this bug by checking whether whenever the time is out, the watchdog will be enabled in future. **w83627ehf-fixed** is the bug fixed version of **w83627ehf**. For **w83697ehf** and **advantech**, we checked the similar property as for **w83627ehf**. **at91rm9200** and **at32ap700x** are Real Time Clock drivers for Linux. In **at91rm9200**, once the *setalarm* function is called in which the alarm is enabled, the interrupt may be disabled.[5] Our tool confirmed the existence of this error. **at91rm9200-fixed** is the bug fixed version of **at91rm9200**. We applied our tool to check **at32ap700x** against the same property as **at91rm9200**. We also found this error. **pcf857x** is a driver for pcf857x, pca857x, and pca967x I2C GPIO expanders. We checked whether the driver correctly sets the number of bits for GPIO expanders. **RSM1**–**RSM5** are examples written by us which are the PDSs encoding the recursive state machine in [26] and are used to check the efficiency of the regular valuations part of our tool. **ieee1394_core_1**–**ieee1394_core_4** are simplified versions of IEEE 1394 driver for Linux. For **ieee1394_core_1**, we checked that whenever the function *hpsb_send_phy_config* is called to send its physical configuration, then the packet of the configuration should be sent by calling the function *hpsb_send_packet* in the function *hpsb_send_packet_and_wait* which will wait the response of sending the packet. This property is expressed in CTL with regular valuations as the formula $AG(call\_hpsb\_send\_phy\_config \rightarrow EF(call\_hpsb\_send\_packet \wedge \Gamma\ hpsb\_send\_packet\_and\_wait3\ \Gamma^*))$, where $\Gamma\ hpsb\_send\_packet\_and\_wait3\ \Gamma^*$ denotes a regular predicate stating the return address of *hpsb_send_packet* is *hpsb_send_packet_and_wait3* and is the second symbol on the stack (Γ denotes the top of the stack can be any symbol and Γ^* denotes that the rest of the stack can be any word), i.e., the function call of *hpsb_send_packet* is made in *hpsb_send_packet_and_wait*. For **ieee1394_core_2**, we checked that if the function *hpsb_send_packet_and_wait* is called, then the function *hpsb_set_packet_complete_task* should not be called until the function *init_completion* is called before the return of *hpsb_send_packet_and_wait*. For **ieee1394_core_3**, we checked that if the function *hpsb_send_packet_and_wait* is called, then the function *hpsb_free_packet* should be called before the return of the function *hpsb_send_packet_and_wait*. For **ieee1394_core_4**, we checked that if the function *send_packet_nocare* is called, then the function *hpsb_send_packet* should be called before the return of the function in which *send_packet_nocare* is called and *hpsb_send_packet* is called in the function *send_packet_nocare*. As described in Fig. 3, our tool could find errors in some of these drivers. We needed regular valuations to express some properties, while "Standard" CTL is not sufficient. All the PDSs and CTL formulas considered in this work and the source code of our tool can be downloaded on the website ftp://222.73.57.93.

7. Related work

Alternating Büchi Pushdown Systems can be seen as non-deterministic Büchi Pushdown Systems over trees. Emptiness of non-deterministic Büchi Pushdown Systems over trees is solved in triple exponential time by Harel and Raz [27]. On the other hand, the emptiness problem of ABPDSs can be seen as solving Büchi games on pushdown systems. [22] proposed an algorithm for computing the winning strategy of Büchi games on pushdown systems. Our work has several differences compared with [22]. The algorithm in [22] has a time complexity which is exponential in $O(|P|^2)$. Moreover, no implementation

[3] http://permalink.gmane.org/gmane.comp.file-systems.btrfs/8072.

[4] https://bugzilla.kernel.org/show_bug.cgi?id=15558.

[5] https://bugzilla.kernel.org/show_bug.cgi?id=11112.

of this algorithm is given in [22]. Also, [22] gives only the proof sketch of some theorems about correctness and termination of substitution. While our work considers the emptiness problem of ABPDSs, our algorithm runs in time exponential in $5|P|$. We also implement our techniques in a tool and give the details of the proofs. These technical proofs are very important to understand why the algorithm works. [7] considers the emptiness problem in Alternating Parity Pushdown Automata. The emptiness problem of nondeterministic parity pushdown tree automata is investigated in [12,28,29]. ABPDSs can be seen as a subclass of these Automata. For ABPDSs, our algorithm is more general than the ones in these works since it allows to characterize and compute the set of configurations from which the ABPDS has an accepting run, whereas the other algorithms allow only to check emptiness. Moreover, the emptiness of ABPDSs is known to be EXPTIME-complete and our algorithm is asymptotically optimal compared with known algorithms.

Model-checking pushdown systems against branching time temporal logics has already been intensively investigated in the literature. Several algorithms have been proposed. Walukiewicz [14] showed that CTL model checking is EXPTIME-complete for PDSs. The complexity of our algorithm matches this bound. CTL corresponds to a fragment of the alternation-free μ-calculus and of CTL*. Model checking full μ-calculus for PDSs has been considered in [15,8–10]. These algorithms allow only to determine whether a given configuration satisfies the property. They cannot compute the set of all the configurations where the formula holds. As far as CTL is concerned, our algorithm is more general since it allows to compute a finite automaton that characterizes the set of all such configurations. Moreover, the complexity of our algorithm is comparable to the ones of [15,8–10] when applied to CTL, it is even better than [9,10].

[11,6] consider the global model-checking of PDSs against the more expressive modal μ-calculus, i.e., they symbolically compute the set of configurations that satisfy the formula which is regular. They reduce this problem to the membership problem in two-way alternating parity tree automata. [6] considers also μ-calculus model-checking with regular valuations. These algorithms are more complex, technically more complicated and less intuitive than our procedure. Indeed, the complexity of [11,6] is $(|\varphi| \cdot |P| \cdot |\Delta| \cdot |\Gamma|)^{O(|P| \cdot |\Delta| \cdot |\varphi|)^2}$, whereas our complexity is $O(|P|^2 \cdot |\varphi|^3 \cdot (|P| \cdot |\Gamma| + |\Delta|) \cdot |\Gamma| \cdot 2^{5|P||\varphi|})$. [18,19] present another technique for global model-checking of PDSs against μ-calculus. Their complexity is $O(2^{O(|P| \cdot |\varphi| \cdot d_\varphi)})$ where d_φ denotes the nesting depth of the fixed points of the formula φ. We showed in [21] that our tool is much more efficient than PDSolver under the benchmark considered in [21].

In [3], Bouajjani et al. consider alternating pushdown systems (without the Büchi accepting condition). They provide an algorithm to compute a finite automaton representing the *Pre** of a regular set of configurations for these systems. We use this procedure in $loop_2$ of Algorithm 1. [24] showed how to efficiently implement this procedure. We used the ideas in [24] while implementing Algorithm 1. In their paper, Bouajjani et al. applied their *Pre** algorithm to compute the set of PDS configurations that satisfy a given alternation-free μ-calculus formula. Their procedure is more complex than ours. It is exponential in $|P| \cdot |\varphi|^2$ whereas our algorithm is exponential only in $|P| \cdot |\varphi|$, where $|P|$ is the number of states of the PDS and $|\varphi|$ is the size of the formula.

It is well known that the model-checking problem for μ-calculus is polynomially reducible to the problem of solving parity games. Parity games for pushdown systems are considered in [30,31] and are solved in time exponential in $(|P||\varphi|)^2$. As far as CTL model-checking is concerned, our method is simpler, less complex (i.e. exponential in $5|P||\varphi|$), and more intuitive than these algorithms.

Model checking CTL* for PDS is 2EXPTIME-complete (in the size of the formula) [7]. Algorithms for model-checking CTL* specifications for PDSs have been proposed in [5,16,17,7]. [5] considers also CTL* model checking with regular valuations. When applied to CTL formulas, these algorithms are more complex than our techniques. They are double exponential in the size of the formula and exponential in the size of the system; whereas our procedure is only exponential for both sizes (the formula and the system).

LTL model-checking with regular valuations was considered in [16,17]. Their algorithm is based on a reduction to the "standard" LTL model-checking problem for PDSs. The reduction is done by performing a kind of product of the PDS with the different regular automata representing the different constraints on the stack. Compared to these algorithms, our techniques for CTL model-checking with regular valuations are direct, in the sense that they do not necessitate to make the product of the PDS with the different automata of the regular constraints.

Acknowledgements

We gratefully acknowledge the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work was partially supported by the Pujiang Talent Project of the Shanghai Science and Technology Committee (No. 14PJ1403200), the Open Project of Shanghai Key Laboratory of Trustworthy Computing (No. 07dz22304201301), ANR grant (No. ANR-08-SEGI-006), Shanghai Knowledge Service Platform for Trustworthy Internet of Things (No. ZF1213).

References

[1] J. Esparza, J. Knoop, An automata-theoretic approach to interprocedural data-flow analysis, in: FoSSaCS, 1999, pp. 14–30.
[2] J. Esparza, S. Schwoon, A BDD-based model checker for recursive programs, in: CAV, 2001, pp. 324–336.
[3] A. Bouajjani, J. Esparza, O. Maler, Reachability analysis of pushdown automata: application to model-checking, in: CONCUR, 1997, pp. 135–150.
[4] J. Esparza, D. Hansel, P. Rossmanith, S. Schwoon, Efficient algorithms for model checking pushdown systems, in: CAV, 2000, pp. 232–247.
[5] A. Finkel, B. Willems, P. Wolper, A direct symbolic approach to model checking pushdown systems, Electron. Notes Theor. Comput. Sci. 9 (1997) 27–37.

[6] O. Kupferman, N. Piterman, M.Y. Vardi, An automata-theoretic approach to infinite-state systems, in: Essays in Memory of Amir Pnueli, 2010, pp. 202–259.
[7] L. Bozzelli, Complexity results on branching-time pushdown model checking, Theoret. Comput. Sci. 379 (1–2) (2007) 286–297.
[8] O. Burkart, B. Steffen, Model checking the full modal mu-calculus for infinite sequential processes, in: ICALP, 1997, pp. 419–429.
[9] I. Walukiewicz, Pushdown processes: games and model checking, in: CAV, 1996, pp. 62–74.
[10] O. Kupferman, M.Y. Vardi, An automata-theoretic approach to reasoning about infinite-state systems, in: CAV, 2000, pp. 36–52.
[11] N. Piterman, M.Y. Vardi, Global model-checking of infinite-state systems, in: CAV, 2004, pp. 387–400.
[12] O. Kupferman, N. Piterman, M.Y. Vardi, Pushdown specifications, in: LPAR, 2002, pp. 262–277.
[13] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, Cambridge, MA, USA, 1999.
[14] I. Walukiewicz, Model checking CTL properties of pushdown systems, in: FSTTCS, 2000, pp. 127–138.
[15] O. Burkart, B. Steffen, Composition, decomposition and model checking of pushdown processes, Nordic J. Comput. 2 (2) (1995) 89–125.
[16] J. Esparza, A. Kucera, S. Schwoon, Model checking LTL with regular valuations for pushdown systems, Inform. and Comput. 186 (2) (2003) 355–376.
[17] J. Esparza, A. Kucera, S. Schwoon, Model-checking LTL with regular valuations for pushdown systems, in: TACS, 2001, pp. 316–339.
[18] M. Hague, C.-H.L. Ong, Analysing mu-calculus properties of pushdown systems, in: SPIN, 2010, pp. 187–192.
[19] M. Hague, C.-H.L. Ong, A saturation method for the modal μ-calculus over pushdown systems, Inform. and Comput. 209 (5) (2011) 799–821.
[20] S. Schwoon, Model-checking pushdown systems, Ph.D. thesis, Technische Universität München, 2002.
[21] F. Song, T. Touili, PuMoC: a CTL model-checker for sequential programs, in: ASE, 2012, pp. 346–349.
[22] T. Cachat, Symbolic strategy synthesis for games on pushdown graphs, in: ICALP, 2002, pp. 704–715.
[23] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, Pacific J. Math. 5 (2) (1955) 285–309.
[24] D. Suwimonteerabuth, S. Schwoon, J. Esparza, Efficient algorithms for alternating pushdown systems with an application to the computation of certificate chains, in: ATVA, 2006, pp. 141–153.
[25] S. Qadeer, D. Wu, Kiss: keep it simple and sequential, in: PLDI 04: Programming Language Design and Implementation, 2004, pp. 14–24.
[26] R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T.W. Reps, M. Yannakakis, Analysis of recursive state machines, ACM Trans. Program. Lang. Syst. 27 (4) (2005) 786–818.
[27] D. Harel, D. Raz, Deciding emptiness for stack automata on infinite trees, Inform. and Comput. 113 (2) (1994) 278–299.
[28] L. Bozzelli, A. Murano, A. Peron, Pushdown module checking, in: LPAR, 2005, pp. 504–518.
[29] L. Bozzelli, A. Murano, A. Peron, Pushdown module checking, Form. Methods Syst. Des. 36 (1) (2010) 65–95.
[30] T. Cachat, Uniform solution of parity games on prefix-recognizable graphs, Electron. Notes Theor. Comput. Sci. 68 (6) (2003) 71–84.
[31] O. Serre, Note on winning positions on pushdown games with [omega]-regular conditions, Inform. Process. Lett. 85 (6) (2003) 285–291.