Theoretical Computer Science 411 (2010) 3460–3468

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Verifying parallel programs with dynamic communication structures

Tayssir Touili *, Mohamed Faouzi Atig

LIAFA, CNRS & Univ. Paris Diderot, Case 7014, 75205 Paris 13, France

ARTICLE INFO

Keywords:
Model checking
Concurrent programs
Reachability analysis
Pushdown Networks
Recognizable sets

ABSTRACT

We address the verification problem of networks of communicating pushdown systems modeling communicating parallel programs with procedure calls. Processes in such networks can read the control state of the other processes according to a given communication structure (specifying the observability rights between processes). The reachability problem of such models is undecidable in general. First, we define a class of networks that effectively preserves recognizability (hence, its reachability problem is decidable). Then, we consider networks where the communication structure can change dynamically during the execution according to a *phase* graph. The reachability problem for these dynamic networks being undecidable in general, we define a subclass for which it becomes decidable. Then, we consider reachability when the switches in the communication structures are bounded. We show that this problem is undecidable even for one switch. We define a natural class of models for which this problem is decidable. This class can be used in the definition of an efficient semi-decision procedure for the analysis of the general model of dynamic networks. Our techniques allowed to find bugs in two versions of a Windows NT Bluetooth driver.

1. Introduction

Verification of concurrent software is a difficult task in the model-checking community. Indeed, concurrent programs include various complex features such as (1) the presence of recursive procedure calls, which can lead to an unbounded number of calls, and (2) concurrency and synchronization between parallel processes. It is well known that checking whether a given control point is reachable is undecidable for programs with recursive procedures and synchronisation statements. During the last few years, several authors have addressed this issue. Different models of these programs have been proposed and analysed.

Pushdown systems have been proposed as an adequate formalism to describe *pure sequential recursive programs* [1–3]. This allows to represent the potentially infinite configurations of recursive programs in a symbolic manner using regular languages [4,5,2]. Thus, a natural approach that allows to reason about multithreaded programs is to consider models based on parallel compositions of pushdown systems [6–10]. Unfortunately, such models are undecidable (it suffices to have two communicating pushdown systems to get undecidability).

Recently, we defined in [11] a new model for multithreaded programs based on networks of pushdown systems. Our model consists of a finite number of parallel processes, each of them corresponding to a pushdown system, and where each process can read the control states of the other ones according to a given *communication structure* specifying the observation rights between processes. Such networks (called PDNs in this paper) are obviously Turing powerful when cyclic communication structures are allowed. We restricted ourselves in [11] to networks with *acyclic* communication structures. In order to represent infinite sets of configurations, we considered symbolic representation structures based

* Corresponding author. Tel.: +33 144275419; fax: +33 144276849.
E-mail addresses: touili@liafa.jussieu.fr (T. Touili), atig@liafa.jussieu.fr (M.F. Atig).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.05.028

on (multidimensional) finite-state automata defining recognizable and rational sets of vectors of words. (Recognizable sets are sets definable as finite unions of products of regular languages). We showed in [11] that (1) reachability is decidable for acyclic networks, that (2) such networks do not preserve recognizability, and (3) we defined a subclass of such networks for which we were able to effectively characterize the reachable configurations by a rational set.

In this work, we go further with this model. First, we define a natural subclass called *stable* acyclic PDNs and prove that it effectively preserves recognizability. Then, we consider networks with dynamic changes in the communication structure according to a *phase graph*, where each phase corresponds to an acyclic PDN. The phase graph specifies the possible switches between a finite number of phases, and the constraints on the configurations under which the system can move from a phase to another. We call this new model MAPN (for Multiphase Acyclic Pushdown Networks). MAPN is a natural model to represent programs where the communication structure between processes can change dynamically.

We show that reachability in MAPN can be reduced to reachability in (possibly cyclic) PDNs. Thus, MAPN has an undecidable reachability problem (even if each communication structure in each phase is acyclic) if we allow cyclic phase graphs. In fact, we prove that the reachability problem is undecidable as soon as we allow one phase switch (and even if communication structures are acyclic).

Then, we define two classes of MAPNs for which reachability becomes decidable. We derive from this a bounded phase-switch analysis procedure for the general MAPN model. For that, we show that it is possible to decompose each given MAPN into an equivalent model where each phase corresponds to a stable acyclic PDN. Finally, we show how the bounded phase-switch analysis of MAPN allows to define a semi-algorithm to decide reachability for *general* PDNs (even cyclic ones). This result generalizes the algorithms proposed in [7,9,12] for bounded context-switch analysis. Indeed, our notion of phase is more general than the notions of context used in these works in the sense that, if we encode our model in those proposed in [7,9,12], one single phase according to our definition may correspond to an unbounded number of context switches in their models. Thus, our bounded phase analysis may allow an arbitrary number of context switches (in the sense of [7,9,12]).

Our MAPN model is a natural model to represent programs where the communication structure between processes can change dynamically. Our PDN model can also be used to describe concurrent programs with synchronisation and procedure calls such as e.g. two versions of a Windows NT Bluetooth driver. Our techniques can be applied to find the bugs of this driver reported in [13,10].

Related work: Recently, several models based on rewriting systems have been considered to model multithreaded programs [14–19]. While these models allow to model dynamic thread creation, they do not allow communication between processes.

In [8], we have introduced a model based on networks of pushdown systems called CDPN. While this model allows dynamic creation of processes, it allows only a restricted form of synchronisation where a process has the right to read only the control states of its immediate sons (i.e., the processes it has created).

[20] considers bounded phase reachability in multi-stack systems, where in each phase the system can pop from one stack, and push on some number of stacks. In our model, we allow the manipulation of different stacks in a single phase. However, since the communication relation in the different phases of a MAPN is fixed, our model cannot simulate phase switches in the sense of [20].

Networks of pushdown systems communicating via message passing [6,10], locks [21,22], or channels [23–25] have been considered. Pushdown Networks with these kinds of communications can also be described in our PDN model.

2. Networks of communicating pushdown systems

A PushDown Network (PDN for short) is given by a tuple $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ where $R \subseteq \{(i, j) \mid 1 \le i, j \le n, i \ne j\}$ is a binary relation defining the communication structure of the network (R defines a directed graph whose nodes are $1, \ldots, n$), and for every $i \in \{1, \ldots, n\}$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ is a communicating pushdown system such that P_i is a finite set of control states, Γ_i is a finite stack alphabet, and Δ_i is a set of transition rules of the form:

$$\phi : (p, \gamma) \hookrightarrow (p', w)$$

where $p, p' \in P_i$ are two control states, $\gamma \in \Gamma_i$ is the symbol popped from the stack, $w \in \Gamma_i^*$ is the string pushed in the stack, and $\phi \subseteq \bigcup_{(i,j)\in R} P_j$ is a set of constraints over the current control states of the other observed processes.

A *local configuration* of a process in the network, say $\mathcal{P}_i$, is a word $p_i w_i \in P_i \Gamma_i^*$ where p_i is a state and w_i is a stack content. A *configuration* of the network N is a vector $(p_1 w_1, \ldots, p_n w_n) \in \prod_{i=1}^n P_i \Gamma_i^*$, where $p_i w_i$ is the local configuration of $\mathcal{P}_i$.

We define a *transition relation* $\Longrightarrow_N$ between configurations. We have $(p_1 w_1, \ldots, p_n w_n) \Longrightarrow_N (p'_1 w'_1, \ldots, p'_n w'_n)$ if and only if there is an index $i \in \{1, \ldots, n\}$ such that:

- there is a rule $\phi : (p, \gamma) \hookrightarrow (p', w) \in \Delta_i$ and there exists a word $u \in \Gamma_i^*$ such that $p_i = p$, $p'_i = p'$, $w_i = \gamma u$, $w'_i = wu$, and for every $j \in \{1, \ldots, n\}$, if $(i, j) \in R$, then $p_j \in \phi$.
- $\forall j \in \{1, \ldots, n\}$ such that $i \ne j$, $p_j = p'_j$ and $w_j = w'_j$.

Let $\Longrightarrow_N^*$ denote the reflexive transitive closure of $\Longrightarrow_N$. Given a configuration c, the set of immediate successors of c is $post_N(c) = \{c' \in \prod_{i=1}^n P_i \Gamma_i^* : c \Longrightarrow_N c'\}$. This notation can be generalized straightforwardly to sets of configurations. Let $post_N^*$ denote the reflexive-transitive closure of $post_N$.

Intuitively, a network $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ can be seen as a collection of "standard" pushdown systems that observe each other according to the structure R: $(i, j) \in R$ means that process $\mathcal{P}_i$ observes (reads) the states of process $\mathcal{P}_j$. If a rule $\phi : (p, \gamma) \hookrightarrow (p', w)$ is in Δ_i, then process $\mathcal{P}_i$ can apply the "standard" pushdown rule $(p, \gamma) \hookrightarrow (p', w)$ iff every process $\mathcal{P}_j$ for j such that $(i, j) \in R$ is in a state $p_j \in \phi \cap P_j$. The network is in the configuration $(p_1w_1, \ldots, p_nw_n)$ means that each pushdown system $\mathcal{P}_i$ is in configuration p_iw_i, i.e., is in control state p_i and has w_i in its stack.

A network $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ is *acyclic* (resp. *cyclic*) if the graph of its relation R is acyclic (resp. cyclic). A network consisting of a single process $N = (\mathcal{P}, \emptyset)$ will simply be denoted by $\mathcal{P}$ and corresponds to the standard pushdown system $\mathcal{P}$.

3. Symbolic representation of PDN configurations

Let $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ be a PDN where $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$. Since a configuration of N can be seen as a word of dimension n in $P_1\Gamma_1^* \times \cdots \times P_n\Gamma_n^*$, a natural way to represent *infinite* sets of PDN configurations is to consider *recognizable* languages. Let $\Sigma_1, \ldots, \Sigma_n$ be n finite alphabets. A n-dim word over $\Sigma_1, \ldots, \Sigma_n$ is an element of $\Sigma_1^* \times \cdots \times \Sigma_n^*$. A n-dim language is a (possibly infinite) set of n-dim words. A n-dim language L is *recognizable* if it is a finite union of products of n regular languages (i.e. $L = \bigcup_{j=1}^{m} L(A_1^j) \times \cdots \times L(A_n^j)$ for some $m \in \mathbb{N}$, where A_i^j is a finite state automaton over Σ_i). Notice that for $n = 1$, recognizable languages correspond precisely to regular languages.

It is well known that for any dimension $n \geq 1$, the class of recognizable languages is closed under boolean operations and that the emptiness problem of recognizable languages is decidable.

4. Reachability analysis of PDNs

The reachability problem between sets of configurations C_1 and C_2 for a PDN N is to determine whether there are two configurations $c_1 \in C_1$ and $c_2 \in C_2$ such that $c_1 \Longrightarrow_N^* c_2$. It is easy to see that a PDN with two processes and a cyclic communication structure is Turing powerful:

Proposition 4.1 ([11]). *The reachability problem of PDNs is undecidable.*

Hence, we restrict ourselves, in a first step, to acyclic PDNs, we show later how this provides a semi-algorithm for the analysis of general PDNs (even cyclic ones). We showed in [11] that acyclic PDNs do not preserve recognizability. In this section, we go further and define conditions under which acyclic PDNs preserve recognizability.

Definition 4.1. Let $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ be an acyclic PDN where for every $i \in \{1, \ldots, n\}$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$. For $i \in \{1, \ldots, n\}$, let ρ_i be a binary relation in $P_i \times P_i$ defined by: $(p, p') \in \rho_i$ iff there exists in Δ_i a rule of the form $\phi : (p, \gamma) \hookrightarrow (p', w)$. Let ρ_i^* be the reflexive-transitive closure of ρ_i.

N is *stable* iff for every $(i, j) \in R$ and every $p, p' \in P_j$, if $(p, p') \in \rho_j^*$ and $(p', p) \in \rho_j^*$, then for every rule $\phi : (q, \gamma) \hookrightarrow (q', w)$ in Δ_i, $p \in \phi$ iff $p' \in \phi$.

Intuitively, N is *stable* means that if $\mathcal{P}_j$ can go from a state p to a state p' and then back to p, for some index $j \in \{1, \ldots, n\}$; then if $(i, j) \in R$ (i.e., if $\mathcal{P}_i$ observes $\mathcal{P}_j$), the rules of Δ_i do not distinguish between the states p and p'.

We show the first main result of our paper: stable acyclic networks effectively preserve recognizability, meaning if N is a stable acyclic PDN and C is a recognizable set of configurations, then $post_N^*(C)$ is an effectively recognizable set:

Theorem 4.1. *Let $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ be a stable acyclic PDN and C be a recognizable set of configurations. Then, $post_N^*(C)$ is an effectively recognizable set.*

Proof. Let us first recall that standard pushdown systems effectively preserve regularity [4,2]. The construction underlying Theorem 4.1 is based on the iterative applications of the standard $post^*$ algorithm for standard pushdown systems [4,2] for each pushdown component in the network. The stability of the network guarantees the termination of the iterative procedure.

We give in what follows a construction of recognizable reachability sets for stable networks. For the sake of simplicity, we consider a *stable* network containing two processes $N = (\mathcal{P}_1, \mathcal{P}_2, R)$, where $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ for $i = 1, 2$. The construction can be extended easily to the general case of an arbitrary number of processes. There are two cases for R since it is acyclic: either $R = \emptyset$ or $R = (2, 1)$ (the case where $R = (1, 2)$ being symmetrical). The first case is trivial, it corresponds to the case where the processes are independent of each other. Let us then consider the case where $R = (2, 1)$ (i.e., process 2 observes process 1). Let $C \subseteq P_1\Gamma_1^* \times P_2\Gamma_2^*$ be a recognizable set of configurations of N, and let $A \subseteq P_1\Gamma_1^*$ and $B \subseteq P_2\Gamma_2^*$ be two recognizable sets such that $C = (A, B)$ (in this proof, we use (A, B) to denote $A \times B$). Our goal is to show that the set of configurations $post_N^*(C)$ is recognizable.

Let $P_1 = \{p_1, \ldots, p_n\}$ and S be the set of sequences $\sigma = p_{i_1}, p_{i_2}, \ldots, p_{i_k}$ such that for every $j < k$, $p_{i_j} \in P_1$, $(p_{i_j}, p_{i_{j+1}}) \in \rho_1^*$, and the p_{i_j}'s are distinct. For every $p \in P_1$, let Δ_2^p be the set of rules $(q, \gamma) \hookrightarrow (q', w)$ such that there exists a rule $\phi : (q, \gamma) \hookrightarrow (q', w)$ in Δ_2 with $p \in \phi$, i.e., Δ_2^p is the set of rules of $\mathcal{P}_2$ (without constraints) that can be applied if the first component $\mathcal{P}_1$ is in state p.

First, we need to introduce the following notations: if $C_1 \subseteq P_1\Gamma_1^*$ is a recognizable set of configurations, we denote by $C_1(p)$ the recognizable set of configurations in C_1 having p as a control state, i.e. $C_1(p) = C_1 \cap p\Gamma_1^*$. Moreover, if $C_2 \subseteq P_2\Gamma_2^*$ is a recognizable set, we denote by $\Delta_2^p(C_2)$ the recognizable set $post^*_{\mathcal{P}_2(p)}(C_2)$, where $\mathcal{P}_2(p) = (P_2, \Gamma_2, \Delta_2^p)$ is the "standard" pushdown system having Δ_2^p as set of rules, i.e., $\Delta_2^p(C_2)$ contains the set of configurations that can be obtained by applying to the configurations in C_2 the rules of Δ_2^p. This recognizable set can be computed using the standard saturation procedure of pushdown systems given in [4,2].

The idea behind our construction is the following: We first start by computing $post^*_{\mathcal{P}_1}(A)$ of the first component $\mathcal{P}_1$ since it is independent of the component $\mathcal{P}_2$, we obtain a recognizable set A' for $\mathcal{P}_1$, and the pair (A', B) for the network N. Notice that (A, B) is a subset of (A', B) since $A \subseteq A'$. This set (A', B) contains all the configurations (p_1w_1, p_2w_2) such that p_1w_1 is a successor by $\mathcal{P}_1$ of a configuration in A, and p_2w_2 is a configuration in B. Then, we need to consider the successors of the second component as well. Since the application of the rules of Δ_2 depends on the current state of $\mathcal{P}_1$, we proceed as follows: For every state $p \in P_1$, we consider the configurations of A' that are in state p (i.e., $A'(p)$). These configurations can be coupled with the configurations of $\mathcal{P}_2$ obtained by applying Δ_2^p. Therefore, we obtain all the pairs $\big(A'(p), \Delta_2^p(B)\big)$ for all $p \in P_1$. Next, we can apply the rules of $\Delta_2^{p'}$ for a state $p' \neq p$ to $\Delta_2^p(B)$ iff $\mathcal{P}_1$ can move some configurations in $A'(p)$ to configurations with control state p'. We obtain then the set $\left(post^*_{\mathcal{P}_1}\big(A'(p)\big)(p'), \Delta_2^{p'}\big(\Delta_2^p(B)\big)\right)$, where p and p' are such that $\mathcal{P}_1$ can move from state p to state p'. Now, we need to apply another set of rules $\Delta_2^{p''}$ to these configurations $\Delta_2^{p'}\big(\Delta_2^p(B)\big)$ (of course if there are configurations in $post^*_{\mathcal{P}_1}\big(A'(p)\big)(p')$ that can move to p''), etc. This technique is guaranteed to terminate. Indeed, in the sequence above for example, the sets of rules Δ_2^p and $\Delta_2^{p'}$ need not to be executed again. Indeed, suppose Δ_2^p could be applied after $\Delta_2^{p''}$ for example, this means that process $\mathcal{P}_1$ can move from state p to p' and then to p'' and then back to p. Since the network is *stable*, this means that $\Delta_2^p = \Delta_2^{p'} = \Delta_2^{p''}$, and therefore, there is no need to apply Δ_2^p again since it will not add any new configuration.

More precisely, let $A' = post^*_{\mathcal{P}_1}(A)$ be the successors of the configurations in A for process $\mathcal{P}_1$. For every sequence $\sigma = p_{i_1}, p_{i_2}, \ldots, p_{i_k}$ in S, let $A^\sigma_{i_1}$ be the recognizable set $A'(p_{i_1})$, $A^\sigma_{i_1,i_2}$ be the recognizable set $post^*_{\mathcal{P}_1}\big(A^\sigma_{i_1}\big)(p_{i_2})$, and for every $j \leq k$, $A^\sigma_{i_1,\ldots,i_j} = post^*_{\mathcal{P}_1}\big(A^\sigma_{i_1,\ldots,i_{j-1}}\big)(p_{i_j})$. The sets A' and the $A^\sigma_{i_1,\ldots,i_j}$'s can be computed using the standard saturation procedure given in [4].

Then, it follows from the discussion above that $post^*_N(C)$ is recognized by the union of the following recognizable sets: (A, B); (A', B); $\big(A'(p), \Delta_2^p(B)\big)$ for all $p \in P_1$; and for every sequence $s = p_{i_1}, p_{i_2}, \ldots, p_{i_k}$ in S, and every $j \leq k$,

$$\left(A^\sigma_{i_1,\ldots,i_j}, \Delta_2^{p_{i_j}}\left(\Delta_2^{p_{i_{j-1}}}\left(\cdots\left(\Delta_2^{p_{i_1}}(B)\right)\right)\right)\right).$$

The above proof can be extended to the case where we have n processes. □

5. Multiphase Acyclic Pushdown Networks

In this work, we go further and extend the model of acyclic PDNs by allowing dynamic changes in the definition of the network. This section is devoted to the definition of this new model.

A *Multiphase* Acyclic Pushdown Network (MAPN) is given by a tuple $M = (N_1, \ldots, N_m, T)$ where for every $j \in \{1, \ldots, m\}$, $N_j = (\mathcal{P}_1^j, \ldots, \mathcal{P}_n^j, R_j)$ is an acyclic PDN where for $i \in \{1, \ldots, n\}$, $\mathcal{P}_i^j = (P_i, \Gamma_i, \Delta_i^j)$. T is a set of transitions of the form (N_i, Φ, N_j) where $i, j \in \{1, \ldots, m\}$ and $\Phi \subseteq \prod_{k \leq n} P_k\Gamma_k^*$ is a recognizable set of configurations.

We can think of the network N_j as an acyclic network over the processes $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$, where each process $\mathcal{P}_i$ ($i \in \{1, \ldots, n\}$) executes only the rules Δ_i^j, and where these processes observe each other according to the structure R_j. T is a *phase graph*: a transition $(N_i, \Phi, N_j) \in T$ means that if the acyclic PDN N_i is in a configuration $(p_1w_1, \ldots, p_nw_n) \in \Phi$, then the network can move from a phase where the processes behave according to the network N_i to a phase where they behave according to N_j, i.e., from N_i to N_j.

Let $\mathcal{G}$ be the underlying graph of T, i.e., $(i, j) \in \mathcal{G}$ iff there exists in T a transition of the form (N_i, Φ, N_j). We say that T is cyclic (resp. acyclic) iff $\mathcal{G}$ is cyclic (resp. acyclic). The network M is said to be cyclic (resp. acyclic) iff T is cyclic (resp. acyclic).

An *indexed configuration* of the MAPN is a pair $\langle(p_1w_1, \ldots, p_nw_n), i\rangle$ where $(p_1w_1, \ldots, p_nw_n) \in \prod_{k=1}^n P_k\Gamma_k^*$, and $i \in \{1, \ldots, m\}$. The index i records the current phase of the network. A *configuration* of the MAPN is a tuple $(p_1w_1, \ldots, p_nw_n) \in \prod_{k=1}^n P_k\Gamma_k^*$.

We define a *transition relation* $\Rightarrow_M$ between indexed configurations as follows: $\langle(p_1w_1, \ldots, p_nw_n), i\rangle \Rightarrow_M \langle(p_1'w_1', \ldots, p_n'w_n'), j\rangle$ if and only if:

- $(p_1w_1, \ldots, p_nw_n) = (p_1'w_1', \ldots, p_n'w_n')$, and there is $(N_i, \Phi, N_j) \in T$ such that $(p_1w_1, \ldots, p_nw_n) \in \Phi$,
- $(p_1w_1, \ldots, p_nw_n) \Longrightarrow_{N_j} (p_1'w_1', \ldots, p_n'w_n')$ and $i = j$.

We extend $\Rightarrow_M$ to configurations in $\prod_{k=1}^{n} P_k \Gamma_k^*$ as follows: $(p_1 w_1, \ldots, p_n w_n) \Rightarrow_M (p'_1 w'_1, \ldots, p'_n w'_n)$ iff there exist two phase indices i and j in $\{1, \ldots, m\}$ such that $\langle (p_1 w_1, \ldots, p_n w_n), i\rangle \Rightarrow_M \langle (p'_1 w'_1, \ldots, p'_n w'_n), j\rangle$. Let $\Rightarrow^*_M$ denote the reflexive transitive closure of $\Rightarrow_M$. Let C be a set of (indexed) configurations. We define $post_M(C)$ and $post^*_M(C)$ in the usual manner. Let C be a set of indexed configurations. C is said to be recognizable if and only if the set $C_j = \{(p_1 w_1, \ldots, p_n w_n) | \langle (p_1 w_1, \ldots, p_n w_n), j\rangle \in C\}$ is recognizable for every j, $1 \le j \le m$. As usual, the reachability problem between two sets of (indexed) configurations C_1 and C_2, for a MAPN M, is to determine whether there are two (indexed) configurations $c_1 \in C_1$ and $c_2 \in C_2$ such that $c_1 \Rightarrow^*_M c_2$.

6. The reachability problem for MAPNs

In this section, we study the reachability problem for the model MAPN. First, we show that reachability for PDN is polynomially reducible to reachability in MAPN. Thus, reachability is undecidable in general for MAPNs. Then, we define two MAPN subclasses for which reachability becomes decidable.

Theorem 6.1. *The reachability problem for PDNs is polynomially reducible to its corresponding problem for MAPNs.*

Proof. Let $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ be a PDN where for every $i \in \{1, \ldots, n\}$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ is a communicating pushdown system. It is easy to see that if N is acyclic, then it can be seen as a MAPN with one phase. If N is cyclic, we construct a MAPN $M = (N_1, \ldots, N_n, T)$ such that the reachability problem for N is polynomially reducible to its corresponding problem for M. The idea consists in decomposing N into n acyclic subnetworks $N_1, \ldots, N_n$ such that the behavior of each subnetwork N_j $(1 \le j \le n)$ is also a behavior of N, and such that any behavior of N can be obtained by performing a certain number of switches between the different N_j's. For this, we need to ensure that (1) $R = \bigcup_{1\le j\le n} R_j$ (where R_j is the graph of the subnetwork N_j); and that (2) for every j, $1 \le j \le n$, the subnetwork N_j allows process $\mathcal{P}_j$ to observe all the processes that it can observe in N. This ensures that all the rules of $\mathcal{P}_j$ that can be applied in N can also be applied in M.

Formally, these subnetworks can be computed as follows: $N_j = (\mathcal{P}^j_1, \ldots, \mathcal{P}^j_n, R_j)$ is an acyclic PDN such that R_j is the maximal acyclic relation containing the subset $R \cap (\{j\} \times \{1, \ldots, n\})$ (this ensures that in N_j, $\mathcal{P}_j$ observes all the processes it can observe in N). Moreover, for every $j \in \{1, \ldots, n\}$ and for every $i \in \{1, \ldots, n\}$, $\mathcal{P}^j_i = (P_i, \Gamma_i, \Delta^j_i)$ is a communicating pushdown system such that the set of transition rules Δ^j_i is defined as follows: $\Delta^j_i = \Delta_i$ if $(\{i\} \times \{1, \ldots, n\}) \cap R = (\{i\} \times \{1, \ldots, n\}) \cap R_j$ (i.e., if $\mathcal{P}^j_i$ can observe in N_j all the processes that $\mathcal{P}_i$ observes in N); and $\Delta^j_i = \emptyset$ otherwise. The set of rules of the processes $\mathcal{P}_i$ that do not observe in N_j all the processes that they observe in N is made empty in order to not activate in N_j rules of $\mathcal{P}_i$ that cannot be activated in N. Moreover, M has to allow a switch from any network N_i to any network N_j $(1 \le i, j \le n)$, i.e. $T = \{(N_i, \prod_{\ell=1}^{n} P_l \Gamma_l^*, N_j) \mid i, j \in \{1, \ldots, n\}\}$.

Then, it is clear that $(p_1 w_1, \ldots, p_n w_n) \Longrightarrow^*_N (p'_1 w'_1, \ldots, p'_n w'_n)$ iff $(p_1 w_1, \ldots, p_n w_n) \Longrightarrow^*_M (p'_1 w'_1, \ldots, p'_n w'_n)$. □

As an immediate consequence of Theorem 6.1 and Proposition 4.1 we have:

Proposition 6.1. *The reachability problem is undecidable for MAPNs.*

Unfortunately, we can show that this undecidability holds even for acyclic MAPNs. We show that solving this problem would imply a decision procedure for Post's Correspondence Problem (PCP).

Theorem 6.2. *The reachability problem between two (indexed) configurations is undecidable for acyclic MAPNs. This holds even if the phase graph has a single transition.*

Proof. We show that solving this problem would imply a decision procedure for Post's Correspondence Problem (PCP). Let $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ be two sequences of words over an alphabet Σ, and let $a_1, \ldots, a_n, b_1, \ldots, b_n$ be letters not in Σ.

We construct an MAPN $M = (N_1, N_2, T)$ such that:

- for every $j \in \{1, 2\}$, $N_j = (\mathcal{P}^j_1, \mathcal{P}^j_2, \mathcal{P}^j_3, R_j)$ is an acyclic PDN, where:
  - for $i \in \{1, 2, 3\}$, $\mathcal{P}^j_i = (P_i, \Gamma_i, \Delta^j_i)$ is a communicating pushdown system;
  - $R_1 = \{(1, 2), (2, 3)\}$ and $R_2 = \{(2, 3), (3, 1)\}$ are two acyclic relations.
- $T = \{(N_1, \Phi, N_2)\}$ where $\Phi = P_1 \Gamma^* \times P_2 \Gamma_2^*$.

Then, deciding whether

$$c' \in post^+_{N_2}(post^+_{N_1}(c))$$

for two configurations c and c' would imply a decision procedure for PCP. The idea is as follows:

1. During the first phase, i.e., in N_1 where $\mathcal{P}_1$ observes $\mathcal{P}_2$ who observes $\mathcal{P}_3$; $\mathcal{P}_3$ pushes the words u_i in its stack. During this time, $\mathcal{P}_2$ can put b_i in its stack if the last word put by $\mathcal{P}_3$ is u_i, whereas $\mathcal{P}_1$ can put a_i in its stack if the last letter put by $\mathcal{P}_2$ is b_i. This ensures that if $\mathcal{P}_1$ pushes $a_{i_1} a_{i_2} \cdots a_{i_k}$ in its stack, then necessarily:
   - $\mathcal{P}_2$ has in its stack $b_{l_1} b_{l_2} \cdots b_{l_n}$, and
   - $\mathcal{P}_3$ has in its stack $u_{j_1} u_{j_2} \cdots u_{j_m}$,

such that $i_1i_2\cdots i_k$ is a subsequence of $l_1l_2\cdots l_n$ which is a subsequence of $j_1j_2\cdots j_m$. This is due to the fact that since $\mathcal{P}_1$ observes $\mathcal{P}_2$, it can be slower than $\mathcal{P}_2$ in pushing the a_i's; and similarly, since $\mathcal{P}_2$ observes $\mathcal{P}_3$, it can be slower than $\mathcal{P}_3$ in pushing the b_i's.

2. During the second phase, i.e., in N_2 where $\mathcal{P}_2$ observes $\mathcal{P}_3$, who observes $\mathcal{P}_1$; $\mathcal{P}_1$ can pop the a_l's. $\mathcal{P}_3$ pops the word v_j from its stack if the last letter popped by $\mathcal{P}_1$ is a_j and process $\mathcal{P}_2$ pops the letter b_i if the last word popped by $\mathcal{P}_3$ is v_i. This ensures that if $\mathcal{P}_1$ has popped $a_{h_1}a_{h_2}\cdots a_{h_s}$ from its stack, then $\mathcal{P}_3$ has popped $v_{g_1}v_{g_2}\cdots v_{g_r}$ and $\mathcal{P}_2$ has popped $b_{f_1}b_{f_2}\cdots b_{f_z}$ such that $f_1f_2\cdots f_z$ is a subsequence of $g_1g_2\cdots g_r$ which is a subsequence of $h_1h_2\cdots h_s$.

The two items above infer that from a configuration where the three processes have empty stacks, we can reach a configuration where the three processes have empty stacks by first executing N_1 and then N_2 iff the sequences of indices $h_1h_2\cdots h_s, g_1g_2\cdots g_r, f_1f_2\cdots f_z, i_1i_2\cdots i_k, l_1l_2\cdots l_n$, and $j_1j_2\cdots j_m$ are the same, and $u_{i_1}u_{i_2}\cdots u_{i_k} = v_{i_1}v_{i_2}\cdots v_{i_k}$. □

6.1. Reachability for finitely-constrained MAPNs

We can show that reachability becomes decidable for MAPNs when the constraints in the phase graph are finite sets of configurations.

Definition 6.1. A MAPN $M = (N_1, \ldots, N_m, T)$ is called *finitely-constrained* if T is a set of transitions of the form (N_i, Φ, N_j) where $i, j \in \{1, \ldots, m\}$ and $\Phi \subseteq \prod_{k\le n} P_k\Gamma_k^*$ is a *finite* set of configurations.

In [11], we showed that the reachability problem between two recognizable sets of configurations for acyclic PDNs is decidable. Thanks to this result, we show that in finitely-constrained MAPNs, reachability can be reduced to reachability in a finite graph:

Proposition 6.2. *The reachability problem between recognizable sets of (indexed) configurations is decidable for finitely-constrained MAPNs.*

Proof. Let $M = (N_1, \ldots, N_m, T)$ be a finitely-constrained MAPN. Let C and C' be two recognizable sets of indexed configurations of M (the case where C and C' are sets of configurations is similar). For every $j \in \{1, \ldots, m\}$, let $C_j = \{(p_1w_1, \ldots, p_nw_n) \mid \langle (p_1w_1, \ldots, p_nw_n), j\rangle \in C\}$ and $C'_j = \{(p_1w_1, \ldots, p_nw_n) \mid \langle (p_1w_1, \ldots, p_nw_n), j\rangle \in C'\}$. We show that the reachability problem of M is reducible to the reachability problem for a finite directed graph $\mathcal{T}$. We sketch hereafter the construction of the directed graph $\mathcal{T}$. For each transition $(N_i, \Phi, N_j) \in T$ and for each configuration $c \in \Phi$, the graph $\mathcal{T}$ has a node $n_{(i,c,j)}$. For each set C_j (resp. C'_j), with $j \in \{1, \ldots, m\}$, the graph $\mathcal{T}$ has a node n_{C_j} (resp. $n_{C'_j}$). The set of direct edges of $\mathcal{T}$ is defined as the smallest set satisfying the following conditions:

- For every pair of nodes $n_{(i,c,j)}$ and $n_{(j,c',k)}$, $i, j, k \in \{1, \ldots, m\}$, there is an edge from the node $n_{(i,c,j)}$ to the node $n_{(j,c',k)}$ iff $c \Longrightarrow^*_{N_j} c'$ (which is decidable thanks to [11]).
- For every pair of nodes n_{C_i} and $n_{(i,c,j)}$, $i, j \in \{1, \ldots, m\}$, there is an edge from the node n_{C_i} to the node $n_{(i,c,j)}$ iff the configuration c is reachable from the set C_i by the acyclic PDN N_i (which is also decidable thanks to [11]).
- For every pair of nodes $n_{(i,c,j)}$ and $n_{C'_j}$, $i, j \in \{1, \ldots, m\}$, there is an edge from the node $n_{(i,c,j)}$ to the node $n_{C'_j}$ iff the set of configurations C'_j is reachable from the configuration c by the acyclic PDN N_j.
- For every pair of nodes n_{C_i} and $n_{C'_i}$, $i \in \{1, \ldots, m\}$, there is an edge from the node n_{C_i} to the node $n_{C'_i}$ iff the set of configurations C'_i is reachable from the set C_i by the acyclic PDN N_i.

Then, it is clear that there is a path in the directed graph $\mathcal{T}$ from a node n_{C_i}, for some $i \in \{1, \ldots, m\}$, to a node $n_{C'_j}$, for some $j \in \{1, \ldots, m\}$, iff the set of configurations C' is reachable by M from the set of configurations C. □

6.2. Reachability for stable acyclic MAPNs

In this section, we define the class of *stable* acyclic MAPNs and show that it effectively preserves recognizability. Hence, its reachability problem is decidable.

Definition 6.2. An MAPN $M = (N_1, \ldots, N_m, T)$ is *stable* if for every $j \in \{1, \ldots, m\}$, $N_j = (\mathcal{P}^j_1, \ldots, \mathcal{P}^j_n, R_j)$ is a stable acyclic PDN.

We show that *stable* acyclic MAPNs effectively preserve recognizability. This is due to the fact that (1) stable acyclic PDNs effectively preserve recognizability, and (2) the phase graphs for acyclic MAPNs are acyclic. This allows to obtain the reachability set for *stable* acyclic MAPNs by successively applying the algorithm underlying Theorem 4.1 a finite number of times.

Theorem 6.3. *Let $M = (N_1, \ldots, N_m, T)$ be a* stable *acyclic MAPN and let C be a recognizable set of (indexed) configurations of M. Then $post^*_M(C)$ is effectively recognizable.*

Proof. We give the proof for recognizable *indexed* configurations. The same proof can also be applied for recognizable configurations. Let C be a recognizable set of indexed configurations of M. For every $j \in \{1, \ldots, m\}$, let $C_j = \{(p_1 w_1, \ldots, p_n w_n) \mid \langle (p_1 w_1, \ldots, p_n w_n), j\rangle \in C\}$. Then, $post^*_M(C)$ can be computed as follows:

- we take all the sequences of indices $i_1, \ldots, i_{k+1} \in \{1, \ldots, m\}$ such that for every $\ell \in \{1, \ldots, k\}$, T contains a transition of the form: $(N_{i_\ell}, \Phi_\ell, N_{i_{\ell+1}})$, and
- we compute $post^*_{N_{i_{k+1}}}\left(\widetilde{post}^*_{N_{i_k}}\left(\cdots\left(\widetilde{post}^*_{N_{i_1}}(C_{i_1})\right)\right)\right)$, where $\widetilde{post}^*_{N_{i_\ell}}(L) = post^*_{N_{i_\ell}}(L) \cap \Phi_\ell$ for every set $L \subseteq \prod_{i=1}^n P_i \Gamma_i^*$.

Since T is acyclic, the i_j's are all different, i.e., for $j \neq l$, $i_j \neq i_l$. Therefore, there exists a finite number of possible such sequences $i_1, \ldots, i_{k+1}$; and it suffices to take the union over all these computed sets. These sets can be computed because (1) for every $s \in \{1, \ldots, m\}$, N_s preserves effectively recognizability since it is a stable acyclic PDN, (2) Φ_s is a recognizable set, and (3) recognizable sets are effectively closed under intersection. □

Since recognizable sets are effectively closed under intersection, we get:

Corollary 6.1. *The reachability problem between recognizable sets of (indexed) configurations is decidable for stable acyclic MAPNs.*

7. Bounded phase switch reachability for MAPNs

We consider in this section the reachability problem for *general* MAPNs. Since this problem is undecidable, we consider *bounded switch* reachability, where the number of switches between the different phases (the different networks N_i) is bounded.

Definition 7.1. Let $M = (N_1, \ldots, N_m, T)$ be a MAPN where for every $j \in \{1, \ldots, m\}$, $N_j = (\mathcal{P}^j_1, \ldots, \mathcal{P}^j_n, R_j)$ is an acyclic PDN. We define the k-switch transition relation between indexed configurations inductively as follows:

- $\langle (p_1 w_1, \ldots, p_n w_n), i\rangle \stackrel{0}{\Longrightarrow}_M \langle (p'_1 w'_1, \ldots, p'_n w'_n), j\rangle$ if and only if $i = j$ and $(p_1 w_1, \ldots, p_n w_n) \Longrightarrow^*_{N_i} (p'_1 w'_1, \ldots, p'_n w'_n)$.
- $\langle (p_1 w_1, \ldots, p_n w_n), i\rangle \stackrel{k+1}{\Longrightarrow}_M \langle (p'_1 w'_1, \ldots, p'_n w'_n), j\rangle$ if and only if there is an indexed configuration $\langle (p''_1 w''_1, \ldots, p''_n w''_n), l\rangle$ such that: $\langle (p_1 w_1, \ldots, p_n w_n), i\rangle \stackrel{k}{\Longrightarrow}_M \langle (p''_1 w''_1, \ldots, p''_n w''_n), l\rangle$; $\langle (p''_1 w''_1, \ldots, p''_n w''_n), l\rangle \Rightarrow_M \langle (p''_1 w''_1, \ldots, p''_n w''_n), j\rangle$; and $(p''_1 w''_1, \ldots, p''_n w''_n) \Longrightarrow^*_{N_j} (p'_1 w'_1, \ldots, p'_n w'_n)$.

$\stackrel{k}{\Longrightarrow}_M$ is extended to configurations as follows: $(p_1 w_1, \ldots, p_n w_n) \stackrel{k}{\Longrightarrow}_M (p'_1 w'_1, \ldots, p'_n w'_n)$ iff there exist two phase indices i and j such that $\langle (p_1 w_1, \ldots, p_n w_n), i\rangle \stackrel{k}{\Longrightarrow}_M \langle (p'_1 w'_1, \ldots, p'_n w'_n), j\rangle$.

The k-bounded switch reachability problem for MAPNs between two sets of (indexed) configurations C and C' consists in determining whether there are $c \in C$ and $c' \in C'$ such that $c \stackrel{k}{\Longrightarrow}_M c'$. Intuitively, this means that $c \stackrel{k}{\Longrightarrow}_M c'$ iff the (indexed) configuration c' can be reached from c after switching at most k times the phase of the network according to the phase graph T. In this case, we say that c' is k-bounded reachable from c.

Unfortunately, even k-bounded switch reachability is undecidable for cyclic as well as acyclic MAPNs. Indeed, it is easy to see that performing k-bounded reachability in M amounts to performing "unrestricted" reachability in the acyclic network defined by $(N_1, \ldots, N_m, T_k)$, where T_k is obtained by considering all the possible paths of T having at most k transitions. Therefore, it follows from Theorem 6.2 that:

Corollary 7.1. *The k-bounded reachability problem between recognizable sets of (indexed) configurations is undecidable for MAPNs. This holds even for $k = 1$.*

However, it follows from Corollary 6.1 and the observation above that:

Corollary 7.2. *The k-bounded switch reachability problem between recognizable sets of (indexed) configurations is decidable for* stable *MAPNs.*

The result above can be used to construct a semi-decision procedure for the k-bounded switch reachability problem for *general* MAPNs. Let $M = (N_1, \ldots, N_m, T)$ be a MAPN, the idea consists in taking advantage of the fact that k-bounded switch reachability is decidable for *stable* networks. To do so, we compute a stable network $M' = (N'_1, \ldots, N'_{m'}, T')$ s.t. the processes in M' have the same behaviors as those in M but can perform more phase switches. This ensures that given two configurations c and c', $c \stackrel{k}{\Longrightarrow}_{M'} c'$ infers that there exists k' such that $c \stackrel{k'}{\Longrightarrow}_M c'$. This gives the semi-decision procedure since we can decide k-bounded reachability for M' thanks to its stability.

Theorem 7.1. *Let M be a MAPN. Then, we can compute a stable MAPN M' such that for every recognizable sets C and C' of (indexed) configurations, if C' is k-bounded reachable from C by M, there exists $k' \geq k$ such that C' is k'-bounded reachable from C by M'.*

Proof. Let $M = (N_1, \ldots, N_m, T)$ be a MAPN. To compute the stable network M', the idea consists in decomposing every acyclic PDN N_j ($j \le m$) into stable subnetworks $N_j^1, \ldots, N_j^{i_j}$ such that the behavior of each subnetwork N_j^l is also a behavior of N_j, and such that any behavior of N_j can be obtained by performing a certain number of switches between the different N_j^l's.

These subnetworks can be computed as follows: $N_l^j = ((\mathcal{P}_1^j)_l, \ldots, (\mathcal{P}_n^j)_l, R_j)$ where for $1 \le i \le n$, $(\mathcal{P}_i^j)_l = (P_i^j, \Gamma_i, (\Delta_i^j)_l)$ s.t. $(\Delta_i^j)_l \subseteq \Delta_i^j$, and the obtained network N_l^j is stable. In other words, the (N_j^l)'s are obtained by restricting the set of the pushdown rules of the network N_j in order to get a network that satisfies the stability condition. Moreover, we make sure that whenever a rule $\phi : (p, \gamma) \hookrightarrow (p', w)$ can be fired in N_j, then there exists an index l such that the same rule can be fired in N_j^l (this condition is easy to satisfy by imposing that $\Delta_i^j = \bigcup_{l=1}^{i_j} (\Delta_i^j)_l$). M' is then the network $M' = (N_1^1, \ldots, N_1^{i_1}, \ldots, N_m^1, \ldots, N_m^{i_m}, T')$, where T' is defined as follows.

- For every $j \in \{1, \ldots, m\}$, T' has to allow a switch from any network N_j^h to any network N_j^l ($h, l \in \{1, \ldots, i_j\}$). This ensures that the behaviors that can occur in N_j without any switch in M can also be performed by performing a certain number of switches between the different N_j^l's in M'.
- Moreover, T' needs to keep the switches allowed by T, i.e., if T allows to move from a given N_j to another N_l (i.e., (N_j, Φ, N_l) is in T), then T' should also allow to move from any N_j subnetwork N_j^h to any N_l subnetwork $N_l^{h'}$, for $h \le i_j$ and $h' \le i_l$ (of course, while respecting the constraints Φ).

Note that the new phase graph T' is cyclic even if T is acyclic. □

8. A semi-algorithm for the reachability problem for general PDNs

We show in this section how we can use the previous results on bounded phase switch reachability for MAPNs to derive a semi-algorithm to check reachability for general PDNs (even cyclic ones). Let $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ be a PDN, where for i, $1 \le i \le n$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ is a communicating pushdown system. The construction underlying Theorem 6.1 produces a MAPN M such that reachability in N can be reduced to reachability in M. Let C and C' be two recognizable sets of configurations of N. Then, if C' is reachable from C in N, there exists an index k such that C' is k-bounded reachable from C in M. Thus, the semi-algorithm given in the previous section can be used to check reachability in N, and thus in PDNs.

This technique generalizes the algorithms proposed in [7,9,12] for bounded context-switch analysis, where the analysis is performed by bounding the number of interleavings between the different processes of the network. Indeed, our notion of phase is more general than the notions of context used in these works in the sense that, if we encode our model in those proposed in [7,9,12], one single phase according to our definition may correspond to an unbounded number of context switches in their models. Thus, our bounded phase analysis may allow an arbitrary number of context switches (in the sense of [7,9,12]). We give in the next paragraph an example of a system where this holds.

8.1. Bounded-phase vs. bounded-context reachability

In the following, we construct a family $N_1, N_2, N_3, \ldots$ of stable acyclic PDNs (corresponding to one phase MAPNs) such that deciding the reachability problem for a network N_k ($k \ge 1$) using the algorithms in [7,9,12] needs at least k context-switches; whereas we can decide this in one step since stable acyclic PDNs effectively preserve recognizability. Formally, for every $k \ge 1$, the network N_k is defined by the tuple $(\mathcal{P}_1^k, \mathcal{P}_2^k, R)$ where for every $i \in \{1, 2\}$, $\mathcal{P}_i^k$ is a communicating pushdown system and $R = \{(1, 2)\}$ is the communication structure ($\mathcal{P}_1^k$ can observe the control states of $\mathcal{P}_2^k$). For every $k \ge 1$ and for every $i \in \{1, 2\}$, the process $\mathcal{P}_i^k$ is defined by the tuple $(P_i^k, \Gamma_i^k, \Delta_i^k)$ where: (1) $P_i^k = \{p_i^1, \ldots, p_i^{k+1}\}$ is a finite set of control states, (2) $\Gamma_i^k = \{\gamma_i^1, \ldots, \gamma_i^{k+1}\}$ is a finite set of stack symbols, and (3) Δ_i^k is a finite set of transition rules. The set Δ_2^k is the smallest set such that for every $1 \le j \le k$, $\mathcal{P}_2^k$ has a rule that moves the control state from p_2^j to p_2^{j+1} and replaces the topmost symbol of the stack γ_2^j by the symbol γ_2^{j+1}, i.e. $\Delta_2^k = \{(p_2^j, \gamma_2^j) \hookrightarrow (p_2^{j+1}, \gamma_2^{j+1}) \mid 1 \le j \le k\}$. The process $\mathcal{P}_1^k$ starts executing from the control state p_1^{k+1} and the stack content γ_1^1. For every $1 \le j \le k$, if the control state of $\mathcal{P}_2^k$ is p_2^j, $\mathcal{P}_1^k$ can push in its stack the symbol γ_1^{j+1}, i.e. the rule $\{p_2^j\} : (p_1^{k+1}, \gamma_1^j) \hookrightarrow (p_1^{k+1}, \gamma_1^{j+1}\gamma_1^j)$ is in Δ_1^k. Then, for every $\ell \in \{2, \ldots, k+1\}$, if the control state of $\mathcal{P}_2^k$ is p_2^{k+1}, $\mathcal{P}_1^k$ can move its control state from p_1^ℓ to $p_1^{\ell-1}$ while popping the symbol γ_1^ℓ from the stack, i.e. the rule $\{p_2^{k+1}\} : (p_1^\ell, \gamma_1^\ell) \hookrightarrow (p_1^{\ell-1}, \epsilon)$ is in Δ_1^k. Thus, deciding whether the configuration $(p_1^1\gamma_1^1, p_2^{k+1}\gamma_2^{k+1})$ is reachable by N_k from the initial configuration $(p_1^{k+1}\gamma_1^1, p_2^1\gamma_2^1)$ needs at least k context-switches (in the sense of [7,9,12]).

8.2. A case study: a bluetooth driver in Windows NT

We used our PDN model to describe two versions of the Bluetooth driver in Windows NT, and we used our techniques to find the bugs of this driver that were reported in [13,10]. The bugs that are found are data race bugs described as reachability queries. We found the bugs after 8 phase switches for the first version, and 14 phase switches for the second version.

9. Conclusion and applications

In this paper, we consider networks of communicating pushdown systems where the processes can read the control states of the other ones according to a given communication structure. Reachability in such a model being undecidable, we consider networks with *acyclic* communication graphs. We define the class of *stable* acyclic PDNs and show that it effectively preserves recognizability. Then, we consider networks with *dynamic* changes of the communication structures (MAPNs). This model being Turing powerful, we give conditions under which reachability or bounded-phase reachability become decidable for MAPNs, and give a semi-algorithm to decide bounded-phase reachability for general MAPNs and PDNs. Our MAPN and PDN models can be used to describe concurrent programs. For example, it can model two versions of a Windows NT Bluetooth driver. Our techniques can be applied to find the bugs of these drivers reported in [13,10].

References

[1] J. Esparza, J. Knoop, An automata-theoretic approach to interprocedural data-flow analysis, in: FoSSaCS, in: LNCS, vol. 1578, Springer, 1999, pp. 14–30.
[2] J. Esparza, S. Schwoon, A BDD-based model checker for recursive programs, in: CAV, in: LNCS, vol. 2102, Springer, 2001, pp. 324–336.
[3] T. Reps, S. Schwoon, S. Jha, Weighted pushdown systems and their application to interprocedural dataflow analysis, in: SAS, in: LNCS, vol. 2694, Springer, 2003, pp. 189–213.
[4] A. Bouajjani, J. Esparza, O. Maler, Reachability analysis of pushdown automata: application to model-checking, in: CONCUR, in: LNCS, vol. 1243, Springer, 1997, pp. 135–150.
[5] A. Finkel, B. Willems, P. Wolper, A direct symbolic approach to model checking pushdown systems, Electron. Notes Theor. Comput. Sci. 9 (1997).
[6] A. Bouajjani, J. Esparza, T. Touili, A generic approach to the static analysis of concurrent programs with procedures, in: POPL, ACM, 2003, pp. 62–73.
[7] S. Qadeer, J. Rehof, Context-bounded model checking of concurrent software, in: TACAS, in: LNCS, vol. 3440, Springer, 2005, pp. 93–107.
[8] A. Bouajjani, M. Müller-Olm, T. Touili, Regular symbolic analysis of dynamic networks of pushdown systems, in: CONCUR, in: LNCS, vol. 3653, Springer, 2005, pp. 473–487.
[9] A. Bouajjani, J. Esparza, S. Schwoon, J. Strejcek, Reachability analysis of multithreaded software with asynchronous communication, in: FSTTCS, in: LNCS, vol. 3821, Springer, 2005, pp. 348–359.
[10] S. Chaki, E. Clarke, N. Kidd, T. Reps, T. Touili, Verifying concurrent message-passing C programs with recursive calls, in: TACAS, in: LNCS, vol. 3920, Springer, 2006, pp. 334–349.
[11] M.F. Atig, A. Bouajjani, T. Touili, On the reachability analysis of acyclic networks of pushdown systems, in: CONCUR, in: LNCS, vol. 5201, Springer, 2008, pp. 356–371.
[12] D. Suwimonteerabuth, J. Esparza, S. Schwoon, Symbolic context-bounded analysis of multithreaded Java programs, in: SPIN, 2008, pp. 270–287.
[13] S. Qadeer, D. Wu, Kiss: keep it simple and sequential, in: PLDI, ACM, 2004, pp. 14–24.
[14] D. Lugiez, P. Schnoebelen, The regular viewpoint on pa-processes, in: CONCUR, in: LNCS, vol. 1466, Springer, 1998, pp. 50–66.
[15] J. Esparza, A. Podelski, Efficient algorithms for pre* and post* on interprocedural parallel flow graphs, in: POPL, ACM, 2000, pp. 1–11.
[16] H. Seidl, B. Steffen, Constraint-based inter-procedural analysis of parallel programs, in: ESOP, in: LNCS, vol. 1782, Springer, 2000, pp. 351–365.
[17] M. Müller-Olm, Variations on constants, Habilitation thesis, Dortmund University, 2002.
[18] A. Bouajjani, T. Touili, Reachability analysis of process rewrite systems, in: FSTTCS, in: LNCS, vol. 2914, Springer, 2003, pp. 74–87.
[19] A. Bouajjani, T. Touili, On computing reachability sets of process rewrite systems, in: RTA, in: LNCS, vol. 3467, Springer, 2005, pp. 484–499.
[20] S.L. Torre, P. Madhusudan, G. Parlato, A robust class of context-sensitive languages, in: LICS, IEEE Computer Society, 2007, pp. 161–170.
[21] V. Kahlon, F. Ivancic, A. Gupta, Reasoning about threads communicating via locks, in: CAV, in: LNCS, vol. 3576, Springer, 2005, pp. 505–518.
[22] V. Kahlon, A. Gupta, On the analysis of interacting pushdown systems, in: POPL, ACM, 2007, pp. 303–314.
[23] R. Chadha, M. Viswanathan, Decidability results for well-structured transition systems with auxiliary storage, in: CONCUR, in: LNCS, vol. 4703, Springer, 2007, pp. 136–150.
[24] S.L. Torre, P. Madhusudan, G. Parlato, Context-bounded analysis of concurrent queue systems, in: TACAS, in: LNCS, vol. 4963, Springer, 2008, pp. 299–314.
[25] A. Heußner, J. Leroux, A. Muscholl, G. Sutre, Reachability analysis of communicating pushdown systems, in: FOSSACS, in: LNCS, vol. 6014, Springer, 2010, pp. 267–281.