BCARET Model Checking for Malware Detection

Huu-Vu Nguyen(✉) and Tayssir Touili(✉)

LIPN, CNRS and University Paris 13, Villetaneuse, France
nguyen@lipn.univ-paris13.fr, touili@lipn.univ-paris13.fr

Abstract. The number of malware is growing fast recently. Traditional malware detectors based on signature matching and code emulation are easy to bypass. To overcome this problem, model-checking appears as an efficient approach that has been extensively applied for malware detection in recent years. Pushdown systems were proposed as a natural model for programs, as they allow to take into account the program's stack into the model. CARET and BCARET were proposed as formalisms for malicious behavior specification since they can specify properties that require matchings of calls and returns which is crucial for malware detection. In this paper, we propose to use BCARET for malicious behavior specification. Since BCARET formulas for malicious behaviors are huge, we propose to extend BCARET with variables, quantifiers and predicates over the stack. Our new logic is called SBPCARET. We reduce the malware detection problem to the model checking problem of PDSs against SBPCARET formulas, and we propose an efficient algorithm to model check SBPCARET formulas for PDSs.

1 Introduction

The number of malware is growing fast in recent years. Traditional approaches including signature matching and code emulation are not efficient enough to detect malwares. While attackers can use obfuscation techniques to hide their malware from the signature based malware detectors easily, the code emulation approaches can only track programs in certain execution paths due to the limited execution time. To overcome these limitations, model-checking appears as an efficient approach for malware detection, since model-checking allows to check the behaviors of a program in all its execution traces without executing it.

A lof of works have been investigated to apply model-checking for malware detection [2–4, 7, 10–12]. [4] proposed to use finite state graphs to model the program and use the temporal logic CTPL to specify malicious behaviours. However, finite graphs are not exact enough to model programs, as they don't allow to take into account the program's stack into the model. Indeed, the program's stack is usually used by malware writers for code obfuscation as explained in [5]. In addition, in binary codes and assembly programs, parameters are passed to functions by pushing them on the stack before the call is made. The values of these parameters are used to determine whether the program is malicious or not [6]. Therefore, being able to record the program's stack is critical for malware detection. To this aim, [10–13] proposed to use pushdown systems to model programs, and defined extensions of LTL and CTL (called SLTPL and SCTPL) to precisely and compactly describe malicious behaviors. However, these logics cannot specify properties that require matchings of calls and returns, which is crucial to describe malicious behaviours [8]. Let us consider the typical behaviour of a spyware to illustrate this. The typical behaviour of a spyware is seeking personal information (emails, bank account information,...) on local drives by searching files that match specific conditions. To do that, it has to search directories of the host to look for interesting files whose names match a certain condition. If a file is found, the spyware will invoke a payload to steal the information, then continue looking for the remaining matching files. If a folder is found, it will pass into the folder path and continue investigating the folder recursively. To obtain this behavior, the spyware first calls the API $FindFirstFileA$ 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 $FindFirstFileA$ is successful or not. When the function call fails, the spyware will call the API $GetLastError$. Otherwise, when the function call succeeds, a search handle h will be returned by $FindFirstFileA$. 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 function $FindNextFileA$ using h as first parameter to look for the remaining matching files. This behavior cannot be described by LTL or CTL since it requires to express that the return value of the API function $FindFirstFileA$ should be used as input to the function $FindNextFileA$.

CARET was introduced to express linear-temporal properties that involve matchings of calls and returns [1] and CARET model-checking for PDSs was considered [6,7]. However, the above behaviour cannot be described by CARET since it is a branching-time property. To specify that behaviour naturally and intuitively, BCARET was introduced to express these branching-time properties that involve matchings of calls and returns [8]. Using BCARET, the above behavior can be expressed by the following formula:

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

where the $\bigvee$ is taken over all possible memory addresses d that contain the values of search handles h in the program, EX^a is a BCARET operator saying that "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 stating that "eventually in all runs, in the same procedural context".

In binary codes and assembly programs, the return value of an API function is placed in the register eax. Therefore, the return value of $FindFirstFileA$ is the value of the register eax at the corresponding return-point of the call. Then, the subformula (call(FindFirstFileA) $\wedge$ $EX^a(eax = d)$) expresses that there is a call to the API function *FindFirstFileA* whose return value is d (the abstract successor of a call is its corresponding return-point). A call to *FindNextFileA* requires a search handle h as parameter and h must be put on top of the program's stack (as parameters are passed through the stack in assembly programs). To express that d is on top of the program stack, we use the regular expression $d\Gamma^*$. Thus, the subformula [call(FindNextFileA) $\wedge$ $d\Gamma^*$] states that the API *FindNextFileA* is invoked with d as parameter (d stores the information of the search handle h). Therefore, φ_{sb} states that there is a call to the function $FindFirstFileA$ whose return value is d (the search handle), then, in all runs starting from that call, there will be either a call to the API $GetLastError$ or a call to the API function $FindFirstFileA$ or a call to the function $FindNextFileA$ in which d is used as a parameter.

However, it can be seen that this formula is huge, since it considers the disjunction (of different BCARET formulas) over all possible memory addresses d which contain the information of search handles h in the program. To represent it in a more compact fashion, we follow the idea of [4,6,10,12] and extend BCARET with variables, quantifiers, and predicates over the stack. We call our new logic SBPCARET. The above formula can be concisely described by a SBPCARET formula as follows:

$$\varphi'_{sb} = \exists \boldsymbol{x} EF^g\Bigg(call(FindFirstFileA) \wedge EX^a(eax = x) \wedge AF^a \\ \Bigg(call(GetLastError) \vee call(FindFirstFileA) \\ \vee \Big(call(FindNextFileA) \wedge x\Gamma^*\Big)\Bigg)\Bigg)$$

Thus, we propose in this work to use pushdown systems (PDSs) to model programs, and SBPCARET formulas to specify malicious behaviors.We reduce the malware detection problem to the model checking problem of PDSs against SBPCARET formulas, and we propose an efficient algorithm to check whether a PDS satisfies a SBPCARET formula. Our algorithm is based on a reduction to the emptiness problem of Symbolic Alternating Büchi Pushdown Systems. This latter problem is already solved in [10].

The rest of paper is organized as follows. In Sect. 2, we recall the definitions of Pushdown Systems. Section 3 introduces our logic SBPCARET. Model checking SBPCARET for PDSs is presented in Sect. 4. Finally, we conclude in Sect. 5.

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 [9]. 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 [8], 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 $\gamma \xrightarrow{call\ \ proc} \gamma_2$. 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 $\perp$.

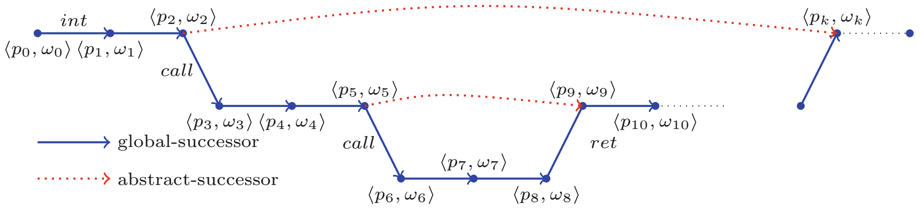

int
$\langle p_2, \omega_2\rangle$
$\langle p_k, \omega_k\rangle$
$\langle p_0, \omega_0\rangle$ $\langle p_1, \omega_1\rangle$
call
$\langle p_5, \omega_5\rangle$
$\langle p_9, \omega_9\rangle$
$\langle p_3, \omega_3\rangle$ $\langle p_4, \omega_4\rangle$
$\langle p_{10}, \omega_{10}\rangle$
global-successor
call
ret
$\langle p_7, \omega_7\rangle$
abstract-successor
$\langle p_6, \omega_6\rangle$
$\langle p_8, \omega_8\rangle$

Fig. 1. Two kinds of successors on a run

For example, in Fig. 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 Fig. 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.

3 Malicious Behaviour Specification

In this section, we define the Stack Branching temporal Predicate logic of CAlls and RETurns (SBPCARET) as an extension of BCARET [8] with variables and regular predicates over the stack contents. The predicates contain variables that can be quantified existentially or universally. Regular predicates are expressed by regular variable expressions and are used to describe the stack content of PDSs.

3.1 Environments, Predicates and Regular Variable Expressions

Let $\mathcal{X} = \{x_1, ..., x_n\}$ be a finite set of variables over a finite domain $\mathcal{D}$. Let $B : \mathcal{X} \cup \mathcal{D} \rightarrow \mathcal{D}$ be an environment that associates each variable $x \in \mathcal{X}$ with a value $d \in \mathcal{D}$ s.t $B(d) = d$ for every $d \in \mathcal{D}$. Let $B[x \leftarrow d]$ be an environment obtained from B such that $B[x \leftarrow d](x) = d$ and $B[x \leftarrow d](y) = B(y)$ for every $y \neq x$. Let $Abs_x(B) = \{B' \in \mathcal{B} \mid \forall y \in \mathcal{X}, y \neq x, B(y) = B'(y)\}$ be the function that abstracts away the value of x. Let $\mathcal{B}$ be the set of all environments.

Let $AP = \{a, b, c, ...\}$ be a finite set of atomic propositions. Let $AP_{\mathcal{D}}$ be a finite set of atomic predicates of the form $b(\alpha_1, ..., \alpha_m)$ such that $b \in AP$ and $\alpha_i \in \mathcal{D}$ for every $1 \leq i \leq m$. Let $AP_{\mathcal{X}}$ be a finite set of atomic predicates $b(\alpha_1, ..., \alpha_n)$ such that $b \in AP$ and $\alpha_i \in \mathcal{X} \cup \mathcal{D}$ for every $1 \leq i \leq n$.

Let $\mathcal{P} = (P, \Gamma, \Delta)$ be a Labelled PDS. A Regular Variable Expression (RVE) e over $\mathcal{X} \cup \Gamma$ is defined by $e ::= \epsilon \mid a \in \mathcal{X} \cup \Gamma \mid e + e \mid e.e \mid e^*$. The language $L(e)$ of a RVE e is a subset of $P \times \Gamma^* \times \mathcal{B}$ and is defined as follows:

- $L(\epsilon) = \{(\langle p, \epsilon\rangle, B) \mid p \in P, B \in \mathcal{B}\}$
- for $x \in \mathcal{X}$, $L(x) = \{(\langle p, \gamma\rangle, B) \mid p \in P, \gamma \in \Gamma, B \in \mathcal{B}$ s.t $B(x) = \gamma\}$
- for $\gamma \in \Gamma$, $L(\gamma) = \{(\langle p, \gamma\rangle, B) \mid p \in P, B \in \mathcal{B}\}$
- $L(e_1.e_2) = \{(\langle p, \omega'\omega''\rangle, B) \mid (\langle p, \omega'\rangle, B) \in L(e_1); (\langle p, \omega''\rangle, B) \in L(e_2)\}$
- $L(e^*) = \{(\langle p, \omega\rangle, B) \mid \omega \in \{v \in \Gamma^* \mid (\langle p, v\rangle, B) \in L(e)\}^*\}$.

3.2 The Stack Branching Temporal Predicate Logic of CAlls and RETurns - SBPCARET

A SBPCARET formula is a BCARET formula where predicates and RVEs are used as atomic propositions and where quantifiers are applied to variables. For technical reasons, we assume w.l.o.g. that formulas are written in positive normal form, where negations are applied only to atomic predicates, and we use the *release operator* R as the dual of the until operator U. From now on, we fix a finite set of variables $\mathcal{X}$, a finite set of atomic propositions AP, a finite domain $\mathcal{D}$, and a finite set of RVEs $\mathcal{V}$. A SBPCARET formula is defined as follows, where $v \in \{g, a\}$, $x \in \mathcal{X}$, $e \in \mathcal{V}$, $b(\alpha_1, ..., \alpha_n) \in AP_{\mathcal{X}}$:

$$\begin{aligned}\varphi := &\, true \mid false \mid b(\alpha_1, ..., \alpha_n) \mid \neg b(\alpha_1, ..., \alpha_n) \mid e \mid \neg e \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \forall x \varphi \mid \\ &\exists x \varphi \mid EX^v \varphi \mid AX^v \varphi \mid E[\varphi U^v \varphi] \mid A[\varphi U^v \varphi] \mid E[\varphi R^v \varphi] \mid A[\varphi R^v \varphi]\end{aligned}$$

Let $\lambda : P \longrightarrow 2^{AP_{\mathcal{D}}}$ be a labelling function which associates each control location to a set of atomic predicates. Let φ be a SBPCARET formula over AP. Let $\langle p, \omega\rangle$ be a configuration of $\mathcal{P}$. Then we say that $\mathcal{P}$ satisfies φ at $\langle p, \omega\rangle$ (denoted by $\langle p, \omega\rangle \models_\lambda \varphi$) iff there exists an environment $B \in \mathcal{B}$ such that $\langle p, \omega\rangle$ satisfies φ under B (denoted by $\langle p, \omega\rangle \models_\lambda^B \varphi$). The satisfiability relation of a SBPCARET formula φ at a configuration $\langle p_0, \omega_0\rangle$ under the environment B w.r.t. the labelling function λ, denoted by $\langle p_0, \omega_0\rangle \vDash_\lambda^B \varphi$, is defined inductively as follows:

- $\langle p_0, \omega_0\rangle \vDash_\lambda^B true$ for every $\langle p_0, \omega_0\rangle$
- $\langle p_0, \omega_0\rangle \nvDash_\lambda^B false$ for every $\langle p_0, \omega_0\rangle$
- $\langle p_0, \omega_0\rangle \vDash_\lambda^B b(\alpha_1, ..., \alpha_n)$, iff $b(B(\alpha_1), ..., B(\alpha_n)) \in \lambda(p_0)$

- $\langle p_0, \omega_0\rangle \models^B_\lambda \neg b(\alpha_1, ..., \alpha_n)$, iff $b(B(\alpha_1), ..., B(\alpha_n)) \notin \lambda(p_0)$
- $\langle p_0, \omega_0\rangle \models^B_\lambda e$ iff $(\langle p_0, \omega_0\rangle, B) \in L(e)$
- $\langle p_0, \omega_0\rangle \models^B_\lambda \neg e$ iff $(\langle p_0, \omega_0\rangle, B) \notin L(e)$
- $\langle p_0, \omega_0\rangle \models^B_\lambda \varphi_1 \vee \varphi_2$ iff $(\langle p_0, \omega_0\rangle \models^B_\lambda \varphi_1$ or $\langle p_0, \omega_0\rangle \models^B_\lambda \varphi_2)$
- $\langle p_0, \omega_0\rangle \models^B_\lambda \varphi_1 \wedge \varphi_2$ iff $(\langle p_0, \omega_0\rangle \models^B_\lambda \varphi_1$ and $\langle p_0, \omega_0\rangle \models^B_\lambda \varphi_2)$
- $\langle p_0, \omega_0\rangle \models^B_\lambda \forall x\varphi$ iff for every $d \in \mathcal{D}$, $\langle p_0, \omega_0\rangle \models^{B[x\leftarrow d]}_\lambda \varphi$
- $\langle p_0, \omega_0\rangle \models^B_\lambda \exists x\varphi$ iff there exists $d \in \mathcal{D}$, $\langle p_0, \omega_0\rangle \models^{B[x\leftarrow d]}_\lambda \varphi$
- $\langle p_0, \omega_0\rangle \models^B_\lambda EX^g\varphi$ iff there exists a global-successor $\langle p', \omega'\rangle$ of $\langle p_0, \omega_0\rangle$ such that $\langle p', \omega'\rangle \models^B_\lambda \varphi$
- $\langle p_0, \omega_0\rangle \models^B_\lambda AX^g\varphi$ iff $\langle p', \omega'\rangle \models^B_\lambda \varphi$ for every global-successor $\langle p', \omega'\rangle$ of $\langle p_0, \omega_0\rangle$
- $\langle p_0, \omega_0\rangle \models^B_\lambda E[\varphi_1 U^g \varphi_2]$ 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. $\exists i \geq 0$, $\langle p_i, \omega_i\rangle \models^B_\lambda \varphi_2$ and for every $0 \leq j < i$, $\langle p_j, \omega_j\rangle \models^B_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models^B_\lambda A[\varphi_1 U^g \varphi_2]$ 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 $\langle p_0, \omega_0\rangle$, $\exists i \geq 0$, $\langle p_i, \omega_i\rangle \models^B_\lambda \varphi_2$ and for every $0 \leq j < i$, $\langle p_j, \omega_j\rangle \models^B_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models^B_\lambda E[\varphi_1 R^g \varphi_2]$ 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 $\langle p_i, \omega_i\rangle \not\models^B_\lambda \varphi_2$ then there exists $0 \leq j < i$ s.t. $\langle p_j, \omega_j\rangle \models^B_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models^B_\lambda A[\varphi_1 R^g \varphi_2]$ 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 $\langle p_i, \omega_i\rangle \not\models^B_\lambda \varphi_2$ then there exists $0 \leq j < i$ s.t. $\langle p_j, \omega_j\rangle \models^B_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models^B_\lambda EX^a\varphi$ iff there exists an abstract-successor $\langle p', \omega'\rangle$ of $\langle p_0, \omega_0\rangle$ such that $\langle p', \omega'\rangle \models^B_\lambda \varphi$
- $\langle p_0, \omega_0\rangle \models^B_\lambda AX^a\varphi$ iff $\langle p', \omega'\rangle \models^B_\lambda \varphi$ for every abstract-successor $\langle p', \omega'\rangle$ of $\langle p_0, \omega_0\rangle$
- $\langle p_0, \omega_0\rangle \models^B_\lambda E[\varphi_1 U^a \varphi_2]$ 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. $\exists i \geq 0$, $\langle p_i, \omega_i\rangle \models^B_\lambda \varphi_2$ and for every $0 \leq j < i$, $\langle p_j, \omega_j\rangle \models^B_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models^B_\lambda A[\varphi_1 U^a \varphi_2]$ 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 $\mathcal{P}$, $\exists i \geq 0$, $\langle p_i, \omega_i\rangle \models^B_\lambda \varphi_2$ and for every $0 \leq j < i$, $\langle p_j, \omega_j\rangle \models^B_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models^B_\lambda E[\varphi_1 R^a \varphi_2]$ 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 $\langle p_i, \omega_i\rangle \not\models^B_\lambda \varphi_2$ then there exists $0 \leq j < i$ s.t. $\langle p_j, \omega_j\rangle \models^B_\lambda \varphi_1$
- $\langle p_0, \omega_0\rangle \models^B_\lambda A[\varphi_1 R^a \varphi_2]$ 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 $\langle p_i, \omega_i\rangle \not\models^B_\lambda \varphi_2$ then there exists $0 \leq j < i$ s.t. $\langle p_j, \omega_j\rangle \models^B_\lambda \varphi_1$

Other SBPCARET 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 SBPCARET formula φ, the closure $Cl(\varphi)$ is the set of all subformulae of φ, including φ. Let $AP^+(\varphi) = \{b(\alpha_1, ..., \alpha_n) \in AP_\mathcal{X} \mid b(\alpha_1, ..., \alpha_n) \in Cl(\varphi)\}$; $AP^-(\varphi) = \{b(\alpha_1, ..., \alpha_n) \in AP_\mathcal{X} \mid \neg b(\alpha_1, ..., \alpha_n) \in Cl(\varphi)\}$, $Reg^+(\varphi) = \{e \in \mathcal{V} \mid e \in Cl(\varphi)\}$, $Reg^-(\varphi) = \{e \in \mathcal{V} \mid \neg e \in Cl(\varphi)\}$.

4 SBPCARET Model-Checking for Pushdown Systems

In this section, we show how to do SBPCARET model-checking for PDSs. Let then $\mathcal{P}$ be a PDS, φ be a SBPCARET formula, and $\mathcal{V}$ be the set of RVEs occurring in φ. We follow the idea of [10] and use Variable Automata to represent RVEs.

4.1 Variable Automata

Given a PDS $\mathcal{P} = (P, \Gamma, \Delta)$ s.t. $\Gamma \subseteq \mathcal{D}$, a Variable Automaton (VA) [10] is a tuple $(Q, \Gamma, \delta, s, F)$, where Q is a finite set of states, Γ is the input alphabet, $s \in Q$ is an initial state; $F \subseteq Q$ is a finite set of accepting states; and δ is a finite set of transition rules of the form $p \xrightarrow{\alpha} \{q_1, ..., q_n\}$ where α can be x, $\neg x$, or γ, for any $x \in \mathcal{X}$ and $\gamma \in \Gamma$.

Let $B \in \mathcal{B}$. A run of VA on a word $\gamma_1, ..., \gamma_m$ under B is a tree of height m whose root is labelled by the initial state s, and each node at depth k labelled by a state q has h children labelled by $p_1, ..., p_h$ respectively, such that:

- either $q \xrightarrow{\gamma_k} \{p_1, ..., p_h\} \in \delta$ and $\gamma_k \in \Gamma$;
- or $q \xrightarrow{x} \{p_1, ..., p_h\} \in \delta$, $x \in \mathcal{X}$ and $B(x) = \gamma_k$;
- or $q \xrightarrow{\neg x} \{p_1, ..., p_h\} \in \delta$, $x \in \mathcal{X}$ and $B(x) \neq \gamma_k$.

A branch of the tree is accepting iff the leaf of the branch is an accepting state. A run is accepting iff all its branches are accepting. A word $\omega \in \Gamma^*$ is accepted by a VA under an environment $B \in \mathcal{B}$ iff the VA has an accepting run on the word ω under the environment B.

The language of a VA M, denoted by $L(M)$, is a subset of $(P \times \Gamma^*) \times \mathcal{B}$. $(\langle p, \omega \rangle, B) \in L(M)$ iff M accepts the word ω under the environment B.

Theorem 1. *[10] For every regular expression $e \in \mathcal{V}$, we can compute in polynomial time a Variable Automaton M s.t. $L(M) = L(e)$.*

Theorem 2. *[10] VAs are closed under boolean operations.*

4.2 Symbolic Alternating Büchi Pushdown Systems (SABPDSs)

Definition 2. *A Symbolic Alternating Büchi Pushdown System (SABPDS) is a tuple $\mathcal{BP} = (P, \Gamma, \Delta, F)$, where P is a set of control locations, $\Gamma \subseteq \mathcal{D}$ is stack alphabet, $F \subseteq P \times 2^{\mathcal{B}}$ is a set of accepting control locations and Δ is a finite set of transitions of the form $\langle p, \gamma \rangle \overset{\mathbb{R}}{\hookrightarrow} \{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\}$ where $p \in P$, $\gamma \in \Gamma$, for every $1 \leq i \leq n$: $p_i \in P$, $\omega_i \in \Gamma^*$; and $\mathbb{R} : (\mathcal{B})^n \to 2^{\mathcal{B}}$ is a function that maps a tuple of environments $(B_1, ..., B_n)$ to a set of environments.*

A configuration of a SABPDS $\mathcal{BP}$ is a tuple $\langle [\![p, B]\!], \omega \rangle$, where $p \in P$ is the current control location, $B \in \mathcal{B}$ is an environment and $\omega \in \Gamma^*$ is the current stack content. Let $\langle p, \gamma \rangle \overset{\mathbb{R}}{\hookrightarrow} \{\langle p_1, \omega_1 \rangle, ..., \langle p_n, \omega_n \rangle\}$ be a rule of Δ, then, for every $\omega \in$

Γ^*, $B, B_1, ..., B_n \in \mathcal{B}$, if $B \in \mathbb{R}(B_1, ..., B_n)$, then the configuration $\langle [\![p, B]\!], \gamma\omega \rangle$ (resp. $\{\langle [\![p_1, B_1]\!], \omega_1\omega \rangle, ..., \langle [\![p_n, B_n]\!], \omega_n\omega \rangle\}$) is an immediate predecessor (resp. successor) of $\{\langle [\![p_1, B_1]\!], \omega_1\omega \rangle, ..., \langle [\![p_n, B_n]\!], \omega_n\omega \rangle\}$ (resp. $\langle [\![p, B]\!], \gamma\omega \rangle$).

A run ρ of a SABPDS $\mathcal{BP}$ starting form an initial configuration $\langle [\![p_0, B_0]\!], \omega_0 \rangle$ is a tree whose root is labelled by $\langle [\![p_0, B_0]\!], \omega_0 \rangle$, and whose other nodes are labelled by elements in $P \times \mathcal{B} \times \Gamma^*$. If a node of ρ is labelled by a configuration $\langle [\![p, B]\!], \omega \rangle$ and has n children labelled by $\langle [\![p_1, B_1]\!], \omega_1 \rangle, ..., \langle [\![p_n, B_n]\!], \omega_n \rangle$ respectively, then, $\langle [\![p, B]\!], \omega \rangle$ must be a predecessor of $\{\langle [\![p_1, B_1]\!], \omega_1 \rangle, ..., \langle [\![p_n, B_n]\!], \omega_n \rangle\}$ in $\mathcal{BP}$. A path of a run ρ is an infinite sequence of configurations $c_0c_1c_2...$ 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 $\Rightarrow_{\mathcal{BP}}$: $2^{(P\times\mathcal{B})\times\Gamma^*} \rightarrow 2^{(P\times\mathcal{B})\times\Gamma^*}$ as follows: (1) $c \Rightarrow_{\mathcal{BP}} \{c\}$ for every $c \in P \times \mathcal{B} \times \Gamma^*$, (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 3. *[10] The membership problem of SABPDS can be solved effectively.*

Functions of $\mathbb{R}$. In what follows, we define several functions of $\mathbb{R}$ which will be used in the next sections. These functions were first defined in [10].

1. $id(B) = \{B\}$. This is the identity function.
2.
$$equal(B_1, ..., B_n) = \begin{cases} B_1 & \text{if } B_i = B_j \text{ for every } 1 \leq i, j \leq n; \\ \emptyset & \text{otherwise} \end{cases}$$

   This function checks whether all the environments are equal and returns $\{B_1\}$ (which is also equal to B_i for every i). Otherwise, it returns the emptyset.
3.
$$meet^x_{\{c_1,...,c_n\}}(B_1, ..., B_n) = \begin{cases} Abs_x(B_1) & \text{if } B_i(x) = c_i \text{ for } 1 \leq i \leq n, \\ & \text{ and } B_i(y) = B_j(y) \text{ for } y \neq x, 1 \leq i, j \leq n; \\ \emptyset & \text{otherwise} \end{cases}$$

   This function checks whether (1) $B_i(x) = c_i$ for every $1 \leq i \leq n$ (2) for every $y \neq x$; every $1 \leq i, j \leq n$ $B_i(y) = B_j(y)$. If the conditions are satisfied, it returns $Abs_x(B_1)$[1], otherwise it returns the emptyset.

[1] $Abs_x(B_1)$ is as defined in Sect. 3.1.

4.
$$join_c^x(B_1,...,B_n) = \begin{cases} B_1 & \text{if } B_i(x) = c \text{ for } 1 \leq i \leq n \\ & \text{ and } B_i = B_j \text{ for } 1 \leq i,j \leq n; \\ \emptyset & \text{otherwise} \end{cases}$$

This function checks whether $B_i(x) = c$ for every i. If this condition is satisfied, $equal(B_1,...,B_n)$ is returned, otherwise, the emptyset is returned.

5.
$$join_c^{\neg x}(B_1,...,B_n) = \begin{cases} B_1 & \text{if } B_i(x) \neq c \text{ for } 1 \leq i \leq n \\ & \text{ and } B_i = B_j \text{ for } 1 \leq i,j \leq n; \\ \emptyset & \text{otherwise} \end{cases}$$

This function checks whether $B_i(x) \neq c$ for every i. If this condition is satisfied, $equal(B_1,...,B_n)$ is returned, otherwise, the emptyset is returned.

4.3 From SBPCARET Model Checking of PDSs to the Membership Problem in SABPDSs

Let $\mathcal{P} = (P, \Gamma, \Delta)$ be a PDS. We suppose w.l.o.g. that $\mathcal{P}$ has a bottom stack symbol $\sharp$ that is never popped from the stack. Let AP be a set of atomic propositions. Let φ be a SBPCARET formula over AP, $\lambda : P \longrightarrow 2^{AP_{\mathcal{D}}}$ be a labelling function. Given a configuration $\langle p_0, \omega_0 \rangle$, we propose in this section an algorithm to check whether $\langle p_0, \omega_0 \rangle \models_\lambda \varphi$, i.e., whether there exists an environment B s.t. $\langle p_0, \omega_0 \rangle \models_\lambda^B \varphi$. Intuitively, we compute an SABPDS $\mathcal{BP}_\varphi$ s.t. $\langle p, \omega \rangle \models_\lambda^B \varphi$ iff $\langle [\![(\!| p, \varphi |\!), B]\!], \omega \rangle \in \mathcal{L}(\mathcal{BP}_\varphi)$ for every $p \in P$, $\omega \in \Gamma^*$, $B \in \mathcal{B}$. Then, to check if $\langle p_0, \omega_0 \rangle \models_\lambda \varphi$, we will check whether there exists a $B \in \mathcal{B}$ s.t. $\langle [\![(\!| p_0, \varphi |\!), B]\!], \omega_0 \rangle \in \mathcal{L}(\mathcal{BP}_\varphi)$.

Let $Reg^+(\varphi) = \{e_1, ..., e_k\}$ and $Reg^-(\varphi) = \{e_{k+1}, ..., e_m\}$. Using Theorems 1 and 2; for every $1 \leq i \leq k$, we can compute a VA $M_{e_i} = (Q_{e_i}, \Gamma, \delta_{e_i}, s_{e_i}, F_{e_i})$ s.t. $L(M_{e_i}) = L(e_i)$. In addition, for every $k+1 \leq j \leq m$, we can compute a VA $M_{\neg e_j} = (Q_{\neg e_j}, \Gamma, \delta_{\neg e_j}, s_{\neg e_j}, F_{\neg e_j})$ s.t. $L(M_{\neg e_j}) = (P \times \Gamma^*) \times \mathcal{B} \setminus L(e_j)$. Let $\mathcal{M}$ be the union of all these automata, $\mathcal{S}$ and $\mathcal{F}$ be respectively the union of all states and final states of these automata.

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

- $P' = P \cup (P \times Cl(\varphi)) \cup \mathcal{S} \cup \{p_\perp\}$
- $\Gamma' = \Gamma \cup (\Gamma \times Cl(\varphi)) \cup \{\gamma_\perp\}$
- $F = F_1 \cup F_2 \cup F_3 \cup F_4$ where
  - $F_1 = \{ [\![(\!| p, b(\alpha_1,...,\alpha_n) |\!), \beta]\!] \mid b(\alpha_1,...,\alpha_n) \in AP^+(\varphi)$, and $\beta = \{B \in \mathcal{B} \mid b(B(\alpha_1),...,B(\alpha_n)) \in \lambda(p)\}$
  - $F_2 = \{ [\![(\!| p, \neg b(\alpha_1,...,\alpha_n) |\!), \beta]\!] \mid b(\alpha_1,...,\alpha_n) \in AP^-(\varphi)$, and $\beta = \{B \in \mathcal{B} \mid b(B(\alpha_1),...,B(\alpha_n)) \notin \lambda(p)\}$

- $F_3 = P \times Cl_R(\varphi) \times \mathcal{B}$ where $Cl_R(\varphi)$ is the set of formulas of $Cl(\varphi)$ in the form $E[\varphi_1 R^v \varphi_2]$ or $A[\varphi_1 R^v \varphi_2]$ $(v \in \{g, a\})$
- $F_4 = \mathcal{F} \times \mathcal{B}$

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

($\hbar$1) If $\phi = b(\alpha_1, ..., \alpha_n)$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{id} \langle (\!|p, \phi|\!), \gamma\rangle \in \Delta'$

($\hbar$2) If $\phi = \neg b(\alpha_1, ..., \alpha_n)$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{id} \langle (\!|p, \phi|\!), \gamma\rangle \in \Delta'$

($\hbar$3) If $\phi = \phi_1 \wedge \phi_2$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p, \phi_1|\!), \gamma\rangle, \langle (\!|p, \phi_2|\!), \gamma\rangle] \in \Delta'$

($\hbar$4) If $\phi = \phi_1 \vee \phi_2$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{id} \langle (\!|p, \phi_1|\!), \gamma\rangle \in \Delta'$ and $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{id} \langle (\!|p, \phi_2|\!), \gamma\rangle \in \Delta'$

($\hbar$5) If $\phi = \exists x \phi_1$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{meet^x_{\{c\}}} \langle (\!|p, \phi_1|\!), \gamma\rangle \in \Delta'$ for every $c \in \mathcal{D}$

($\hbar$6) If $\phi = \forall x \phi_1$, then, $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{meet^x_{\mathcal{D}}} [\langle (\!|p, \phi_1|\!), \gamma\rangle, ..., \langle (\!|p, \phi_1|\!), \gamma\rangle] \in \Delta'$ where $\langle (\!|p, \phi_1|\!), \gamma\rangle$ is repeated m times in the right-hand side, where m is the number of elements in $\mathcal{D}$

($\hbar$7) If $\phi = EX^g \phi_1$, then
$\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{id} \langle (\!|q, \phi_1|\!), \omega\rangle \in \Delta'$ for every $\langle p, \gamma\rangle \xrightarrow{t} \langle q, \omega\rangle \in \Delta$

($\hbar$8) If $\phi = AX^g \phi_1$, then,
$\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|q_1, \phi_1|\!), \omega_1\rangle, ..., \langle (\!|q_n, \phi_1|\!), \omega_n\rangle] \in \Delta'$, where for every $1 \leq i \leq n$, $\langle p, \gamma\rangle \xrightarrow{t} \langle q_i, \omega_i\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p, \gamma\rangle \xrightarrow{t} \langle q, \omega\rangle$ that have $\langle p, \gamma\rangle$ on the left hand side.

($\hbar$9) If $\phi = EX^a \phi_1$, then,

(a) $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{id} \langle q, \gamma' (\!|\gamma'', \phi_1|\!)\rangle \in \Delta'$ for every $\langle p, \gamma\rangle \xrightarrow{call} \langle q, \gamma'\gamma''\rangle \in \Delta$

(b) $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{id} \langle (\!|q, \phi_1|\!), \omega\rangle \in \Delta'$ for every $\langle p, \gamma\rangle \xrightarrow{int} \langle q, \omega\rangle \in \Delta$

(c) $\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{id} \langle p_\perp, \gamma_\perp\rangle \in \Delta'$ for every $\langle p, \gamma\rangle \xrightarrow{ret} \langle q', \epsilon\rangle \in \Delta$

($\hbar$10) If $\phi = AX^a \phi_1$, then,
$\langle (\!|p, \phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle p_1, \gamma'_1 (\!|\gamma''_1, \phi_1|\!)\rangle, ..., \langle p_m, \gamma'_m (\!|\gamma''_m, \phi_1|\!)\rangle, \langle (\!|q_1, \phi_1|\!), \omega_1\rangle, ..., \langle (\!|q_n, \phi_1|\!), \omega_n\rangle, \langle p_\perp, \gamma_\perp\rangle, ..., \langle p_\perp, \gamma_\perp\rangle] \in \Delta'$, where $\langle p_\perp, \gamma_\perp\rangle$ is repeated k times in the right-hand side s.t.:

(a) for every $1 \leq i \leq m$, $\langle p, \gamma\rangle \xrightarrow{call} \langle p_i, \gamma'_i\gamma''_i\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p, \gamma\rangle \xrightarrow{call} \langle q, \gamma'\gamma''\rangle$ that have $\langle p, \gamma\rangle$ on the left hand side.

(b) for every $1 \leq i \leq n$, $\langle p, \gamma\rangle \xrightarrow{int} \langle q_i, \omega_i\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p, \gamma\rangle \xrightarrow{int} \langle q, \omega\rangle$ that have $\langle p, \gamma\rangle$ on the left hand side.

(c) for every $1 \leq i \leq k$, $\langle p, \gamma\rangle \xrightarrow{ret} \langle q'_i, \epsilon\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p, \gamma\rangle \xrightarrow{ret} \langle q', \epsilon\rangle$ that have $\langle p, \gamma\rangle$ on the left hand side.

($\hbar$11) If $\phi = E[\phi_1 U^g \phi_2]$, then,

(a) $\langle (\!|p,\phi|\!), \gamma\rangle \overset{id}{\hookrightarrow} \langle (\!|p,\phi_2|\!), \gamma\rangle \in \Delta'$

(b) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_1|\!), \gamma\rangle, \langle (\!|q,\phi|\!), \omega\rangle] \in \Delta'$ for every $\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta$

(ℏ12) If $\phi = E[\phi_1 U^a \phi_2]$, then,

(a) $\langle (\!|p,\phi|\!), \gamma\rangle \overset{id}{\hookrightarrow} \langle (\!|p,\phi_2|\!), \gamma\rangle \in \Delta'$

(b) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_1|\!), \gamma\rangle, \langle q, \gamma'(\!|\gamma'',\phi|\!)\rangle] \in \Delta'$ for every $\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma'\gamma''\rangle \in \Delta$

(c) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_1|\!), \gamma\rangle, \langle (\!|q,\phi|\!), \omega\rangle] \in \Delta'$ for every $\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle \in \Delta$

(d) $\langle (\!|p,\phi|\!), \gamma\rangle \overset{id}{\hookrightarrow} \langle p_\bot, \gamma_\bot\rangle \in \Delta'$ for every $\langle p,\gamma\rangle \xrightarrow{ret} \langle q',\epsilon\rangle \in \Delta$

(ℏ13) If $\phi = A[\phi_1 U^g \phi_2]$, then,

(a) $\langle (\!|p,\phi|\!), \gamma\rangle \overset{id}{\hookrightarrow} \langle (\!|p,\phi_2|\!), \gamma\rangle \in \Delta'$

(b) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_1|\!), \gamma\rangle; \langle (\!|q_1,\phi|\!), \omega_1\rangle, ..., \langle (\!|q_n,\phi|\!), \omega_n\rangle] \in \Delta'$ where for every $1 \leq i \leq n$,$\langle p,\gamma\rangle \xrightarrow{t} \langle q_i,\omega_i\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle$ that have $\langle p,\gamma\rangle$ on the left hand side.

(ℏ14) If $\phi = A[\phi_1 U^a \phi_2]$, then,

(a) $\langle (\!|p,\phi|\!), \gamma\rangle \overset{id}{\hookrightarrow} \langle (\!|p,\phi_2|\!), \gamma\rangle \in \Delta'$

(b) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_1|\!), \gamma\rangle; \langle p_1, \gamma_1'(\!|\gamma_1'',\phi|\!)\rangle, ..., \langle p_m, \gamma_m'(\!|\gamma_m'',\phi|\!)\rangle;$ $\langle (\!|q_1,\phi|\!), \omega_1\rangle, ..., \langle (\!|q_n,\phi|\!), \omega_n\rangle, \langle p_\bot,\gamma_\bot\rangle, ..., \langle p_\bot,\gamma_\bot\rangle] \in \Delta'$, where $\langle p_\bot,\gamma_\bot\rangle$ is repeated k times in the right-hand side s.t.:

- for every $1 \leq i \leq m$, $\langle p,\gamma\rangle \xrightarrow{call} \langle p_i,\gamma_i'\gamma_i''\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma'\gamma''\rangle$ that have $\langle p,\gamma\rangle$ on the left hand side.
- for every $1 \leq i \leq n$, $\langle p,\gamma\rangle \xrightarrow{int} \langle q_i,\omega_i\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle$ that have $\langle p,\gamma\rangle$ on the left hand side.
- for every $1 \leq i \leq k$, $\langle p,\gamma\rangle \xrightarrow{ret} \langle q_i',\epsilon\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p,\gamma\rangle \xrightarrow{ret} \langle q',\epsilon\rangle$ that have $\langle p,\gamma\rangle$ on the left hand side.

(ℏ15) If $\phi = E[\phi_1 R^g \phi_2]$, then, we add to Δ' the rule:

(a) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!), \gamma\rangle, \langle (\!|p,\phi_1|\!), \gamma\rangle] \in \Delta'$

(b) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!), \gamma\rangle, \langle (\!|q,\phi|\!), \omega\rangle] \in \Delta'$ for every $\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta$

(ℏ16) If $\phi = A[\phi_1 R^g \phi_2]$, then, we add to Δ' the rule:

(a) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!), \gamma\rangle, \langle (\!|p,\phi_1|\!), \gamma\rangle] \in \Delta'$

(b) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!), \gamma\rangle; \langle (\!|q_1,\phi|\!), \omega_1\rangle, ..., \langle (\!|q_n,\phi|\!), \omega_n\rangle] \in \Delta'$ where for every $1 \leq i \leq n$, $\langle p,\gamma\rangle \xrightarrow{t} \langle q_i,\omega_i\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle$ that have $\langle p,\gamma\rangle$ on the left hand side.

($\hbar$17) If $\phi = E[\phi_1 R^a \phi_2]$, then,

(a) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!),\gamma\rangle, \langle (\!|p,\phi_1|\!),\gamma\rangle] \in \Delta'$

(b) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!),\gamma\rangle, \langle q,\gamma'(\!|\gamma'',\phi|\!)\rangle] \in \Delta'$ for every $\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma'\gamma''\rangle \in \Delta$

(c) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!),\gamma\rangle, \langle (\!|q,\phi|\!),\omega\rangle] \in \Delta'$ for every $\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle \in \Delta$

(d) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{id} \langle p_\perp,\gamma_\perp\rangle \in \Delta'$ for every $\langle p,\gamma\rangle \xrightarrow{ret} \langle q',\epsilon\rangle \in \Delta$

($\hbar$18) If $\phi = A[\phi_1 R^a \phi_2]$, then,

(a) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!),\gamma\rangle, \langle (\!|p,\phi_1|\!),\gamma\rangle] \in \Delta'$

(b) $\langle (\!|p,\phi|\!), \gamma\rangle \xhookrightarrow{equal} [\langle (\!|p,\phi_2|\!),\gamma\rangle; \langle p_1,\gamma_1'(\!|\gamma_1'',\phi|\!)\rangle, ..., \langle p_m,\gamma_m'(\!|\gamma_m'',\phi|\!)\rangle;$ $\langle (\!|q_1,\phi|\!),\omega_1\rangle, ..., \langle (\!|q_n,\phi|\!),\omega_n\rangle, \langle p_\perp,\gamma_\perp\rangle, ..., \langle p_\perp,\gamma_\perp\rangle] \in \Delta'$, where $\langle p_\perp,\gamma_\perp\rangle$ is repeated k times in the right-hand side s.t.:

- for every $1 \le i \le m$, $\langle p,\gamma\rangle \xrightarrow{call} \langle p_i,\gamma_i'\gamma_i''\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p,\gamma\rangle \xrightarrow{call} \langle q,\gamma'\gamma''\rangle$ that have $\langle p,\gamma\rangle$ on the left hand side.
- for every $1 \le i \le n$, $\langle p,\gamma\rangle \xrightarrow{int} \langle q_i,\omega_i\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p,\gamma\rangle \xrightarrow{int} \langle q,\omega\rangle$ that have $\langle p,\gamma\rangle$ on the left hand side.
- for every $1 \le i \le k$, $\langle p,\gamma\rangle \xrightarrow{ret} \langle q_i',\epsilon\rangle \in \Delta$ and these transitions are all the transitions of Δ that are in the form $\langle p,\gamma\rangle \xrightarrow{ret} \langle q',\epsilon\rangle$ that have $\langle p,\gamma\rangle$ on the left hand side.

($\hbar$19) for every $\langle p,\gamma\rangle \xrightarrow{ret} \langle q,\epsilon\rangle \in \Delta$:

– $\langle q, (\!|\gamma'',\phi_1|\!)\rangle \xhookrightarrow{id} \langle (\!|q,\phi_1|\!),\gamma''\rangle \in \Delta'$ for every $\gamma'' \in \Gamma$, $\phi_1 \in Cl(\varphi)$

($\hbar$20) $\langle p_\perp,\gamma_\perp\rangle \xhookrightarrow{id} \langle p_\perp,\gamma_\perp\rangle \in \Delta'$

($\hbar$21) for every $\langle p,\gamma\rangle \xrightarrow{t} \langle q,\omega\rangle \in \Delta$: $\langle p,\gamma\rangle \xhookrightarrow{id} \langle q,\omega\rangle \in \Delta'$

($\hbar$22) If $\phi = e$, e is a regular expression, then, $\langle (\!|p,\phi|\!),\gamma\rangle \xhookrightarrow{id} \langle s_e,\gamma\rangle \in \Delta'$

($\hbar$23) If $\phi = \neg e$, e is a regular expression, then, $\langle (\!|p,\phi|\!),\gamma\rangle \xhookrightarrow{id} \langle s_{\neg e},\gamma\rangle \in \Delta'$

($\hbar$24) for every transition $q \xrightarrow{\alpha} \{q_1,..,q_n\}$ in $\mathcal{M}$: $\langle q,\gamma\rangle \xhookrightarrow{\mathbb{R}} [\langle q_1,\epsilon\rangle, ..., \langle q_n,\epsilon\rangle] \in \Delta'$, where:

(a) $\mathbb{R} = equal$ iff $\alpha = \gamma$

(b) $\mathbb{R} = join_\gamma^x$ iff $\alpha = x \in \mathcal{X}$

(c) $\mathbb{R} = join_\gamma^{\neg x}$ iff $\alpha = \neg x$ and $x \in \mathcal{X}$

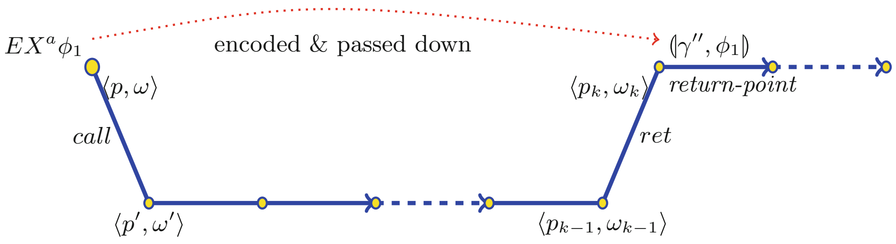
($\hbar$25) for every $q \in \mathcal{F}$, $\langle q,\sharp\rangle \xhookrightarrow{id} \langle q,\sharp\rangle \in \Delta'$

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

- If $\phi = b(\alpha_1, ..., \alpha_n)$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \models_\lambda^B \phi$ iff $b(B(\alpha_1), ..., B(\alpha_n)) \in \lambda(p)$. Thus, for such a B, $\mathcal{BP}_\varphi$ should have an accepting run from $\langle [\![(\!| p, b(\alpha_1, ..., \alpha_n) |\!), B]\!], \omega \rangle$ iff $b(B(\alpha_1), ..., B(\alpha_n)) \in \lambda(p)$. This is ensured by the transition rules in **($\hbar$1)** which add a loop at $\langle [\![(\!| p, b(\alpha_1, ..., \alpha_n) |\!), B]\!], \omega \rangle$ and the fact that $[\![(\!| p, b(\alpha_1, ..., \alpha_n) |\!), B]\!] \in F$ (because it is in F_1). The function *id* in **($\hbar$1)** ensures that the environments before and after are the same.
- If $\phi = \neg b(\alpha_1, ..., \alpha_n)$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \models_\lambda^B \phi$ iff $b(B(\alpha_1), ..., B(\alpha_n)) \notin \lambda(p)$. Thus, for such a B, $\mathcal{BP}_\varphi$ should have an accepting run from $\langle [\![(\!| p, \neg b(\alpha_1, ..., \alpha_n) |\!), B]\!], \omega \rangle$ iff $b(B(\alpha_1), ..., B(\alpha_n)) \notin \lambda(p)$. This is ensured by the transition rules in **($\hbar$2)** which add a loop at $\langle [\![(\!| p, \neg b(\alpha_1, ..., \alpha_n) |\!), B]\!], \omega \rangle$ and the fact that $[\![(\!| p, \neg b(\alpha_1, ..., \alpha_n) |\!), B]\!] \in F$ (because it is in F_2). The function *id* in **($\hbar$2)** ensures that the environments before and after are the same.
- If $\phi = \phi_1 \wedge \phi_2$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \models_\lambda^B \phi$ iff ($\langle p, \omega \rangle \models_\lambda^B \phi_1$ and $\langle p, \omega \rangle \models_\lambda^B \phi_2$). This is ensured by the transition rules in **($\hbar$3)** stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, \phi_1 \wedge \phi_2 |\!), B]\!], \omega \rangle$ iff $\mathcal{BP}_\varphi$ has an accepting run from both $\langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle$ and $\langle [\![(\!| p, \phi_2 |\!), B]\!], \omega \rangle$. **($\hbar$4)** is similar to **($\hbar$3)**.
- If $\phi = \exists x \phi_1$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \models_\lambda^B \phi$ iff there exists $c \in \mathcal{D}$ s.t. $\langle p, \omega \rangle \models_\lambda^{B[x \leftarrow c]} \phi_1$. This is ensured by the transition rules in **($\hbar$5)** stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, \exists x \phi_1 |\!), B]\!], \omega \rangle$ iff there exists $c \in \mathcal{D}$ s.t. $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, \phi_1 |\!), B[x \leftarrow c]]\!], \omega \rangle$ since $B \in meet^x_{\{c\}}(B[x \leftarrow c])$
- If $\phi = \forall x \phi_1$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \models_\lambda^B \phi$ iff for every $c \in \mathcal{D}$, $\langle p, \omega \rangle \models_\lambda^{B[x \leftarrow c]} \phi_1$. This is ensured by the transition rules in **($\hbar$6)** stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, \forall x \phi_1 |\!), B]\!], \omega \rangle$ iff for every $c \in \mathcal{D}$, $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, \phi_1 |\!), B[x \leftarrow c]]\!], \omega \rangle$ since if $\mathcal{D} = \{c_1, ..., c_m\}$, then, $B \in meet^x_{\mathcal{D}}(B[x \leftarrow c_1], ..., B[x \leftarrow c_m])$
- If $\phi = EX^g \phi_1$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \models_\lambda^B \phi$ iff there exists an immediate successor $\langle p', \omega' \rangle$ of $\langle p, \omega \rangle$ s.t. $\langle p', \omega' \rangle \models_\lambda^B \phi_1$. This is ensured by the transition rules in **($\hbar$7)** stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, EX^g \phi_1 |\!), B]\!], \omega \rangle$ 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 $\langle [\![(\!| p', \phi_1 |\!), B]\!], \omega' \rangle$. **($\hbar$8)** is similar to **($\hbar$7)**.
- If $\phi = E[\phi_1 U^g \phi_2]$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \models_\lambda^B \phi$ iff $\langle p, \omega \rangle \models_\lambda^B \phi_2$ or ($\langle p, \omega \rangle \models_\lambda^B \phi_1$ and there exists an immediate successor $\langle p', \omega' \rangle$ of $\langle p, \omega \rangle$ s.t. $\langle p', \omega' \rangle \models_\lambda^B \phi$). This is ensured by the transition rules in **($\hbar$11)** stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, E[\phi_1 U^g \phi_2] |\!), B]\!], \omega \rangle$ iff $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, \phi_2 |\!), B]\!], \omega \rangle$ (by the rules in **($\hbar$11)(a)** or ($\mathcal{BP}_\varphi$ has an accepting run from both $\langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle$ and $\langle [\![(\!| p', \phi |\!), B]\!], \omega' \rangle$ where $\langle p', \omega' \rangle$ is an immediate successor of $\langle p, \omega \rangle$) (by the rules in **($\hbar$11)(b)**). **($\hbar$13)** is similar to **($\hbar$11)**.
- If $\phi = E[\phi_1 R^g \phi_2]$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \models_\lambda^B \phi$ iff ($\langle p, \omega \rangle \models_\lambda^B \phi_2$ and $\langle p, \omega \rangle \models_\lambda^B \phi_1$) or ($\langle p, \omega \rangle \models_\lambda^B \phi_2$ and there exists an immediate successor $\langle p', \omega' \rangle$ of $\langle p, \omega \rangle$ s.t. $\langle p', \omega' \rangle \models_\lambda^B \phi$). This is ensured by the transition rules in **($\hbar$15)** stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, E[\phi_1 R^g \phi_2] |\!), B]\!], \omega \rangle$ iff $\mathcal{BP}_\varphi$

has an accepting run from both $\langle [\![(\!|p, \phi_2|\!), B]\!], \omega \rangle$ and $\langle [\![(\!|p, \phi_1|\!), B]\!], \omega \rangle$ (by the rules in **($\hbar$15)(a)**); or $\mathcal{BP}_\varphi$ has an accepting run from both $\langle [\![(\!|p, \phi_2|\!), B]\!], \omega \rangle$ and $[\![(\!|p', \phi|\!), B]\!], \omega' \rangle$ where $\langle p', \omega' \rangle$ is an immediate successor of $\langle p, \omega \rangle$ (by the rules in **($\hbar$15)(b)**). 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 $[\![(\!|p, \phi|\!), B]\!]$ to the Büchi accepting condition F (via the subset F_3 of F). **($\hbar$16)** is similar to **($\hbar$15)**.

– If $\phi = EX^a\phi_1$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \vDash_\lambda^B \phi$ iff there exists an abstract-successor $\langle p_k, \omega_k \rangle$ of $\langle p, \omega \rangle$ s.t. $\langle p_k, \omega_k \rangle \vDash_\lambda^B \phi_1$ (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 this.



$EX^a\phi_1$
encoded & passed down
$(\!|\gamma'', \phi_1|\!)$
$\langle p, \omega \rangle$
$\langle p_k, \omega_k \rangle$
return-point
call
ret
$\langle p', \omega' \rangle$
$\langle p_{k-1}, \omega_{k-1} \rangle$

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

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

- If $\langle p, \omega \rangle \Rightarrow_\mathcal{P} \langle p', \omega' \rangle$ corresponds to a call statement. Let us consider Fig. 2 to explain this case. $\langle [\![(\!|p, \phi|\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![(\!|p_k, \phi_1|\!), B]\!], \omega_k \rangle$ is ensured by the rules in **($\hbar$9)(a)**, the rules in **($\hbar$21)** and the rules in **($\hbar$19)** as follows: rules in **($\hbar$9)(a)** allow to record ϕ_1 in the return point of the call, rules in **($\hbar$21)** allow to mimic the run of the PDS $\mathcal{P}$ and rules in **($\hbar$19)** 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 Sect. 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 $\langle [\![(\!|p, \phi|\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![(\!|p_k, \phi_1|\!), B]\!], \omega_k \rangle$, 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 in **($\hbar$9)(a)** stating that $\langle (\!|p, EX^a\phi_1|\!), \gamma\rangle \overset{id}{\hookrightarrow} \langle p', \gamma'(\!|\gamma'', \phi_1|\!)\rangle \in \Delta'$. This allows to record ϕ_1 in the corresponding return point of the stack. After that, the rules in **($\hbar$21)** 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 $\mathcal{BP}_\varphi$: $\langle [\![p', B]\!], \gamma'(\!|\gamma'', \phi_1|\!)\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_{k-1}, B]\!], \beta(\!|\gamma'', \phi_1|\!)\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_k, B]\!], (\!|\gamma'', \phi_1|\!)\omega''\rangle$ using the rules of **($\hbar$21)**. At the return-point, we extract ϕ_1 from the stack and encode it into p_k by adding the transition rules in **($\hbar$19)** $\langle p_k, (\!|\gamma'', \phi_1|\!)\rangle \overset{id}{\hookrightarrow} \langle (\!|p_k, \phi_1|\!), \gamma''\rangle$. Therefore, we obtain that $\langle [\![(\!|p, \phi|\!), B]\!], \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![(\!|p_k, \phi_1|\!), B]\!], \omega_k\rangle$. 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 **($\hbar$9)(b)**, we get that $\langle [\![(\!|p, EX^a\phi_1|\!), B]\!], \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![(\!|p', \phi_1|\!), B]\!], \omega'\rangle$. Therefore, $\langle [\![(\!|p, EX^a\phi_1|\!), B]\!], \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![(\!|p_k, \phi_1|\!), B]\!], \omega_k\rangle$. 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 $\langle [\![(\!|p, EX^a\phi_1|\!), B]\!], \omega\rangle$, we need to ensure that the path of $\mathcal{BP}_\varphi$ reflecting the possibility in (A1) that $\langle p_k,\omega_k\rangle \vDash^B_\lambda \phi_1$ 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 in **($\hbar$9)(c)** to allow $\langle [\![(\!|p, EX^a\phi_1|\!), B]\!], \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_\perp, B]\!], \gamma_\perp\omega''\rangle$. Since a run of $\mathcal{BP}_\varphi$ includes only infinite paths, we equip these trap configurations with self-loops by the transition rules in **($\hbar$20)**, i.e., $\langle [\![p_\perp, B]\!], \gamma_\perp\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_\perp, B]\!], \gamma_\perp\omega''\rangle$. As a result, we obtain a corresponding path in $\mathcal{BP}_\varphi$: $\langle [\![(\!|p, EX^a\phi_1|\!), B]\!], \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_\perp, B]\!], \gamma_\perp\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_\perp, B]\!], \gamma_\perp\omega''\rangle$. Note that this path is not accepted by $\mathcal{BP}_\varphi$ because $[\![p_\perp, B]\!] \notin F$.

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, $\langle [\![(\!|p, EX^a\phi_1|\!), B]\!], \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![(\!|p_k, \phi_1|\!), B]\!], \omega_k\rangle$; otherwise $\langle [\![(\!|p, EX^a\phi_1|\!), B]\!], \omega\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_\perp, B]\!], \gamma_\perp\omega''\rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_\perp, B]\!], \gamma_\perp\omega''\rangle$ which is not accepted by $\mathcal{BP}_\varphi$. Therefore, (A1) is ensured by the transition rules in **($\hbar$9)** stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!|p, EX^a\phi_1|\!), B]\!], \omega\rangle$ 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 $\langle [\![(\!|p_k, \phi_1|\!), B]\!], \omega_k\rangle$.

– If $\phi = AX^a\phi_1$: this case is ensured by the transition rules in **($\hbar$10)** together with **($\hbar$19)** and **($\hbar$21)**. The intuition of **($\hbar$10)** is similar to that of **($\hbar$9)**.

– If $\phi = E[\phi_1 U^a \phi_2]$, then, for every $\omega \in \Gamma^*$, $\langle p, \omega \rangle \vDash_\lambda^B \phi$ iff $\langle p, \omega \rangle \vDash_\lambda^B \phi_2$ or $(\langle p, \omega \rangle \vDash_\lambda^B \phi_1$ and there exists an abstract successor $\langle p_k, \omega_k \rangle$ of $\langle p, \omega \rangle$ s.t. $\langle p_k, \omega_k \rangle \vDash_\lambda^B \phi)$ (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 $\langle p, \omega \rangle$, $\langle [\![(\!| p, \phi |\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \{ \langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle, \langle [\![(\!| p_k, \phi |\!), B]\!], \omega_k \rangle \}$ where $B \in \mathcal{B}$. There are two possibilities:

- If $\langle p, \omega \rangle \Rightarrow_{\mathcal{P}} \langle p', \omega' \rangle$ corresponds to a call statement. From the rules in **($\hbar$12)(b)**, we get that $\langle [\![(\!| p, \phi |\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \{ \langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle, \langle p', \omega' \rangle \}$ where $\langle p', \omega' \rangle$ is the immediate successor of $\langle p, \omega \rangle$. Thus, to ensure that $\langle [\![(\!| p, \phi |\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \{ \langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle, \langle [\![(\!| p_k, \phi |\!), B]\!], \omega_k \rangle \}$, we only need to ensure that $\langle p', \omega' \rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![(\!| p_k, \phi |\!), B]\!], \omega_k \rangle$. As for the case $\phi = EX^a \phi_1$, $\langle p', \omega' \rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![(\!| p_k, \phi |\!), B]\!], \omega_k \rangle$ is ensured by the rules in **($\hbar$21)** and the rules in **($\hbar$19)**: rules in **($\hbar$21)** allow to mimic the run of the PDS $\mathcal{P}$ before the return and rules in **($\hbar$19)** 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 **($\hbar$12)(c)**, we get that $\langle [\![(\!| p, E[\phi_1 U^a \phi_2] |\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \{ \langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle, \langle [\![(\!| p', \phi |\!), B]\!], \omega' \rangle \}$. Therefore, $\langle [\![(\!| p, E[\phi_1 U^a \phi_2] |\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \{ \langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle, \langle [\![(\!| p_k, \phi |\!), B]\!], \omega_k \rangle \}$. In other words, $\mathcal{BP}_\varphi$ has an accepting run from both $\langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle$ and $\langle [\![(\!| p_k, \phi |\!), B]\!], \omega_k \rangle$ 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 $\langle p_k, \omega_k \rangle = \perp$. 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 $\perp$. Note that $\perp$ does not satisfy any formula, i.e., $\perp$ does not satisfy ϕ. Therefore, from $\langle [\![(\!| p, E[\phi_1 U^a \phi_2] |\!), B]\!], \omega \rangle$, we need to ensure that the path reflecting the possibility in (A2) that $(\langle p, \omega \rangle \vDash_\lambda^B \phi_1$ and $\langle p_k, \omega_k \rangle \vDash_\lambda^B \phi)$ is not accepted by $\mathcal{BP}_\varphi$. This is ensured as for the case $\phi = EX^a \phi_1$ by the transition rules in **($\hbar$12)(d)**.

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, $\langle [\![(\!| p, E[\phi_1 U^a \phi_2] |\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \{ \langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle, \langle [\![(\!| p_k, E[\phi_1 U^a \phi_2] |\!), B]\!], \omega_k \rangle \}$; otherwise $\langle [\![(\!| p, E[\phi_1 U^a \phi_2] |\!), B]\!], \omega \rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_\perp, B]\!], \gamma_\perp \omega'' \rangle \Rightarrow_{\mathcal{BP}_\varphi} \langle [\![p_\perp, B]\!], \gamma_\perp \omega'' \rangle$ which is not accepted by $\mathcal{BP}_\varphi$. Therefore, (A2) is ensured by the transition rules in **($\hbar$12)** stating that $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, E[\phi_1 U^a \phi_2] |\!), B]\!], \omega \rangle$ iff $\mathcal{BP}_\varphi$ has an accepting run from $\langle [\![(\!| p, \phi_2 |\!), B]\!], \omega \rangle$; or $\mathcal{BP}_\varphi$ has an accepting run from both $\langle [\![(\!| p, \phi_1 |\!), B]\!], \omega \rangle$ and $\langle [\![(\!| p_k, E[\phi_1 U^a \phi_2] |\!), B]\!], \omega_k \rangle$ 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 $\phi = A[\phi_1 U^a \phi_2]$, $\phi = A[\phi_1 R^a \phi_2]$ are similar to the previous case.

– If $\phi = e (e \in \mathcal{V})$. Given $p \in P$, $e \in \mathcal{V}$, $\omega \in \Gamma^*$, we get that the SABPDS $\mathcal{BP}_\varphi$ should accept $\langle [\![(\!| p, e |\!), B]\!], \omega \rangle$ iff $(\langle p, \omega \rangle, B) \in L(M_e)$. To check whether $(\langle p, \omega \rangle, B) \in L(M_e)$, we let $\mathcal{BP}_\varphi$ go to state s_e, the initial state corresponding to p in M_e by adding rules in **($\hbar$22)**; and then, from this state, we will check whether ω is accepted by M_e under B. This is ensured by the transition rules in **($\hbar$24)** and **($\hbar$25)**. **($\hbar$24)** lets $\mathcal{BP}_\varphi$ mimic a run of M_e on ω under B, which includes three possibilities:

- if $\mathcal{BP}_\varphi$ is in a state $[\![q, B]\!]$ with γ on the top of the stack where $\gamma \in \Gamma$, and if $q \xrightarrow{\gamma} \{q_1, ..., q_n\}$ is a transition rule in M_e, then, $\mathcal{BP}_\varphi$ will move to states $[\![q_1, B]\!], ..., [\![q_n, B]\!]$ and pop γ from its stack. Note that popping γ allows us to check the rest of the word. This is ensured by the rules corresponding to **($\hbar$24)(a)**. Then function *equal* ensures that all these environments are the same.
- if $\mathcal{BP}_\varphi$ is in a state $[\![q, B]\!]$ with γ on the top of the stack, and if $q \xrightarrow{x} \{q_1, ..., q_n\}$ is a transition rule in M_e where $x \in \mathcal{X}$, then, $\mathcal{BP}_\varphi$ can mimic a run of M_e under B iff $B(x) = \gamma$. If this condition is guaranteed, $\mathcal{BP}_\varphi$ will move to states $[\![q_1, B]\!], ..., [\![q_n, B]\!]$ and pop γ from its stack. Again, popping γ allows us to check the rest of the word. This is ensured by the rules corresponding to **($\hbar$24)(b)**. Then function $join^x_\gamma$ ensures that all these environments are the same B and $B(x) = \gamma$.
- Similar to **($\hbar$24)(b)**, **($\hbar$24)(c)** deals with the cases where $q \xrightarrow{\neg x} \{q_1, ..., q_n\}$ is a transition rule in M_e where $x \in \mathcal{X}$.

In each VA M_e, a configuration is accepted if the run with the word ω reaches a 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 should add $F_e \times \mathcal{B}$ as a set of accepting control locations in $\mathcal{BP}_\varphi$. This is why F_4 is a set of accepting control locations. In addition, since $\mathcal{BP}_\varphi$ only recognizes infinite paths, **($\hbar$25)** adds a loop on every configuration $\langle [\![q, B]\!], \sharp \rangle$ where $q \in F_e$.

– If $\phi = \neg e (e \in \mathcal{V})$. This case is ensured by the transition rules in **($\hbar$23)**, **($\hbar$24)** and **($\hbar$25)**. The intuition behind this case is similar to the case $\phi = e$.

We can show that:

Theorem 4. *Given a PDS $\mathcal{P} = (P, \Gamma, \Delta)$, a set of atomic propositions AP, a labelling function $\lambda : AP_\mathcal{D} \to 2^P$ and a SBPCARET formula φ, we can compute an SABPDS $\mathcal{BP}_\varphi$ such that for every configuration $\langle p, \omega \rangle$, for every $B \in \mathcal{B}$, $\langle p, \omega \rangle \models^B_\lambda \varphi$ iff $\mathcal{BP}_\varphi$ has an accepting run from the configuration $\langle [\![(\!| p, \varphi |\!), B]\!], \omega \rangle$.*

5 Conclusion

In this paper, we present a new logic SBPCARET and show how it can precisely and succinctly specify malicious behaviors. We then propose an efficient algorithm for SBPCARET model-checking for PDSs. Our algorithm is based on reducing the model checking problem to the emptiness problem of Symbolic Alternating Büchi Pushdown Systems.

References

1. Alur, R., Etessami, K., Madhusudan, P.: A temporal logic of nested calls and returns. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 467–481. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24730-2_35
2. Bergeron, J., Debbabi, M., Desharnais, J., Erhioui, M.M., Lavoie, Y., Tawbi, N.: Static detection of malicious code in executable programs. Int. J. Req. Eng. **184–189**, 79 (2001)
3. Christodorescu, M., Jha, S.: Static analysis of executables to detect malicious patterns. In: Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12, SSYM 2003, Berkeley, CA, USA, p. 12. USENIX Association (2003)
4. Kinder, J., Katzenbeisser, S., Schallhart, C., Veith, H.: Detecting malicious code by model checking. In: Julisch, K., Kruegel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 174–187. Springer, Heidelberg (2005). https://doi.org/10.1007/11506881_11
5. Lakhotia, A., Kumar, E.U., Venable, M.: A method for detecting obfuscated calls in malicious binaries. IEEE Trans. Softw. Eng. **31**(11), 955–968 (2005)
6. Nguyen, H.-V., Touili, T.: CARET model checking for malware detection. In: SPIN 2017 (2017)
7. Nguyen, H.-V., Touili, T.: CARET model checking for pushdown systems. In: SAC 2017 (2017)
8. Nguyen, H.-V., Touili, T.: Branching temporal logic of calls and returns for pushdown systems. In: Furia, C.A., Winter, K. (eds.) IFM 2018. LNCS, vol. 11023, pp. 326–345. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98938-9_19
9. Schwoon, S.: Model-checking pushdown systems. Dissertation, Technische Universität München, München (2002)
10. Song, F., Touili, T.: Pushdown model checking for malware detection. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 110–125. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28756-5_9
11. Song, F., Touili, T.: Efficient malware detection using model-checking. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012. LNCS, vol. 7436, pp. 418–433. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32759-9_34
12. Song, F., Touili, T.: LTL model-checking for malware detection. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 416–431. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36742-7_29
13. Song, F., Touili, T.: PoMMaDe: pushdown model-checking for malware detection. In: SIGSOFT 2013 (2013)