On Symbolic Verification of Weakly Extended PAD

Ahmed Bouajjani[a,1], Jan Strejček[b,2], and Tayssir Touili[a,3]

[a] *LIAFA, CNRS and University of Paris 7, France*

[b] *Faculty of Informatics, Masaryk University, Brno, Czech Republic*

Abstract

We consider the verification problem of a class of infinite-state systems called wPAD. These systems can be used to model programs with (possibly recursive) procedure calls and dynamic creation of parallel processes. They correspond to PAD models extended with an acyclic finite-state control unit, where PAD models can be seen as combinations of prefix rewrite systems (pushdown systems) with context-free multiset rewrite systems (synchronization-free Petri nets). Recently, we have presented symbolic reachability techniques for the class of PAD based on the use of a class of unranked tree automata. In this paper, we generalize our previous work to the class wPAD which is strictly larger than PAD. This generalization brings a positive answer to an open question on decidability of the model checking problem for wPAD against EF logic. Moreover, we show how symbolic reachability analysis of wPAD can be used in (under) approximate analysis of Synchronized PAD, a (Turing) powerful model for multithreaded programs (with unrestricted synchronization between parallel processes). This leads to a pragmatic approach for detecting the presence of erroneous behaviors in these models based on the bounded reachability paradigm where the notion of bound considered here is the number of synchronization actions.

Keywords: rewrite systems, infinite-state systems, symbolic reachability analysis, model checking

1 Introduction

Reasoning about software systems requires the consideration of powerful models which are in general infinite-state, i.e., they may have an infinite number of reachable configurations. Sources of complexity, and of infinity of the state space, may be related to either *data manipulation* such as the use of variables over infinite data domains, dynamic and unbounded-size data structures, etc, or to *complex control primitives* such as procedures calls, (unbounded) dynamic creation of concurrent processes, etc. One popular approach to handle this complexity is to combine abstraction methods with model-checking. Techniques such as predicate abstraction

[1] Email: abou@liafa.jussieu.fr

[2] Partly supported by the research centre Institute for Theoretical Computer Science (ITI), project No. 1M0545. The paper has been written during J. Strejček's postdoc stay in LaBRI, Univeristé Bordeaux 1. Email: strejcek@fi.muni.cz

[3] Email: touili@liafa.jussieu.fr

allows to deal with aspects such as data manipulation and to generate abstract models over finite data domains. Then, the so obtained abstract models can be analyzed automatically using model checking algorithms, provided that such algorithms exist for the considered class of abstract models. This is the case obviously when abstract models are finite-state. However, as said above, in order to take into account complex control primitives such as procedure calls and process creation, finite state models are not expressive enough. For instance, in the case of sequential programs with recursive procedure calls, the needed abstract models are (unbounded-stack) pushdown systems, and for programs with dynamic creation of communicating finite-state processes, natural models are (unbounded) Petri nets. Fortunately, there exist several algorithmic techniques (e.g., reachability analysis, model-checking) which have been developed for the analysis and the verification of these infinite-state models.

In this paper, we consider the case of programs which may contain both (recursive) procedure calls and dynamic creation of processes (threads). One possible approach to model such systems is to combine pushdown systems with Petri nets. This corresponds to the use of *Process Rewrite Systems* (PRS) introduced in [17]. These models can be seen indeed as combinations of prefix rewrite systems and multiset rewrite systems. The relevance of PRS in program modeling have been discussed for instance in [8,9,7,1,2]. Subclasses of PRS which are of particular interest for program modeling are for instance the class of PA processes, and the larger class of PAD processes generalizing both PA and pushdown processes and corresponding to synchronization-free PRS (i.e., models where parallel composition is not allowed in the left-hand-side of the rewrite rules). Processes in these classes allow indeed to model systems with procedure calls and parbegin-parend blocks (i.e., launching a number of parallel threads, and wait for their termination before proceeding). PAD allow in addition return values from sequential procedure calls.

Richard Mayr has shown that the *reachability problem* (whether a given state is reachable from another given state) for PRS is decidable using a reduction to the reachability problem of Petri nets [17]. To get practical verification algorithms, symbolic reachability algorithms have been investigated for significant subclasses of PRS such as PA [16,9] and PAD [1,2]. These algorithms use (various kinds of) tree automata to represent (regular) infinite sets of configurations (i.e., process terms). In particular, we have provided in [2] a generic construction allowing to compute the set of (forward or backward) reachable configurations of any subclass of PRS built from the combination of prefix rewrite systems with an effectively semilinear class of multiset rewrite systems (i.e., a class of systems for which reachability sets are always semilinear and effectively computable). We have shown that this leads to a symbolic reachability analysis algorithm for PAD processes in a certain normal form.

The PRS formalism is not Turing powerful due to a subtle restriction on the way synchronization is done between parallel processes. Roughly speaking, the semantics of PRS implies that synchronization can only be allowed between parallel processes with empty stacks.

In order to extend the modeling power of PRS, one approach is to add synchronization by rendez-vous (à la CCS), which leads to a Turing powerful model called *synchronized PRS* [21]. Similarly, PAD can be extended to *synchronized PAD* (which is also a Turing powerful model). Approximate analysis algorithms for these models using abstraction techniques have been proposed in [21].

Another approach for enhancing the modeling power of PRS (and PAD) consists in adding global control states. The new models, called sePRS [11], can be seen as parallel product of a PRS with a finite-state automaton representing a global control. Obviously, sePRS are Turing powerful since they allow communication between recursive parallel processes through the global control state. However, if the structure of the control automaton is *weak*, which means that all its loops are self-loops, then it can be proved that the obtained models, called wPRS, have a decidable reachability problem [12] (the proof employs decidability of the reachability problem for Petri nets). Similarly, if we add control states to PAD processes, we obtain Turing powerful models, but the extension of PAD with weak control automata leads to models, called wPAD, having a decidable reachability problem, and interestingly, which can be proven to be strictly more powerful (w.r.t. strong bisimulation) than PAD [13].

In this paper we extend the results on symbolic reachability analysis presented in [2]. While [2] deals only with PAD processes in a certain normal form (now called *canonic PAD*), here we show that the set of reachability states are computable and effectively representable even for (general) wPAD systems. To do this, we employ symbolic representations based on so-called *commutative-hedge automata* (CH-automata), allowing to define sets of process terms modulo the associativity of sequential composition, and the associativity-commutativity of the parallel composition. We show that these representations are effectively closed under the computation of the *post** and *pre** images (i.e., computation of all successors and all predecessors) for wPAD, as well as under the *post* and *pre* images (i.e., computation of immediate successors and predecessors) for the *whole class of wPRS*.

Further, we solve the *global model-checking* problem of wPAD against the EF logic. We consider a variant of EF logic which generalizes the standard action-based EF logic by the use of atomic propositions corresponding to (potentially infinite) sets of configurations which are definable using CH-automata. We prove that for every formula in this logic, it is possible to construct a (CH-automata based) representation of the set of all configurations (in a given wPAD) satisfying this formula. This result closes an open problem formulated in [14] concerning the model-checking problem of wPAD. Notice that global model-checking is a more general problem than deciding whether a given configuration satisfies a given formula.

Finally, we show that our results concerning symbolic reachability analysis of wPAD can be used in the analysis of *synchronized PAD* (SPAD) with a bounded number of synchronizations. This leads to an approximate analysis procedure for SPAD based on computing *under* approximations of their reachability sets by considering only reachable configurations up to some fixed number of synchronizations. Such approximate analysis method for SPAD can be used in practice to establish the *existence* of erroneous behaviors, following the approach advocated in [18]. It constitutes a complementary approach to the abstract analysis (provided for SPAD in [21]), which is based on considering *upper* approximations of the set of possible behaviors and which is useful for establishing the *absence* of erroneous behaviors.

2 Preliminaries

2.1 Process terms

Let *Const* $= \{X, \ldots\}$ be a set of *process constants.* For every $C \subseteq \mathit{Const}$, the set T_C of *process terms* over C is defined by the abstract syntax $t ::= 0 \mid X \mid t \odot t \mid t \| t$, where 0 is the *idle term*, $X \in C$ is a process constant; and $\odot$ and $\|$ mean *sequential* and *parallel compositions* respectively.

We use ω to denote in a generic way $\odot$ or $\|$. We denote by $\overline{\omega}$ the operator $\odot$ (resp. $\|$) if $\omega = \|$ (resp. $\omega = \odot$). Process terms are considered modulo the following algebraic properties: associativity of $\odot$, associativity and commutativity of $\|$, and neutrality of 0 w.r.t. both $\odot$ and $\|$, i.e. $0 \odot t = t \odot 0 = t \| 0 = t$. Let $\simeq$ be the equivalence relation on T induced by these properties.

We distinguish four *classes of process terms* as:

1 – terms consisting of a single process constant only, in particular $0 \notin 1$,

S – *sequential* terms - terms without parallel composition, e.g. $X \odot Y \odot Z$,

P – *parallel* terms - terms without sequential composition, e.g. $X \| Y \| Z$,

G – *general* terms - terms without any restrictions, e.g. $(X \odot (Y \| Z)) \| W$.

Process terms in *canonical form* are terms t defined by:

$$\begin{aligned} t &::= 0 \mid s \mid p \\ s &::= X \mid p_1 \odot p_2 \odot \ldots \odot p_n,\ n \geq 2 \\ p &::= X \mid s_1 \| s_2 \| \ldots \| s_n,\ n \geq 2 \end{aligned}$$

It can easily be seen that every term has an $\simeq$-equivalent term in canonical form. In the following we work with terms in canonical form.

Term t is called *seq-term* if $t = 0$, or $t = X$ for a constant X, or $t = p_1 \odot p_2 \odot \ldots \odot p_n$ where $n \geq 2$. In the last case, the term is also called $\odot$*-rooted term.* Further, t is called *flat seq-term* if $t = X_1 \odot X_2 \odot \ldots \odot X_n$ for $n \geq 0$ (the case $n = 0$ corresponds to the term 0, and the case $n = 1$ corresponds to a process constant X). By analogy we define *par-terms*, $\|$*-rooted terms*, and *flat par-terms.*

2.2 Process Rewrite Systems and weak extension

Let $M = \{o, p, q, \ldots\}$ be an ordered set of *control states* and $\mathit{Act} = \{a, b, c, \ldots\}$ be a set of *actions.* Let $\alpha, \beta \in \{1, S, P, G\}$ be classes of process terms such that $\alpha \subseteq \beta$. An *(α, β)-wPRS (weakly extended process rewrite system)* R is a finite set of *rewrite rules* of the form $(p, t_1) \overset{a}{\hookrightarrow} (q, t_2)$, where $t_1 \in \alpha$, $t_1 \neq 0$, $t_2 \in \beta$, $p, q \in M$, $p \leq q$, and $a \in \mathit{Act}$. By $M(R)$, $\mathit{Const}(R)$, and $\mathit{Act}(R)$ we denote sets of control states, process constants, and actions occurring in rewrite rules of R.

An (α, β)-wPRS R induces a labelled transition system the states of which are pairs (p, t) such that $p \in M(R)$ is a control state and $t \in \beta$ is a process term over $\mathit{Const}(R)$. The transition relation $\longrightarrow_R$ is the least relation satisfying the following inference rules:

$$\frac{((p, t_1) \overset{a}{\hookrightarrow} (q, t_2)) \in R}{(p, t_1) \xrightarrow{a}_R (q, t_2)} \qquad \frac{(p, t_1) \xrightarrow{a}_R (q, t_2)}{(p, t_1 \| t) \xrightarrow{a}_R (q, t_2 \| t)} \qquad \frac{(p, t_1) \xrightarrow{a}_R (q, t_2)}{(p, t_1 \odot t) \xrightarrow{a}_R (q, t_2 \odot t)}$$

We extend the transition relation to finite words over *Act* in a standard way. The reflexive and transitive closure of $\rightarrow_R$ is denoted by $\xrightarrow{*}_R$. To shorten our notation we write *pt* in lieu of (p,t).

An (α,β)-wPRS where $M(R)$ is a singleton is called *(α,β)-PRS (process rewrite system).* In such systems we omit the single control state from rules and states.

Instead of (S,G)-PRS, (S,G)-wPRS, (G,G)-PRS, and (G,G)-wPRS we use more readable names PAD, wPAD, PRS, and wPRS respectively. Let us note that the classes PAD and wPAD subsume widely known models of infinite-state systems as *pushdown processes* (PDA), *basic parallel processes* (BPP), and *process algebras* (PA). The classes PRS and wPRS subsume also *Petri nets* (PN). More information about expressiveness of (α,β)-wPRS and (α,β)-wPRS can be found in [13,12].

Given a state *pt* of a wPRS R, we define

$$Post_R(pt) = \{p't' \mid pt \xrightarrow{a}_R p't' \text{ for some } a\} \qquad Post^*_R(pt) = \{p't' \mid pt \xrightarrow{*}_R p't'\}$$
$$Pre_R(pt) = \{p't' \mid p't' \xrightarrow{a}_R pt \text{ for some } a\} \qquad Pre^*_R(pt) = \{p't' \mid p't' \xrightarrow{*}_R pt\}$$

The sets $Post^*_R(pt)$ and $Pre^*_R(pt)$ are called *(forward and backward) reachability sets*. The sets $Post_R(pt)$ and $Pre_R(pt)$ are called *1-step (forward and backward) reachability sets*. These definitions and notations can be extended to sets of states in the obvious manner.

2.3 Canonic PRS

A *canonic PRS R* is a set of rewrite rules of the forms:

$$X_1 \odot X_2 \odot \ldots \odot X_n \overset{a}{\hookrightarrow} Y_1 \odot Y_2 \odot \ldots \odot Y_m \tag{1}$$
$$X_1 \| X_2 \| \ldots \| X_n \overset{a}{\hookrightarrow} Y_1 \| Y_2 \| \ldots \| Y_m \tag{2}$$

where $n,m \geq 0$. Rules of the form (1) and (2) are called $\odot$*-rules* and $\|$*-rules* respectively. By R_ω we denote the set of all ω-rules of R. Note that the sets $R_\|$ and $R_\odot$ do not have to be disjoint as some rules (e.g. $X \overset{a}{\hookrightarrow} Y$) are of both types. Let $\alpha,\beta \in \{1,S,P,G\}$ be classes of process terms. A canonic PRS is called *canonic (α,β)-PRS* if every rule $t_1 \overset{a}{\hookrightarrow} t_2$ of R satisfies $t_1 \in \alpha$ and $t_2 \in \beta$. Finally, *canonic PAD* stands for canonic (S,G)-PRS.

Note that a canonic PRS does not have to be a PRS as we allow rules with 0 on the left-hand side. Further, the definition of canonic (α,β)-PRS does not require that $\alpha \subseteq \beta$. The meaning of $Const(R), \rightarrow_R, Post_R, Pre_R, \ldots$ remains the same.

Given a canonic (α,β)-PRS R, by R^{-1} we denote the canonic (β,α)-PRS with rules obtained by swapping the left-hand and right-hand sides of the rules of R. Notice that for every set of process terms L, $Pre_R(L) = Post_{R^{-1}}(L)$ and $Pre^*_R(L) = Post^*_{R^{-1}}(L)$.

The problem of computing reachability sets of PRS systems can be transformed into the same problem for canonic PRS using the following theorem. The proof of this theorem employs a variant of the standard construction given in [17]. However, our theorem differs from the one of [17] in several aspects. In particular, (1) we transform an (α,β)-PRS into a canonic (α,β)-PRS, which is not the case of Mayr's transformation, and (2) in contrast to the original theorem in [17], our theorem states that the same transformation of R works for *all* terms over a given set of process constants.

A *term substitution* h is a function on process terms satisfying $h(0) = 0$ and $h(t_1 \,\omega\, \ldots \,\omega\, t_n) = h(t_1) \,\omega\, \ldots \,\omega\, h(t_n)$ for all finite sequences $t_1, \ldots, t_n$ of terms and for both $\omega = \odot, \|$. In other words, a term substitution is fully specified by its values on process constants. We say that a term subsitution h is *finite* if the set $\{X \mid h(X) \neq X\}$ of process constants is finite.

Theorem 2.1 *For every (α,β)-PRS system R and every set of process constants C we can construct a canonic (α,β)-PRS system R' and a finite term substitution h, such that for every t_1, t_2 over $C \cup Const(R)$ and every $a \in Act(R)$ we have:*

(i) $t_1 \xrightarrow{a}_R t_2$ *iff there exists* t'_1, t'_2 *satisfying* $h(t'_1) = t_1$, $h(t'_2) = t_2$, *and* $t'_1 \xrightarrow{a}_{R'} t'_2$,

(ii) $t_1 \xrightarrow{*}_R t_2$ *iff there exists* t'_1, t'_2 *satisfying* $h(t'_1) = t_1$, $h(t'_2) = t_2$, *and* $t'_1 \xrightarrow{*}_{R'} t'_2$.

Proof. Let $size(t \overset{a}{\hookrightarrow} t')$ be the number of occurrences of $\odot$ and $\|$ in terms t and t'. Given any PRS R, let k_i be the number of rules $r \in R$ that are neither $\odot$-rules nor $\|$-rules and $size(r) = i$. Thus, R is canonic PRS iff $k_i = 0$ for every i. In this case, let $n = 0$. Otherwise, let n be the largest i such that $k_i \neq 0$ (n exists as the set of rules is finite). We define $norm(R)$ to be the pair (n, k_n).

First we describe a procedure transforming an (α,β)-PRS R into an (α,β)-PRS R' and defining finite term substitution h such that $norm(R') < norm(R)$ (with respect to the lexicographical ordering) and for every terms t_1, t_2 over $C \cup Const(R)$ and every $a \in Act(R)$ the following equivalences hold:

(i) $t_1 \xrightarrow{a}_R t_2 \iff$ there exists t'_1, t'_2 satisfying $h(t'_i) = t_i$ and $t'_1 \xrightarrow{a}_{R'} t'_2$

(ii) $t_1 \xrightarrow{*}_R t_2 \iff$ there exists t'_1, t'_2 satisfying $h(t'_i) = t_i$ and $t'_1 \xrightarrow{*}_{R'} t'_2$

In this proof we assume that $\odot$ is left-associative. It means that the term $X \odot Y \odot Z$ is seen as $(X \odot Y) \odot Z$ and so its subterms are X, Y, Z, and $X \odot Y$, but not $Y \odot Z$. Let us assume that R is not canonic PRS. Let $\tau \notin Act(R)$ be a fresh action. We set $h(X) = X$ for every $X \in C \cup Const(R)$ and $R' = R$. Let $r = (s_1 \overset{a}{\hookrightarrow} s_2)$ be a rule of R' that is neither $\odot$-rule nor $\|$-rule and has the maximal *size*. There are three cases:

(i) s_1 is ω-rooted and s_2 is $\overline{\omega}$-rooted. In R' we replace the rule r by rules $s_1 \overset{\tau}{\hookrightarrow} Z$, $Z \overset{a}{\hookrightarrow} s_2$, where $Z \notin C \cup Const(R)$ is a fresh process constant. We set $h(Z) = s_1$. Clearly, the considered equivalences holds.

(ii) s_1, s_2 are par-terms and at least one of them is not flat. Let t be an $\odot$-rooted subterm of s_1 or s_2. We modify R' in two steps. First, in all left-hand and right-hand sides of all rules, we replace every occurrence of t by a fresh process constant $Z \notin C \cup Const(R)$. Further, we add the rule $Z \overset{\tau}{\hookrightarrow} t$ and if $t \in \alpha$ then we add also the rule $t \overset{\tau}{\hookrightarrow} Z$. We set $h(Z) = t$.

We say that an occurrence of subterm t of term s is *active*, if a rule $t \overset{\tau}{\hookrightarrow} Z$ can be applied on s such that the occurrence of t is replaced by Z. The occurrence is *inactive* otherwise. Note that an occurrence of t in s is inactive iff it is a subterm of the right component of some sequential composition.

Clearly, the first equivalence and the implication "$\Longleftarrow$" of the second equivalence hold. In order to prove the remaining implication, we show that every transition $l_1 \xrightarrow{a}_R l_2$ (where l_1, l_2 are terms over $C \cup \mathit{Const}(R)$) corresponds to a transition sequence $l'_1 \overset{\tau^* a \tau^*}{\longrightarrow}_{R'} l'_2$, where l'_1, l'_2 are l_1, l_2 with all inactive occurrences of t replaced by Z. Let us assume that the transition $l_1 \xrightarrow{a}_R l_2$ is generated by a rule $l \overset{a}{\hookrightarrow} l'$. Each occurrence of t in l_1 modified by the rule is either active, or it is inactive (and thus replaced by Z in l'_1) and completely contained in l (due to the left-associativity of $\odot$). Hence, we can apply the rule $t \overset{\tau}{\hookrightarrow} Z$ to all occurrences of t in l'_1 which are going to be modified by the rule of R' corresponding to $l \overset{a}{\hookrightarrow} l'$ (i.e. the same rule with all occurrences of t replaced by Z). This corresponding rule is applied afterwards.

The situation with occurrences of t appearing in l_2 after the application of the considered rule is similar. Each occurrence of t in l_2 created by the rule is either active, or it is inactive and completely contained in l. Hence, after application of the corresponding rule of R', we apply the rule $Z \overset{\tau}{\hookrightarrow} t$ to all active occurrences of Z to reach l'_2.

(iii) s_1, s_2 are seq-terms and at least one of them is not flat. This case is a direct analogy of the previous one.

Note that $\mathit{norm}(R') < \mathit{norm}(R)$ and R' belongs to (α, β)-PRS class. After finitely many (say n) applications of this procedure, a given (α, β)-PRS R is transformed into a canonic (α, β)-PRS R'. Let h_i be the finite term substitution defined in i-th application of the procedure. We set $h = h_1 \circ h_2 \circ \ldots \circ h_n$. It is now easy to see that this canonic PRS R' and finite term substitution h satisfy the equivalences formulated in the theorem. $\square$

3 Automata-based symbolic representations

In order to perform reachability analysis of PRS, we need representation structures for (infinite) sets of process terms. For this purpose, we use a class of tree-automata, called *commutative hedge automata* [2], which recognize sets of trees modulo associativity / associativity-commutativity. These automata extend both (1) bottom-up tree automata over ranked alphabets [5], and (2) hedge automata recognizing sets of unbounded width trees [3].

3.1 Preliminaries

Presburger arithmetic is the first order logic of integers with addition and linear ordering. Given a formula φ, we denote by $FV(\varphi)$ the set of its free variables. Let $FV(\varphi) = \{x_1, \ldots, x_n\}$. Then, a vector $\boldsymbol{u} = (u_1, \ldots, u_n) \in \mathbb{Z}^n$ satisfies φ, written $\boldsymbol{u} \models \varphi$, if $\varphi(\boldsymbol{u}) = \varphi[x_i \leftarrow u_i]$ is true. Each formula φ defines a set of integer vectors $[\![\varphi]\!] = \{\boldsymbol{u} \in \mathbb{Z}^n \mid \boldsymbol{u} \models \varphi\}$. Presburger formulas define *semilinear sets* of integer vectors, i.e., finite union of sets of the form $\{\boldsymbol{x} \in \mathbb{Z}^n \mid \exists k_1, \ldots, k_n \in \mathbb{Z}, \boldsymbol{x} = \boldsymbol{v}_0 + k_1 \boldsymbol{v}_1 \cdots + k_n \boldsymbol{v}_m\}$, where $\boldsymbol{v}_i \in \mathbb{Z}^n$, for $1 \leq i \leq m$ (see [10]).

Given a word w over an alphabet $\Sigma = \{a_1, \ldots, a_n\}$, the *Parikh image* of w, denoted $\mathit{Parikh}(w)$, is the vector $(|w|_{a_1}, \ldots, |w|_{a_n})$. This definition can be generalized to sets of words (languages) over Σ in the obvious manner.

As usual, a set of words is *regular* if it is definable by a finite-state automaton. The notion of regularity can be transfered straightforwardly to sets of flat seq-terms. Similarly, the notion of semilinearity can be transfered to sets of flat par-term by associating with a term $X_1\|\cdots\|X_n$ the vector $\mathit{Parikh}(X_1\cdots X_n)$.

In the sequel, we will represent by γ a *constraint* which is either a regular language or a Presburger formula. We say that a word $w = a_1a_2\ldots a_n$ *satisfies* the constraint γ if $w \in \gamma$ (resp. $\mathit{Parikh}(w) \models \gamma$) when γ is a language (resp. a formula).

3.2 Commutative Hedge Automata

Let $\Sigma = \Sigma' \cup \Sigma_A$ be a finite alphabet, where Σ' is a ranked alphabet, and Σ_A is a finite set of associative operators. We assume that Σ' and Σ_A are disjoint. For $k \geq 0$, let Σ_k denote the set of elements of Σ' of rank k.

3.2.1 Σ-Terms:

Let $\mathcal{X}$ be a fixed countable set of variables $\{x_1, x_2, \ldots\}$. The set $T_\Sigma[\mathcal{X}]$ of Σ-terms over $\mathcal{X}$ is the smallest set such that:

- $\Sigma_0 \cup \mathcal{X} \subseteq T_\Sigma[\mathcal{X}]$,
- for $k \geq 1$, if $f \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma[\mathcal{X}]$, then $f(t_1, \ldots, t_k) \in T_\Sigma[\mathcal{X}]$,
- if $f \in \Sigma_A$, $t_1, \ldots, t_n \in T_\Sigma[\mathcal{X}]$ for some $n \geq 1$, and $\mathit{root}(t_i) \neq f$ for every $1 \leq i \leq n$, then $f(t_1, \ldots, t_n) \in T_\Sigma[\mathcal{X}]$, where $\mathit{root}(\sigma) = \sigma$ if $\sigma \in \Sigma_0 \cup \mathcal{X}$, and $\mathit{root}\big(g(u_1, \ldots, u_m)\big) = g$.

Note that if $f \in \Sigma_A$, we only consider terms of the form $f(t_1, \ldots, t_n)$ such that for every i, the root of t_i is different from f. Indeed, since f is associative, $f\big(t_1, \ldots, t_{i-1}, f(u_1, \ldots, u_m), t_{i+1}, \ldots, t_n\big)$ is equivalent to the term $f(t_1, \ldots, t_{i-1}, u_1, \ldots, u_m, t_{i+1}, \ldots, t_n)$.

Terms without variables are called *ground terms*. Let T_Σ be the set of ground terms of $T_\Sigma[\mathcal{X}]$. A term t in $T_\Sigma[\mathcal{X}]$ is *linear* if each variable occurs at most once in t. A *context* C is a linear term of $T_\Sigma[\mathcal{X}]$. Let $t_1, \ldots, t_n$ be terms of T_Σ, then $C[t_1, \ldots, t_n]$ denotes the term obtained by replacing in the context C the occurrence of the variable x_i by the term t_i, for each $1 \leq i \leq n$.

3.2.2 Definition of CH-automata:

Let us consider that $\Sigma_A = \Sigma'_A \cup \Sigma'_{AC}$ where Σ'_{AC} is a set of associative and commutative operators. We assume that Σ'_A and Σ'_{AC} are disjoint. Then, a CH-automaton is a tuple $\mathcal{A} = (Q, \Sigma, F, \Delta)$ where:

- Q is a union of disjoint finite sets of states $Q' \cup \bigcup_{f \in \Sigma_A} Q_f$,
- $F \subseteq Q$ is a set of final states,
- Δ is a set of rules of the form:

(i) $a \rightarrow q$, where $q \in Q', a \in \Sigma_0$,
(ii) $f(q_1, \ldots, q_k) \rightarrow q$, where $f \in \Sigma_k, q \in Q'$, and $q_i \in Q$,
(iii) $q \rightarrow q'$, where $(q, q') \in Q' \times Q' \cup \bigcup_{f \in \Sigma_A} Q_f \times Q_f$,
(iv) $f(\mathit{Reg}) \rightarrow q$, where $f \in \Sigma'_A$, $\mathit{Reg} \subseteq (Q \setminus Q_f)^*$ is a regular language given by a finite-state automaton, and $q \in Q_f$,

(v) $f(\varphi) \rightarrow q$, where $f \in \Sigma'_{AC}$, $q \in Q_f$, and φ is a Presburger formula such that $FV(\varphi) = \{x_q \mid q \in Q \setminus Q_f\}$.

We define a *move relation* $\rightarrow_\Delta$ between ground terms in $T_{\Sigma \cup Q}$ as follows: for every two terms t and t', we have $t \rightarrow_\Delta t'$ iff there exist a context C and a rule $r \in \Delta$ such that $t = C[s]$, $t' = C[s']$, and:

- $r = a \rightarrow q$, with $s = a$ and $s' = q$, or
- $r = q \rightarrow q'$, with $s = q$ and $s' = q'$, or
- $r = f(q_1, \ldots, q_k) \rightarrow q$, with $s = f(q_1, \ldots, q_k)$ and $s' = q$, or
- $r = f(Reg) \rightarrow q$, with $f \in \Sigma'_A$, $s = f(q_1, \ldots, q_n)$, $q_1 \cdots q_n \in Reg$, and $s' = q$, or
- $r = f(\varphi) \rightarrow q$, with $f \in \Sigma'_{AC}$, $s = f(q_1, \ldots, q_n)$, $Parikh(q_1 \cdots q_n) \models \varphi$, and $s' = q$.

Let $\xrightarrow{*}_\Delta$ denote the reflexive-transitive closure of $\rightarrow_\Delta$. A ground term $t \in T_\Sigma$ is *accepted by a state* q if $t \xrightarrow{*}_\Delta q$. Let $L_q = \{t \in T_\Sigma \mid t \xrightarrow{*}_\Delta q\}$. A ground term $t \in T_\Sigma$ is *accepted by the automaton* $\mathcal{A}$ if it is accepted by some final state $q \in F$. The CH-language of $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of all ground terms accepted by $\mathcal{A}$.

We have the following fact [4,15,19,20,2]:

Theorem 3.1 *The class of languages recognized by CH-automata is effectively closed under boolean operations, term substitutions and inverse of finite term substitutions. Moreover, the emptiness problem of CH-automata is decidable.*

3.3 CH-automata for PRS process terms

We consider process terms as trees and use CH-automata to represent sets of such trees. Indeed, for any finite set $C \subseteq Const$, the set T_C of process terms can be seen as the set of Σ-terms T_Σ where $\Sigma_0 = \{0\} \cup C$, $\Sigma'_A = \{\odot\}$, and $\Sigma'_{AC} = \{\|\}$.

Sets of process terms are recognized by CH-automata $\mathcal{A} = (Q, \Sigma, F, \Delta)$ such that (1) Q is the disjoint union $Q = Q' \cup Q_\odot \cup Q_\|$ where Q' is itself the disjoint union $Q' = Q_0 \cup Q_-$, and (2) the rules in Δ are of the form: (a) $X \rightarrow q$, where $q \in Q_-$, $X \in Const$, (b) $0 \rightarrow q$, where $q \in Q_0$, (c) $q \rightarrow q'$, where $(q, q') \in (Q_0)^2 \cup (Q_-)^2 \cup (Q_\odot)^2 \cup (Q_\|)^2$, (d) $\odot(Reg) \rightarrow q$, where $Reg \subseteq \big(Q \setminus (Q_\odot \cup Q_0)\big)^*$ is a regular language and $q \in Q_\odot$, and (e) $\|(\varphi) \rightarrow q$, where $q \in Q_\|$ and φ is a Presburger formula such that $FV(\varphi) = \{x_q \mid q \in Q \setminus (Q_\| \cup Q_0)\}$.

In other words, the states in $Q_\odot$ (resp. $Q_\|$) recognize trees whose root is $\odot$ (resp. $\|$). The states in Q_- recognize constants in C, and the states in Q_0 recognize 0.

4 Computing 1-step reachability sets for canonic PRS

Let us consider a canonic PRS $R = R_\odot \cup R_\|$ and let $\mathcal{A} = (Q, \Sigma, F, \Delta)$ be a CH-automaton recognizing a set L of process terms. We show that the sets $Post_R(L)$ and $Pre_R(L)$ are effectively representable and computable by CH-automata.

For a given canonic PRS R' and a given set of terms L_1, we write $R'(L_1)$ as an abbreviation for $Post_{R'}(L_1)$. In the following we use the fact that given a regular set L_2 of flat seq-terms, the set $R'_\odot(L_2)$ is again regular and easily constructible.

The same holds for any semilinear sets L_3 of flat par-terms and $R'_{\|}(L_3)$.

We construct a CH-automaton $\mathcal{A}' = (\widetilde{Q}, \Sigma, \widetilde{F}, \widetilde{\Delta})$ which recognizes $R(L)$, where $\widetilde{Q}$ is the set of states, $\widetilde{F}$ is the set of final states, and $\widetilde{\Delta}$ is the set of rules. Let C be a finite set of process constants such that $C \supseteq \mathit{Const}(R)$ and $L \subseteq T_C$.

4.1 The set of states

The set of states $\widetilde{Q}$ includes the set of states Q of $\mathcal{A}$ and contains new states q_X, which are assumed to accept precisely the singletons $\{X\}$ (i.e., $L_{q_X} = \{X\}$), for each $X \in C$. Let Q_R be the set of states $Q_R = \{q_X \mid X \in C\}$. In addition, the set $\widetilde{Q}$ contains states which recognize the set $R(L_q)$ of *immediate successors* of terms in L_q for each $q \in Q \cup Q_R$. In order to ensure (during the construction) that the recognized trees are always in canonical form, we need to partition the sets of recognized trees according to their types (given by their root).

We associate with each $q \in Q \cup Q_R$ different states $(q, -)$, $(q, 0)$, $(q, \odot)$, and $(q, \|)$ recognizing immediate successors of terms in L_q which are respectively constants in C, null (equal to 0), $\odot$-rooted terms, and $\|$-rooted terms.

Let $Q = Q_0 \cup Q_- \cup Q_\odot \cup Q_\|$. We consider that the set $\widetilde{Q}$ is equal to the union of the following sets: (1) $\widetilde{Q}_0 = Q_0 \cup \{(q,0) \mid q \in Q \cup Q_R\}$, (2) $\widetilde{Q}_- = Q_- \cup Q_R \cup \{(q,-) \mid q \in Q \cup Q_R\}$, and (3) $\widetilde{Q}_\omega = Q_\omega \cup \{(q,\omega) \mid q \in Q \cup Q_R\}$, for $\omega \in \{\odot, \|\}$. Moreover, we consider that $\widetilde{F} = \{(q,-), (q,0), (q,\odot), (q,\|) \mid q \in F\}$.

4.2 Rewrite system over the alphabet of states

Rules in CH-automata (of the forms $\omega(\gamma) \to q$) involve constraints on sequences of *states*, whereas the systems $R_\odot$ and $R_\|$ are defined over the alphabet of process constants. Therefore, we define the systems $S_\odot = \alpha(R_\odot)$ and $S_\| = \alpha(R_\|)$ where α is the substitution such that $\alpha(X) = q_X$, for every $X \in C$ (extended in the standard way to terms, rules, and sets of rules).

4.3 The set of transition rules

The set $\widetilde{\Delta}$ is defined as the smallest set of transition rules which (1) contains Δ, (2) contains the set of rules $X \to q_X$ for every $X \in \mathit{Const}$, and (3) is such that:

(β_1) **Closure rules: successors of process constants and 0:**
(a) If $X \xrightarrow{*}_\Delta q$, then $\omega\big(S_\omega(q_X)\big) \to (q,\omega) \in \widetilde{\Delta}$,
(b) If $0 \xrightarrow{*}_\Delta q$, then $\omega\big(S_\omega(0)\big) \to (q,\omega) \in \widetilde{\Delta}$.
The rule (a) says that if X is in L_q, then all its immediate ω-successors obtained by applying once the system R_ω are also immediate successors of L_q. The rule (b) says the same thing for successors of 0.

(β_2) **Closure rule: successors of ω-rooted terms:** If $\omega(\gamma) \to p \in \Delta$, then $\omega\big(S_\omega(\sigma(\gamma))\big) \to (p,\omega) \in \widetilde{\Delta}$, where σ is the substitution such that $\forall q \in Q \cup Q_R$, $\sigma(q) = \{q\} \cup \{q_X \mid X \xrightarrow{*}_\Delta q\} \cup \{0 \mid 0 \xrightarrow{*}_\Delta q\}$.
This rule says that if $\omega(X_1,\ldots,X_n) \in L_p$ and $\omega(X'_1,\ldots,X'_m) \in R_\omega\big(\omega(X_1,\ldots,X_n)\big)$, then $\omega(X'_1,\ldots,X'_m)$ is a ω-successor of L_p.

(β_3) **Propagation rule:** If $\omega(\gamma) \to p \in \Delta$, then $\omega\big(E_\omega(\gamma)\big) \to (p, \omega) \in \widetilde{\Delta}$, where E is a canonic PRS defined as $E = \{q \hookrightarrow (q, -), q \hookrightarrow (q, \odot), q \hookrightarrow (q, \|)\}$.
The rule says that if $\odot(t_1, \ldots, t_n) \in L_p$ and t'_1 is a successor of t_1, then $\odot(t'_1, \ldots, t_n)$ is a successor of L_p. Moreover, if $\|(t_1, \ldots, t_n) \in L_p$ and t'_i is a successor of t_i, then $\|(t_1, \ldots, t'_i, \ldots, t_n)$ is a successor of L_p.
Note that we need to distinguish between $E_\|(\gamma)$ and $E_\odot(\gamma)$ to ensure that the prefix-rewrite strategy of the $\odot$ is correctly taken into account.

(β_4) **Term flattening rules:**
(a) If $\omega(\gamma) \to (q, \omega) \in \widetilde{\Delta}$ and $q' \in \gamma$, then $q' \to (q, -) \in \widetilde{\Delta}$ if $q' \in \widetilde{Q}_-$, and $q' \to (q, \overline{\omega}) \in \widetilde{\Delta}$ if $q' \in \widetilde{Q}_{\overline{\omega}}$.
(b) If $\omega(\gamma) \to (q, \omega) \in \widetilde{\Delta}$ and $0 \in \gamma$, then $0 \to (q, 0) \in \widetilde{\Delta}$.
The rules say that if $\omega(t)$ is a successor of L_q, then t is also a successor of L_q.

Now we prove that the construction is correct.

Lemma 4.1 *For every process term t, and every $q \in Q \cup Q_R$ we have:*

(1) $t \xrightarrow{*}_{\widetilde{\Delta}} (q, 0)$ iff $t \in Post_R(L_q)$ and $t = 0$,

(2) $t \xrightarrow{*}_{\widetilde{\Delta}} (q, -)$ iff $t \in Post_R(L_q)$ and $t \in C$,

(3) $t \xrightarrow{*}_{\widetilde{\Delta}} (q, \omega)$ iff $t \in Post_R(L_q)$ and $root(t) = \omega$, for $\omega \in \{\odot, \|\}$.

Proof. We consider the (more complicated) left-to-right direction. The proof is by structural induction on t:

- $t = X \xrightarrow{*}_{\widetilde{\Delta}} (q, -)$ (the case where $t = 0 \xrightarrow{*}_{\widetilde{\Delta}} (q, 0)$ is similar). Note that the rules of $\widetilde{\Delta}$ do not allow derivations of the form $X \xrightarrow{*}_{\widetilde{\Delta}} (q, 0)$ or $X \xrightarrow{*}_{\widetilde{\Delta}} (q, w)$.
  Such a derivation has necessarily the following form:
$$X \to_{\widetilde{\Delta}} q_X \to_{\widetilde{\Delta}} (q, -)$$
  where the rule $q_X \to_{\widetilde{\Delta}} (q, -)$ is a β_4-rule. There are three cases:
  (i) There exists $w \in \{\odot, ||\}$, such that $w(\gamma) \to_\Delta q$, $\omega\big(S_\omega(\sigma(\gamma))\big) \to (q, \omega)$ is in $\widetilde{\Delta}$, and $q_X \in S_\omega(\sigma(\gamma))$. Suppose that $\omega = \odot$, the other case where $\omega = ||$ is analogous. This means that there exists $q_{X_1} \cdots q_{X_n} \in \sigma(\gamma)$ such that $q_X \in S_\odot(q_{X_1} \cdots q_{X_n})$. This means that $X \in R_\odot\big(\odot(X_1, \ldots, X_n)\big)$. Since $q_{X_1} \cdots q_{X_n} \in \sigma(\gamma)$ and $\odot(\gamma) \to_\Delta q$, it follows that $\odot(X_1, \ldots, X_n) \in L_q$. Therefore, $X \in R_\odot(L_q)$, i.e., $X \in Post_R(L_q)$.
  (ii) There exists a constant Y such that $Y \xrightarrow{*}_\Delta q$, $\omega\big(S_\omega(q_Y)\big) \to (q, \omega)$ is in $\widetilde{\Delta}$, and $q_X \in S_\omega(q_Y)$. Suppose here also that $w = \odot$, the other case where $w = ||$ is analogous. This means that $q_X \in S_\odot(q_Y)$, and that $X \in R_\odot(Y)$. Since $Y \in L_q$, it follows that $X \in R_\odot(L_q)$, i.e., $X \in Post_R(L_q)$.
  (iii) $0 \xrightarrow{*}_\Delta q$, $\omega\big(S_\omega(0)\big) \to (q, \omega)$ is in $\widetilde{\Delta}$,and $q_X \in S_\omega(0)$. Suppose here also that $w = \odot$, the other case where $w = ||$ is analogous. This means that $q_X \in S_\odot(0)$, and that $X \in R_\odot(0)$. Since $0 \in L_q$, it follows that $X \in R_\odot(L_q)$, i.e., $X \in Post_R(L_q)$.

- $t = \odot(t_1, \ldots, t_n) \xrightarrow{*}_{\widetilde{\Delta}} (q, \odot)$. The case where $t = ||(t_1, \ldots, t_n) \xrightarrow{*}_{\widetilde{\Delta}} (q, ||)$ is similar. There are three cases:

(i) There exist n constants $X_1, \ldots, X_n$ such that

$$t = \odot(t_1, \ldots, t_n) \xrightarrow{*}_{\widetilde{\Delta}} \odot(q_{X_1}, \ldots, q_{X_n}) \rightarrow_{\widetilde{\Delta}} (q, \odot).$$

In this case, every t_i is necessarily equal to the constant X_i. Then, the $\widetilde{\Delta}$-rule $\odot(\mathit{Reg}) \rightarrow (q, \odot)$, where $q_{X_1} \cdots q_{X_n} \in \mathit{Reg}$ is either a β_1 or a β_2 rule. Let us consider the case where it is a β_1-rule, the other case being similar. Let then X be a constant such that $X \xrightarrow{*}_{\Delta} q$ and $\mathit{Reg} = S_{\odot}(q_X)$. Since $q_{X_1} \cdots q_{X_n} \in \mathit{Reg}$, this means as previously that $\odot(X_1, \ldots, X_n) \in R_{\odot}(X)$, i.e., since $X \in L_q$ that $\odot(t_1, \ldots, t_n) = \odot(X_1, \ldots, X_n) \in \mathit{Post}_R(L_q)$.

(ii) There exist k constants $X_1, \ldots, X_k$ and $n-k$ states $q_{k+1}, \ldots, q_n$ in Q such that

$$t = \odot(t_1, \ldots, t_n) \xrightarrow{*}_{\widetilde{\Delta}} \odot(q_{X_1}, \ldots, q_{X_k}, q_{k+1}, \ldots, q_n) \rightarrow_{\widetilde{\Delta}} (q, \odot).$$

In this case, for every i, $1 \leq i \leq k$, t_i is necessarily equal to the constant X_i, and for every i, $k+1 \leq i \leq n$, $t_i \in L_{q_i}$. Then, the $\widetilde{\Delta}$-rule $\odot(\mathit{Reg}) \rightarrow (q, \odot)$, where $q_{X_1} \cdots q_{X_k} q_{k+1} \cdots q_n \in \mathit{Reg}$ is necessarily a β_2 rule. Let then $\odot(\mathit{Reg}') \rightarrow q$ be a rule in Δ such that $\mathit{Reg} = S_{\odot}(\sigma(\mathit{Reg}'))$. Since $q_{X_1} \cdots q_{X_k} q_{k+1} \cdots q_n \in \mathit{Reg}$, it follows that there exists $q_{Y_1} \cdots q_{Y_m} q_{k+1} \cdots q_n \in \sigma(\mathit{Reg}')$ such that $q_{X_1} \cdots q_{X_k} q_{k+1} \cdots q_n \in S_{\odot}(q_{Y_1} \cdots q_{Y_m} q_{k+1} \cdots q_n)$, and therefore that $q_{X_1} \cdots q_{X_k} \in S_{\odot}(q_{Y_1} \cdots q_{Y_m})$, and hence that $\odot(X_1, \ldots, X_k) \in R_{\odot}\big(\odot(Y_1, \ldots, Y_m)\big)$, and that $\odot(X_1, \ldots, X_k, t_{k+1}, \ldots, t_n) \in R_{\odot}\big(\odot(Y_1, \ldots, Y_m, t_{k+1}, \ldots, t_n)\big)$.

Since $q_{Y_1} \cdots q_{Y_m} q_{k+1} \cdots q_n \in \sigma(\mathit{Reg}')$, we get that $\odot(Y_1, \ldots, Y_m, t_{k+1}, \ldots, t_n) \in L_q$, and since $\odot(X_1, \ldots, X_k, t_{k+1}, \ldots, t_n) \in R_{\odot}\big(\odot(Y_1, \ldots, Y_m, t_{k+1}, \ldots, t_n)\big)$, it follows that $\odot(X_1, \ldots, X_k, t_{k+1}, \ldots, t_n) \in \mathit{Post}_R(L_q)$. Therefore

$$t = \odot(t_1, \ldots, t_n) = \odot(X_1, \ldots, X_k, t_{k+1}, \ldots, t_n) \in \mathit{Post}_R(L_q)$$

(iii) There exist n states $q_1, \ldots, q_n$ where at least one q_i is of the form $(p, ||)$ or $(p, -)$ where

$$t = \odot(t_1, \ldots, t_n) \xrightarrow{*}_{\widetilde{\Delta}} \odot(q_1, \ldots, q_n) \rightarrow_{\widetilde{\Delta}} (q, \odot)$$

In this case, the last rule that is applied during the derivation is necessarily a β_3-rule. Then, β_3 implies that for every i, $2 \leq i \leq n$, $q_i \in Q$, and that it is the state q_1 that is of the form $(p_1, ||)$ (the case where it is of the form $(p_1, -)$ is similar). More precisely, it implies that there exist a rule $\odot(\mathit{Reg}) \rightarrow q$ in Δ and a rule $\odot(\mathit{Reg}') \rightarrow (q, \odot)$ in $\widetilde{\Delta}$ such that $p_1 q_2 \cdots q_n \in \mathit{Reg}$ and $(p_1, ||) q_2 \cdots q_n \in \mathit{Reg}'$.

By structural induction, it follows that $t_1 \in \mathit{Post}_R(L_{p_1})$. Let then $t_1' \in L_{p_1}$ be such that $t_1 \in \mathit{Post}_R(t_1')$. It follows that $\odot(t_1, \ldots, t_n) \in \mathit{Post}_R\big(\odot(t_1', \ldots, t_n)\big)$, and since for i, $2 \leq i \leq n$, $t_i \in L_{q_i}$ we have:

$$\odot(t_1', \ldots, t_n) \xrightarrow{*}_{\Delta} \odot(p_1, q_2, \ldots, q_n) \rightarrow_{\Delta} q$$

It follows then that $t = \odot(t_1, \ldots, t_n) \in \mathit{Post}_R(L_q)$.

□

Therefore, we have:

Theorem 4.2 *For every canonic PRS R and every CH-automaton $\mathcal{A}$, we have $Post_R\big(L(\mathcal{A})\big) = L(\mathcal{A}')$.*

As $Pre_R(L) = Post_{R^{-1}}(L)$, the previous construction can also be used to compute 1-step *backward* reachability sets.

5 Computing reachability sets for PAD and wPAD

In this section, we solve the problem of computing both reachability sets and 1-step reachability sets for PAD and wPAD systems. Computing reachability sets is difficult for PRS in general. One of the reasons is that already the reachability sets of Petri nets are not semilinear. In [2] we show that the reachability sets of a given canonic PRS system R can be effectively computed provided the underlying multiset rewrite system $R_{\|}$ is effectively semilinear. This is, for example, the case of canonic PAD systems due to the result of [6] concerning context-free multiset rewrite systems (BPP processes).

Theorem 5.1 ([2]) *Let $\mathcal{A}$ be a CH-automaton recognizing a set of process terms and R be a canonic PAD. Then the sets $Post^*_R(L(\mathcal{A}))$ and $Pre^*_R(L(\mathcal{A}))$ are computable and effectively representable by CH-automata.*

Using this theorem and the results of the previous section, we get the following.

Theorem 5.2 *For every PAD R and every CH-automaton $\mathcal{A}$, the sets $Post_R(L(\mathcal{A}))$, $Pre_R(L(\mathcal{A}))$, $Post^*_R(L(\mathcal{A}))$, and $Pre^*_R(L(\mathcal{A}))$ are computable and effectively representable by CH-automata.*

Proof. Theorem 2.1 implies that for every PAD R and every set of terms L, there exists a canonic PAD R' and a finite term substitution h such that $Post^*_R(L) = h(Post^*_{R'}(h^{-1}(L)))$ and $Post_R(L) = h(Post_{R''}(h^{-1}(L)))$, where R'' is the set R' restricted to rules labelled with actions of $Act(R)$. Hence, CH-automata representing the sets $Post^*_R(L(\mathcal{A}))$ and $Post_R(L(\mathcal{A}))$ are constructible due to closure properties of CH-automata and Theorems 5.1 and 4.2. The proof for $Pre^*_R(L(\mathcal{A}))$ and $Pre_R(L(\mathcal{A}))$ is analogous. $\square$

Now we show that the previous theorem holds for wPAD as well. Recall that states of wPAD are pairs pt of a control state p and a term t. The sets of such states can be represented by *CHA-mappings*.

Definition 5.3 Let R be a wPRS. A *CHA-mapping* Λ is a mapping assigning to each control state $p \in M(R)$ a CH-automaton $\Lambda(p)$. A *CHA-mapping* Λ represents the set of states $L(\Lambda) = \{pt \mid p \in M(R), t \in L(\Lambda(p))\}$.

Theorem 5.4 *For every wPAD R and every CHA-mapping Λ, the sets $Post_R(L(\Lambda))$, $Pre_R(L(\Lambda))$, $Post^*_R(L(\Lambda))$, and $Pre^*_R(L(\Lambda))$ are computable and effectively representable by CHA-mappings.*

Proof. Let R be a wPAD. For each pair of control states $p, q \in M(R)$ we set $R_{p,q} = \{t_1 \overset{a}{\hookrightarrow} t_2 \mid pt_1 \overset{a}{\hookrightarrow} qt_2 \text{ is a rule of } R\}$. Note that each $R_{p,q}$ is a PAD system.

CHA-mapping Λ_1 representing $Post_R(L(\Lambda))$ is defined as follows. For each $q \in M(R)$, $\Lambda_1(q)$ is an CH-automaton satisfying

$$L(\Lambda_1(q)) = \bigcup_{p \in M(R)} Post_{R_{p,q}}\big(L(\Lambda(p))\big).$$

CHA-mapping Λ_2 representing $Post^*_R(L(\Lambda))$ is defined inductively with respect to ordering $<$ on set $M(R)$ of control states. For every minimal element r of $M(R)$, $\Lambda_2(r)$ is a CH-automaton satisfying $L(\Lambda_2(r)) = Post^*_{R_{r,r}}\big(L(\Lambda(r))\big)$. For non-minimal element q of $M(R)$, $\Lambda_2(q)$ is a CH-automaton satisfying

$$L(\Lambda_2(q)) = Post^*_{R_{q,q}}\Big(L(\Lambda(q)) \cup \bigcup_{p<q} Post_{R_{p,q}}\big(L(\Lambda_2(p))\big)\Big).$$

CHA-mappings Λ_1, Λ_2 are constructible due to Theorem 5.2 and the fact that CH-automata are closed under union. The proof for $Pre_R(L(\Lambda))$ and $Pre^*_R(L(\Lambda))$ is analogous. $\square$

As mentioned in [2], the generic algorithm presented there can employ known algorithms computing semilinear overapproximations of reachability sets for Petri nets in order to compute overapproximations of reachability sets for general canonic PRS systems. If we use this approximative algorithm for canonic PRS instead of exact algorithm for canonic PAD system in Theorems 5.2 and 5.4, we get an algorithm computing overapproximations of reachability sets for general wPRS systems. Note that 1-step reachability sets for wPRS systems can still be computed precisely as Theorems 5.2 and 5.4 hold even for (w)PRS if we restrict our attention only to 1-step reachability sets.

6 Model checking of wPAD against EF logic

This section presents a straightforward application of Theorem 5.4. We consider a variant of EF logic combining both action-based and state-based approaches. We show that the global model checking problem of wPAD systems against this logic is decidable.

Formulae of EF logic are defined as

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle a \rangle \varphi \mid \mathsf{EF}\varphi,$$

where A ranges over set AP of *atomic propositions* and a ranges over Act. Here, formulae are interpreted over states of wPRS systems. For each atomic proposition A, let $V(A)$ denotes its valuation, i.e. the set of states where A holds. We define when a state pt of a given wPRS system R *satisfies* a formula φ, written $R, pt \models \varphi$, by induction on the structure of φ.

$$
\begin{array}{lll}
R, pt \models A & \text{iff} & pt \in V(A) \\
R, pt \models \neg\varphi & \text{iff} & R, pt \not\models \varphi \\
R, pt \models \varphi_1 \wedge \varphi_2 & \text{iff} & R, pt \models \varphi_1 \text{ and } R, pt \models \varphi_2 \\
R, pt \models \langle a \rangle \varphi & \text{iff} & \exists qt' \text{ such that } pt \xrightarrow{a}_R qt' \text{ and } R, qt' \models \varphi \\
R, pt \models \mathsf{EF}\varphi & \text{iff} & \exists qt' \text{ such that } pt \xrightarrow{*}_R qt' \text{ and } R, qt' \models \varphi
\end{array}
$$

Theorem 6.1 *For every wPAD system R and every EF formula φ over atomic propositions with valuations given by CHA-mappings, the set of states of R satisfying φ is computable and effectively representable by a CHA-mapping.*

Proof. The theorem follows directly from Theorem 5.4 and closure properties of CH-automata. Here we mention just the induction step corresponding to operator $\langle a \rangle$. Let $\varphi = \langle a \rangle \psi$ and let CHA-mapping Λ recognizes all states satisfying ψ. We construct a CHA-mapping Λ', which recognizes all states where φ holds, to satisfy $L(\Lambda') = Pre_{R_a}\big(L(\Lambda)\big)$, where R_a is the set R restricted to rules with label a. Such a CHA-mapping Λ' is constructible due to Theorem 5.4. □

This theorem gives a positive answer to open questions formulated in [14], namely whether model checking of wBPP, wPA, and wPAD systems against action-based EF logic is decidable. Our result is tight as model checking of state extended PAD (defined as wPAD where rules may not respect the ordering on control states) against EF logic is already undecidable. In fact, the problem is undecidable even for the subclass of state extended PAD called *multiset automata* and EF formulae with the only atomic proposition *true* (this can be proved by the arguments of [6] showing that model checking of Petri nets against EF logic is undecidable).

7 Bounded reachability analysis of synchronized PAD

First we give a precise definition of synchronized (α, β)-PRS. Let *Act* be a disjoint union $Async \cup Sync \cup \{\tau\}$. We assume that to each $a \in Sync$ corresponds a co-action $\tilde{a} \in Sync$ such that $\tilde{\tilde{a}} = a$. Intuitively, *Sync* is the set of all *synchronization actions*, i.e. actions which must be performed simultaneously with their corresponding co-actions. A *synchronized (α, β)-PRS R* is defined as standard (α, β)-PRS. Instead of 'synchronized (α, β)-PRS' we use shorter names like SPAD, SPRS, etc.

Let $\rightarrow$ be the least transition relation over terms satisfying the inference rules

$$
\frac{(t_1 \overset{b}{\hookrightarrow} t_2) \in R}{t_1 \xrightarrow{b} t_2} \qquad
\frac{t_1 \xrightarrow{b} t_2}{t_1 \| t \xrightarrow{b} t_2 \| t} \qquad
\frac{t_1 \xrightarrow{b} t_2}{t_1 \odot t \xrightarrow{b} t_2 \odot t} \qquad
\frac{t_1 \xrightarrow{a} t_1' \quad t_2 \xrightarrow{\tilde{a}} t_2'}{t_1 \| t_2 \xrightarrow{\tau} t_1' \| t_2'}
$$

where a ranges over *Sync* and b ranges over *Act*. An transition step induced by the last rule is called *synchronization*. The transition relation $\rightarrow_R$ is then defined as the restriction of $\rightarrow$ to transitions labelled with actions of $Act \setminus Sync$.

Now we present a technique for computing underapproximations of these sets in style of [18]. Given a synchronized (α, β)-PRS R and $n > 0$, we construct an (α, β)-wPRS R_n which mimics (prefixes of) all behaviors of R with at most n

synchronizations. The system R_n uses control states $0 < 1 < \ldots < 3n$. For every rewrite rule $r = (t_1 \overset{b}{\hookrightarrow} t_2)$ of R, let $Z_r \notin \mathit{Const}(R)$ be a fresh process constant. If

- $b \in \mathit{Async} \cup \{\tau\}$ then we add to R_n the rule $(3i)t_1 \overset{b}{\hookrightarrow} (3i)t_2$ for every $0 \leq i \leq n$.
- $b = a \in \mathit{Sync}$ the we add to R_n rules $(3i)t_1 \overset{\tau'}{\hookrightarrow} (3i{+}1)Z_r$ and $(3i{+}2)Z_r \overset{\tau'}{\hookrightarrow} (3i{+}3)t_2$ for every $0 \leq i < n$.
- $b = \tilde{a} \in \mathit{Sync}$ the we add to R_n rule $(3i+1)t_1 \overset{\tau}{\hookrightarrow} (3i+2)t_2$ for every $0 \leq i < n$.

Intuitively, every synchronization via actions $a, \tilde{a}$ is replaced by a sequence of actions $\tau'\tau\tau'$. The changes of control states prevents interleaving of this sequence with other actions. Moreover, use of fresh process constants ensures that the rules under a and $\tilde{a}$ are applied on different parts of the current term.

Let R be an SPAD and L be a set of states represented by a CH-automaton. Theorem 5.4 says that we can construct a CHA-mapping Λ such that $L(\Lambda) = \mathit{Post}^*_{R_n}(\{0\} \times L)$. Obviously, the set $\bigcup_{0 \leq i \leq n} L(\Lambda(3i))$ is an underapproximation of $\mathit{Post}^*_R(L)$. Further, with increasing n, we can compute better approximations of this set. Moreover, if for n and $n+1$ the computed underapproximations are the same, we know that we have exactly the set $\mathit{Post}^*_R(L)$.

The same technique can be employed to underapproximate the set $\mathit{Pre}^*_R(L)$. The sets $\mathit{Post}_R(L)$ and $\mathit{Pre}_R(L)$ can be computed precisely using a similar approach.

8 Conclusion

We have presented an automata-based symbolic reachability analysis algorithm for the class of wPAD systems. This algorithm is based on the use of a class of unranked tree automata (called CH-automata) which can recognize sets of configurations closed under the algebraic properties of the sequential and parallel composition. We used the reachability analysis algorithm, together with one-step successor computation (and boolean operations on CH-automata), in order to define an algorithm for the global model checking of wPAD against the EF logic with regular atomic predicates. These results generalize those proved in [2] concerning the class of (canonic) PAD systems, which is a strict subclass of wPAD, pushing the known decidability limit of EF model checking further up in the (se/w)PRS hierarchy, and answering open questions left in [14].

We have also shown that the symbolic reachability algorithm for wPAD can be used to compute under approximations of the set of reachable configurations of synchronized PAD (SPAD), a (Turing) powerful model introduced in [21] for modeling multithreaded programs (with dynamic creation of communicating processes and procedure calls).

References

[1] Bouajjani, A. and T. Touili, *Reachability Analysis of Process Rewrite Systems*, in: *Proc. of FSTTCS 2003*, LNCS **2914** (2003), pp. 74–87.

[2] Bouajjani, A. and T. Touili, *On computing reachability sets of process rewrite systems*, in: *Proceedings of RTA 2005*, LNCS **3467** (2005), pp. 484–499.

[3] Bruggemann-Klein, A., M. Murata and D. Wood, *Regular tree and regular hedge languages over unranked alphabets*, Research report (2001).

[4] Colcombet, T., *Rewriting in the partial algebra of typed terms modulo ac*, in: *Proceedings of INFINITY'02*, ENTCS **68** (2002).

[5] Comon, H., M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi, *Tree automata techniques and applications*, Available on: http://www.grappa.univ-lille3.fr/tata (1997).

[6] Esparza, J., *Decidability of model checking for infinite-state concurrent systems*, Acta Informatica **34** (1997), pp. 85–107.

[7] Esparza, J., *Grammars as processes*, in: *Formal and Natural Computing*, LNCS **2300** (2002).

[8] Esparza, J. and J. Knoop, *An automata-theoretic approach to interprocedural dataflow analysis*, in: *Proceedings of FOSSACS'99*, LNCS **1578**, 1999, pp. 14–30.

[9] Esparza, J. and A. Podelski, *Efficient algorithms for pre* and post* on interprocedural parallel flow graphs*, in: *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POLP'00)* (2000), pp. 1–11.

[10] Harrison, M. A., "Introduction to Formal Language Theory," Addison-Wesley, 1978.

[11] Jančar, P., A. Kučera and R. Mayr, *Deciding bisimulation-like equivalences with finite-state processes*, Theor. Comput. Sci. **258** (2001), pp. 409–433.

[12] Křetínský, M., V. Řehák and J. Strejček, *Extended process rewrite systems: Expressiveness and reachability*, in: *Proceedings of CONCUR'04*, LNCS **3170** (2004), pp. 355–370.

[13] Křetínský, M., V. Řehák and J. Strejček, *On extensions of process rewrite systems: Rewrite systems with weak finite-state unit*, in: *Proceedings of INFINITY'03*, ENTCS **98** (2004), pp. 75–88.

[14] Křetínský, M., V. Řehák and J. Strejček, *Reachability of Hennessy-Milner properties for weakly extended PRS*, in: *Proceedings of FSTTCS 2005*, LNCS **3821** (2005), pp. 213–224.

[15] Lugiez, D., *Counting and equality constraints for multitree automata*, in: *Proceedings of FoSSaCS 2003*, LNCS **2620** (2003), pp. 328–342.

[16] Lugiez, D. and P. Schnoebelen, *The regular viewpoint on PA-processes*, in: *Proc. of CONCUR'98*, LNCS **1466** (1998), pp. 50–66.

[17] Mayr, R., *Process rewrite systems*, Information and Computation **156** (2000), pp. 264–286.

[18] Qadeer, S. and J. Rehof, *Context-bounded model checking of concurrent software*, in: *Proceedings of TACAS'2005*, LNCS **3440** (2005), pp. 93–107.

[19] Seidl, H., T. Schwentick and A. Muscholl, *Numerical document queries*, in: *Proceedings of PODS'03* (2003), pp. 155–166.

[20] Touili, T., "Analyse symbolique de systèmes infinis basée sur les automates: Application à la vérification de systèmes paramétrés et dynamiques," Ph.D. thesis, University of Paris 7 (2003).

[21] Touili, T., *Dealing with communication for dynamic multithreaded recursive programs*, in: *Proceedings of VISSAS'05*, 2005.