Branching Temporal Logic of Calls and Returns for Pushdown Systems

Huu Vu Nguyen, Tayssir Touili

Branching Temporal Logic of Calls and Returns for Pushdown Systems

Huu-Vu Nguyen[1], Tayssir Touili[2]

[1] LIPN, CNRS and University Paris 13, France
[2] CNRS, LIPN and University Paris 13, France

Abstract. Pushdown Systems (PDSs) are a natural model for sequential programs with (recursive) procedure calls. In this work, we define the Branching temporal logic of CAlls and RETurns (BCARET) that allows to write branching temporal formulas while taking into account the matching between calls and returns. We consider the model-checking problem of PDSs against BCARET formulas with "standard" valuations (where an atomic proposition holds at a configuration c or not depends only on the control state of c, not on its stack) as well as regular valuations (where the set of configurations in which an atomic proposition holds is regular). We show that these problems can be effectively solved by a reduction to the emptiness problem of Alternating Büchi Pushdown Systems. We show that our results can be applied for malware detection.

1 Introduction

Pushdown Systems (PDSs) are a natural model for sequential programs with (recursive) procedure calls. Thus, it is very important to have model-checking algorithms for PDSs. A lot of work focuses on proposing verification algorithms for PDSs, e.g, for both linear temporal logic (LTL and its extensions) [6,10,9,11,12,18] and branching temporal logic (CTL and its extensions) [6,7,8,19,16]. However, LTL and CTL are not always adequate to specify properties. Indeed, some properties need to talk about matching between calls and returns. Thus, CARET (a temporal logic of calls and returns) was introduced by Alur et al [5]. This logic allows to write linear temporal logic formulas while taking into account matching of calls and returns. Later, VP-μ (also named NT-μ in other works of the same authors) [2,3,4], a branching-time temporal logic that allows to talk about matching between calls and returns, was introduced. VP-μ can be seen as an extension of the modal μ-calculus which allows to talk about matching of calls and returns.

In [2], the authors proposed an algorithm to model-check VP-μ formulas for Recursive State Machines (RSMs) [1]. RSMs can be seen as a natural model to represent sequential programs with (recursive) procedure calls. Each procedure is modelled as a module. The invocation to a procedure is modelled as a *call* node; the return from a module corresponds to a *ret* node; and the remaining statements are considered as internal nodes in the RSMs. Thus, RSMs are a

good formalism to model sequential programs written in structured programming languages like C or Java. However, they become non suitable for modelling binary or assembly programs; since, in these programs, explicit push and pop instructions can occur. This makes impossible the use of RSMs to model assembly programs and binary codes directly (whereas Pushdown Systems can model binary codes in a natural way [17]). Model checking binary and assembly programs is very important. Indeed, sometimes, only the binary code is available. Moreover, malicious programs are often executables, i.e., binary codes. Thus, it is very important to be able to model check binary and assembly programs against branching-time formulas with matchings between calls and returns. One can argue that from a binary/assembly program, one can compute a PDS as described in [17] and then apply the translation in [1] to obtain a RSM and then apply the VP-μ model-checking algorithm of [2] on this RSM. However, by doing so, we loose the explicit manipulation of the program's stack. Explicit push and pop instructions are not represented in a natural way anymore, and the stack of the RSM does not correspond to the stack of the assembly program anymore. Thus, it is not possible to state intuitive formulas that correspond to properties of the program's behaviors on the obtained RSM. Especially, when these formulas talk about the content of the program's stack. Thus, it is very important to have a *direct* algorithm for model-checking a branching-time temporal logic with matching of calls and returns for PDSs.

However, VP-μ is a heavy formalism that can't be used by novice users. Indeed, VP-μ can be seen as an extension of the modal μ calculus with several modalities $\langle loc\rangle$, $[loc]$, $\langle call\rangle$, $[call]$, $\langle ret\rangle$, $[ret]$ that allow to distinguish between calls, returns, and other statements (neither calls nor returns). Writing a simple specification in VP-μ is complicated. For example, the following simple property stating that "the configuration e can be reached in the same procedural context as the current configuration" can be described (as shown in [2]) by the complex VP-μ formula $\varphi'_2 = \mu X(e \vee \langle loc\rangle X \vee \langle call\rangle \varphi'_3\{X\})$ where $\varphi'_3 = \mu Y(\langle ret\rangle R_1 \vee \langle loc\rangle Y \vee \langle call\rangle Y\{Y\})$. Thus, we need to define a more intuitive branching-time temporal logic (in the style of CTL) that allow to talk naturally and intuitively about matching calls and returns.

Therefore, we define in this work the Branching temporal logic of CAlls and RETurns BCARET. BCARET can be seen as an extension of CTL with operators that allow to talk about matchings between calls and returns. Using BCARET, the above reachability property can be described in a simple way by the formula $EF^a e$ where EF^a is a BCARET operator that means "there exists a run on which eventually in the future in the same procedural context". We consider the model-checking problem of PDSs against BCARET formulas with "standard" valuations (where an atomic proposition holds at a configuration c or not depends only on the control state of c, not on its stack) as well as regular valuations (where the set of configurations in which an atomic proposition holds is a regular set of configurations). We show that these problems can be effectively solved by a reduction to the emptiness problem of Alternating Büchi Pushdown Systems (ABPDSs). The latter problem can be solved effectively in [16]. Note that the regular valuation case cannot be solved by translating the PDSs to RSMs

since as said previously, by doing the translation of PDSs to obtain RSMs, we loose the structure of the program's stack.

The rest of the paper is organized as follows. In Section 2, we define Labelled Pushdown Systems. In Section 3, we define the logic BCARET. Section 4 presents applications of BCARET in specifying malicious behaviours. Our algorithm to reduce BCARET model-checking to the membership problem of ABPDSs is presented in Section 5. Section 6 discusses the model-checking problem for PDSs against BCARET formulas with regular valuations. Finally, we conclude in Section 7. The proofs can be found in the full version of the paper [13].

2 Pushdown Systems: A model for sequential programs

Pushdown systems is a natural model that was extensively used to model sequential programs. Translations from sequential programs to PDSs can be found e.g. in [15]. As will be discussed in the next section, to precisely describe malicious behaviors as well as context-related properties, we need to keep track of the call and return actions in each path. Thus, as done in [14], we adapt the PDS model in order to record whether a rule of a PDS corresponds to a *call*, a *return*, or another instruction. We call this model a *Labelled Pushdown System*. We also extend the notion of *run* in order to take into account matching returns of calls.

Definition 1. *A Labelled Pushdown System (PDS) $\mathcal{P}$ is a tuple $(P, \Gamma, \Delta, \sharp)$, where P is a finite set of control locations, Γ is a finite set of stack alphabet, $\sharp \notin \Gamma$ is a bottom stack symbol and Δ is a finite subset of $((P \times \Gamma) \times (P \times \Gamma^*) \times \{call, ret, int\})$. If $((p,\gamma),(q,\omega),t) \in \Delta$ $(t \in \{call, ret, int\})$, we also write $\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta$. Rules of Δ are of the following form, where $p \in P, q \in P, \gamma, \gamma_1, \gamma_2 \in \Gamma$, and $\omega \in \Gamma^*$:*

- *(r_1):* $\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma_1\gamma_2\rangle$
- *(r_2):* $\langle p,\gamma\rangle \xrightarrow{ret} \langle q,\epsilon\rangle$
- *(r_3):* $\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle$

Intuitively, a rule of the form $\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma_1\gamma_2\rangle$ corresponds to a call statement. Such a rule usually models a statement of the form $\gamma \xrightarrow{call \quad proc} \gamma_2$. In this rule, γ is the control point of the program where the function call is made, γ_1 is the entry point of the called procedure, and γ_2 is the return point of the call. A rule r_2 models a return, whereas a rule r_3 corresponds to a *simple* statement (neither a call nor a return). A configuration of $\mathcal{P}$ is a pair $\langle p,\omega\rangle$, where p is a control location and $\omega \in \Gamma^*$ is the stack content. For technical reasons, we suppose w.l.o.g. that the bottom stack symbol $\sharp$ is never popped from the stack, i.e., there is no rule in the form $\langle p,\sharp\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta$ $(t \in \{call, ret, int\})$. $\mathcal{P}$ defines a transition relation $\Rightarrow_{\mathcal{P}}$ $(t \in \{call, ret, int\})$ as follows: If $\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle$, then for every $\omega' \in \Gamma^*$, $\langle p,\gamma\omega'\rangle \Rightarrow_{\mathcal{P}} \langle q,\omega\omega'\rangle$. In other words, $\langle q,\omega\omega'\rangle$ is an immediate successor of $\langle p,\gamma\omega'\rangle$. Let $\overset{*}{\Rightarrow}_{\mathcal{P}}$ be the reflexive and transitive closure of $\Rightarrow_{\mathcal{P}}$.

A run of $\mathcal{P}$ from $\langle p_0, \omega_0 \rangle$ is a sequence $\langle p_0, \omega_0 \rangle \langle p_1, \omega_1 \rangle \langle p_2, \omega_2 \rangle ...$ where $\langle p_i, \omega_i \rangle \in P \times \Gamma^*$ s.t. for every $i \geq 0$, $\langle p_i, \omega_i \rangle \Rightarrow_{\mathcal{P}} \langle p_{i+1}, \omega_{i+1} \rangle$. Given a configuration $\langle p, \omega \rangle$, let $Traces(\langle p, \omega \rangle)$ be the set of all possible runs starting from $\langle p, \omega \rangle$.

2.1 Global and abstract successors

Let $\pi = \langle p_0, \omega_0 \rangle \langle p_1, \omega_1 \rangle ...$ be a run starting from $\langle p_0, \omega_0 \rangle$. Over π, two kinds of successors are defined for every position $\langle p_i, \omega_i \rangle$:

- *global-successor*: The global-successor of $\langle p_i, \omega_i \rangle$ is $\langle p_{i+1}, \omega_{i+1} \rangle$ where $\langle p_{i+1}, \omega_{i+1} \rangle$ is an immediate successor of $\langle p_i, \omega_i \rangle$.
- *abstract-successor*: The abstract-successor of $\langle p_i, \omega_i \rangle$ is determined as follows:
  - If $\langle p_i, \omega_i \rangle \Rightarrow_{\mathcal{P}} \langle p_{i+1}, \omega_{i+1} \rangle$ corresponds to a call statement, there are two cases: (1) if $\langle p_i, \omega_i \rangle$ has $\langle p_k, \omega_k \rangle$ as a corresponding return-point in π, then, the abstract successor of $\langle p_i, \omega_i \rangle$ is $\langle p_k, \omega_k \rangle$; (2) if $\langle p_i, \omega_i \rangle$ does not have any corresponding return-point in π, then, the abstract successor of $\langle p_i, \omega_i \rangle$ is $\perp$.
  - If $\langle p_i, \omega_i \rangle \Rightarrow_{\mathcal{P}} \langle p_{i+1}, \omega_{i+1} \rangle$ corresponds to a *simple* statement, the abstract successor of $\langle p_i, \omega_i \rangle$ is $\langle p_{i+1}, \omega_{i+1} \rangle$.
  - If $\langle p_i, \omega_i \rangle \Rightarrow_{\mathcal{P}} \langle p_{i+1}, \omega_{i+1} \rangle$ corresponds to a return statement, the abstract successor of $\langle p_i, \omega_i \rangle$ is defined as $\perp$.

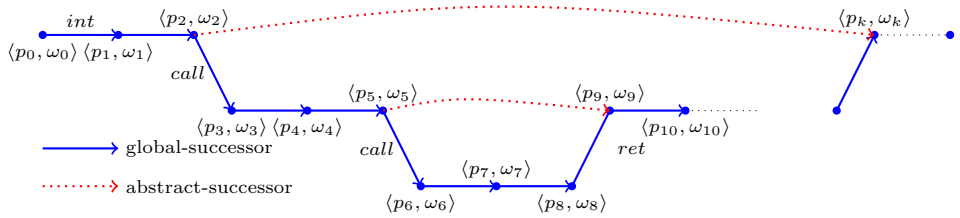

Fig. 1: Two kinds of successors on a run

For example, in Figure 1:

- The global-successors of $\langle p_1, \omega_1 \rangle$ and $\langle p_2, \omega_2 \rangle$ are $\langle p_2, \omega_2 \rangle$ and $\langle p_3, \omega_3 \rangle$ respectively.
- The abstract-successors of $\langle p_2, \omega_2 \rangle$ and $\langle p_5, \omega_5 \rangle$ are $\langle p_k, \omega_k \rangle$ and $\langle p_9, \omega_9 \rangle$ respectively.

Let $\langle p, \omega \rangle$ be a configuration of a PDS $\mathcal{P}$. A configuration $\langle p', \omega' \rangle$ is defined as a global-successor of $\langle p, \omega \rangle$ iff $\langle p', \omega' \rangle$ is a global-successor of $\langle p, \omega \rangle$ over a run $\pi \in Traces(\langle p, \omega \rangle)$. Similarly, a configuration $\langle p', \omega' \rangle$ is defined as an abstract-successor of $\langle p, \omega \rangle$ iff $\langle p', \omega' \rangle$ is an abstract-successor of $\langle p, \omega \rangle$ over a run $\pi \in Traces(\langle p, \omega \rangle)$

A *global-path* of $\mathcal{P}$ from $\langle p_0, \omega_0 \rangle$ is a sequence $\langle p_0, \omega_0 \rangle \langle p_1, \omega_1 \rangle \langle p_2, \omega_2 \rangle ...$ where $\langle p_i, \omega_i \rangle \in P \times \Gamma^*$ s.t. for every $i \geq 0$, $\langle p_{i+1}, \omega_{i+1} \rangle$ is a global-successor of $\langle p_i, \omega_i \rangle$. Similarly, an *abstract-path* of $\mathcal{P}$ from $\langle p_0, \omega_0 \rangle$ is a sequence $\langle p_0, \omega_0 \rangle \langle p_1, \omega_1 \rangle \langle p_2, \omega_2 \rangle ...$ where $\langle p_i, \omega_i \rangle \in P \times \Gamma^*$ s.t. for every $i \geq 0$, $\langle p_{i+1}, \omega_{i+1} \rangle$ is an abstract-successor of $\langle p_i, \omega_i \rangle$. For instance, in Figure 1, $\langle p_0, \omega_0 \rangle \langle p_1, \omega_1 \rangle \langle p_2, \omega_2 \rangle \langle p_3, \omega_3 \rangle \langle p_4, \omega_4 \rangle \langle p_5, \omega_5 \rangle ...$ is a global-path, while $\langle p_0, \omega_0 \rangle \langle p_1, \omega_1 \rangle \langle p_2, \omega_2 \rangle \langle p_k, \omega_k \rangle ...$ is an abstract-path.

2.2 Multi Automata

Definition 2. *[6] Let $\mathcal{P} = (P, \Gamma, \Delta, \natural)$ be a PDS. A $\mathcal{P}$-Multi-Automaton (MA for short) is a tuple $\mathcal{A} = (Q, \Gamma, \delta, I, Q_f)$, where Q is a finite set of states, $\delta \subseteq Q \times \Gamma \times Q$ is a finite set of transition rules, $I = P \subseteq Q$ is a set of initial states, $Q_f \subseteq Q$ is a set of final states.*

The transition relation $\rightarrow_\delta \subseteq Q \times \Gamma^* \times Q$ is defined as follows:

- $q \xrightarrow{\epsilon}_\delta q$ for every $q \in Q$
- $q \xrightarrow{\gamma}_\delta q'$ if $(q, \gamma, q') \in \delta$
- if $q \xrightarrow{\omega}_\delta q'$ and $q' \xrightarrow{\gamma}_\delta q''$, then, $q \xrightarrow{\omega\gamma}_\delta q''$

$\mathcal{A}$ recognizes a configuration $\langle p, \omega \rangle$ where $p \in P$, $\omega \in \Gamma^*$ iff $p \xrightarrow{\omega}_\delta q$ for some $q \in Q_f$. The language of $\mathcal{A}$, $L(\mathcal{A})$, is the set of all configurations which are recognized by $\mathcal{A}$. A set of configurations is *regular* if it is recognized by some Multi-Automaton.

3 Branching Temporal Logic of Calls and Returns - BCARET

In this section, we define the Branching temporal logic of CAlls and RETurns BCARET. For technical reasons, we assume w.l.o.g. that BCARET formulas are given in positive normal form, i.e. negations are applied only to atomic propositions. To do that, we use the release operator R as a dual of the until operator U.

Definition 3. ***Syntax of BCARET***
Let AP be a finite set of atomic propositions, a BCARET formula φ is defined as follows, where $b \in \{g, a\}$, $e \in AP$:

$$\varphi ::= true \mid false \mid e \mid \neg e \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid EX^b\varphi \mid AX^b\varphi \mid E[\varphi U^b \varphi] \mid A[\varphi U^b \varphi] \mid E[\varphi R^b \varphi] \mid A[\varphi R^b \varphi]$$

Let $\mathcal{P} = (P, \Gamma, \Delta, \natural)$ be a PDS, $\lambda : AP \rightarrow 2^{P \times \Gamma^*}$ be a labelling function that assigns to each atomic proposition $e \in AP$ a set of configurations of $\mathcal{P}$. The satisfiability relation of a BCARET formula φ at a configuration $\langle p_0, \omega_0 \rangle$ w.r.t. the labelling function λ, denoted by $\langle p_0, \omega_0 \rangle \vDash_\lambda \varphi$, is defined inductively as follows:

- $\langle p_0, \omega_0\rangle \models_\lambda true$ for every $\langle p_0, \omega_0\rangle$
- $\langle p_0, \omega_0\rangle \not\models_\lambda false$ for every $\langle p_0, \omega_0\rangle$
- $\langle p_0, \omega_0\rangle \models_\lambda e$ $(e \in AP)$ iff $\langle p_0, \omega_0\rangle \in \lambda(e)$
- $\langle p_0, \omega_0\rangle \models_\lambda \neg e$ $(e \in AP)$ iff $\langle p_0, \omega_0\rangle \notin \lambda(e)$
- $\langle p_0, \omega_0\rangle \models_\lambda \varphi_1 \vee \varphi_2$ iff $(\langle p_0, \omega_0\rangle \models_\lambda \varphi_1$ or $\langle p_0, \omega_0\rangle \models_\lambda \varphi_2)$
- $\langle p_0, \omega_0\rangle \models_\lambda \varphi_1 \wedge \varphi_2$ iff $(\langle p_0, \omega_0\rangle \models_\lambda \varphi_1$ and $\langle p_0, \omega_0\rangle \models_\lambda \varphi_2)$
- $\langle p_0, \omega_0\rangle \models_\lambda EX^g\varphi$ iff there exists a global-successor $\langle p', \omega'\rangle$ of $\langle p_0, \omega_0\rangle$ such that $\langle p', \omega'\rangle \models_\lambda \varphi$
- $\langle p_0, \omega_0\rangle \models_\lambda AX^g\varphi$ iff $\langle p', \omega'\rangle \models_\lambda \varphi$ for every global-successor $\langle p', \omega'\rangle$ of $\langle p_0, \omega_0\rangle$
- $\langle p_0, \omega_0\rangle \models_\lambda E[\varphi_1 U^g \varphi_2]$ iff there exists a global-path $\pi = \langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle\langle p_2, \omega_2\rangle...$ of $\mathcal{P}$ starting from $\langle p_0, \omega_0\rangle$ s.t. $\exists i \geq 0$, $\langle p_i, \omega_i\rangle \models_\lambda \varphi_2$ and for every $0 \leq j < i$, $\langle p_j, \omega_j\rangle \models_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models_\lambda A[\varphi_1 U^g \varphi_2]$ iff for every global-path $\pi = \langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle\langle p_2, \omega_2\rangle...$of $\mathcal{P}$ starting from $\langle p_0, \omega_0\rangle$, $\exists i \geq 0$, $\langle p_i, \omega_i\rangle \models_\lambda \varphi_2$ and for every $0 \leq j < i$, $\langle p_j, \omega_j\rangle \models_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models_\lambda E[\varphi_1 R^g \varphi_2]$ iff there exists a global-path $\pi = \langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle\langle p_2, \omega_2\rangle...$ of $\mathcal{P}$ starting from $\langle p_0, \omega_0\rangle$ s.t. for every $i \geq 0$, if $\langle p_i, \omega_i\rangle \not\models_\lambda \varphi_2$ then there exists $0 \leq j < i$ s.t. $\langle p_j, \omega_j\rangle \models_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models_\lambda A[\varphi_1 R^g \varphi_2]$ iff for every global-path $\pi = \langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle\langle p_2, \omega_2\rangle...$ of $\mathcal{P}$ starting from $\langle p_0, \omega_0\rangle$, for every $i \geq 0$, if $\langle p_i, \omega_i\rangle \not\models_\lambda \varphi_2$ then there exists $0 \leq j < i$ s.t. $\langle p_j, \omega_j\rangle \models_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models_\lambda EX^a\varphi$ iff there exists an abstract-successor $\langle p', \omega'\rangle$ of $\langle p_0, \omega_0\rangle$ such that $\langle p', \omega'\rangle \models_\lambda \varphi$
- $\langle p_0, \omega_0\rangle \models_\lambda AX^a\varphi$ iff $\langle p', \omega'\rangle \models_\lambda \varphi$ for every abstract-successor $\langle p', \omega'\rangle$ of $\langle p_0, \omega_0\rangle$
- $\langle p_0, \omega_0\rangle \models_\lambda E[\varphi_1 U^a \varphi_2]$ iff there exists an abstract-path $\pi = \langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle\langle p_2, \omega_2\rangle...$ of $\mathcal{P}$ starting from $\langle p_0, \omega_0\rangle$ s.t. $\exists i \geq 0$, $\langle p_i, \omega_i\rangle \models_\lambda \varphi_2$ and for every $0 \leq j < i$, $\langle p_j, \omega_j\rangle \models_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models_\lambda A[\varphi_1 U^a \varphi_2]$ iff for every abstract-path $\pi = \langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle\langle p_2, \omega_2\rangle...$ of $\mathcal{P}$, $\exists i \geq 0$, $\langle p_i, \omega_i\rangle \models_\lambda \varphi_2$ and for every $0 \leq j < i$, $\langle p_j, \omega_j\rangle \models_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models_\lambda E[\varphi_1 R^a \varphi_2]$ iff there exists an abstract-path $\pi = \langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle\langle p_2, \omega_2\rangle...$ of $\mathcal{P}$ starting from $\langle p_0, \omega_0\rangle$ s.t. for every $i \geq 0$, if $\langle p_i, \omega_i\rangle \not\models_\lambda \varphi_2$ then there exists $0 \leq j < i$ s.t. $\langle p_j, \omega_j\rangle \models_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models_\lambda A[\varphi_1 R^a \varphi_2]$ iff for every abstract-path $\pi = \langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle\langle p_2, \omega_2\rangle...$ of $\mathcal{P}$ starting from $\langle p_0, \omega_0\rangle$, for every $i \geq 0$, if $\langle p_i, \omega_i\rangle \not\models_\lambda \varphi_2$ then there exists $0 \leq j < i$ s.t. $\langle p_j, \omega_j\rangle \models_\lambda \varphi_1$

Other BCARET operators can be expressed by the above operators: $EF^g\varphi = E[true\ U^g\varphi]$, $EF^a\varphi = E[true\ U^a\varphi]$, $AF^g\varphi = A[true\ U^g\varphi]$, $AF^a\varphi = A[trueU^a\varphi]$,...

Closure. Given a BCARET formula φ, the closure $Cl(\varphi)$ is the set of all sub-formulae of φ, including φ.

Regular Valuations. We talk about regular valuations when for every $e \in AP$, $\lambda(e)$ is a regular language.

Remark 1. CTL can be seen as the subclass of BCARET where the operators $EX^a\varphi, AX^a\varphi, E[\varphi U^a\varphi], A[\varphi U^a\varphi], E[\varphi R^a\varphi], A[\varphi R^a\varphi]$ are not considered.

4 Application

In this section, we show how BCARET can be used to describe branching-time malicious behaviors. More malicious behaviors can be found in the full version of the paper [13].

Spyware Behavior. The typical behaviour of a spyware is hunting for personal information (emails, bank account information,...) on local drives by searching files matching certain conditions. To do that, it has to search directories of the host to look for interesting files whose names match a specific condition. When a file is found, the spyware will invoke a payload to steal the information, then continue looking for the remaining matching files. When a folder is found, it will enter the folder path and continue scanning that folder recursively. To achieve this behavior, the spyware first calls the API function $FindFirstFileA$ to search for the first matching file in a given folder path. After that, it has to check whether the call to the API function $FindFirstFileA$ succeeds or not. If the function call fails, the spyware will call the function $GetLastError$. Otherwise, if the function call is successful, $FindFirstFileA$ will return a search handle h. There are two possibilities in this case. If the returned result is a folder, it will call the API function $FindFirstFileA$ again to search for matching results in the found folder. If the returned result is a file, it will call the API function $FindNextFileA$ using h as first parameter to look for the remaining matching files. This behavior cannot be expressed by LTL or CTL because it requires to express that the return value of the function $FindFirstFileA$ should be used as input to the API function $FindNextFileA$. It cannot be described by CARET neither (because this is a branching-time property). Using BCARET, the above behavior can be expressed by the following formula:

$$\varphi_{sb} = \bigvee_{d \in D} EF^g \Big(call(FindFirstFileA) \wedge EX^a(eax = d) \wedge AF^a \Big(call(GetLastError) \vee call(FindFirstFileA) \vee \Big(call(FindNextFileA) \wedge d\Gamma^* \Big) \Big) \Big)$$

where the $\bigvee$ is taken over all possible memory addresses d which contain the values of search handles h in the program, EX^a is a BCARET operator that means "next in some run, in the same procedural context"; EF^g is the standard CTL EF operator (eventually in some run), while AF^a is a BCARET operator that means "eventually in all runs, in the same procedural context".

In binary codes and assembly programs, the return value of an API function is put in the register eax. Thus, the return value of $FindFirstFileA$ is the value of eax at its corresponding return-point. Then, the subformula (call(FindFirstFileA)$\wedge$ $EX^a(eax = d)$) states that there is a call to the API $FindFirstFileA$ and the

return value of this function is d (the abstract successor of a call is its corresponding return-point). When *FindNextFileA* is invoked, it requires a search handle as parameter and this search handle must be put on top of the program stack (since parameters are passed through the stack in assembly). The requirement that d is on top of the program stack is expressed by the regular expression $d\Gamma^*$. Thus, the subformula $[\text{call(FindNextFileA)} \wedge d\Gamma^*]$ expresses that *FindNextFileA* is called with d as parameter (d stores the information of the search handle). Therefore, φ_{sb} expresses then that there is a call to the API $FindFirstFileA$ with the return value d (the search handle), then, in all runs starting from that call, there will be either a call to the API function $GetLastError$ or a call to the function $FindFirstFileA$ or a call to the function $FindNextFileA$ in which d is used as a parameter.

To detect spyware, [14] used the following CARET formula:
$\varphi'_{sb} = \bigvee_{d \in D} F^g(\text{call(FindFirstFileA)} \wedge X^a(eax = d) \wedge F^a(\text{call(FindNextFileA)} \wedge d\Gamma^*))$

It can be seen that this CARET formula φ'_{sb} is not as precise as the BCARET formula φ_{sb}, as it does not deal with the case when the returned result of $FindFirstFileA$ is a folder or an error. Thus, this CARET formula φ'_{sb} may lead to false alarms that can be avoided using our BCARET formula φ_{sb}. BCARET can deal with it because BCARET is a branching-time temporal logic. For example, AF^a allows us to take into account all possible abstract-paths from a certain state in the computation tree. By using AF^a, φ_{sb} can deal with different returned values of $FindFirstFileA$ as presented above.

5 BCARET Model-Checking for Pushdown Systems

In this section, we consider "standard" BCARET model-checking for pushdown systems where an atomic proposition holds at a configuration c or not depends only on the control state of c, not on its stack.

5.1 Alternating Büchi Pushdown Systems (ABPDSs).

Definition 4. *An Alternating Büchi Pushdown System (ABPDS) is a tuple $\mathcal{BP} = (P, \Gamma, \Delta, F)$, where P is a set of control locations, Γ is the stack alphabet, $F \subseteq P$ is a set of accepting control locations and Δ is a transition function that maps each element of $P \times \Gamma$ with a positive boolean formula over $P \times \Gamma^*$.*

A configuration of $\mathcal{BP}$ is a pair $\langle p, \omega \rangle$, where $p \in P$ is the current control location and $\omega \in \Gamma^*$ is the current stack content. Without loss of generality, we suppose that the boolean formulas of ABPDSs are in disjunctive normal form $\bigvee_{j=1}^{n} \bigwedge_{i=1}^{m_j} \langle p_i^j, \omega_i^j \rangle$. Then, we can see Δ as a subset of $(P \times \Gamma) \times 2^{P \times \Gamma^*}$ by rewriting the rules of Δ in the form $\langle p, \gamma \rangle \rightarrow \bigvee_{j=1}^{n} \bigwedge_{i=1}^{m_j} \langle p_i^j, \omega_i^j \rangle$ as n rules of the form $\langle p, \gamma \rangle \rightarrow \{\langle p_1^j, \omega_1^j \rangle, ..., \langle p_{m_j}^j, \omega_{m_j}^j \rangle\}$, where $1 \leq j \leq n$. Let $\langle p, \gamma \rangle \rightarrow \{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\}$ be a rule of Δ, then, for every $\omega \in \Gamma^*$, the configuration $\langle p, \gamma\omega \rangle$(resp. $\{\langle p_1, \omega_1\omega \rangle, ..., \langle p_n, \omega_n\omega \rangle\}$) is an immediate predecessor (resp. successor) of $\{\langle p_1, \omega_1\omega \rangle, ..., \langle p_n, \omega_n\omega \rangle\}$ (resp. $\langle p, \gamma\omega \rangle$).

A run ρ of $\mathcal{BP}$ starting form an initial configuration $\langle p_0, \omega_0 \rangle$ is a tree whose root is labelled by $\langle p_0, \omega_0 \rangle$, and whose other nodes are labelled by elements in $P \times \Gamma^*$. If a node of ρ is labelled by a configuration $\langle p, \omega \rangle$ and has n children labelled by $\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle$ respectively, then, $\langle p, \omega \rangle$ must be a predecessor of $\{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\}$ in $\mathcal{BP}$. A path of a run ρ is an infinite sequence of configurations $c_0 c_1 c_2 ...$ s.t. c_0 is the root of ρ and c_{i+1} is one of the children of c_i for every $i \geq 0$. A path is accepting iff it visits infinitely often configurations with control locations in F. A run ρ is accepting iff every path of ρ is accepting. The language of $\mathcal{BP}$, $\mathcal{L}(\mathcal{BP})$, is the set of configurations c s.t. $\mathcal{BP}$ has an accepting run starting from c.

$\mathcal{BP}$ defines the reachability relation $\Rightarrow_{\mathcal{BP}_\varphi}$ as follows: (1) $c \Rightarrow_{\mathcal{BP}} \{c\}$ for every $c \in P \times \Gamma^*$, (2) $c \Rightarrow_{\mathcal{BP}} C$ if C is an immediate successor of c; (3) if $c \Rightarrow_{\mathcal{BP}} \{c_1, c_2, ..., c_n\}$ and $c_i \Rightarrow_{\mathcal{BP}} C_i$ for every $1 \leq i \leq n$, then $c \Rightarrow_{\mathcal{BP}} \bigcup_{i=1}^{n} C_i$. Given $c_0 \Rightarrow_{\mathcal{BP}} C'$, then, $\mathcal{BP}$ has an accepting run from c_0 iff $\mathcal{BP}$ has an accepting run from c' for every $c' \in C'$.

Theorem 1. *[16] Given an ABPDS $\mathcal{BP} = (P, \Gamma, \Delta, F)$, for every configuration $\langle p, \omega \rangle \in P \times \Gamma^*$, whether or not $\langle p, \omega \rangle \in \mathcal{L}(\mathcal{BP})$ can be decided in time $\mathcal{O}(|P|^2.|\Gamma|.(|\Delta|2^{5|P|} + 2^{|P|}|\omega|))$.*

5.2 From BCARET model checking of PDSs to the membership problem in ABPDSs

Let $\mathcal{P} = (P, \Gamma, \Delta, \sharp)$ be a pushdown system with an initial configuration c_0. Given a set of atomic propositions AP, let φ be a BCARET formula. Let $f : AP \rightarrow 2^P$ be a function that associates each atomic proposition with a set of control states, and $\lambda_f : AP \rightarrow 2^{P \times \Gamma^*}$ be a labelling function s.t. for every $e \in AP$, $\lambda_f(e) = \{\langle p, \omega \rangle \mid p \in f(e), \omega \in \Gamma^*\}$. In this section, we propose an algorithm to check whether $c_0 \vDash_{\lambda_f} \varphi$. Intuitively, we construct an Alternating Büchi Pushdown System $\mathcal{BP}_\varphi$ which recognizes a configuration c iff $c \vDash_{\lambda_f} \varphi$. Then to check whether $c_0 \vDash_{\lambda_f} \varphi$, we will check if $c_0 \in \mathcal{L}(\mathcal{BP}_\varphi)$. The membership problem of an ABPDS can be solved effectively by Theorem 1.

Let $\mathcal{BP}_\varphi = (P', \Gamma', \Delta', F)$ be the ABPDS defined as follows:

- $P' = P \cup (P \times Cl(\varphi)) \cup \{p_\perp\}$
- $\Gamma' = \Gamma \cup (\Gamma \times Cl(\varphi)) \cup \{\gamma_\perp\}$
- $F = F_1 \cup F_2 \cup F_3$ where
  - $F_1 = \{(\!|p, e|\!) \mid e \in Cl(\varphi), e \in AP \text{ and } p \in f(e)\}$
  - $F_2 = \{(\!|p, \neg e|\!) \mid \neg e \in Cl(\varphi), e \in AP \text{ and } p \notin f(e)\}$
  - $F_3 = \{P \times Cl_R(\varphi)\}$ where $Cl_R(\varphi)$ is the set of formulas of $Cl(\varphi)$ in the form $E[\varphi_1 R^b \varphi_2]$ or $A[\varphi_1 R^b \varphi_2]$ $(b \in \{g, a\})$

The transition relation Δ' is the smallest set of transition rules defined as follows: $\Delta \subseteq \Delta'$ and for every $p \in P$, $\phi \in Cl(\varphi)$, $\gamma \in \Gamma$, $b \in \{g, a\}$ and $t \in \{call, ret, int\}$:

(α1) If $\phi = e$, $e \in AP$ and $p \in f(e)$, then, $\langle (\!|p, \phi|\!), \gamma \rangle \rightarrow \langle (\!|p, \phi|\!), \gamma \rangle \in \Delta'$

($\alpha 2$) If $\phi = \neg e$, $e \in AP$ and $p \notin f(e)$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow \langle (\!|p, \phi|\!), \gamma\rangle \in \Delta'$

($\alpha 3$) If $\phi = \phi_1 \wedge \phi_2$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow \langle (\!|p, \phi_1|\!), \gamma\rangle \wedge \langle (\!|p, \phi_2|\!), \gamma\rangle \in \Delta'$

($\alpha 4$) If $\phi = \phi_1 \vee \phi_2$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow \langle (\!|p, \phi_1|\!), \gamma\rangle \vee \langle (\!|p, \phi_2|\!), \gamma\rangle \in \Delta'$

($\alpha 5$) If $\phi = EX^g\phi_1$, then $(\!|p, \phi|\!), \gamma\rangle \rightarrow \bigvee_{\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta} \langle (\!|q, \phi_1|\!), \omega\rangle \in \Delta'$ where $t \in \{call, int, ret\}$

($\alpha 6$) If $\phi = AX^g\phi_1$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow \bigwedge_{\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta} \langle (\!|q, \phi_1|\!), \omega\rangle \in \Delta'$

($\alpha 7$) If $\phi = EX^a\phi_1$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow h_1 \vee h_2 \vee h_3 \in \Delta'$, where
- $h_1 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma_1\gamma_2\rangle \in \Delta} \langle q, \gamma_1 (\!|\gamma_2, \phi_1|\!)\rangle$
- $h_2 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle \in \Delta} \langle (\!|q, \phi_1|\!), \omega\rangle$
- $h_3 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{ret} \langle q,\epsilon\rangle \in \Delta} \langle p_\perp, \gamma_\perp\rangle$

($\alpha 8$) If $\phi = AX^a\phi_1$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow h_1 \wedge h_2 \wedge h_3 \in \Delta'$, where
- $h_1 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma_1\gamma_2\rangle \in \Delta} \langle q, \gamma_1 (\!|\gamma_2, \phi_1|\!)\rangle$
- $h_2 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle \in \Delta} \langle (\!|q, \phi_1|\!), \omega\rangle$
- $h_3 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{ret} \langle q,\epsilon\rangle \in \Delta} \langle p_\perp, \gamma_\perp\rangle$

($\alpha 9$) If $\phi = E[\phi_1 U^g \phi_2]$, then,
$\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow \langle (\!|p, \phi_2|\!), \gamma\rangle \vee \bigvee_{\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta} (\langle (\!|p, \phi_1|\!), \gamma\rangle \wedge \langle (\!|q, \phi|\!), \omega\rangle) \in \Delta'$

($\alpha 10$) If $\phi = E[\phi_1 U^a \phi_2]$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow \langle (\!|p, \phi_2|\!), \gamma\rangle \vee h_1 \vee h_2 \vee h_3 \in \Delta'$, where
- $h_1 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma_1\gamma_2\rangle \in \Delta} \langle (\!|p, \phi_1|\!), \gamma\rangle \wedge \langle q, \gamma_1 (\!|\gamma_2, \phi|\!)\rangle$
- $h_2 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle \in \Delta} \langle (\!|p, \phi_1|\!), \gamma\rangle \wedge \langle (\!|q, \phi|\!), \omega\rangle$
- $h_3 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{ret} \langle q,\epsilon\rangle \in \Delta} \langle p_\perp, \gamma_\perp\rangle$

($\alpha 11$) If $\phi = A[\phi_1 U^g \phi_2]$, then,
$\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow \langle (\!|p, \phi_2|\!), \gamma\rangle \vee \bigwedge_{\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta} (\langle (\!|p, \phi_1|\!), \gamma\rangle \wedge \langle (\!|q, \phi|\!), \omega\rangle) \in \Delta'$

($\alpha 12$) If $\phi = A[\phi_1 U^a \phi_2]$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow \langle (\!|p, \phi_2|\!), \gamma\rangle \vee (h_1 \wedge h_2 \wedge h_3) \in \Delta'$, where
- $h_1 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma_1\gamma_2\rangle \in \Delta} \langle (\!|p, \phi_1|\!), \gamma\rangle \wedge \langle q, \gamma_1 (\!|\gamma_2, \phi|\!)\rangle$
- $h_2 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle \in \Delta} \langle (\!|p, \phi_1|\!), \gamma\rangle \wedge \langle (\!|q, \phi|\!), \omega\rangle$
- $h_3 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{ret} \langle q,\epsilon\rangle \in \Delta} \langle p_\perp, \gamma_\perp\rangle$

($\alpha 13$) If $\phi = E[\phi_1 R^g \phi_2]$, then, we add to Δ' the rule:
$\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow (\langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle (\!|p, \phi_1|\!), \gamma\rangle) \vee (\bigvee_{\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta} (\langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle (\!|q, \phi|\!), \omega\rangle)$

($\alpha 14$) If $\phi = A[\phi_1 R^g \phi_2]$, then, we add to Δ' the rule:
$\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow (\langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle (\!|p, \phi_1|\!), \gamma\rangle) \vee (\bigwedge_{\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta} (\langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle (\!|q, \phi|\!), \omega\rangle)$

($\alpha 15$) If $\phi = E[\phi_1 R^a \phi_2]$: $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow (\langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle (\!|p, \phi_1|\!), \gamma\rangle) \vee h_1 \vee h_2 \vee h_3 \in \Delta'$, where
- $h_1 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma_1\gamma_2\rangle \in \Delta} \langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle q, \gamma_1 (\!|\gamma_2, \phi|\!)\rangle$
- $h_2 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle \in \Delta} \langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle (\!|q, \phi|\!), \omega\rangle$
- $h_3 = \bigvee_{\langle p,\gamma\rangle \xrightarrow{ret} \langle q,\epsilon\rangle \in \Delta} \langle p_\perp, \gamma_\perp\rangle$

($\alpha 16$) If $\phi = A[\phi_1 R^a \phi_2]$, $\langle (\!|p, \phi|\!), \gamma\rangle \rightarrow (\langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle (\!|p, \phi_1|\!), \gamma\rangle) \vee (h_1 \wedge h_2 \wedge h_3) \in \Delta'$, where
- $h_1 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma_1\gamma_2\rangle \in \Delta} \langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle q, \gamma_1 (\!|\gamma_2, \phi|\!)\rangle$

- $h_2 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle \in \Delta} \langle (\!|p, \phi_2|\!), \gamma\rangle \wedge \langle (\!|q, \phi|\!), \omega\rangle$
- $h_3 = \bigwedge_{\langle p,\gamma\rangle \xrightarrow{ret} \langle q,\epsilon\rangle \in \Delta} \langle p_\perp, \gamma_\perp\rangle$

(α17) for every $\langle p, \gamma\rangle \xrightarrow{ret} \langle q, \epsilon\rangle \in \Delta$:
- $\langle q, (\!|\gamma'', \phi_1|\!)\rangle \rightarrow \langle (\!|q, \phi_1|\!), \gamma''\rangle \in \Delta'$ for every $\gamma'' \in \Gamma$, $\phi_1 \in Cl(\varphi)$

(α18) $\langle p_\perp, \gamma_\perp\rangle \rightarrow \langle p_\perp, \gamma_\perp\rangle \in \Delta'$

Roughly speaking, the ABPDS $\mathcal{BP}_\varphi$ is a kind of product between $\mathcal{P}$ and the BCARET formula φ which ensures that $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p, \varphi|\!), \omega\rangle$ iff the configuration $\langle p, \omega\rangle$ satisfies φ. The form of the control locations of $\mathcal{BP}_\varphi$ is $(\!|p, \phi|\!)$ where $\phi \in Cl(\varphi)$. Let us explain the intuition behind our construction:

- If $\phi = e \in AP$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega\rangle \vDash_{\lambda_f} \phi$ iff $p \in f(e)$. In other words, $\mathcal{BP}_\varphi$ should have an accepting run from $\langle (\!|p, e|\!), \omega\rangle$ iff $p \in f(e)$. This is ensured by the transition rules in (α1) which add a loop at $\langle (\!|p, e|\!), \omega\rangle$ where $p \in f(e)$ and the fact that $(\!|p, e|\!) \in F$.
- If $\phi = \neg e$ ($e \in AP$), then, for every $\omega \in \Gamma^*$, $\langle p, \omega\rangle \vDash_{\lambda_f} \phi$ iff $p \notin f(e)$. In other words, $\mathcal{BP}_\varphi$ should have an accepting run from $\langle (\!|p, \neg e|\!), \omega\rangle$ iff $p \notin f(e)$. This is ensured by the transition rules in (α2) which add a loop at $\langle (\!|p, \neg e|\!), \omega\rangle$ where $p \notin f(e)$ and the fact that $(\!|p, \neg e|\!) \in F$.
- If $\phi = \phi_1 \wedge \phi_2$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega\rangle \vDash_{\lambda_f} \phi$ iff ($\langle p, \omega\rangle \vDash_{\lambda_f} \phi_1$ and $\langle p, \omega\rangle \vDash_{\lambda_f} \phi_2$). This is ensured by the transition rules in (α3) stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p, \phi_1 \wedge \phi_2|\!), \omega\rangle$ iff $\mathcal{BP}_\varphi$ has an accepting run from both $\langle (\!|p, \phi_1|\!), \omega\rangle$ and $\langle (\!|p, \phi_2|\!), \omega\rangle$. ($\alpha$4) is similar to ($\alpha$3).
- If $\phi = E[\phi_1 U^g \phi_2]$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega\rangle \vDash_{\lambda_f} \phi$ iff $\langle p, \omega\rangle \vDash_{\lambda_f} \phi_2$ or ($\langle p, \omega\rangle \vDash_{\lambda_f} \phi_1$ and there exists an immediate successor $\langle p', \omega'\rangle$ of $\langle p, \omega\rangle$ s.t. $\langle p', \omega'\rangle \vDash_{\lambda_f} \phi$). This is ensured by the transition rules in (α9) stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p, E[\phi_1 U^g \phi_2]|\!), \omega\rangle$ iff $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p, \phi_2|\!), \omega\rangle$ or ($\mathcal{BP}_\varphi$ has an accepting run from both $\langle (\!|p, \phi_1|\!), \omega\rangle$ and $\langle (\!|p', \phi|\!), \omega'\rangle$ where $\langle p', \omega'\rangle$ is an immediate successor of $\langle p, \omega\rangle$). (α11) is similar to (α9).
- If $\phi = E[\phi_1 R^g \phi_2]$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega\rangle \vDash_{\lambda_f} \phi$ iff ($\langle p, \omega\rangle \vDash_{\lambda_f} \phi_2$ and $\langle p, \omega\rangle \vDash_{\lambda_f} \phi_1$) or ($\langle p, \omega\rangle \vDash_{\lambda_f} \phi_2$ and there exists an immediate successor $\langle p', \omega'\rangle$ of $\langle p, \omega\rangle$ s.t. $\langle p', \omega'\rangle \vDash_{\lambda_f} \phi$). This is ensured by the transition rules in (α13) stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p, E[\phi_1 R^g \phi_2]|\!), \omega\rangle$ iff $\mathcal{BP}_\varphi$ has an accepting run from both $\langle (\!|p, \phi_2|\!), \omega\rangle$ and $\langle (\!|p, \phi_1|\!), \omega\rangle$; or $\mathcal{BP}_\varphi$ has an accepting run from both $\langle (\!|p, \phi_2|\!), \omega\rangle$ and $(\!|p', \phi|\!), \omega'\rangle$ where $\langle p', \omega'\rangle$ is an immediate successor of $\langle p, \omega\rangle$. In addition, for R^g formulas, the *stop* condition is not required, i.e, for a formula $\phi_1 R^g \phi_2$ that is applied to a specific run, we don't require that ϕ_1 must eventually hold. To ensure that the runs on which ϕ_2 always holds are accepted, we add $(\!|p, \phi|\!)$ to the Büchi accepting condition F (via the subset F_3 of F). (α14) is similar to (α13).
- If $\phi = EX^g \phi_1$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega\rangle \vDash_{\lambda_f} \phi$ iff there exists an immediate successor $\langle p', \omega'\rangle$ of $\langle p, \omega\rangle$ s.t. $\langle p', \omega'\rangle \vDash_{\lambda_f} \phi_1$. This is ensured by the transition rules in (α5) stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p, EX^g \phi_1|\!), \omega\rangle$ iff there exists an immediate successor $\langle p', \omega'\rangle$ of $\langle p, \omega\rangle$ s.t. $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p', \phi_1|\!), \omega'\rangle$. ($\alpha$6) is similar to ($\alpha$5).

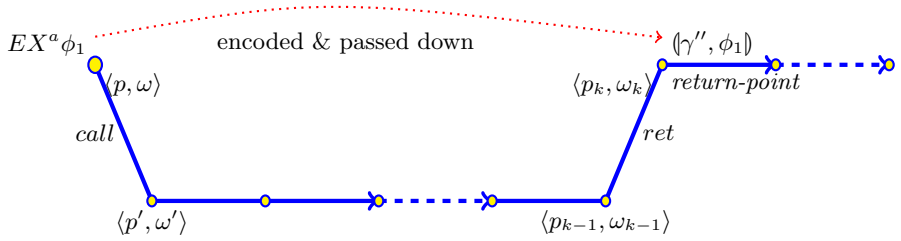

Fig. 2: $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$ corresponds to a call statement

– If $\phi = EX^a\phi_1$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega\rangle \vDash_{\lambda_f} \phi$ iff there exists an abstract-successor $\langle p_k, \omega_k\rangle$ of $\langle p, \omega\rangle$ s.t. $\langle p_k, \omega_k\rangle \vDash_{\lambda_f} \phi_1$ (A1) . Let $\pi \in Traces(\langle p, \omega\rangle)$ be a run starting from $\langle p, \omega\rangle$ on which $\langle p_k, \omega_k\rangle$ is the abstract-successor of $\langle p, \omega\rangle$. Over π, let $\langle p', \omega'\rangle$ be the immediate successor of $\langle p, \omega\rangle$. In what follows, we explain how we can ensure (A1).

1. Firstly, we show that for every abstract-successor $\langle p_k, \omega_k\rangle \neq \bot$ of $\langle p, \omega\rangle$, $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p_k, \phi_1|\!), \omega_k\rangle$. There are two possibilities:

- If $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$ corresponds to a call statement. Let us consider Figure 2 to explain this case. $\langle (\!|p, \phi|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p_k, \phi_1|\!), \omega_k\rangle$ is ensured by rules corresponding to h_1 in (α7), the rules in $\Delta \subseteq \Delta'$ and the rules in (α17) as follows: rules corresponding to h_1 in (α7) allow to record ϕ_1 in the return point of the call, rules in $\Delta \subseteq \Delta'$ allow to mimic the run of the PDS $\mathcal{P}$ and rules in (α17) allow to extract and put back ϕ_1 when the return-point is reached. In what follows, we show in more details how this works: Let $\langle p, \gamma\rangle \xrightarrow{call} \langle p', \gamma'\gamma''\rangle$ be the rule associated with the transition $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$, then we have $\omega = \gamma\omega''$ and $\omega' = \gamma'\gamma''\omega''$. Let $\langle p_{k-1}, \omega_{k-1}\rangle \Rightarrow_{\mathcal{P}} \langle p_k, \omega_k\rangle$ be the transition that corresponds to the *ret* statement of this call on π. Let then $\langle p_{k-1}, \beta\rangle \xrightarrow{ret} \langle p_k, \epsilon\rangle \in \Delta$ be the corresponding return rule. Then, we have necessarily $\omega_{k-1} = \beta\gamma''\omega''$, since as explained in Section 2, γ'' is the return address of the call. After applying this rule, $\omega_k = \gamma''\omega''$. In other words, γ'' will be the topmost stack symbol at the corresponding return point of the call. So, in order to ensure that $\langle (\!|p, \phi|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p_k, \phi_1|\!), \omega_k\rangle$, we proceed as follows: At the call $\langle p, \gamma\rangle \xrightarrow{call} \langle p', \gamma'\gamma''\rangle$, we encode the formula ϕ_1 into γ'' by the rule corresponding to h_1 in (α7) stating that $\langle (\!|p, EX^a\phi_1|\!), \gamma\rangle \rightarrow \langle p', \gamma'(\!|\gamma'', \phi_1|\!)\rangle \in \Delta'$. This allows to record ϕ_1 in the corresponding return point of the stack. After that, the rules in $\Delta \subseteq \Delta'$ allow $\mathcal{BP}_\varphi$ to mimic the run π of $\mathcal{P}$ from $\langle p', \omega'\rangle$ till the corresponding return-point of this call, where $(\!|\gamma'', \phi_1|\!)$ is the topmost stack symbol. More specifically, the following sequence of $\mathcal{P}$: $\langle p', \gamma'\gamma''\omega''\rangle \overset{*}{\Rightarrow}_{\mathcal{P}} \langle p_{k-1}, \beta\gamma''\omega''\rangle \overset{*}{\Rightarrow}_{\mathcal{P}} \langle p_k, \gamma''\omega''\rangle$ will be mimicked by the following sequence of $\mathcal{BP}_\varphi$: $\langle (\!|p', \gamma'(\!|\gamma'', \phi_1|\!)\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_{k-1}, \beta(\!|\gamma'', \phi_1|\!)\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_k, (\!|\gamma'', \phi_1|\!)\omega''\rangle$ using the rules of Δ. At the return-point, we extract ϕ_1 from the stack and encode it into p_k

by adding the transition rules in (α17) $\langle p_k, (\!|\gamma'', \phi_1|\!)\rangle \rightarrow \langle (\!|p_k, \phi_1|\!), \gamma''\rangle$. Therefore, we obtain that $\langle (\!|p, \phi|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p_k, \phi_1|\!), \omega_k\rangle$. The property holds for this case.

- If $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$ corresponds to a simple statement. Then, the abstract successor of $\langle p, \omega\rangle$ is its immediate successor $\langle p', \omega'\rangle$. Thus, we get that $\langle p_k, \omega_k\rangle = \langle p', \omega'\rangle$. From the transition rules corresponding to h_2 in (α7), we get that $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p', \phi_1|\!), \omega'\rangle$. Therefore, $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p_k, \phi_1|\!), \omega_k\rangle$. The property holds for this case.

2. Now, let us consider the case where $\langle p_k, \omega_k\rangle$, the abstract successor of $\langle p, \omega\rangle$, is $\perp$. This case occurs when $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$ corresponds to a return statement. Then, one abstract successor of $\langle p, \omega\rangle$ is $\perp$. Note that $\perp$ does not satisfy any formula, i.e., $\perp$ does not satisfy ϕ_1. Therefore, from $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle$, we need to ensure that the path of $\mathcal{BP}_\varphi$ reflecting the possibility in (A1) that $\langle p_k, \omega_k\rangle \vDash_{\lambda_f} \phi_1$ is not accepted. To do this, we exploit additional trap configurations. We use $p_\perp$ and $\gamma_\perp$ as trap control location and trap stack symbol to obtain these trap configurations. To be more specific, let $\langle p, \gamma\rangle \xrightarrow{ret} \langle p', \epsilon\rangle$ be the rule associated with the transition $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$, then we have $\omega = \gamma\omega''$ and $\omega' = \omega''$. We add the transition rule corresponding to h_3 in (α7) to allow $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_\perp, \gamma_\perp\omega''\rangle$. Since a run of $\mathcal{BP}_\varphi$ includes only infinite paths, we equip these trap configurations with self-loops by the transition rules in (α18), i.e., $\langle p_\perp, \gamma_\perp\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_\perp, \gamma_\perp\omega''\rangle$. As a result, we obtain a corresponding path in $\mathcal{BP}_\varphi$: $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_\perp, \gamma_\perp\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_\perp, \gamma_\perp\omega''\rangle$. Note that this path is not accepted by $\mathcal{BP}_\varphi$ because $p_\perp \notin F$.

In summary, for every abstract-successor $\langle p_k, \omega_k\rangle$ of $\langle p, \omega\rangle$, if $\langle p_k, \omega_k\rangle \neq \perp$, then, $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p_k, \phi_1|\!), \omega_k\rangle$; otherwise $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_\perp, \gamma_\perp\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_\perp, \gamma_\perp\omega''\rangle$ which is not accepted by $\mathcal{BP}_\varphi$. Therefore, (A1) is ensured by the transition rules in (α7) stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p, EX^a\phi_1|\!), \omega\rangle$ iff there exists an abstract successor $\langle p_k, \omega_k\rangle$ of $\langle p, \omega\rangle$ s.t. $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!|p_k, \phi_1|\!), \omega_k\rangle$.

– If $\phi = AX^a\phi_1$: this case is ensured by the transition rules in (α8) together with (α17) and $\Delta \subseteq \Delta'$. The intuition of (α8) is similar to that of (α7).
– If $\phi = E[\phi_1 U^a \phi_2]$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega\rangle \vDash_{\lambda_f} \phi$ iff $\langle p, \omega\rangle \vDash_{\lambda_f} \phi_2$ or ($\langle p, \omega\rangle \vDash_{\lambda_f} \phi_1$ and there exists an abstract successor $\langle p_k, \omega_k\rangle$ of $\langle p, \omega\rangle$ s.t. $\langle p_k, \omega_k\rangle \vDash_{\lambda_f} \phi$) (A2) . Let $\pi \in Traces(\langle p, \omega\rangle)$ be a run starting from $\langle p, \omega\rangle$ on which $\langle p_k, \omega_k\rangle$ is the abstract-successor of $\langle p, \omega\rangle$. Over π, let $\langle p', \omega'\rangle$ be the immediate successor of $\langle p, \omega\rangle$.

1. Firstly, we show that for every abstract-successor $\langle p_k, \omega_k\rangle \neq \perp$ of $\langle p, \omega\rangle$, $\langle (\!|p, \phi|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \{\langle (\!|p, \phi_1|\!), \omega\rangle, \langle (\!|p_k, \phi|\!), \omega_k\rangle\}$. There are two possibilities:

- If $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$ corresponds to a call statement. From the rules corresponding to h_1 in (α10), we get that $\langle (\!|p, \phi|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \{\langle (\!|p, \phi_1|\!), \omega\rangle, \langle p', \omega'\rangle\}$ where $\langle p', \omega'\rangle$ is the immediate successor of $\langle p, \omega\rangle$. Thus, to ensure that $\langle (\!|p, \phi|\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \{\langle (\!|p, \phi_1|\!), \omega\rangle, \langle (\!|p_k, \phi|\!), \omega_k\rangle\}$, we only need to ensure that $\langle p', \omega'\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p_k, \phi|\!), \omega_k\rangle$. As for the case $\phi = EX^a\phi_1$, $\langle p', \omega'\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle (\!|p_k, \phi|\!), \omega_k\rangle$ is ensured by the rules in $\Delta \subseteq \Delta'$ and the rules in (α17):

rules in $\Delta \subseteq \Delta'$ allow to mimic the run of the PDS $\mathcal{P}$ before the return and rules in (α17) allow to extract and put back ϕ_1 when the return-point is reached.

- If $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$ corresponds to a simple statement. Then, the abstract successor of $\langle p, \omega\rangle$ is its immediate successor $\langle p', \omega'\rangle$. Thus, we get that $\langle p_k, \omega_k\rangle = \langle p', \omega'\rangle$. From the transition rules corresponding to h_2 in (α10), we get that $\langle (\!| p, E[\phi_1 U^a \phi_2] |\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \{\langle (\!| p, \phi_1 |\!), \omega\rangle, \langle (\!| p', \phi |\!), \omega'\rangle\}$. Therefore, $\langle (\!| p, E[\phi_1 U^a \phi_2] |\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \{\langle (\!| p, \phi_1 |\!), \omega\rangle, \langle (\!| p_k, \phi |\!), \omega_k\rangle\}$. In other words, $\mathcal{BP}_\varphi$ has an accepting run from both $\langle (\!| p, \phi_1 |\!), \omega\rangle$ and $\langle (\!| p_k, \phi |\!), \omega_k\rangle$ where $\langle p_k, \omega_k\rangle$ is an abstract successor of $\langle p, \omega\rangle$. The property holds for this case.

2. Now, let us consider the case where $\langle p_k, \omega_k\rangle = \bot$. As explained previously, this case occurs when $\langle p, \omega\rangle \Rightarrow_{\mathcal{P}} \langle p', \omega'\rangle$ corresponds to a return statement. Then, the abstract successor of $\langle p, \omega\rangle$ is $\bot$. Note that $\bot$ does not satisfy any formula, i.e., $\bot$ does not satisfy ϕ. Therefore, from $\langle (\!| p, E[\phi_1 U^a \phi_2] |\!), \omega\rangle$, we need to ensure that the path reflecting the possibility in (A2) that ($\langle p, \omega\rangle \vDash_{\lambda_f} \phi_1$ and $\langle p_k, \omega_k\rangle \vDash_{\lambda_f} \phi$) is not accepted by $\mathcal{BP}_\varphi$. This is ensured as for the case $\phi = EX^a\phi_1$ by the transition rules corresponding to h_3 in (α10).

In summary, for every abstract-successor $\langle p_k, \omega_k\rangle$ of $\langle p, \omega\rangle$, if $\langle p_k, \omega_k\rangle \neq \bot$, then, $\langle (\!| p, E[\phi_1 U^a \phi_2] |\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \{\langle (\!| p, \phi_1 |\!), \omega\rangle, \langle (\!| p_k, E[\phi_1 U^a \phi_2] |\!), \omega_k\rangle\}$; otherwise $\langle (\!| p, E[\phi_1 U^a \phi_2] |\!), \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_\bot, \gamma_\bot \omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle p_\bot, \gamma_\bot \omega''\rangle$ which is not accepted by $\mathcal{BP}_\varphi$. Therefore, (A2) is ensured by the transition rules in (α10) stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!| p, E[\phi_1 U^a \phi_2] |\!), \omega\rangle$ iff $\mathcal{BP}_\varphi$ has an accepting run from $\langle (\!| p, \phi_2 |\!), \omega\rangle$; or $\mathcal{BP}_\varphi$ has an accepting run from both $\langle (\!| p, \phi_1 |\!), \omega\rangle$ and $\langle (\!| p_k, E[\phi_1 U^a \phi_2] |\!), \omega_k\rangle$ where $\langle p_k, \omega_k\rangle$ is an abstract successor of $\langle p, \omega\rangle$.

– The intuition behind the rules corresponding to the cases $\phi = A[\phi_1 U^a \phi_2]$, $\phi = E[\phi_1 R^a \phi_2]$, $\phi = A[\phi_1 R^a \phi_2]$ are similar to the previous cases.

The Büchi accepting condition. The elements of the Büchi accepting condition set F of $\mathcal{BP}_\varphi$ ensure the liveness requirements of until-formulas on infinite global paths, infinite abstract paths as well as on finite abstract paths.

– With regards to infinite global paths, the fact that the liveness requirement ϕ_2 in $E[\phi_1 U^g \phi_2]$ is eventually satisfied in $\mathcal{P}$ is ensured by the fact that $(\!| p, E[\phi_1 U^g \phi_2] |\!)$ doesn't belong to F. Note that $\langle p, \omega\rangle \vDash_{\lambda_f} E[\phi_1 U^g \phi_2]$ iff $\langle p, \omega\rangle \vDash_{\lambda_f} \phi_2$ or there exists a global-successor $\langle p', \omega'\rangle$ s.t. ($\langle p, \omega\rangle \vDash_{\lambda_f} \phi_1$ and $\langle p', \omega'\rangle \vDash_{\lambda_f} E[\phi_1 U^g \phi_2]$). Because ϕ_2 should hold eventually, to avoid the case where a run of $\mathcal{BP}_\varphi$ always carries $E[\phi_1 U^g \phi_2]$ and never reaches ϕ_2, we don't set $(\!| p, E[\phi_1 U^g \phi_2] |\!)$ as an element of the Büchi accepting condition set. This guarantees that the accepting run of $\mathcal{BP}_\varphi$ must visit some control locations in $(\!| p, \phi_2 |\!)$ which ensures that ϕ_2 will eventually hold. The liveness requirements of $A[\phi_1 U^g \phi_2]$ are ensured as for the case of $E[\phi_1 U^g \phi_2]$.
– With regards to infinite abstract paths, the fact that the liveness requirement ϕ_2 in $E[\phi_1 U^a \phi_2]$ is eventually satisfied in $\mathcal{P}$ is ensured by the fact that

$(\!|p, E[\phi_1 U^a \phi_2]|\!)$ doesn't belong to F. The intuition behind this case is similar to the intuition of $E[\phi_1 U^g \phi_2]$. The liveness requirements of $A[\phi_1 U^a \phi_2]$ are ensured as for the case of $E[\phi_1 U^a \phi_2]$.

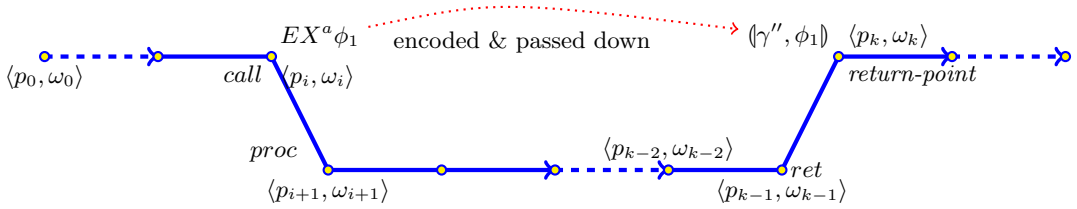

Fig. 3: $\langle p_i, \omega_i \rangle$ finally reach its corresponding return-point

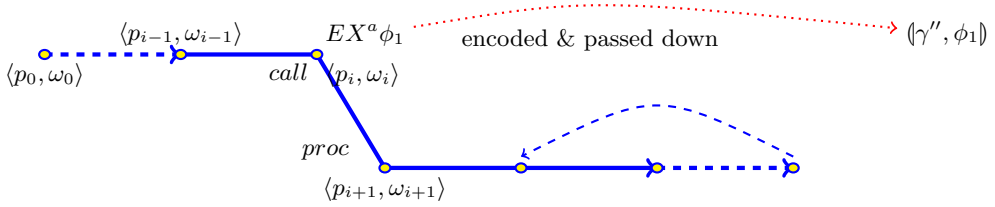

Fig. 4: $\langle p_i, \omega_i \rangle$ never reach its corresponding return-point

– With regards to finite abstract paths $\langle p_0, \omega_0 \rangle \langle p_1, \omega_1 \rangle ... \langle p_m, \omega_m \rangle$ where $\langle p_m, \omega_m \rangle \Rightarrow_{\mathcal{P}} \langle p_{m+1}, \omega_{m+1} \rangle$ corresponds to a return statement, the fact that the liveness requirement ϕ_2 in $E[\phi_1 U^g \phi_2]$ is eventually satisfied in $\mathcal{P}$ is ensured by the fact that $p_\perp$ doesn't belong to F. Look at Figure 3 for an illustration. In this figure, for every $i+1 \leq u \leq k-1$, the abstract path starting from $\langle p_u, \omega_u \rangle$ is finite because the abstract successor of $\langle p_{k-1}, \omega_{k-1} \rangle$ is $\perp$ since $\langle p_{k-1}, \omega_{k-1} \rangle \Rightarrow_{\mathcal{P}} \langle p_k, \omega_k \rangle$ corresponds to a return statement. Suppose that we want to check whether $\langle p_{k-1}, \omega_{k-1} \rangle \vDash_{\lambda_f} E[\phi_1 U^a \phi_2]$, then, we get that $\langle p_{k-1}, \omega_{k-1} \rangle \vDash_{\lambda_f} E[\phi_1 U^a \phi_2]$ iff $\langle p_{k-1}, \omega_{k-1} \rangle \vDash_{\lambda_f} \phi_2$ or there exists an abstract-successor $\langle p', \omega' \rangle$ s.t. ($\langle p_{k-1}, \omega_{k-1} \rangle \vDash_{\lambda_f} \phi_1$ and $\langle p', \omega' \rangle \vDash_{\lambda_f} E[\phi_1 U^a \phi_2]$). Since ϕ_2 should eventually hold, ϕ_2 should hold at $\langle p_{k-1}, \omega_{k-1} \rangle$ because the abstract-successor of $\langle p_{k-1}, \omega_{k-1} \rangle$ on this abstract-path is $\perp$. To ensure this, we move $\langle p_{k-1}, \omega_{k-1} \rangle$ to the trap configuration $\langle p_\perp, \gamma_\perp \rangle$ and add a loop here by the transition rule (α18). In addition, we don't set $p_\perp$ as an element of the Büchi accepting condition set, which means that $\langle p_{k-1}, \omega_{k-1} \rangle \vDash_{\lambda_f} E[\phi_1 U^a \phi_2]$ iff $\langle p_{k-1}, \omega_{k-1} \rangle \vDash_{\lambda_f} \phi_2$ by the transition rules in (α10). This ensures the liveness requirement ϕ_2 in $E[\phi_1 U^a \phi_2]$ is eventually satisfied.

– With regards to finite abstract paths $\langle p_0, \omega_0\rangle\langle p_1, \omega_1\rangle ... \langle p_m, \omega_m\rangle$ where $\langle p_m, \omega_m\rangle \Rightarrow_{\mathcal{P}} \langle p_{m+1}, \omega_{m+1}\rangle$ corresponds to a call statement but this call never reaches its corresponding return-point, the fact that the liveness requirement ϕ_2 in $E[\phi_1 U^g \phi_2]$ is eventually satisfied in $\mathcal{P}$ is ensured by the fact that $p \notin F$. Look at Figure 4 where the procedure *proc* never terminates. In this figure, for every $0 \leq u \leq i$, the abstract path starting from $\langle p_u, \omega_u\rangle$ is finite. Suppose that we want to check whether $\langle p_i, \omega_i\rangle \vDash_{\lambda_f} E[\phi_1 U^a \phi_2]$, then, we get that $\langle p_i, \omega_i\rangle \vDash_{\lambda_f} E[\phi_1 U^a \phi_2]$ iff $\langle p_i, \omega_i\rangle \vDash_{\lambda_f} \phi_2$ or there exists an abstract-successor $\langle p', \omega'\rangle$ s.t. ($\langle p_i, \omega_i\rangle \vDash_{\lambda_f} \phi_1$ and $\langle p', \omega'\rangle \vDash_{\lambda_f} E[\phi_1 U^a \phi_2]$). Since ϕ_2 should eventually hold, ϕ_2 should hold at $\langle p_i, \omega_i\rangle$ because the abstract-successor of $\langle p_i, \omega_i\rangle$ on this abstract-path is $\perp$. As explained above, at $\langle p_i, \omega_i\rangle$, we will encode the formula $E[\phi_1 U^a \phi_2]$ into the stack and mimic the run of $\mathcal{P}$ on $\mathcal{BP}_\varphi$ until it reaches the corresponding return-point of the call. In other words, if the call is never reached, the run of $\mathcal{BP}_\varphi$ will infinitely visit the control locations of $\mathcal{P}$. To ensure this path unaccepted, we don't set $p \in P$ as an element of the Büchi accepting condition set, which means that $\langle p_i, \omega_i\rangle \vDash_{\lambda_f} E[\phi_1 U^a \phi_2]$ iff $\langle p_i, \omega_i\rangle \vDash_{\lambda_f} \phi_2$ by the transition rules in (α10). This ensures the liveness requirement ϕ_2 in $E[\phi_1 U^a \phi_2]$ is eventually satisfied.

Thus, we can show that (the proof can be found in the full version of the paper [13]):

Theorem 2. *Given a PDS $\mathcal{P} = (P, \Gamma, \Delta, \sharp)$, a set of atomic propositions AP, a labelling function $f : AP \to 2^P$ and a BCARET formula φ, we can compute an ABPDS $\mathcal{BP}_\varphi$ such that for every configuration $\langle p, \omega\rangle$, $\langle p, \omega\rangle \vDash_{\lambda_f} \varphi$ iff $\mathcal{BP}_\varphi$ has an accepting run from the configuration $\langle (\!| p, \varphi |\!), \omega\rangle$*

The number of control locations of $\mathcal{BP}_\varphi$ is at most $\mathcal{O}(|P||\varphi|)$, the number of stack symbols is at most $\mathcal{O}(|\Gamma||\varphi|)$ and the number of transitions is at most $\mathcal{O}(|P||\Gamma||\Delta||\varphi|)$. Therefore, we get from Theorems 1 and 2:

Theorem 3. *Given a PDS $\mathcal{P} = (P, \Gamma, \Delta, \sharp)$, a set of atomic propositions AP, a labelling function $f : AP \to 2^P$ and a BCARET formula φ, for every configuration $\langle p, \omega\rangle \in P \times \Gamma^*$, whether or not $\langle p, \omega\rangle$ satisfies φ can be solved in time $\mathcal{O}(|P|^2|\varphi|^3.|\Gamma|(|P||\Gamma||\Delta|.|\varphi|.2^{5|P||\varphi|} + 2^{|P||\varphi|}.|\omega|))$*

6 BCARET model-checking for PDSs with regular valuations

Up to now, we have considered the *standard* model-checking problem for BCARET, where the validity of an atomic proposition depends only on the control state, not on the stack. In this section, we go further and consider model-checking with regular valuations where the set of configurations in which an atomic proposition holds is a regular set of configurations (see Section 3 for a formal definition of regular valuations).

6.1 From BCARET model checking of PDSs with regular valuations to the membership problem in ABPDSs

Given a pushdown system $\mathcal{P} = (P, \Gamma, \Delta, \sharp)$, and a set of atomic propositions AP, let φ be a BCARET formula over AP, $\lambda : AP \to 2^{P \times \Gamma^*}$ be a labelling function s.t. for every $e \in AP$, $\lambda(e)$ is a regular set of configurations. Given a configuration c_0, we propose in this section an algorithm to check whether $c_0 \vDash_\lambda \varphi$. Intuitively, we compute an ABPDS $\mathcal{BP}'_\varphi$ s.t. $\mathcal{BP}'_\varphi$ recognizes a configuration c of $\mathcal{P}$ iff $c \vDash_\lambda \varphi$. Then, to check if c_0 satisfies φ, we will check whether $\mathcal{BP}'_\varphi$ recognizes c_0.

For every $e \in AP$, since $\lambda(e)$ is a regular set of configurations, let $M_e = (Q_e, \Gamma, \delta_e, I_e, F_e)$ be a multi-automaton s.t. $L(M_e) = \lambda(e)$, $M_{\neg e} = (Q_{\neg e}, \Gamma, \delta_{\neg e}, I_{\neg e}, F_{\neg e})$ be a multi-automaton s.t. $L(M_{\neg e}) = P \times \Gamma^* \setminus \lambda(e)$, which means $M_{\neg e}$ will recognize the complement of $\lambda(e)$ that is the set of configurations in which e doesn't hold. Note that for every $e \in AP$, the initial states of M_e and $M_{\neg e}$ are the control locations $p \in \mathcal{P}$. Thus, to distinguish between the initial states of these two automata, we will denote the initial state corresponding to the control location p in M_e (resp. $M_{\neg e}$) by p_e (resp. $p_{\neg e}$). Let $AP^+(\varphi) = \{e \in AP \mid e \in Cl(\varphi)\}$ and $AP^-(\varphi) = \{e \in AP \mid \neg e \in Cl(\varphi)\}$.

Let $\mathcal{BP}'_\varphi = (P'', \Gamma'', \Delta'', F')$ be the ABPDS defined as follows:

- $P'' = P \cup P \times Cl(\varphi) \cup \{p_\perp\} \cup \bigcup_{e \in AP^+(\varphi)} Q_e \cup \bigcup_{e \in AP^-(\varphi)} Q_{\neg e}$
- $\Gamma'' = \Gamma \cup (\Gamma \times Cl(\varphi)) \cup \{\gamma_\perp\}$
- $F' = F_1 \cup F_2 \cup F_3$ where
  - $F_1 = \bigcup_{e \in AP^+(\varphi)} F_e$
  - $F_2 = \bigcup_{e \in AP^-(\varphi)} F_{\neg e}$
  - $F_3 = \{P \times Cl_R(\varphi)\}$ where $Cl_R(\varphi)$ is the set of formulas of $Cl(\varphi)$ in the form $E[\varphi_1 R^b \varphi_2]$ or $A[\varphi_1 R^b \varphi_2]$ $(b \in \{g, a\})$

The transition relation Δ'' is the smallest set of transition rules defined as follows: $\Delta \subseteq \Delta''$, $\Delta'_0 \subseteq \Delta''$ where Δ'_0 is the transitions of Δ' that are created by the rules from ($\alpha 3$) to ($\alpha 18$) and such that:

($\beta 1$) for every $p \in P$, $e \in AP^+(\varphi)$, $\gamma \in \Gamma$: $\langle (\!| p, e |\!), \gamma \rangle \to \langle p_e, \gamma \rangle \in \Delta''$
($\beta 2$) for every $p \in P$, $e \in AP^-(\varphi)$, $\gamma \in \Gamma$: $\langle (\!| p, \neg e |\!), \gamma \rangle \to \langle p_{\neg e}, \gamma \rangle \in \Delta''$
($\beta 3$) for very $(q_1, \gamma, q_2) \in (\bigcup_{e \in AP^+(\varphi)} \delta_e) \cup (\bigcup_{e \in AP^-(\varphi)} \delta_{\neg e})$: $\langle q_1, \gamma \rangle \to \langle q_2, \epsilon \rangle \in \Delta''$
($\beta 4$) for very $q \in (\bigcup_{e \in AP^+(\varphi)} F_e) \cup (\bigcup_{e \in AP^-(\varphi)} F_{\neg e})$: $\langle q, \sharp \rangle \to \langle q, \sharp \rangle \in \Delta''$

Intuitively, we compute the ABPDS $\mathcal{BP}'_\varphi$ such that $\mathcal{BP}'_\varphi$ has an accepting run from $\langle (\!| p, \phi |\!), \omega \rangle$ iff the configuration $\langle p, \omega \rangle$ satisfies ϕ according to the regular labellings M_e for every $e \in AP$. The only difference with the previous case of standard valuations, where an atomic proposition holds at a configuration depends only on the control location of that configuration, not on its stack, comes from the interpretation of the atomic proposition e. This is why Δ'' contains Δ and Δ'_0 (which are the transitions of $\mathcal{BP}_\varphi$ that don't consider the atomic propositions). Here the rules $(\beta_1) - (\beta_4)$ deal with the cases e, $\neg e$ $(e \in AP)$. Given $p \in P$, $\phi = e \in AP$, $\omega \in \Gamma^*$, we get that the ABPDS $\mathcal{BP}'_\varphi$ should accept

$\langle (\!|p, e|\!), \omega \rangle$ iff $\langle p, \omega \rangle \in L(M_e)$. To check whether $\langle p, \omega \rangle \in L(M_e)$, we let $\mathcal{BP}'_\varphi$ go to state p_e, the initial state corresponding to p in M_e by adding rules in (β1); and then, from this state, we will check whether ω is accepted by M_e. This is ensured by the transition rules in (β3) and (β4). (β3) lets $\mathcal{BP}'_\varphi$ mimic a run of M_e on ω, i.e., if $\mathcal{BP}'_\varphi$ is in a state q_1 with γ on the top of the stack, and if (q_1, γ, q_2) is a transition rule in M_e, then, $\mathcal{BP}'_\varphi$ will move to state q_2 and pop γ from its stack. Note that popping γ allows us to check the rest of the word. In M_e, a configuration is accepted if the run with the word ω reaches the final state in F_e; i.e., if $\mathcal{BP}'_\varphi$ reaches a state $q \in F_e$ with an empty stack, i.e., with a stack containing the bottom stack symbol $\sharp$. Thus, we add F_e as a set of accepting control locations in $\mathcal{BP}'_\varphi$. Since $\mathcal{BP}'_\varphi$ only recognizes infinite paths, (β4) adds a loop on every configuration $\langle q, \sharp \rangle$ where $q \in F_e$. The intuition behind the transition rules in (β2) is similar to that of (β1). They correspond to the case where $\phi = \neg e$.

Theorem 4. *Given a PDS $\mathcal{P} = (P, \Gamma, \Delta, \sharp)$, a set of atomic propositions AP, a regular labelling function $\lambda : AP \to 2^{P \times \Gamma^*}$ and a BCARET formula φ, we can compute an ABPDS $\mathcal{BP}'_\varphi$ such that for every configuration $\langle p, \omega \rangle$, $\langle p, \omega \rangle \vDash_\lambda \varphi$ iff $\mathcal{BP}'_\varphi$ has an accepting run from the configuration $\langle (\!|p, \varphi|\!), \omega \rangle$*

The number of control locations of $\mathcal{BP}'_\varphi$ is at most $\mathcal{O}(|P||\varphi| + k)$ where $k = \sum_{e \in AP^+(\varphi)} |Q_e| + \sum_{e \in AP^-(\varphi)} |Q_{\neg e}|$, the number of stack symbols is at most $\mathcal{O}(|\Gamma||\varphi|)$ and the number of transitions is at most $\mathcal{O}(|P||\Gamma||\Delta||\varphi| + d)$ where $d = \sum_{e \in AP^+(\varphi)} |\delta_e| + \sum_{e \in AP^-(\varphi)} |\delta_{\neg e}|$. Therefore, we get from Theorems 1 and 4:

Theorem 5. *Given a PDS $\mathcal{P} = (P, \Gamma, \Delta, \sharp)$, a set of atomic propositions AP, a regular labelling function $\lambda : AP \to 2^{P \times \Gamma^*}$ and a BCARET formula φ, for every configuration $\langle p, \omega \rangle \in P \times \Gamma^*$, whether or not $\langle p, \omega \rangle$ satisfies φ can be solved in time $\mathcal{O}((|P||\varphi| + k)^2 . |\Gamma||\varphi|((|P||\Gamma||\Delta||\varphi| + d) . 2^{5(|P||\varphi|+k)} + 2^{|P||\varphi|+k} . |\omega|))$*

7 Conclusion

In this paper, we introduce the Branching temporal logic of CAlls and RETurns BCARET and show how it can be used to describe malicious behaviors that CARET and other specification formalisms cannot. We present an algorithm for "standard" BCARET model checking for PDSs where whether a configuration of a PDS satisfies an atomic proposition or not depends only on the control location of that configuration. Moreover, we consider BCARET model-checking for PDSs with regular valuations where the set of configurations on which an atomic proposition holds is a regular language. Our approach is based on reducing these problems to the emptiness problem of Alternating Büchi Pushdown Systems.

References

1. Rajeev Alur, Michael Benedikt, Kousha Etessami, Patrice Godefroid, Thomas W. Reps, and Mihalis Yannakakis. Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.*, 2005.
2. Rajeev Alur, Swarat Chaudhuri, and P. Madhusudan. A fixpoint calculus for local and global program flows. In *POPL 2006*.
3. Rajeev Alur, Swarat Chaudhuri, and P. Madhusudan. Languages of nested trees. In *CAV 2006*.
4. Rajeev Alur, Swarat Chaudhuri, and P. Madhusudan. Software model checking using languages of nested trees. *ACM Trans. Program. Lang. Syst.*, 2011.
5. Rajeev Alur, Kousha Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *TACAS 2004*.
6. Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR '97*.
7. Laura Bozzelli. Complexity results on branching-time pushdown model checking. In *VMCAI 2006*.
8. Olaf Burkart and Bernhard Steffen. Model checking the full modal mu-calculus for infinite sequential processes. In *ICALP'97*.
9. Javier Esparza, David Hansel, Peter Rossmanith, and Stefan Schwoon. Efficient algorithms for model checking pushdown systems. In *CAV 2000*.
10. Javier Esparza, Antonín Kucera, and Stefan Schwoon. Model checking LTL with regular valuations for pushdown systems. *Inf. Comput.*, 2003.
11. Alain Finkel, Bernard Willems, and Pierre Wolper. A direct symbolic approach to model checking pushdown systems. *Electr. Notes Theor. Comput. Sci.*, 1997.
12. Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. An automata-theoretic approach to infinite-state systems. In *Time for Verification, Essays in Memory of Amir Pnueli 2010*.
13. Huu-Vu Nguyen and Tayssir Touili. BCARET model checking for pushdown systems. In *https://lipn.univ-paris13.fr/~nguyen/BCARETFullVersion.pdf*.
14. Huu-Vu Nguyen and Tayssir Touili. CARET model checking for pushdown systems. In *SAC 2017*.
15. Stefan Schwoon. *Model-Checking Pushdown Systems*. Dissertation, Technische Universität München, München, 2002.
16. Fu Song and Tayssir Touili. Efficient CTL model-checking for pushdown systems. In *CONCUR 2011*.
17. Fu Song and Tayssir Touili. Efficient malware detection using model-checking. In *FM 2012*.
18. Fu Song and Tayssir Touili. LTL model-checking for malware detection. In *TACAS 2013*.
19. Igor Walukiewicz. Pushdown processes: Games and model checking. In *CAV 1996*.