On the Reachability Analysis of Acyclic Networks of Pushdown Systems

Mohamed Faouzi Atig, Ahmed Bouajjani, and Tayssir Touili

LIAFA, CNRS & Univ. of Paris 7, Case 7014, 75205 Paris 13, France.
{atig,abou,touili}@liafa.jussieu.fr

Abstract. We address the reachability problem in acyclic networks of pushdown systems. We consider communication based either on shared memory or on message passing through unbounded lossy channels. We prove mainly that the reachability problem between recognizable sets of configurations (i.e., definable by a finite union of products of finite-state automata) is decidable for such networks, and that for lossy channel pushdown networks, the channel language is effectively recognizable. This fact holds although the set of reachable configurations (including stack contents) for a network of depth (at least) 2 is not rational in general (i.e., not definable by a multi-tape finite automaton). Moreover, we prove that for a network of depth 1, the reachability set is rational and effectively constructible (under an additional condition on the topology for lossy channel networks).

1 Introduction

The verification of concurrent programs is an important and highly challenging problem. It is well known that basic analysis problems (such as control point reachability) on concurrent programs with (recursive) procedure calls are undecidable in general (even for programs with boolean data). A lot of efforts have been nevertheless devoted recently to the development of (1) precise analysis algorithms of specific formal models of some classes of programs [23, 17, 9, 10, 8, 28, 12, 22, 19], and of (2) generic analysis techniques based on computing upper or lower approximations of the set of potential behaviors (see, e.g., [7, 13, 25, 6, 29]).

In this paper we address the issue of analyzing concurrent programs with *asynchronous acyclic communication*, i.e., programs where the communication relation between parallel processes is a *directed acyclic graph* (and therefore the information flows only in one direction between any two processes, but it can follow several paths). We consider two kinds of asynchronous communication mechanisms: communication using a shared memory, and communication through unreliable fifo channels. We define models of such concurrent programs, and we investigate the decidability of their reachability problem, as well as the issue of characterizing and constructing a finite representation of their sets of reachable configurations.

The program models we define are based on *networks of communicating pushdown systems*. It is indeed well admitted that pushdown systems are adequate models for sequential programs with recursive procedure calls [16, 27], and therefore, its is natural to model concurrent programs as parallel communicating pushdown processes.

We start by considering asynchronous communication with shared memory. We define a model, called $\mathrm{APN}_{\mathsf{obs}}$, which consists in a finite collection of pushdown processes communicating according to an *acyclic observation relation* R: a process P can observe (i.e., read and test) the control state of a process Q if $(P,Q) \in R$. Intuitively, the control state of a process Q represents the content of the variables owned by Q (i.e., variables that can be accessed both in read and write modes by Q), and the fact that P can observe

Q means that P has the right to read the variables owned by Q. (It is easy to show that considering cyclic observation relations leads to Turing powerful models.)

The configurations of an $\mathrm{APN}_{\mathsf{obs}}$ with n processes $P_1, \ldots, P_n$ are n-dim vectors of words of the form $p_i w_i$, where p_i (resp. w_i) represents the control state (resp. the stack content) of the process P_i. Therefore, sets of configurations are n-dim languages (i.e., sets of n-dim vectors of words), and we need to consider finite representations for such languages. Two natural and well-known classes of n-dim languages are (1) the class of *rational* languages definable by multi-tape finite automata, and (2) the class of *recognizable* languages which are finite unions of products of regular (1-dim) word languages. (Every recognizable language is a rational one, but the converse is not true in general. For instance, the set $\{(a^n, b^n) : n \geq 0\}$ is rational but not recognizable.)

Then, we are interested in the problem of deciding whether, given two sets of configurations S_1 and S_2, supposed to be recognizable or rational (effectively represented, respectively, either by a multi-tape automaton, or by a finite collection of vectors of finite automata), it is possible to reach a configuration in S_2 from a configuration in S_1. We are also interested in the nature of the sets of reachable configurations in the sense that we want to determine whether these sets are recognizable, rational, or outside these classes. This is important for understanding the expressive power of the models, and for tackling analysis problems such as the reachability problem stated above.

We show that the set of reachable configurations in an $\mathrm{APN}_{\mathsf{obs}}$ from a single configuration is *not* rational in general, and that this fact holds as soon as the graph of the observation relation is of depth 2. Moreover, we prove that the reachability problem from a single configuration to a rational set of configurations is *undecidable* for such networks. Nevertheless, we establish several positive results when the source and target sets in the reachability problem are recognizable. First, we prove that for networks of depth 1, the set of reachable configurations from a recognizable set (which is not recognizable in general) is actually always *rational* and effectively constructible. (From this result follows immediately the decidability of the reachability problem between recognizable sets for networks of depth 1.) Furthermore, we show that for the general case (i.e., networks of *any* depth), the reachability problem between recognizable sets of configurations is *decidable*, although the reachability sets for these models are not always rational as mentioned above.

Then, we pursue our study by considering message-passing communication. We introduce a model called $\mathrm{APN}_{\mathsf{lc}}$ which consists in an *acyclic* network of pushdown processes communicating through *unbounded lossy FIFO channels*. (Again, it can be shown that cyclic lossy channel pushdown networks are Turing powerful). The $\mathrm{APN}_{\mathsf{lc}}$ models are actually more general than the $\mathrm{APN}_{\mathsf{obs}}$ models: every $\mathrm{APN}_{\mathsf{obs}}$ can indeed be simulated by an $\mathrm{APN}_{\mathsf{lc}}$, whereas the converse holds (only) when the topology is linear.

A configuration of an $\mathrm{APN}_{\mathsf{lc}}$ consists in a vector of local configurations of each of the processes in the network (i.e., the vector of their control states and stack contents), and a vector of words corresponding the contents of all the communication channels. Then, we address for the $\mathrm{APN}_{\mathsf{lc}}$ models the same questions (reachability problem, characterization of the reachability sets) as for the $\mathrm{APN}_{\mathsf{obs}}$ models.

Clearly, negative results stating undecidability or non recognizability/rationality for $\mathrm{APN}_{\mathsf{obs}}$ models can be transfered to $\mathrm{APN}_{\mathsf{lc}}$ models. Moreover, we show that, contrary to the case of $\mathrm{APN}_{\mathsf{lc}}$ models, even for networks of depth 1 the set of reachable configurations is *not rational* in general. However, we prove that for networks of depth 1, the set of reachable configurations is rational and effectively constructible when the undirected graph of the communication relation is a forest (i.e., the undirected graph has no cycle). Moreover, we prove that the reachability problem between recognizable sets of configurations is actually *decidable* for the *whole* class of $\mathrm{APN}_{\mathsf{lc}}$ models. We also prove that the channel language of these models, i.e., the *projection* of the set of their reachable configurations on control states and on channels contents (abstracting away the stack contents), is an *effectively constructible* recognizable set.

For lack of space, detailed proofs are omitted here. They can be found in [4].

Related work: The automatic verification of concurrent programs is a very active research topic. Several models for such programs have been proposed recently, and their verification problems have been investigated. Decidability results in this context appear for instance in [23, 9, 10, 8, 20, 11, 28, 29]. These results do not cover the class of models we consider in this paper.

A form of acyclic observation between pushdown systems was introduced in [8]. In that work, a process can observe the states of the processes it has created (dynamic creation is allowed in that work); however, the process cannot distinguish between different states in a control loop of any observed process. This (relatively strong) restriction guarantees that the set of (backward) reachable configurations (of the models defined in [8]) is definable using a finite tree automaton. In the context of our present work, we can show that a similar restriction guarantees in fact the recognizability of the set of reachable configurations.

In [29], the decidability of the reachability problem is established for a finite number of computation phases in multi-stack systems, where in each phase the system can pop from one distinguished stack, and push on some number of stacks. Thus, for each pair of stacks s_1 and s_2, the alternation between phases where s_1 is popped while s_2 is pushed, and phases where the converse holds, is bounded. In our models, it is possible to have an unbounded number of such alternations. On the other hand, since the communication relation in our models is fixed, our models cannot simulate phase switches in the sense of [29]. Actually, we can prove that in our $\mathrm{APN}_{\mathsf{obs}}$ models, switching between different communication relations leads to an undecidable model, even for one single switch [4]. These facts show that our models are not comparable with the ones defined in [29].

Many works have addressed the verification problem of lossy FIFO channel systems with a finite control structure (see, e.g., [3, 2]). For this case, the reachability problem is decidable (without any restriction on the topology of the network) and the proof is based on the theory of monotonic systems w.r.t. a well-quasi ordering on the configuration space [1, 18]. Arguments based on this theory are not applicable to our models (due to the stacks). Moreover, the channel language of finite-control lossy channel systems is recognizable but non constructible in general [24]. We prove here that the channel language is effectively constructible for acyclic network with a pushdown-definable control. Also, the complexity of the reachability problem for finite-control lossy channel systems is nonprimitive recursive [26]. In our case, we have established a primitive recursive upper-bound.

Acyclic networks of FIFO channel systems have been considered in [14] where the authors prove that for two finite-control processes communicating with one perfect channel and one lossy channel the reachability problem is decidable.

In [12], the authors consider acyclic networks of pushdown systems communicating through lossy FIFO channels. They prove the decidability of the reachability problem under the restriction that each process can read a message from a channel *only* when its stack is empty. Our result on $\mathrm{APN}_{\mathsf{lc}}$ is more general since we do not have such a restriction in our models.

Networks of pushdown systems communicating through *perfect* FIFO-channel systems have been considered in [21]. The reachability problem is decidable under the assumption that there is finite number of phases where in each of these phases one process is allowed to run, and this process can read from one single *input channel*, but *only if its stack is empty* (like in [12]), and to send messages to other output channels *different from the input channel*. Actually, the authors show that these models can be simulated by the ones they have considered in [29]. Again, these models are not comparable with our $\mathrm{APN_{lc}}$ models. In particular, we do not have the restriction that a message reception can occur if the stack is empty. Moreover, in the framework of [21], since the channels are perfect, allowing a process to receive messages from two different channels leads to a Turing powerful model, whereas reception from several channels is allowed in our case ($\mathrm{APN_{lc}}$ can have any directed acyclic topology). On the other hand, in our framework channels are supposed to be lossy, and the communication relation is fixed.

2 Preliminaries

2.1 Languages and finite automata

Let Σ be a finite alphabet. We denote by Σ^* (resp. Σ^+) the set of all *words* (resp. non empty words) over Σ, and by ε the empty word. A language is a (possibly infinite) set of words. Let $w = a_1 \ldots a_n$ be a word in Σ^*, then the reverse of w is the word $w^R = a_n \ldots a_1$. 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.

Given two n-dim words $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$, their concatenation is defined by $\mathbf{uv} = (u_1v_1, \ldots, u_nv_n)$. Let $\Sigma_\varepsilon = \Sigma_1 \cup \{\varepsilon\} \times \cdots \times \Sigma_n \cup \{\varepsilon\}$. It easy to see that every n-dim word is a concatenation of n-dim words in Σ_ε.

Given an alphabet Σ, we denote by $\preceq \subseteq \Sigma^* \times \Sigma^*$ the *subword relation* defined as follows: for every $u = a_1 \cdots a_n \in \Sigma^*$, and every $v = b_1 \cdots b_m \in \Sigma^*$, $u \preceq v$ iff $\exists i_1, \ldots, i_n \in \{1, \ldots, m\}$ such that $i_1 < i_2 < \ldots < i_n$ and $\forall j \in \{1, \ldots, n\}, a_j = b_{i_j}$. The relation $\preceq$ is generalized in a pointwise manner to n-dim words as follows: Let $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$ be two n-dim words, $\mathbf{u} \preceq \mathbf{v}$ iff for every i, $1 \leq i \leq n$, $u_i \preceq v_i$. $\prec$ is the strict subword relation: $\mathbf{u} \prec \mathbf{v}$ iff $\mathbf{u} \preceq \mathbf{v}$ and $\mathbf{u} \neq \mathbf{v}$.

Given a (n-dim) language $L \subseteq \Sigma_1^* \times \cdots \times \Sigma_n^*$, the *upward closure* (resp. *downward closure*) of L (w.r.t. $\preceq$) is the set $L\!\uparrow$ (resp. $L\!\downarrow$) $= \{\mathbf{u} \in \Sigma_1^* \times \cdots \times \Sigma_n^* \ : \ \exists \mathbf{v} \in L.\ \mathbf{v} \preceq \mathbf{u}$ (resp. $\mathbf{u} \preceq \mathbf{v}$) $\}$. A (n-dim) language L is upward closed (resp. downward closed) iff $L\!\uparrow = L$ (resp. $L\!\downarrow = L$).

A *n-tape automaton* over $\Sigma_1, \ldots, \Sigma_n$ is a tuple $T = (Q, \Sigma_1, \ldots, \Sigma_n, \delta, I, F)$ where Q is a finite set of states, $\delta \subseteq Q \times \Sigma_\varepsilon \times Q$ is a labeled transition relation, $I \in Q$ is a set of initial states, and $F \subseteq Q$ is a set of final states. Given a *n-tape* automaton $T = (Q, \Sigma_1, \ldots, \Sigma_n, \delta, I, F)$ and a state $q \in Q$, let T^q denote the *n-tape* automaton $(Q, \Sigma_1, \ldots, \Sigma_n, \delta, \{q\}, F)$. A run of T over $\mathbf{w} \in \Sigma_1^* \times \cdots \times \Sigma_n^*$ is a states sequence $q_0 q_1 \cdots q_m \in Q^+$ such that (1) $q_0 \in I$, (2) $\exists \mathbf{u_0}, \ldots, \mathbf{u_{m-1}} \in \Sigma_\varepsilon$. $\forall i \in \{0, \ldots, m-1\}$. $(q_i, \mathbf{u_i}, q_{i+1}) \in \delta$ and $\mathbf{u_0} \cdots \mathbf{u_{m-1}} = \mathbf{w}$. The run is *accepting* if $q_m \in F$. The language of T, denoted $L(T)$, is the set of n-dim words for which there is an accepting run of T.

A n-dim language is *rational* if it is definable as the language of some n-tape automaton. Note that 1-tape automata are the usual finite-state word automata. Their languages are commonly known to be *regular*. 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 an automaton over Σ_i). The class of rational languages subsumes strictly the class of recognizable languages. For instance, the set $\{(a^n, b^n) : n \geq 0\}$ is rational but not recognizable, whereas $\{(a^n, b^m) : n, m \geq 0\}$ is recognizable, and $\{(a^n, b^n c^n) : n \geq 0\}$ is not rational.

Let us recall some well known facts about these classes of languages (see, e.g., [5]), and fix some notations. First, the class of recognizable languages, for any dimension

$n \geq 1$, is closed under boolean operations. On the other hand, for every $n \geq 2$, the class of n-dim rational languages is closed under union, but not under complementation, nor intersection. However, the intersection of a rational language with a recognizable language is rational. Moreover, the emptiness problem of n-tape automata is decidable, and the same holds for the inclusion problem of recognizable languages. However, the inclusion problem is undecidable for rational languages (for $n \geq 2$).

Rational languages are also closed under projection, defined as follows: Given a n-tape automaton T over $\Sigma_1, \ldots, \Sigma_n$, and a set of indices $\iota \subset \{1, \ldots, n\}$, the projection of T on ι, denoted $\Pi_\iota(T)$, is the automaton obtained by erasing all the tapes which are not in ι (if ι has k indices, then $\Pi_\iota(T)$ is a k-tape automaton). Rational languages are also closed under composition: Let T and T' be two multi-tape automata on, respectively, the alphabets $\Sigma_1, \ldots, \Sigma_n$ and $\Sigma'_1, \ldots, \Sigma'_m$, and let $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ be two indices s.t. $\Sigma'_j = \Sigma_i$. Then, it is possible to construct a $(n+m-1)$-tape automaton $T \circ_{(i,j)} T'$ which accepts $(w_1, \ldots, w_n, w'_1, \ldots, w'_{j-1}, w'_{j+1}, \ldots, w'_m)$ iff $(w_1, \ldots, w_n) \in L(T)$ and $(w'_1, \ldots, w_{j-1}, w_i, w_{j+1}, \ldots, w'_m) \in L(T')$, i.e. the composition corresponding to the synchronization of the i^{th} tape of T with the j^{th} tape of T'.

In the same manner, we define the composition operator $\diamond_{(i,j)}$, between two multi-tape automata T and T', that checks whether the content of the j^{th} tape of T' (say w'_j) is a prefix of the content of the i^{th} tape of T (say w_i). If this holds, the i^{th} tape of $T \diamond_{(i,j)} T'$ contains the word w s.t. $w_i = w'_j w$. Formally, let T and T' be two tape automata on, respectively, the alphabets $\Sigma_1, \ldots, \Sigma_n$ and $\Sigma'_1, \ldots, \Sigma'_m$, and let $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ such that $\Sigma'_j = \Sigma_i$. Then, it is possible to construct a $(n+m-1)$-tape automaton $T \diamond_{(i,j)} T'$ that accepts the set of vectors $(w_1, \ldots, w_{i-1}, w, w_{i+1}, \ldots, w_n, w'_1, \ldots, w'_{j-1}, w'_{j+1}, \ldots, w'_m)$ if $\exists w_i, w'_j \in \Sigma_i^*$ such that: (1) $(w_1, \ldots, w_n) \in L(T)$, (2) $(w'_1, \ldots, w'_m) \in L(T')$, and (3) $w_i = w'_j w$.

Proposition 1. *Let $T = (Q, \Sigma_1, \ldots, \Sigma_n, \delta, I, F)$ be a n-tape automaton that accepts the rational language L (i.e. $L = L(T)$), then, the sets $L\downarrow$ and $L\uparrow$ are effectively recognizable. Moreover, the set $L\downarrow$ (resp. $L\uparrow$) can be accepted by an union of $((|\Sigma_1|+1) \times \cdots \times (|\Sigma_n|+1))^{|Q|}$ products of n finite state automata with at most $|Q|$ states.*

2.2 Labeled pushdown systems

A Labeled Pushdown System (LPDS) is defined by a tuple $\mathcal{P} = (P, \Sigma, \Gamma, \Delta)$ where P is a finite set of states, Σ is the input alphabet (actions), Γ is the stack alphabet, and Δ is a finite set of transition rules of the form $\langle p, u \rangle \overset{a}{\hookrightarrow} \langle p', u' \rangle$; where: (1) $p, p' \in P$, (2) $a \in \Sigma \cup \{\varepsilon\}$, and (3) $u, u' \in \Gamma^*$ s.t. either (*i*) $|u| = 1$ and $u' = \varepsilon$ (pop operation), (*ii*) $u = \varepsilon$ and $|u'| = 1$ (push operation), or (*iii*) $u = u' = \varepsilon$ (no operation on the stack).

A *configuration* of an LPDS is a word $pw \in P\Gamma^*$ where p is a state and w is a stack content. In particular, configurations of the form $p\varepsilon$ are simply denoted by p. We define a *transition relation* $\overset{a}{\Longrightarrow}_{\mathcal{P}}$ between configurations as follows: $pw \overset{a}{\Longrightarrow}_{\mathcal{P}} p'w'$ if $\exists\ (\langle p, u \rangle \overset{a}{\hookrightarrow} \langle p', u' \rangle) \in \Delta$ and $\exists v \in \Gamma^*$ s.t. $w = uv$ and $w' = u'v$. We generalize the transition relation $\overset{a}{\Longrightarrow}_{\mathcal{P}}$ to sequences of actions in the usual way: $pw \overset{\varepsilon}{\Longrightarrow}_{\mathcal{P}} pw$ for every $pw \in P\Gamma^*$, and given a word $\sigma = a_0 \cdots a_{n-1} \in \Sigma^+$, $pw \overset{\sigma}{\Longrightarrow}_{\mathcal{P}} p'w'$ iff $\exists p_0 w_0, \ldots, p_n w_n \in P\Gamma^*$ s.t. $pw = p_0 w_0$, $p'w' = p_n w_n$, and $p_i w_i \overset{a_i}{\Longrightarrow}_{\mathcal{P}} p_{i+1} w_{i+1}$ for every $i \in \{0, \ldots, n-1\}$.

A set of configurations $C \subseteq P\Gamma^*$ is recognizable iff it is recognized by some finite state automaton (i.e., there exists an automaton $\mathcal{A}$ such that $C = L(\mathcal{A})$).

Given an LPDS $\mathcal{P}$ and two recognizable sets of configurations $C, C' \subseteq P\Gamma^*$, let $Traces_{\mathcal{P}}(C,C') = \{\sigma \in \Sigma^* : \exists (c,c') \in C \times C', c \stackrel{\sigma}{\Longrightarrow}_{\mathcal{P}} c'\}$ be the set of sequences that lead $\mathcal{P}$ from C to C'. Clearly, $Traces_{\mathcal{P}}(C,C')$ is a context-free language, and conversely, every context-free language can be defined as a trace language of some LPDS.

We give hereafter two results concerning the regularity of the downward and the upward closures of context-free languages which will be used later in the paper.

Proposition 2. *Let $\mathcal{P} = (P,\Sigma,\Gamma,\Delta)$ be an LPDS and $p, p' \in P\Gamma^*$ be two configurations. It is possible to construct, in exponential (resp. double exponential) time in $(|P|+|\Sigma|+|\Gamma|)$, a finite state automaton $A = (Q,\Sigma,\delta,I,F)$ such that $L(A) = Traces_{\mathcal{P}}(p,p')\!\downarrow$ (resp. $L(A) = Traces_{\mathcal{P}}(p,p')\!\uparrow$), where in the worst case, $|Q|$ is exponential (resp. doubly exponential) in $(|P|+|\Sigma|+|\Gamma|)$.*

3 Relating reachable configurations with traces in LPDS

We establish hereafter a result that is a key ingredient for the construction of Section 5. More precisely, let $\mathcal{P} = (P,\Sigma,\Gamma,\Delta)$ be a LPDS, and let $p \in P$. We show hereafter that the sets $U(\mathcal{P}, p)$ and $D(\mathcal{P}, p)$ defined below are rational and effectively constructible:

$$U(\mathcal{P},p) = \{(p'w,\sigma) \in P\Gamma^* \times \Sigma^* : \sigma^R \in Traces_{\mathcal{P}}(p,p'w)\!\uparrow\} \tag{1}$$

$$D(\mathcal{P},p) = \{(p'w,\sigma) \in P\Gamma^* \times \Sigma^* : \sigma^R \in Traces_{\mathcal{P}}(p,p'w)\!\downarrow\} \tag{2}$$

Theorem 1. *It is possible to construct a 2-tape automaton $\widetilde{T}(\mathcal{P},p)$ (resp. $\widehat{T}(\mathcal{P},p)$), with a number of states that is at most exponential (resp. doubly exponential) in $|P|+|\Sigma|+|\Gamma|$, such that $L(\widetilde{T}(\mathcal{P},p)) = D(\mathcal{P},p)$ (resp. $\widehat{T}(\mathcal{P},p) = U(\mathcal{P},p)$).*

In the following, we give the intuition of the construction of a 2-tape automaton $\widetilde{T}(\mathcal{P},p)$ that accepts the set $D(\mathcal{P},p)$ (the one for $U(\mathcal{P},p)$ is similar). For that, we define a 2-tape automaton $T = (Q,\Gamma,\Sigma,\delta,I,F)$ with $I = P$, such that (w,σ) is accepted by T from the state p' iff $(p'w,\sigma) \in L(\widetilde{T}(\mathcal{P},p))$, i.e., $L(\widetilde{T}(\mathcal{P},p)) = \bigcup_{p' \in P}(p',\varepsilon)L(T^{p'})$.

The idea behind the construction of T is the following: Suppose there is a computation $p \stackrel{\sigma'}{\Longrightarrow}_{\mathcal{P}} p'\gamma_m \cdots \gamma_1$ of $\mathcal{P}$ with $p, p' \in P$, $\sigma' \in \Sigma^*$, and $\gamma_1, \ldots, \gamma_m \in \Gamma$. Then, this computation can be decomposed as follows: $\exists p_0, p'_0, p_1, p'_1, \ldots, p_m \in P$, $\exists \sigma'_0, \ldots, \sigma'_m \in \Sigma^*$, and $\exists a'_0, \ldots, a'_{m-1} \in (\Sigma \cup \{\varepsilon\})$ such that: (1) $p_0 = p$, (2) $\sigma' = \sigma'_0 a'_0 \sigma'_1 a'_1 \cdots \sigma'_{m-1} a'_{m-1} \sigma'_m$, (3) $\forall i \in \{1, \ldots, m-1\}$, $p_i \stackrel{\sigma'_i}{\Longrightarrow}_{\mathcal{P}} p'_i$ (i.e. $\sigma'_i \in Traces_{\mathcal{P}}(p_i, p'_i)$), $(\langle p'_i, \varepsilon\rangle \stackrel{a'_i}{\Longrightarrow} \langle p_{i+1}, \gamma_{i+1}\rangle) \in \Delta$, and (4) $p_m \stackrel{\sigma'_m}{\Longrightarrow}_{\mathcal{P}} p'$.

Now, let σ be a word over Σ such that $\sigma \preceq \sigma'$. Then, $\exists \sigma_0, \ldots, \sigma_m \in \Sigma^*$ and $\exists a_0, \ldots, a_{m-1} \in (\Sigma \cup \{\varepsilon\})$ such that: (1) $\forall i \in \{0, \ldots, m\}$, $\sigma_i \preceq \sigma'_i$ (i.e. $\sigma_i \in Traces_{\mathcal{P}}(p_i, p'_i)\!\downarrow$), and (2) $\forall i \in \{0, \ldots, m-1\}$, $a_i \preceq a'_i$ (i.e., a_i is either a'_i or ε). Then, we have: $(\gamma_m \cdots \gamma_1, \sigma^R) = (\varepsilon, \sigma_m^R)(\gamma_m, a_{m-1})(\varepsilon, \sigma_{m-1}^R)(\gamma_{m-1}, a_{m-2}) \cdots (\gamma_1, a_0)(\varepsilon, \sigma_0^R)$.

The 2-tape automaton T must recognize such pairs $(\gamma_m \cdots \gamma_1, \sigma^R)$. Therefore to construct T we proceed as follows: (1) For each pair $(q,q') \in P \times P$, we construct the automaton $\mathcal{A}_{(q,q')}$ that recognizes the set $Traces_{\mathcal{P}}(q,q')\!\downarrow$ such that q (resp. q') is its unique initial (resp. final) state. This automaton is effectively constructible due to Proposition 2. (2) Then, we consider automata recognizing the mirror (reverse) language of the automata $\mathcal{A}_{(q,q')}$ and extend them to 2-tape words by adding ε on the first tape on all transitions. (3) Finally, we connect these automata using transitions of the forms $(p_{i+1}, (\gamma_{i+1}, a'_i), p'_i)$ and $(p_{i+1}, (\gamma_{i+1}, \varepsilon), p'_i)$, for every rule $\langle p'_i, \varepsilon\rangle \stackrel{a'_i}{\Longrightarrow} \langle p_{i+1}, \gamma_{i+1}\rangle$ in Δ (note that p_{i+1} is the initial state of $\mathcal{A}_{(p_{i+1}, p'_{i+1})}$, and p'_i the final state of $\mathcal{A}_{(p_i, p'_i)}$).

4 Acyclic Networks of Pushdown Systems

An Acyclic Observation Relation Pushdown Network ($\mathrm{APN}_{\mathsf{obs}}$ for short) is defined by a tuple $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ where $R \subseteq \{(i, j) \,:\, 1 \leq i < j \leq n\}$ is an *antisymmetric* binary relation (R defines an acyclic directed graph whose nodes are $1, \ldots, n$), and $\forall i \in \{1, \ldots, n\}$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ is a (constrained) pushdown system where P_i is a finite set of control states, Γ_i is a finite stack alphabet, and Δ_i is a finite set of transition rules of the form $\phi : \langle p, u\rangle \hookrightarrow \langle p', u'\rangle$ where (1) $\phi \subseteq \bigcup\{P_j \,:\, (i, j) \in R\}$, (2) $p, p' \in P_i$, and (3) $u, u' \in \Gamma_i^*$ such that either (*i*) $|u| = 1$ and $u' = \varepsilon$, (*ii*) $u = \varepsilon$ and $|u'| = 1$, or (*iii*) $u = u' = \varepsilon$.

An $\mathrm{APN}_{\mathsf{obs}}$ is *linear* if the relation R is of the form $R = \{(i, i+1) \,:\, 1 \leq i < n\}$. An $\mathrm{APN}_{\mathsf{obs}}$ N consisting of a single process $\mathcal{P}$, i.e. $N = (\mathcal{P}, \emptyset)$, will be denoted by $\mathcal{P}$.

The *depth* of $\mathcal{P}_i$ in N, denoted $d(i)$, is the length of the longest path starting from i in the graph of the relation R. The depth of N is the maximal depth of its processes.

Our models are networks of pushdown systems where each process can observe other processes according to the relation R: A process $\mathcal{P}_i$ can observe any process $\mathcal{P}_j$ s.t. $(i, j) \in R$. In this case, the execution of a rule of $\mathcal{P}_i$ can be conditioned by the fact that $\mathcal{P}_j$ is at some particular state (specified in the constraint ϕ of the rule).

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 $\mathcal{P}_i$. (Notice that a vector $(p_1, \ldots, p_n) \in \prod_{i=1}^n P_i$ is a configuration where all processes have empty stacks.)

We define a *transition relation* $\Longrightarrow_N$ between configurations as follows: $(p_1 w_1, \ldots, p_n w_n) \Longrightarrow_N (p'_1 w'_1, \ldots, p'_n w'_n)$ if $\exists i \in \{1, \ldots, n\}$, and $\exists(\phi : \langle p, u\rangle \hookrightarrow \langle p', u'\rangle) \in \Delta_i$ such that (1) $p = p_i$ and $p' = p'_i$, (2) $w_i = uv$ and $w'_i = u'v$ for some $v \in \Gamma_i^*$, (3) $\forall j > i$, if $(i, j) \in R$, then $p_j \in \phi$, and (4) $\forall j \neq i.\ p_j = p'_j$ and $w_j = w'_j$. Given a configuration $c \in \prod_{i=1}^n P_i\Gamma_i^*$, 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 definition is generalized to sets of configurations in the usual manner. $post_N^*$ denotes the reflexive-transitive closure of $post_N$.

Proposition 3. *For every* $\mathrm{APN}_{\mathsf{obs}}$ $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$, *the reachability problem can be reduced to the same problem for a linear* $\mathrm{APN}_{\mathsf{obs}}$ $N' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, R')$ *s.t.* $\forall i \in \{1, \ldots, n\}$, $P'_i = \prod_{j=i}^n P_j$.

Remark 1. The extension of $\mathrm{APN}_{\mathsf{obs}}$ by allowing cycles in communication relation leads to Turing powerful models.

Remark 2. $\mathrm{APN}_{\mathsf{obs}}$ of depth zero are collections of independent pushdown systems. Their analysis boils down to the analysis of each of the processes separately. Therefore, we consider in the sequel only networks of depth ≥ 1.

5 Computing *post** images for $\mathrm{APN}_{\mathsf{obs}}$

5.1 Limits of recognizability and rationality

We start by showing that the set of reachable configurations of $\mathrm{APN}_{\mathsf{obs}}$ models is not recognizable in general. Consider for instance the network with a process $\mathcal{P}'$ which observes a process $\mathcal{P}$, and assume that $\mathcal{P}$ is cycling between two states, and that at each cycle, it pushes a symbol a in its stack. $\mathcal{P}'$ tries to mimic $\mathcal{P}$ by pushing a symbol b in its stack after each observable cycle of $\mathcal{P}$. Then, the set of reachable configurations, starting from empty stacks, is $\{(b^m, a^n), : m \leq n\}$ (we omit here the states) which is not a recognizable set.

This shows that even for networks of depth 1, the set of reachable configurations can be non recognizable:

Proposition 4. *Given an APN_{obs} N of depth ≥ 1, and a configuration c of N, the set $post_N^*(c)$ is not recognizable in general.*

Next, we show that for networks of depth ≥ 2 the set of reachable configurations is not rational in general. Consider indeed a network with three processes $\mathcal{P}_1$ observing $\mathcal{P}_2$ which observes $\mathcal{P}_3$. Assume that $\mathcal{P}_3$ cycles on two different states and pushes at each cycle in its stack a symbol a, and then terminates by going to a special state. During the execution of $\mathcal{P}_3$, $\mathcal{P}_2$ mimics $\mathcal{P}_3$ by pushing in its stack a symbol b after each observable cycle, and in the same time, $\mathcal{P}_1$ mimics $\mathcal{P}_2$ by pushing a symbol c at each observable cycle. At the end of this phase, $\mathcal{P}_2$ moves to another state where it starts cycling and popping at each cycle a symbol from its stack. During this new phase, $\mathcal{P}_1$ pushes a d at each observable cycle of $\mathcal{P}_2$. Then, the set of reachable configurations when $\mathcal{P}_2$ has emptied its stack is $\{(d^\ell c^m, \varepsilon, a^n) : \ell \leq n, m \leq n\}$, which is not a rational set.

Proposition 5. *Given an APN_{obs} N of depth ≥ 2, and a configuration c of N, the set $post_N^*(c)$ is not rational in general.*

5.2 Reachability analysis for $\mathbf{APN}_{\mathsf{obs}}$ of depth 1

In the sequel, we prove that the set of reachable configurations for an $\mathrm{APN}_{\mathsf{obs}}$ of depth 1 starting from a recognizable set of configurations is a rational set. Notice that computing the reachability set, starting from a recognizable set, can be reduced to computing the reachability set starting from a configuration of the form $(p_1, \ldots, p_n)$ where all the stacks are empty. This can be done by adding to the pushdown processes of the network rules that create the initial recognizable set. Hence, we consider w.l.o.g. that our initial set of configurations is of the form $(p_1, \ldots, p_n)$.

The case of two processes: We first present the proof for the special case of a network with two processes $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\mathcal{P}_1$ observes $\mathcal{P}_2$, i.e., $N = (\mathcal{P}_1, \mathcal{P}_2, \{(1,2)\})$, where $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ and $C = \{(p_1, p_2)\}$. We consider later the general case.

To show that $post_N^*(C)$ is effectively rational, we proceed as follows: First, we compute a 2-tape automaton T_1 that accepts the set U_1 consisting of pairs $(pw, \sigma^R) \in P_1\Gamma_1^* \times P_2^*$ s.t. $\mathcal{P}_1$ can reach the configuration pw from p_1 *provided that* the observed sequence of states of $\mathcal{P}_2$ is σ. Second, we construct a 2-tape automaton T_2 that accepts the set U_2 of pairs $(p'w', \sigma^R) \in P_2\Gamma_2^* \times P_2^*$ s.t. σ is a sequence of $\mathcal{P}_2$ states that can be observed by $\mathcal{P}_1$ during a computation of $\mathcal{P}_2$ from p_2 to $p'w'$. Then, we have that $(pw, p'w') \in post_N^*(C)$ iff $\exists \sigma \in P_2^*$ such that $(pw, \sigma^R) \in U_1$ and $(p'w', \sigma^R) \in U_2$, which means that the sequence σ of $\mathcal{P}_2$ states, needed to be observed by $\mathcal{P}_1$ in order to reach pw from p_1, can be provided by a computation of $\mathcal{P}_2$ that reaches $p'w'$ from p_2. Hence, a 2-tape automaton T such that $L(T) = post_N^*(C)$ can be obtained by a synchronization operation between T_1 and T_2 on their second tape.

Let us start by proving that $U_1 = U(\widehat{\mathcal{P}}_1, p_1)$, where $\widehat{\mathcal{P}}_1 = (P_1, P_2, \Gamma_1, \Delta_1')$ is an LPDS that labels each transition rule that can be fired by process $\mathcal{P}_1$ by the state of process $\mathcal{P}_2$ required (i.e. $\langle p, u\rangle \overset{p_2}{\hookrightarrow} \langle p', u'\rangle \in \Delta_1'$ iff $\exists(\phi : \langle p, u\rangle \hookrightarrow \langle p', u'\rangle) \in \Delta_1$ such that $p_2 \in \phi$).

It is clear that if there exists a run $p_1 \overset{\sigma}{\Longrightarrow}_{\widehat{\mathcal{P}}_1} pw$ of $\widehat{\mathcal{P}}_1$, then, (pw, σ^R) is in U_1. Moreover, the set U_1 is *upward closed* in the sense that if $\sigma \in P_2^*$ enables a sequence of transition rules of $\mathcal{P}_1$ that reaches pw from p_1 (i.e. $(pw, \sigma^R) \in U_1$), then, $\forall \sigma' \in P_2^*$ such that $\sigma \preceq \sigma'$, $\mathcal{P}_1$ can fire the same sequence of transition rules to reach pw from p_1 (i.e. $(pw, \sigma'^R) \in U_1$). Hence, $U(\widehat{\mathcal{P}}_1, p_1) \subseteq U_1$. For other direction, every *minimal pair*

$(pw, \sigma^R) \in U_1$ (i.e. $\nexists \sigma' \prec \sigma$ s.t. $(pw, \sigma'^R) \in U_1$) must verify that each state of σ enables a transition rule fired by $\mathcal{P}_1$ in order to reach pw from p_1. Then, $p_1 \overset{\sigma}{\Longrightarrow}_{\widehat{\mathcal{P}}_1} pw$ is a run of $\widehat{\mathcal{P}}_1$ and this implies that $U_1 \subseteq U(\widehat{\mathcal{P}}_1, p_1)$.

Let now $\widetilde{\mathcal{P}}_2 = (P_2, P_2, \Gamma_2, \Delta_2')$ be an LPDS such that $\langle p, u \rangle \overset{p'}{\hookrightarrow} \langle p', u' \rangle$ is in Δ_2' iff: (*i*) $\langle p, u \rangle \hookrightarrow \langle p', u' \rangle \in \Delta_2$ or (*ii*) $p = p'$ and $u = u' = \varepsilon$. Then, we have that $U_2 = D(\widetilde{\mathcal{P}}_2, p_2)$. To prove this, let us consider $q_0 w_0 \Longrightarrow_{\mathcal{P}_2} q_1 w_1 \cdots \Longrightarrow_{\mathcal{P}_2} q_m w_m$ a computation of $\mathcal{P}_2$ that reaches the configuration $q_m w_m = p' w'$ from $q_0 w_0 = p_2$. Then, every state q_i in this sequence can (1) either be observed several times by $\mathcal{P}_1$ and can then be used to enable many transitions of $\mathcal{P}_1$, (2) or, the state q_i may not be observed at all by $\mathcal{P}_1$. Hence, the set of sequences of states that can be observed by $\mathcal{P}_1$ during this computation is $(q_0^+ q_1^+ \cdots q_m^+)\downarrow$. The sequences $q_0^* q_1^+ \cdots q_m^+$ are traces of $\widetilde{\mathcal{P}}_2$, where the stuttering property is ensured by the rules (*ii*) that add loops on every state. Indeed, taking the downward closure gives the set we are looking for, i.e., $U_2 = D(\widetilde{\mathcal{P}}_2, p_2)$.

The 2-tape automaton T can be computed as follows: (1) we construct a 2-tape automaton T_1 (resp. T_2) accepting $U(\widehat{\mathcal{P}}_1, p_1)$ (resp. $D(\widetilde{\mathcal{P}}_2, p_2)$). T_1 and T_2 are constructed following the proof of Theorem 1, (2) we compose T_1 and T_2 according to their second tape, and (3) we abstract away the second tape in the composed automaton. Formally, $T = \Pi_{(1,3)}(T_1 \circ_{(2,2)} T_2)$. We prove that $L(T) = post^*(C)$ [4].

The general case: Consider an $\mathrm{APN}_{\mathsf{obs}}$ $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ of depth 1. The construction above can be extended to the general case as follows: Suppose that processes $\mathcal{P}_k, \ldots, \mathcal{P}_n$ are of depth 0 and $\mathcal{P}_1, \ldots, \mathcal{P}_{k-1}$ are of depth 1. Moreover, we suppose w.l.o.g. that the processes of depth 1 can observe all the processes of depth 0 (if $\mathcal{P}_i$ does not observe $\mathcal{P}_j$, we add to the constraints of its rules all the states of $\mathcal{P}_j$ so that it becomes observable by it without introducing any constraints on its behaviour). For each $\mathcal{P}_j$ of depth 1, we compute a 2-tape automaton that recognizes $U(\widehat{\mathcal{P}}_j, p_j)$ such that the first tape contains a reachable configuration $p_j w_j$ of $\mathcal{P}_j$, and the second one contains the sequence $\sigma \in (P_k \times \cdots \times P_n)^*$ of control states of processes $\mathcal{P}_k, \ldots, \mathcal{P}_n$ that are needed by $\mathcal{P}_j$ to reach $p_j w_j$. Then, we compose all these automata by synchronizing them on their second tape to get a k-tape automaton $\widehat{T}$ that recognizes a vector $(p_1 w_1, \ldots, p_{k-1} w_{k-1}, \sigma)$ iff for $j \in \{1, \ldots, k-1\}$, $(p_j w_j, \sigma) \in U(\widehat{\mathcal{P}}_j, p_j)$. For the next step, we need to see $\widehat{T}$ as a n-tape automaton if we write the above vector $(p_1 w_1, \ldots, p_{k-1} w_{k-1}, \sigma)$ as $(p_1 w_1, \ldots, p_{k-1} w_{k-1}, s_k, \ldots, s_n)$ s.t. $\sigma = (s_k, \ldots, s_n)$.

Next, we compute for every $\mathcal{P}_i$ of depth 0 a 2-tape automaton $\widetilde{T}_i$ that recognizes $D(\widetilde{\mathcal{P}}_i, p_i)$. Then, we synchronise all these automata with $\widehat{T}$ (the second tape of $\widetilde{T}_i$ gets synchronized with the component of $\widehat{T}$ that corresponds to the states of P_i). We project then on the components corresponding to the configurations. The obtained n-tape automaton accepts $(p_1 w_1, \ldots, p_n w_n)$ iff its is in the reachability set of N.

Theorem 2. *Let $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ be a $\mathrm{APN}_{\mathsf{obs}}$ of depth 1, and let $C = \{(p_1, \ldots, p_n)\}$ be a configuration. It is possible to construct a n-tape automaton T such that $L(T) = post_N^*(C)$. The number of states of T is double exponential in $\sum_{i=1}^n |P_i| + |\Gamma_i|$.*

6 Solving the reachability problem for $\mathrm{APN}_{\mathsf{obs}}$

We consider in this section the reachability problem between two sets of configurations C_1 and C_2 for a linear $\mathrm{APN}_{\mathsf{obs}}$ $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$, i.e., checking whether $\exists c_2 \in C_2$ and

$\exists c_1 \in C_1$ s.t. $c_1 \Longrightarrow_N^* c_2$. Notice that the linearity property of the network N is not a real restriction thanks to Proposition 3. We show that the reachability problem is : (1) undecidable when C_1 is a singleton and C_2 is a rational set, and (2) decidable when C_1 and C_2 are recognizable sets. The undecidability result is proven by a reduction of PCP.

Theorem 3. *Given a rational set C and a configuration c, the problem of checking whether C is reachable from c is undecidable for APN_{obs}.*

We now consider the reachability between two recognizable sets of configurations C_1 and C_2. Notice that checking whether C_2 is reachable from C_1 can be reduced to checking reachability between two single configurations where all stacks are empty. This can be done by adding to processes of the network: (1) *push* rules that create an initial configuration in C_1, and (2) *pop* rules that check, in a nondeterministic way, if the current configuration belongs to C_2.

Intuitively, to check whether the configuration $(p'_1, \ldots, p'_n)$ is reachable from $(p_1, \ldots, p_n)$, we proceed inductively as follows: We start from index 1, and let A_1 be an automaton recognizing P_1^*. For every $i \in \{1, \ldots, n-1\}$, we construct an automaton $\mathcal{A}_{i+1}$ that recognizes the set of state sequences $\sigma \in P_{i+1}^*$ such that, if $\mathcal{P}_{i+1}$ has a run generating σ (modulo stuttering), then $\mathcal{P}_i$ has a computation from p_i to p'_i generating a sequence (modulo stuttering) in $L(\mathcal{A}_i)$. We consider stuttering since $\mathcal{P}_i$ (resp. $\mathcal{P}_{i-1}$) can observe several times $\mathcal{P}_{i+1}$ (resp. $\mathcal{P}_i$) in the same state. The set $L(\mathcal{A}_{i+1})$ is upward closed since if $\mathcal{P}_i$ requires observing the sequence σ of $\mathcal{P}_{i+1}$ for its computation, then $\mathcal{P}_i$ can perform the same computation if $\mathcal{P}_{i+1}$ generates (modulo stuttering) a sequence σ' such that $\sigma \preceq \sigma'$. We consider an LPDS $\widehat{\mathcal{P}_i}$ such that $Traces_{\widehat{\mathcal{P}_i}}(p_i, p'_i)$ is the set of observation sequences required by $\mathcal{P}_i$. Let $\widehat{\mathcal{P}_i} \otimes \mathcal{A}_i$ be the restriction of $\widehat{\mathcal{P}_i}$ to runs generating sequences in $L(\mathcal{A}_i)$ modulo stuttering. The automaton $\mathcal{A}_{i+1}$ can be obtained by constructing the upward closure of $Traces_{\widehat{\mathcal{P}_i} \otimes \mathcal{A}_i}(p_i, p'_i)$. Then, if $\mathcal{P}_n$ has a computation from p_n to p'_n generating a sequence in $L(\mathcal{A}_n)$, then $(p'_1, \ldots, p'_n)$ is reachable from $(p_1, \ldots, p_n)$ in N.

Theorem 4. *Checking reachability between two configurations for a linear APN_{obs} $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ can be decide in $2(n-1)$-exponential time in $\sum_{i=0}^{n} |P_i| + |\Gamma_i|$.*

We give hereafter the details of the algorithm underlying this theorem.

Definition 1. *$\forall i \in \{1, \ldots, n-1\}$, let $\widehat{\mathcal{P}_i} = (P_i, P_{i+1}, \Gamma_i, \Delta'_i)$ be an LPDS s.t. $\langle p, u\rangle \overset{s}{\hookrightarrow} \langle p', u'\rangle \in \Delta'_i$ iff $\exists (\phi : \langle p, u\rangle \hookrightarrow \langle p', u'\rangle) \in \Delta_i$ such that $s \in \phi$, and let $\widehat{\mathcal{P}_n} = (P_n, P_n, \Gamma_n, \Delta'_n)$ be an LPDS s.t. $\langle p, u\rangle \overset{\varepsilon}{\hookrightarrow} \langle p', u'\rangle \in \Delta'_n$ iff $\langle p, u\rangle \hookrightarrow \langle p', u'\rangle \in \Delta_n$.*

Definition 2. *Given an LPDS $\mathcal{P} = (P, \Sigma, \Gamma, \Delta)$ and an automaton $\mathcal{A} = (A, P, \delta, I, F)$, let $\mathcal{P} \otimes \mathcal{A} = (P \times A, \Sigma, \Gamma, \Delta')$ be the LPDS where Δ' is the set of transition rules such that: (1) $\langle (p, s), \varepsilon\rangle \overset{\varepsilon}{\hookrightarrow} \langle (p, s'), \varepsilon\rangle \in \Delta'$ iff $(s, p, s') \in \delta$, and (2) $\langle (p, s), u\rangle \overset{a}{\hookrightarrow} \langle (p', s'), u'\rangle \in \Delta'$ iff $\langle p, u\rangle \overset{a}{\hookrightarrow} \langle p', u'\rangle \in \Delta$ and $(s, p', s') \in \delta$.*

Now, we are ready to describe formally our decision procedure: To solve the reachability problem between two configurations $(p_1, \ldots, p_n)$ and $(p'_1, \ldots, p'_n)$, we compute inductively a sequence of automata $\mathcal{A}_i = (A_i, P_i, \delta_i, s_i, f_i)$, where s_i is the initial state and f_i is the final state, defined as follows: (1) $\mathcal{A}_1$ recognizes P_1^* *(since there is no constraint on $\mathcal{P}_1$)*, and (2) $\forall i \in \{1, \ldots, n-1\}$, $\mathcal{A}_{i+1}$ recognizes the regular language $Traces_{\widehat{\mathcal{P}_i} \otimes \mathcal{A}_i}((p_i, s_i), (p'_i, f_i)) \uparrow$. Then we obtain that:

Lemma 1. $(p_1,\ldots,p_n) \Longrightarrow_N^* (p'_1,\ldots,p'_n)$ *iff* $Traces_{\widehat{\mathcal{P}}_n \otimes \mathcal{A}_n}((p_n,s_n),(p'_n,f_n)) \neq \emptyset$.

Remark 3. The above algorithm is *top-down*: it starts from process $\mathcal{P}_1$ down to process $\mathcal{P}_n$. We can also use a *bottom-up* algorithm that starts from process $\mathcal{P}_n$ up to process $\mathcal{P}_1$ as follows: For every index i, we compute the set $\mathcal{A}'_i$ of state sequences that $\mathcal{P}_{i+1}$ can perform to go from p_{i+1} to p'_{i+1}. Then, we compute an LPDS that performs the same transitions as $\mathcal{P}_i$ when the sequences of states performed by $\mathcal{P}_{i+1}$ are in $\mathcal{A}'_i$ (remember that $\mathcal{P}_i$ observes $\mathcal{P}_{i+1}$). This LPDS is computed by a kind of product between $\mathcal{A}'_i$ and $\mathcal{P}_i$. $\mathcal{A}'_{i-1}$ can be computed as the downward closure of the language of the product between $\mathcal{P}_i$ and $\mathcal{A}'_i$. We start by computing $\mathcal{A}'_{n-1}$. Then $(p'_1,\ldots,p'_n)$ is reachable from $(p_1,\ldots,p_n)$ iff the pushdown process corresponding to the restriction of $\mathcal{P}_1$ can go from p_1 to p'_1. This bottom-up procedure is in $(n-1)$-exponential time in $\sum_{i=0}^{n}|P_i|+|\Gamma_i|$.

7 Acyclic Lossy Channel Pushdown Networks

We consider now acyclic networks of pushdown systems communicating through unbounded lossy FIFO channels. An Acyclic Lossy Channel Pushdown Network ($\mathsf{APN}_{\mathsf{lc}}$ for short) is a tuple $H=(\mathcal{P}_1,\ldots,\mathcal{P}_n,C,M)$ where: (1) $C \subseteq \{(j,i)\,:\,1\le i<j\le n\}$ is a finite set of unidirectional channels[1], (2) M is a finite set of messages, and (3) $\forall i \in \{1,\ldots,n\}$, $\mathcal{P}_i=(P_i,\Sigma_i,\Gamma_i,\Delta_i)$ is a *communicating* pushdown system, where P_i a finite set of states, $\Sigma_i=(\{!\}\times M\times\{j\,:\,(i,j)\in C\})\cup(\{?\}\times M\times\{j\,:\,(j,i)\in C\})\cup(\{nop\})$ is a finite set of transition labels, Γ_i is a finite stack alphabet, and Δ_i is a finite set of transition rules of the form: $\langle p,u\rangle \overset{a}{\hookrightarrow} \langle p',u'\rangle$ where $a\in\Sigma_i$, $p,p'\in P_i$, and $u,u'\in\Gamma_i^*$ such that either (*i*) $|u|=1$ and $u'=\varepsilon$, (*ii*) $u=\varepsilon$ and $|u'|=1$, or (*iii*) $u=u'=\varepsilon$.

A transition of $\mathcal{P}_i$ labeled by $(!,m,j)$ means " $\mathcal{P}_i$ sends message m via the channel $(i,j)\in C$ to $\mathcal{P}_j$", whereas a transition labeled by $(?,m,j)$ means "$\mathcal{P}_i$ receives message m from the channel $(j,i)\in C$ sent by $\mathcal{P}_j$". A nop corresponds to an internal action.

An $\mathsf{APN}_{\mathsf{lc}}$ is said to be *linear* if C is of the form $\{(i+1,i)\,:\,i\in\{1,\ldots,n-1\}\}$. The depth $d(i)$ of $\mathcal{P}_i$ is its depth in graph of the binary relation $R=\{(i,j):\,(j,i)\in C\}$. The depth of H, $d(H)$ is $\max\{d(i)\,:\,1\le i\le n\}$.

A *configuration* of the $\mathsf{APN}_{\mathsf{lc}}$ H is a vector $\langle p_1w_1,\ldots,p_nw_n,\mathcal{V}\rangle$ where $p_iw_i\in P_i\Gamma_i^*$ is a local configuration of process $\mathcal{P}_i$ and $\mathcal{V}$ is a mapping from C to M^* giving the contents of each channel, i.e., $\mathcal{V}(i,j)$ describes the content of the channel (i,j).

We define a *transition relation* $\Longrightarrow_H$ between configurations as follows: $\langle p_1w_1,\ldots,p_nw_n,\mathcal{V}\rangle \Longrightarrow_H \langle p'_1w'_1,\ldots,p'_nw'_n,\mathcal{V}'\rangle$ iff $\exists i\in\{1,\ldots,n\}$ and $\exists(\langle p,u\rangle \overset{a}{\hookrightarrow} \langle p',u'\rangle)\in\Delta_i$ such that: (1) $p=p_i$ and $p'=p'_i$, (2) $w_i=uv$ and $w'_i=u'v$ for some $v\in\Gamma_i^*$, (3) $\forall j\neq i.\ p_j=p'_j$ and $w_j=w'_j$, and (4) either (*i*) $a=(?,m,k)$ is a *receive* operation; $m\mathcal{V}'(k,i)\preceq\mathcal{V}(k,i)$ and $\mathcal{V}'(j,l)\preceq\mathcal{V}(j,l)$ for every $(j,l)\in C$ s.t. $(j,l)\neq(k,i)$ (message m is read from the channel (k,i), and the contents of all the channels can lose some messages). Or (*ii*) $a=(!,m,k)$ is a *send* operation, and $\mathcal{V}'(i,k)\preceq\mathcal{V}(i,k)m$ and $\mathcal{V}'(j,l)\preceq\mathcal{V}(j,l)$ for every $(j,l)\in C$ s.t. $(j,l)\neq(i,k)$ (m is added to the channel (i,k) that receives the message and all the channels can lose messages). Or (*iii*) $a=nop$, and $\mathcal{V}'(j,l)\preceq\mathcal{V}(j,l)$ for every $(j,l)\in C$ (to express the loss of messages). Let $\Longrightarrow_H^*$ and $post_H^*$ denote respectively the reflexive-transitive closure of $\Longrightarrow_H$ and $post_H$.

Proposition 6. *For every $\mathsf{APN}_{\mathsf{obs}}$ $N=(\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$, the reachability problem can be reduced to the same problem for an $\mathsf{APN}_{\mathsf{lc}}$ $H=(\mathcal{P}'_1,\ldots,\mathcal{P}'_n,M,C)$. Moreover, for every*

[1] Notice that the graph defined by C is acyclic.

linear APN$_{\mathsf{lc}}$ $H = (\mathcal{P}'_1,\ldots,\mathcal{P}'_n,M,C)$*, the reachability problem between two configurations can be reduced to the same problem for a linear APN*$_{\mathsf{obs}}$ $N = (\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$ *such that* $\forall i \in \{1,\ldots,n\}$*,* $P_i = P'_i \times M' \times M'$ *where* $M' = M \cup \{\#\}$ *(with* $\# \notin M$*). Thanks to Proposition 3, we get that*

8 Computing $post^*$ images for APN$_{\mathsf{lc}}$

8.1 Limits of rationality

Contrary to APN$_{\mathsf{obs}}$ models, the set of reachable configurations for an APN$_{\mathsf{lc}}$ of depth 1 is not rational in general. To see that, consider the network $H_1 = (\mathcal{P}_1,\ldots,\mathcal{P}_4, \{(4,1),(3,1),(4,2),(3,2)\},\{m_3,m_4\})$ given by the figure below with an initial configuration $\langle t_1,s_1,q_1,p_1,\mathcal{V}_0\rangle$ with $\mathcal{V}_0(4,1) = \mathcal{V}_0(4,2) = \mathcal{V}_0(3,1) = \mathcal{V}_0(3,2) = \varepsilon$. $\mathcal{P}_4$ (resp. $\mathcal{P}_3$) starts by pushing a symbol a (resp. b) in its stack while sending the same message m_4 (resp. m_3) to $\mathcal{P}_1$ and $\mathcal{P}_2$. During the execution of $\mathcal{P}_3$ and $\mathcal{P}_4$, $\mathcal{P}_2$ (resp. $\mathcal{P}_1$) pushes c (resp. e) in its stack while removing message m_4 (resp. m_3) from the channel $(4,2)$ (resp. $(3,1)$), then, it starts pushing symbol d (resp. f) in its stack while removing message m_3 (resp. m_4) from the channel $(3,2)$ (resp. $(4,1)$). Hence, the set of reachable configurations when all channels are empty is composed of $\langle t_2 f^i e^l, s_2 d^j c^k, q_1 b^m, p_1 a^n, \mathcal{V}_0\rangle$ such that $i,k \leq n$, $l,j \leq m$. This set is not rational. Hence, we have:

Proposition 7. *Given an APN*$_{\mathsf{lc}}$ H *of depth* 1*, and a configuration* c *of* H*, the set* $post^*_H(c)$ *is not rational in general.*

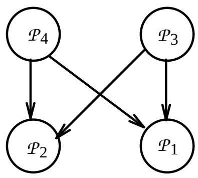
$\langle p_1,\varepsilon\rangle \overset{(!,m_4,1)}{\hookrightarrow} \langle p_2,a\rangle$

$\langle p_2,\varepsilon\rangle \overset{(!,m_4,2)}{\hookrightarrow} \langle p_1,\varepsilon\rangle$

$\langle s_1,\varepsilon\rangle \overset{(?,m_4,4)}{\hookrightarrow} \langle s_1,c\rangle$

$\langle s_1,\varepsilon\rangle \overset{nop}{\hookrightarrow} \langle s_2,\varepsilon\rangle$

$\langle s_2,\varepsilon\rangle \overset{(?,m_3,3)}{\hookrightarrow} \langle s_2,d\rangle$



$\langle q_1,\varepsilon\rangle \overset{(!,m_3,1)}{\hookrightarrow} \langle q_2,b\rangle$

$\langle q_2,\varepsilon\rangle \overset{(!,m_3,2)}{\hookrightarrow} \langle q_1,\varepsilon\rangle$

$\langle t_1,\varepsilon\rangle \overset{(?,m_3,3)}{\hookrightarrow} \langle t_1,e\rangle$

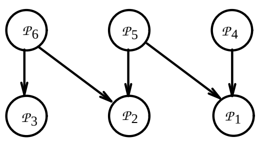
$\langle t_1,\varepsilon\rangle \overset{nop}{\hookrightarrow} \langle t_2,\varepsilon\rangle$

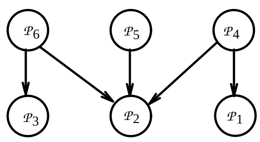
$\langle t_2,\varepsilon\rangle \overset{(?,m_4,4)}{\hookrightarrow} \langle t_2,f\rangle$

8.2 Networks with rational reachability sets

We consider the class of networks of depth 1 such that the undirected graph of the binary relation C is a forest (examples of such networks are given in the figure below). We assume w.l.o.g that initially all stacks and channels are empty.

Theorem 5. *Let* $H = (\mathcal{P}_1,\ldots,\mathcal{P}_n,C,M)$ *be an APN*$_{\mathsf{lc}}$ *of depth* 1 *such that the undirected graph of the binary relation* C *is a forest and let* c *be an initial configuration of* H*. Then, it is possible to construct a* $(n+|C|)$*-tape automaton* T *such that* $L(T) = post^*_H(c)$*. The number of states of* T *is doubly exponential in* $\sum_{j=1}^{n} |P_j| + |\Gamma_j| + |M|$*.*





The case of two processes: To explain the construction underlying this theorem we consider first the case where H contains two processes $\mathcal{P}_1$ and $\mathcal{P}_2$, communicating via a channel $(2,1)$, i.e., $H = (\mathcal{P}_1, \mathcal{P}_2, \{(2,1)\}, M)$. the general case will be treated later. Let $c = \langle p_1, p_2, \mathcal{V}_0 \rangle$ where $\mathcal{V}_0(2,1) = \varepsilon$ be the initial configuration. Next, we will compute a 3-tape automaton T that accepts $post^*_H(c)$. We proceed as follows: First, we compute an LPDS $\breve{\mathcal{P}}_1 = (P_1, M, \Gamma_1, \Delta'_1)$ s.t. $p_1 \stackrel{\sigma}{\Longrightarrow}_{\breve{\mathcal{P}}_1} pw$ means that $\mathcal{P}_1$ reaches the configuration pw from p_1, provided that the sequence of messages consumed is σ. Δ'_1 is defined as follows: (1) $\langle p,u \rangle \stackrel{m}{\hookrightarrow} \langle p',u' \rangle \in \Delta'_1$ iff $\langle p,u \rangle \stackrel{(?,m,2)}{\hookrightarrow} \langle p',u' \rangle \in \Delta_1$, and (2) $\langle p,u \rangle \stackrel{\varepsilon}{\hookrightarrow} \langle p',u' \rangle \in \Delta'_1$ iff $\langle p,u \rangle \stackrel{nop}{\hookrightarrow} \langle p',u' \rangle \in \Delta_1$.

Second, we construct an LPDS $\overline{\mathcal{P}}_2 = (P_2, M, \Gamma_2, \Delta'_2)$ such that $p_2 \stackrel{\sigma}{\Longrightarrow}_{\overline{\mathcal{P}}_2} p'w'$ means that $\mathcal{P}_2$ reaches the configuration $p'w'$ from p_2 while sending in the channel the sequence of messages σ. Δ'_2 is defined as follows: (1) $\langle p,u \rangle \stackrel{m}{\hookrightarrow} \langle p',u' \rangle \in \Delta'_2$ iff $\langle p,u \rangle \stackrel{(!,m,1)}{\hookrightarrow} \langle p',u' \rangle \in \Delta_2$, and (2) $\langle p,u \rangle \stackrel{\varepsilon}{\hookrightarrow} \langle p',u' \rangle \in \Delta'_2$ iff $\langle p,u \rangle \stackrel{nop}{\hookrightarrow} \langle p',u' \rangle \in \Delta_2$.

We want the 3-tape automaton T to accept $(p'w', \sigma, pw) \in P_2\Gamma_2^* \times M^* \times P_1\Gamma_1^*$ iff $\exists \sigma_1, \sigma_2 \in M^*$ s.t. : (1) $p_1 \stackrel{\sigma_1}{\Longrightarrow}_{\breve{\mathcal{P}}_1} pw$, (2) $p_2 \stackrel{\sigma_2}{\Longrightarrow}_{\overline{\mathcal{P}}_2} p'w'$, and (3) $\sigma_1\sigma \preceq \sigma_2$ (the sequence of messages σ_1 is required by $\mathcal{P}_1$ to perform its execution, σ_2 is the sequence of messages sent by $\mathcal{P}_2$, and σ is what remains in the channel after $\mathcal{P}_1$ reads σ_1 from σ_2).

Hence, T can be obtained by computing the 2-tape automata $\widehat{T}(\breve{\mathcal{P}}_1, p_1)$ and $\widetilde{T}(\overline{\mathcal{P}}_2, p_2)$ that accept respectively the sets $U(\breve{\mathcal{P}}_1, p_1)$ and $D(\overline{\mathcal{P}}_2, p_2)$ and then applying the composition operator $\diamond$ between these automata on their second tape (the component corresponding to the channel content). Then, we can prove the following fact:

Lemma 2. *Let $T = \widetilde{T}(\overline{\mathcal{P}}_2, p_2) \diamond_{(2,2)} \widehat{T}(\breve{\mathcal{P}}_1, p_1)$. Then, the 3-tape automaton T accepts the vector $(p'w', \sigma, pw)$ iff $\langle pw, p'w', \mathcal{V} \rangle \in post^*_H(c)$ where $\mathcal{V}(2,1) = \sigma$.*

The general case: The construction above can be extended to any $\mathrm{APN}_{\mathsf{lc}}$ $H = (\mathcal{P}_1, \ldots, \mathcal{P}_n, C, M)$ of depth 1 such that the undirected graph of the binary relation C is a forest. For every $i \in \{1, \ldots, n\}$, let k_i be the number of (input/output) channels of $\mathcal{P}_i$. To compute the reachability set, we proceed as follows: First, for every $\mathcal{P}_i$ of depth 0, we compute a (k_i+1)-tape finite automaton $\widetilde{T}(\overline{\mathcal{P}}_i, p_i)$ that relates the set of reachable configurations of $\mathcal{P}_i$ with the set of message sequences sent by $\mathcal{P}_i$ in each of its k_i-output channels. Second, for every $\mathcal{P}_j$ of depth 1, we compute a (k_j+1)-tape finite automaton $\widehat{T}(\breve{\mathcal{P}}_j, p_j)$ that relates the set of reachable configurations of $\mathcal{P}_j$ with the set of message sequences that $\mathcal{P}_j$ needs to receive from its k_j-input channels. The automata $\widetilde{T}(\overline{\mathcal{P}}_i, p_i)$ and $\widehat{T}(\breve{\mathcal{P}}_j, p_j)$ are constructible due to Theorem 1.

Now, consider a process $\mathcal{P}$ adjacent to a leaf process $\mathcal{Q}$ in the undirected graph of the binary relation C, and assume that there is a channel from $\mathcal{P}$ to $\mathcal{Q}$ (the symmetrical case is similar). Then, we replace the multi-automaton $T_{\mathcal{P}}$ associated with $\mathcal{P}$ by the muti-tape automaton $T'_{\mathcal{P}} = (T_{\mathcal{P}} \diamond T_{\mathcal{Q}})$, and we delete $\mathcal{Q}$ from the network. (This is possible since in our model between any two processes there is at most one channel). We repeat this procedure until we are left with a set of non connected processes. Finally, the product of all their mult-tape automata accepts $post^*_H(c)$.

9 Computing the channels language for $\mathrm{APN}_{\mathsf{lc}}$

We show in this section that the reachability problem for the whole class of $\mathrm{APN}_{\mathsf{lc}}$ is decidable. Moreover, we show that the projection of the reachability set on the channels is an effectively recognizable set.

Let $H = (\mathcal{P}_1, \ldots, \mathcal{P}_n, C, M)$ be an $\mathrm{APN}_{\mathsf{lc}}$, and $c = \langle p_1, \ldots, p_n, \mathcal{V}_0 \rangle$, where $\mathcal{V}_0(i, j) = \varepsilon$ for every channel $(i, j) \in C$, be the initial configuration where all channels and stacks are empty. Let $(p'_1, \ldots, p'_n) \in \prod_{i=1}^{n} P_i$ be a tuple of states of H. We define L_H as follows:

Definition 3. *L_H is the channels language such that $((u_{i,j})_{(i,j)\in C}) \in L_H$ iff $\exists (w_1, \ldots, w_n) \in \Gamma_1^* \times \cdots \times \Gamma_n^*$ s.t. $\langle p'_1 w_1, \ldots, p'_n w_n, \mathcal{V} \rangle \in post_H^*(c)$ and $\mathcal{V}(i, j) = u_{i,j}$.*

Theorem 6. *The channel language L_H is effectively recognizable. Moreover, the set L_H can be accepted by the m-union of products of finite state automata with at most m states, with m is n-exponential in $\sum_{j=1}^{n}(|P_j| + |\Gamma_j|) + |M|$.*

We show hereafter the construction underlying this theorem for the case where H is linear. We explain later how can be extended to an arbitrary acyclic topology. For presentation matters, we suppose first that the system contains three processes $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$. We proceed inductively as follows: First, we start with $\mathcal{P}_3$ and compute the LPDS $\bar{\mathcal{P}}_3 = (P_3, M, \Gamma_3, \Delta'_3)$ as defined previously. Remember that $p_3 \overset{\sigma}{\Longrightarrow}_{\bar{\mathcal{P}}_3} pw$ means that $\mathcal{P}_3$ can reach the configuration pw from p_3, while sending the sequence of messages σ in the channel $(3, 2)$. Then, let $\mathcal{A}_3 = (Q_3, M, \delta_3, s_3, f_3)$ be a finite state automaton that recognizes the regular language $(Traces_{\bar{\mathcal{P}}_3}(p_3, p'_3\Gamma_3^*) \downarrow)$, i.e., the downward closure of messages sequences that are sent by $\mathcal{P}_3$ in order to reach $p'_3 w$ from p_3 for some $w \in \Gamma_3^*$. Therefore, $\mathcal{A}_3$ represents all possible contents of the channel $(3, 2)$ when $\mathcal{P}_3$ reaches $p'_3\Gamma_3^*$ from p_3 (we take the downward closure because the channels are lossy).

Next, we compute an automaton $\mathcal{A}_2$ that represents the sequences of messages that process $\mathcal{P}_2$ can send in the channel $(2, 1)$ while consuming messages from channel $(3, 2)$ (i.e., while consuming messages from $\mathcal{A}_3$). To do this, we compute a kind of product $\mathcal{P}_2 \star \mathcal{A}_3$ between $\mathcal{P}_2$ and $\mathcal{A}_3$ defined as follows:

Definition 4. *Given an index i and an automaton $\mathcal{A} = (Q, M, \delta, s, f)$, let $\mathcal{P}_i \star \mathcal{A} = (P_i \times Q, M, \Gamma_i, \Delta'_i)$ be an LPDS where Δ'_i is defined as follows: (1) $\langle (p, q), u \rangle \overset{\varepsilon}{\hookrightarrow} \langle (p', q'), u' \rangle \in \Delta'_i$ iff $\langle p, u \rangle \overset{(?,m,i+1)}{\hookrightarrow} \langle p', u' \rangle \in \Delta_i$ and $(q, m, q') \in \delta$, (2) $\langle (p, q), u \rangle \overset{m}{\hookrightarrow} \langle (p', q'), u' \rangle \in \Delta'_i$ iff $\langle p, u \rangle \overset{(!,m,i-1)}{\hookrightarrow} \langle p', u' \rangle \in \Delta_i$ and $q = q'$, and (3) $\langle (p, q), u \rangle \overset{\varepsilon}{\hookrightarrow} \langle (p', q'), u' \rangle$ is in Δ'_i iff $\langle p, u \rangle \overset{nop}{\hookrightarrow} \langle p', u' \rangle \in \Delta_i$ and $q = q'$.*

It is clear that $\mathcal{P}_2 \star \mathcal{A}_3$ behaves like $\mathcal{P}_2$, while consuming messages from $\mathcal{A}_3$ (this is ensured by rules (i) that make sure the messages that are consumed are those provided by $\mathcal{P}_3$). Moreover, the language of $\mathcal{P}_2 \star \mathcal{A}_3$ corresponds to the sequences of messages that $\mathcal{P}_2$ sends in the channel $(2, 1)$ (due to rules (ii)). Since $\mathcal{P}_2$ consumes only a part (a prefix) from the channel, for each state $q_3 \in Q_3$ of $\mathcal{A}_3$, $Traces_{\mathcal{P}_2 \star \mathcal{A}_3}((p_2, s_3), (p'_2, q_3)\Gamma_2^*)$ represents the set of messages sequences that are sent by $\mathcal{P}_2$ in order to reach $p'_2 w$ from p_2 for some $w \in \Gamma_2^*$, while consuming a prefix of $\mathcal{A}_3$ that leads to state q_3. In this case, the set of messages sequences that are left in the channel $(3, 2)$ correspond $L(\mathcal{A}_3^{q_3})$.

Let now $\mathcal{A}_{2,q_3}$ be an automaton that accepts $(Traces_{\mathcal{P}_2 \star \mathcal{A}_3}((p_2, s_3), (p'_2, q_3)\Gamma_2^*) \downarrow)$. This means that $\mathcal{A}_{2,q_3}$ represents the content of the channel $(2, 1)$ when the content of the channel $(3, 2)$ is recognized by $\mathcal{A}_3^{q_3}$.

Since $\mathcal{P}_1$ consumes the messages from $\mathcal{A}_{2,q_3}$, for every state $q_2 \in Q_2$ of $\mathcal{A}_{2,q_3}$, we need to check whether it is possible for process $\mathcal{P}_1$ to consume a sequence of messages that correspond to the prefix of $\mathcal{A}_{2,q_3}$ that end in state q_2 in order to go from p_1 to $p'_1 w$ for some $w \in \Gamma_1^*$, i.e., checking whether the set $Traces_{\mathcal{P}_1 \star \mathcal{A}_{2,q_3}}((p_1, s_2), (p'_1, q_2)\Gamma_1^*)$ is empty. If not, $(\mathcal{A}_{2,q_3})^{q_2} \times \mathcal{A}_3^{q_3}$ corresponds to a possible contents of the channels $(3,2)$ and $(2,1)$. It follows that the contents of the channels is given by the union over all the states $q_2 \in Q_2$ and $q_3 \in Q_3$ of automata $(\mathcal{A}_{2,q_3})^{q_2} \times \mathcal{A}_3^{q_3}$ such that $Traces_{\mathcal{P}_1 \star \mathcal{A}_{2,q_3}}((p_1, s_2), (p'_1, q_2)\Gamma_1^*) \neq \emptyset$.

In the case of linear networks with $n \geq 2$ processes, we proceed as follows: First, we compute the finite state automaton $\mathcal{A}_n = (Q_n, M, \delta_n, s_n, f_n)$ that recognizes $Traces^{-}_{\mathcal{P}_n}(p_n, p'_n \Gamma_n^*) \downarrow$. Then, $\forall i \in \{n-1, \ldots, 2\}$ (we start from $n-1$ down to 2), $\forall (q_n, \ldots, q_{i+1}) \in Q_n \times \cdots \times Q_{i+1}$, we compute the finite state automaton $\mathcal{A}_{i,q_n,\ldots,q_{i+1}} = (Q_i, M, \delta_i, s_i, f_i)$ that recognizes $Traces_{\mathcal{P}_i \star \mathcal{A}_{i+1,q_n,\ldots,q_{i+2}}}((p_i, s_i), (p'_i, q_{i+1})\Gamma_i^*) \downarrow$.

Lemma 3. *$\exists (w_1, \ldots, w_n) \in \Gamma_1^* \times \cdots \times \Gamma_n^*$ such that $\langle p'_1 w_1, \ldots, p'_n w_n, \mathcal{V} \rangle \in post_H^*(c)$ with $\mathcal{V}(i+1, i) = u_{(i+1,i)}$ iff $\exists (q_n, \ldots, q_2) \in Q_n \times \cdots \times Q_2$ such that $Traces_{\mathcal{P}_1 \star A_{2,q_n,\ldots,q_3}}((p_1, s_2), (p'_i, q_2)\Gamma_1^*) \neq \emptyset$ and $(u_{(2,1)}, \ldots, u_{(n-1,n)})$ is accepted by $\mathcal{A}^{q_2}_{2,q_n,\ldots,q_3} \times \cdots \times \mathcal{A}_n^{q_n}$.*

For linear network Theorem 6 is an immediate consequence of Lemma 3. The construction above can be easily extended to the general case where H is not linear as follows: For every process $\mathcal{P}_i$ of depth 0, we compute a rational set that characterizes the sequence of messages that can be sent in all the channels related to $\mathcal{P}_i$. Since the channels are lossy, we are interested in the downward closure of this set, which is recognizable by Proposition 1. Then, we proceed upward according to the structure of the channels (this can be done because of acyclicity) as explained above.

Finally, the previous algorithm can be adapted in order to prove the following:

Theorem 7. *The reachability problem between configurations (and therefore between recognizable sets) for APN_{lc} is decidable.*

10 Conclusion

We have proved the decidability of the reachability problem for acyclic networks of pushdown systems with communication mechanisms based on shared memory and message-passing through lossy FIFO channels. Our models constitute an example of infinite-state systems for which the reachability problem is decidable, even though their reachability sets are not rational in general. In our work, we have explored the limits of recognizability/rationality of the reachability sets. We have shown that rationality is lost for depth 2, and for networks with lossy channels, it is lost even for depth 1 (unless the undirected graph of the network is a forest).

Our decidability proofs use automata constructions based on compositional analysis principles. A proof technique we use consists in analyzing the set of reachable configurations in a component of the network, assuming that its environment can provide some input (either an observable computation path, or a sequence of messages). On the other hand, we can analyze the reachable configurations together with the output sequences generated by the computations reaching these configurations. These two kinds of analysis allow to define respectively the input and the output interface of each pushdown process composing the network. Due to the communication mechanisms we consider, these interfaces (which are context-free languages) can be approximated without loss of preciseness (w.r.t. the considered reachability problem) by regular languages that are effectively computable (as upward and downward closures of context-free languages). Then, our proofs consist in showing that the input and output interfaces of a whole network can be "summarized" as a regular language. For that, (1) we isolate the extremal process, P_k say, (2) compute recursively the interface of the rest of the network, (3) compose this (regular) interface with P_k, which leads to a new pushdown system P'_k, and then (4) compute the interface of P'_k. This allows to reduce the reachability problem of our models to solving a sequence of decidable problems on single pushdown systems. (But obviously, this does not mean that our models can be reduced (or simulated) by a single pushdown system.) We believe that this kind of compositional analysis, based on computing upper/under approximate input and output interfaces, could be used for the (approximate) verification of pushdown networks, not only in the acyclic case.

Finally, a natural question which may raise is whether the reachability problem remains decidable if we allow switches between different acyclic communication relations. We prove that even for one of such a switch, the problem becomes undecidable for networks of depth (at least) 2 [4]. The case of networks of depth 1 is left open.

References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite-state systems. In *LICS'96*, pages 313–321, 1996.
2. Parosh Aziz Abdulla, Aurore Collomb-Annichini, Ahmed Bouajjani, and Bengt Jonsson. Using forward reachability analysis for verification of lossy channel systems. *FMSD*, 25(1):39–65, 2004.
3. Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Inf. Comput.*, 127(2):91–101, 1996.
4. M. F. Atig, A. Bouajjani, and T. Touili. On the reachability analysis of acyclic networks of pushdown systems. Technical report, LIAFA lab, April 2008. Available at `http://www.liafa.jussieu.fr/~atig/sub11.pdf`.
5. J. Berstel. Transductions and context-free langages. TeubnerStudienbucher Informatik, 1979.
6. A. Bouajjani, J. Esparza, S. Schwoon, and J. Strejcek. Reachability analysis of multithreaded software with asynchronous communication. In *FSTTCS'05*. LNCS 3821, 2005.
7. A. Bouajjani, J. Esparza, and T. Touili. A generic approach to the static analysis of concurrent programs with procedures. *Int. J. Found. Comput. Sci.*, 14(4):551–, 2003.
8. A. Bouajjani, M. Müller-Olm, and T. Touili. Regular symbolic analysis of dynamic networks of pushdown systems. In *Proc. of CONCUR'05*, volume 3653 of *LNCS*. Springer, 2005.
9. A. Bouajjani and T. Touili. Reachability analysis of process rewrite systems. In *FSTTCS*, volume 2914 of *LNCS*, pages 74–87. Springer, 2003.
10. A. Bouajjani and T. Touili. On computing reachability sets of process rewrite systems. In *Proc. of RTA'05*, volume 3467 of *LNCS*. Springer, 2005.
11. Ahmed Bouajjani and Javier Esparza. Rewriting models of boolean programs. In *RTA*, pages 136–150. LNCS 4098, 2006.
12. R. Chadha and Mahesh Viswanathan. Decidability results for well-structured transition systems with auxiliary storage. In *CONCUR*, pages 136–150. LNCS 4703, 2007.
13. Sagar Chaki, Edmund M. Clarke, Nicholas Kidd, Thomas W. Reps, and Tayssir Touili. Verifying concurrent message-passing c programs with recursive calls. In *TACAS*, pages 334–349. LNCS 3920, 2006.
14. P. Chambard and P. Schnoebelen. Post embedding problem is not primitive recursive, with applications to channel systems. In *FSTTCS'07*, pages 265–276. LNCS 4855, 2007.
15. Bruno Courcelle. On construction obstruction sets of words. *EATCS*, 44:178–185, June 1991.

16. J. Esparza and J. Knoop. An automata-theoretic approach to interprocedural dataflow analysis. In *FOSSACS'99*. LNCS 6578, 1999.
17. J. Esparza and A. Podelski. Efficient algorithms for pre* and post* on interprocedural parallel flow graphs. In *POPL*, pages 1–11, 2000.
18. A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *TCS*, 256(1-2):63–92, 2001.
19. Ranjit Jhala and Rupak Majumdar. Interprocedural analysis of asynchronous programs. In *POPL*. IEEE, 2007.
20. Vineet Kahlon, Franjo Ivancic, and Aarti Gupta. Reasoning about threads communicating via locks. In *CAV*. LNCS 3576, 2005.
21. Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. Context-bounded analysis of concurrent queue systems. In *TACAS08*. LNCS 4963, 2008.
22. Peter Lammich and Markus Müller-Olm. Precise fixpoint-based analysis of programs with thread-creation and procedures. In *CONCUR*. LNCS 4703, 2007.
23. D. Lugiez and Ph. Schnoebelen. The regular viewpoint on PA-processes. In *Proc. of CONCUR'98*, volume 1466 of *LNCS*, pages 50–66. Springer, 1998.
24. Richard Mayr. Undecidable problems in unreliable computations. *Theor. Comput. Sci.*, 1-3(297), 2003.
25. S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS'05*. LNCS 3440, 2005.
26. Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Inf. Process. Lett.*, 83(5), 2002.
27. Stefan Schwoon. *Model-Checking Pushdown Systems*. PhD thesis, Technische Universität München, 2002.
28. Koushik Sen and Mahesh Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. In *CAV*, pages 300–314. LNCS 4144, 2006.
29. Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. A robust class of context-sensitive languages. In *LICS*, pages 161–170. IEEE, 2007.

A Proof of Proposition 1

Proposition 1 *Let $T = (Q, \Sigma_1, \ldots, \Sigma_n, \delta, I, F)$ be a n-tape automaton that accepts the rational language L (i.e. $L = L(T)$), then, the sets $L\downarrow$ and $L\uparrow$ are effectively recognizable. Moreover, the set $L\downarrow$ (resp. $L\uparrow$) can be accepted by an union of $((|\Sigma_1|+1)\times\cdots\times(|\Sigma_n|+1))^{|Q|}$ products of n finite state automata with at most $|Q|$ states.*

Proof (Sketch):

1. **Upward case:** The main idea is to prove that the set of minimal elements of the n-dim language L is a recognizable set. To show that, consider the set $M \subseteq L$ consisting of n-dim word $(w_1, \ldots, w_n)$ accepted by a run $q_0 q_1 \cdots q_m$ of the n-tape automaton T such that: $q_i \neq q_j$ for every $i \neq j$, $q_0 \in I$, and $q_m \in F$. It is clear that:

   Lemma 4. *$\forall (w_1, \ldots, w_n) \in L$, $\exists (w'_1, \ldots, w'_n) \in M$ such that $(w'_1, \ldots, w'_n) \preceq (w_1, \ldots, w_n)$.*

   Hence, we have the following fact $M\uparrow = L\uparrow$.
   On the other hand, the set M contains at most $((|\Sigma_1|+1)\times\cdots\times(|\Sigma_n|+1))^{|Q|}$ elements and we have that $M = \bigcup_{(w_1,\ldots,w_n)\in M}\{w_1\}\times\{w_2\}\times\cdots\times\{w_n\}$, which implies that $M\uparrow = L\uparrow = \bigcup_{(w_1,\ldots,w_n)\in M}\{w_1\}\uparrow\times\{w_2\}\uparrow\times\cdots\times\{w_n\}\uparrow$. Notice that each regular language $w_i\uparrow$ over the alphabet Σ_i can be accepted by a finite state automaton with at most $|Q|$.

2. **Downward case:** Let $\mathbf{w} = (a_0^0, \ldots, a_0^n)(a_1^0, \ldots, a_1^n)\cdots(a_m^0, \ldots, a_m^n)$ be a n-dim word over the alphabets $\Sigma_1, \ldots, \Sigma_n$, then, $\forall i \in \{1, \ldots, n\}$, we define $\alpha_i(w) = \{a_0^i, \ldots, a_m^i\} \subseteq \Sigma_i$ as the set of symbols of Σ_i which appear in the n-dim word $\mathbf{w}$. This definition is generalized straightforwardly to the set of n-dim languages as follows: $\alpha_i(L') = \bigcup_{w\in L'}\alpha_i(w)$.
   To prove that the downward of an effectively rational set is a recognizable one, we need the following fact:

   Fact 1. *Let L' be a n-dim language. Then, $(L')^*\downarrow = (\alpha_1(L'))^*\times\cdots\times(\alpha_n(L'))^*$.*

   Therefore, we need to identify the set of strongly connected compounds of Q that induces the set of language in the previous form. For that, let $C_1, C_2, \ldots, C_k \subseteq Q$ be a sequence of sets such that: (1) $Q = \bigcup_{i=1}^k C_I$, (2) $C_i \cap C_j = \emptyset$ for every $i \neq j$, and (3) for every $i \in \{1, \ldots, k\}$ and for every $q, q' \in C_I$, there exists a run $q_0 q_1 \cdots q_m$ of T such that $q_0 = q$ and $q_m = q'$.
   Then, let $S_1, \ldots, S_k$ be a sequence of n-dim langauges such that for every $i \in \{1, \ldots, k\}$ the set S_i contains the n-dim word $(a_1, \ldots, a_n) \in \Sigma_\varepsilon$ iff there exists a transition $(q, (a_1, \ldots, a_n), q') \in \delta$ such that $q, q' \in C_i$.
   Consider now the set $R \subseteq Q^* \times (\Sigma_1^* \times \cdots \times \Sigma_n^*)$ of *minimal elements* of T such that $(q_0 q_1 \cdots q_m, \mathbf{a_0 a_1 \cdots a_{m-1}}) \in R$, with $\mathbf{a_i} \in \Sigma_\varepsilon$ for every $i \in \{0, \ldots, m-1\}$, iff $q_0 q_1 \cdots q_m$ is a run of T that accepts the n-dim word $\mathbf{a_0 a_1 \cdots a_{m-1}}$ where $q_i \neq q_j$ for every $i \neq j$, $(q_i, \mathbf{a_i}, q_{i+1}) \in \delta$ for every $i \in \{0, \ldots, m-1\}$, $q_0 \in I$, and $q_m \in F$.
   Then, for every $(q_0 q_1 \cdots q_m, \mathbf{a_0 a_1 \cdots a_{m-1}}) \in R$, we can compute the recognizable set $L(q_0 q_1 \cdots q_m, \mathbf{a_0 a_1 \cdots a_{m-1}})$ such that $L(q_0 q_1 \cdots q_m, \mathbf{a_0 a_1 \cdots a_{m-1}}) = L(A_1) \times L(A_2) \times \cdots \times L(A_n)$ where for every $i \in \{1, \ldots, n\}$, $A_i = (Q, \Sigma_i, \delta_i, \{q_0\}, \{q_m\})$ is a finite state automaton over the alphabet Σ_i defined as follows: (1) for every $j \in \{0, \ldots, m-1\}$, $(q_j, \alpha_i(\mathbf{a_j}), q_{j+1})$ and $(q_j, \varepsilon, q_{j+1})$ are in δ_i, and (2) for every $j \in \{0, \ldots, m\}$, for every $\sigma \in S_l$ such that $q_j \in C_l$, $(q_j, \alpha_i(\sigma), q_j) \in \delta_i$. Hence, we can show that:

Lemma 5. $L{\downarrow} = \bigcup_{(q_0q_1\cdots q_m, \mathbf{a_0a_1}\cdots\mathbf{a_{m-1}})\in R} L(q_0q_1\cdots q_m, \mathbf{a_0a_1}\cdots\mathbf{a_{m-1}})$.

Notice that the set R contains at most $((|\Sigma_1|+1)\times\cdots\times(|\Sigma_n|+1))^{|Q|}$ elements.

□

B Context-Free Grammar in Chomsky Normal Form

We recall hereafter the definition of context free grammar in Chomesky normal form and some well known facts.

A context free grammar (CFG) G in Chomsky normal form is a 4-tuple $G = (N,\Sigma,R,S)$ where N is a finite set of non terminal symbols, Σ is a finite set of terminal symbols, disjoint with V, $S \in N$ is the start variable, and R is a finite set of production rules of the form: (1) $A \rightarrow_G BC$, or (2) $A \rightarrow_G \sigma$ or (3) $S \rightarrow_G \varepsilon$, where $A,B,C \in N$, $\sigma \in \Sigma$ a terminal symbol, and neither B nor C may be the start symbol.

Given two strings $u, v \in (N\cup\Sigma)^*$, we say that u yields to v, written as $u \Longrightarrow_G v$, if $\exists(\alpha \rightarrow_G \beta) \in R$ such that $u = u_1\alpha u_2$ and $v = u_1\beta u_2$ for some $u_1,u_2 \in (N\cup\Sigma)^*$. We denote by $\Longrightarrow_G^*$ the transitive and reflexive closure of $\Longrightarrow_G$. Finally, let $L(G) = \{w \in \Sigma^* : S \Longrightarrow_G^* w\}$ be the language generated by a context free grammar G in Chomsky normal form.

Notice that for any context free language L (effectively described by an CFG), then, we can construct an CFG G in Chomesky normal form that generates the same language, i.e. $L = L(G)$.

Now, let us recall a well known property satisfied by context-free languages.

Proposition 8. *(The pumping lemma for context-free language) Let $G = (N,\Sigma,R,S)$ be a context free grammar in Chomsky normal form. Then, for any string $w \in L(G)$ such that $|w| > 2^{|N|}$, we can write $w = rstuv$, for some $r,s,t,u,v \in \Sigma^*$, subject to the following conditions: (1) $|stu| \leq 2^{|N|}$, (2) $su \neq \varepsilon$, and (3) $\forall j \in \mathbb{N}$, $rs^jtu^jv \in L(G)$.*

Moreover, given a context free-language L, the emptiness problem, i.e. whether L is empty, and membership problem, i.e. whether a given string $w \in \Sigma^*$ is in L, are decidable.

C Proof of Proposition 2

Proposition 2 *Let $\mathcal{P} = (P,\Sigma,\Gamma,\Delta)$ be an LPDS and $p, p' \in P\Gamma^*$ be two configurations. It is possible to construct, in exponential (resp. double exponential) time in $(|P|+|\Sigma|+|\Gamma|)$, a finite state automaton $A = (Q,\Sigma,\delta,I,F)$ such that $L(A) = Traces_{\mathcal{P}}(p,p'){\downarrow}$ (resp. $L(A) = Traces_{\mathcal{P}}(p,p'){\uparrow}$), where in the worst case, $|Q|$ is exponential (resp. double exponential) in $(|P|+|\Sigma|+|\Gamma|)$.*

Proof:

Downward Closure. The result concerning the downward closure has been established by Bruno Courcelle in [15]. We recall hereafter the proof and give some hints to compute the complexity.

Let $G = (N,\Sigma,R,S)$ be a context free language in Chomsky normal form such that $L = L(G)$. For every $A \in N$, we let $\mathbf{L}(G,A)$ denote the language generated by the context free grammar (N,Σ,R,A). Moreover, we assume w.l.o.g that $\mathbf{L}(G,A) \neq \emptyset$ for every nonterminal $A \in N$. For every language L', we let:

$\alpha(L')$ = the set of letters (terminal symbols) occurring in L' (hence $\alpha(L') = \emptyset$ iff $L' \subseteq \{\varepsilon\}$).

In addition, for every $L_1, L_2 \subseteq \Sigma^*$, we have the following facts:

1. $\alpha(L_1 \cup L_2) = \alpha(L_1 L_2) = \alpha(L_1) \cup \alpha(L_2)$;
2. $(L_1 \cup L_2){\downarrow} = L_1{\downarrow} \cup L_2{\downarrow}$;
3. $(L_1 L_2){\downarrow} = L_1{\downarrow}\, L_2{\downarrow}$.

For every $m \in (N \cup \Sigma)^*$, we let $\mathbf{L}(G,m)$ denote the language generated by G from m taken as starting string.

For every $A, B \in N$, we let:

1. $B <_1 A$ iff $A \Longrightarrow_G^+ mBm'$ for some $m, m' \in (N \cup \Sigma)^*$,
2. $B <_2 A$ iff $A \Longrightarrow_G^+ mBm'Bm"$ for some $m, m', m" \in (N \cup \Sigma)^*$,
3. $B =_1 A$ iff $B <_1 A <_1 B$.

Fact 2. *if $A <_2 A$, then $\boldsymbol{L}(G,A){\downarrow} = (\alpha(\boldsymbol{L}(G,A)))^*$.*

Fact 3. *if $A =_1 B$, then $\boldsymbol{L}(G,A){\downarrow} = \boldsymbol{L}(G,B){\downarrow}$.*

We now explain how $\mathbf{L}(G,A){\downarrow}$ can be computed for any $A \in N$. If $A <_2 A$ (which is decidable), then Fact 2 yields the answer.

Otherwise, we compute $\mathbf{L}(G,A){\downarrow}$ in terms of the languages $\mathbf{L}(G,B){\downarrow}$ for $B <_1 A$ and $B \neq_1 A$, that we may assume to be given by previously computed finite state automata.

Let $p : A \rightarrow_G m$ be a production rule. We let $R_0(p)$, $R_1(p)$, and $R_2(p)$ be words defined as follows:

- **First case:** m does not contain any nonterminal B such that $B \neq_1 A$. We let $R_0(p) = m$, and $R_1(p)$, $R_2(p)$ be the empty word.
- **Second case:** m contains a unique nonterminal B with $B =_1 A$ and $m = m'Bm"$. We let $R_1(p) = m'$ and $R_2(p) = m"$. (Since we assume that $A \not<_2 A$, the word m cannot contain two occurrences of nonterminals equivalent (i.e. $=_1$) to A.) In this case $R_0(p)$ is the empty word.

Fact 4. *For every $A \in N$ such that $A \not<_2 A$, we have:*

$$\boldsymbol{L}(G,A){\downarrow} = (\cup\alpha(\boldsymbol{L}(G,R_1(p))))^*(\cup\boldsymbol{L}(G,R_0(p)){\downarrow})(\cup\alpha(\boldsymbol{L}(G,R_2(p))))^*$$

where the unions extend to all production rules p with lefthand side B such that $B =_1 A$.

Notice that for every $A \in N$ and production rule $p : A \rightarrow_G m$, we have $R_1(p) \in N$, or $R_2(p) \in N$, or $R_0(p) \in (NN \cup \Sigma \cup \{\varepsilon\})$, since the context free grammar G is in Chomesky normal form.

Since the words $R_0(p)$, $R_1(p)$, and $R_2(p)$ contain only nonterminals C with $C <_1 A$ and $C \neq_1 A$, we have achieved our goal.

Complexity results: We give hereafter some indications about the complexity of the previous algorithm.

1. For every $A, B \in N$, checking whether $B <_1 A$ or $B <_2 A$ can be reduced to the reachability problem for context free language which is decidable in polynomial time in $(|N| + |\Sigma|)$,

2. For every $A \in N$, a finite state automaton (with only two states) that recognizes $(\cup\alpha(\mathbf{L}(G,R_1(p))))^*$ (resp. $(\cup\alpha(\mathbf{L}(G,R_2(p))))^*$) can be constructed in polynomial time in $(|N|+|\Sigma|)$,
3. For every $A \in N$ such that $\nexists B \in N$ where $B <_1 A$, a finite state automaton, with only two states, that recognizes $\mathbf{L}(G,A)$ can be computed in polynomial time in $(|N|+|\Sigma|)$,
4. For every $A \in N$, a finite state automaton, that recognizes $(\cup\mathbf{L}(G,R_0(p))\downarrow)$, will have at most $(\sum_{(B,C<_1 A,B,C\neq_1 A)} k_B k_C + 2)$ states, where k_B (resp. k_C) is the number of states of the automaton that recognizes $\mathbf{L}(G,B)$ (resp. $\mathbf{L}(G,C)$).

Hence, a finite state automaton D that recognizes the downward closure of the context free language L (effectively described by a context free grammar G in Chomesky normal form) can be constructed in exponential time in $(|N|+|\Sigma|)$. Moreover, the number of states of D is (at most) exponential in $(|N|+|\Sigma|)$.

Upward closure. For the upward closure, the result follows from the fact that it is possible to construct a finite set M' that contains the set $M = \{\sigma \in L : \nexists \sigma' \in L.\, \sigma' \prec \sigma\}$ of the minimal elements of $L\uparrow$. This can be done using the following lemma which is an immediate consequence of the standard pumping lemma for context-free languages (see proposition 8).

Lemma 6. *For every context free language L, it is possible to define a bound $m \in \mathbb{N}$ s.t. $\forall \sigma \in L$, if $|\sigma| \geq m$ then $\exists \sigma' \in L$ s.t. $|\sigma'| \leq m$ and $\sigma' \preceq \sigma$.*

Let us consider the finite set $M' = \{\sigma \in L : |\sigma| \leq m\}$, where m is the upper bound given by lemma.6. Then, we can prove the following fact:

Lemma 7. $M \subseteq M'$.

Proof: (Proof by Contradiction.) Assume to the contrary, there is a word $\sigma \in M$ such that $|\sigma| > m$. Thanks to lemma.6, there exists a word $\sigma' \in L$ such that $\sigma' \preceq \sigma$ and $|\sigma'| \leq m$. This fact leaves us with a contradiction from the definition of the set M. Hence, σ is necessarily in M'. $\square$

Consider now the finite state automaton $\mathcal{A} = (Q,\Sigma,\delta,I,F)$ that recognizes exactly the set $M' = \{\sigma \in L : |\sigma| \leq m\}$. Notice that such automaton $\mathcal{A}$ is effectively constructible due to the fact that membership problem is decidable for context-free languages and that M is finite. Let $\mathcal{A}' = (Q,\Sigma,\delta',I,F)$ be the finite state automaton such that $\delta' = \delta \cup \{(q,a,q) : \forall (q,a) \in Q \times \Sigma\}$. Then, we can easily show that $L(\mathcal{A}') = M'\uparrow$. Indeed, we have that:

Lemma 8. $M\uparrow = M'\uparrow = L\uparrow$.

Proof: It is clear that $M\uparrow \subseteq M'\uparrow \subseteq L\uparrow$ since $M \subseteq M' \subseteq L$. On the other hand, for any word σ in $L\uparrow$, there exists a $\sigma' \in L$ such that $\sigma' \preceq \sigma$ and $\nexists \beta \in L.\ \ \beta \prec \sigma'$. Notice that such a word σ' always exists and that $\sigma' \in M$. Since $\sigma' \preceq \sigma$, we have that σ is in $M\uparrow$. Then, we obtain that $L\uparrow \subseteq M\uparrow$. Thus implies that $M\uparrow = M'\uparrow = L\uparrow$.

$\square$

This completes the proof that the set $L\uparrow$ is regular since there exists a finite state automaton $\mathcal{A}'$ such that $L(A)' = M\uparrow = M'\uparrow = L\uparrow$.

Complexity results. We give hereafter some indications about the complexity of the previous algorithm. In fact, we know that the number of words that we need to check is double exponential in $(|N|+|\Sigma|)$ (the size of the context free grammar G in Chomesky normal form that generates L). Moreover, for every word $w \in \sigma^*$, we need to check if $w \in L$ which is decidable for context-free languages in polynomial time in the size of G. Hence, a finite state automaton D, that recognized the upward closure of the context free language, can be constructed in time double exponential in $(|N|+|\Sigma|)$ (the size of G). Indeed, the number of states of D will be double exponential in the size of G which is polynomial in $(|P|+|\Sigma|+|\Gamma|)$. $\square$

D Proof of Theorem 1

Theorem 1 *It is possible to construct a 2-tape automaton $\widetilde{T}(\mathcal{P},p)$ (resp. $\widehat{T}(\mathcal{P},p)$), with a number of states that is at most exponential (resp. doubly exponential) in $|P|+|\Sigma|+|\Gamma|$, such that $L(\widetilde{T}(\mathcal{P},p)) = D(\mathcal{P},p)$ (resp. $\widehat{T}(\mathcal{P},p) = U(\mathcal{P},p)$).*

Proof:

In the following, we provide the construction of 2-tape automaton $\widetilde{T}(\mathcal{P},p)$ which accepts the set $D(\mathcal{P},p)$ (the one for $U(\mathcal{P},p)$ is similar). For that, we define a 2-tape automaton $T = (Q,\Gamma,\Sigma,\delta,I,F)$ with $I = P$, such that (w,σ) is accepted by T from the state p' iff $(p'w,\sigma) \in L(\widetilde{T}(\mathcal{P},p))$, i.e. $L(\widetilde{T}(\mathcal{P},p)) = \bigcup_{p'\in P}(p',\varepsilon)L(T^{p'})$.

The idea behind the construction of T is the following lemma which can be proved by induction on stack's length:

Lemma 9. *Let $p \overset{\sigma'}{\Longrightarrow}_{\mathcal{P}} p'\gamma_m\cdots\gamma_1$ be a computation of $\mathcal{P}$. Then, $\exists p_0,p_0',p_1,p_1',\ldots,p_m \in P$, $\exists \sigma_0',\ldots,\sigma_m' \in \Sigma^*$, and $\exists a_0',\ldots,a_{m-1}' \in \Sigma\cup\{\varepsilon\}$ such that:*

1. *$\forall i \in \{1,\ldots,m-1\}$, $p_i \overset{\sigma_i'}{\Longrightarrow}_{\mathcal{P}} p_i'$ (i.e. $\sigma_i' \in Traces_{\mathcal{P}}(p_i,p_i')$),*
2. *$\forall i \in \{1,\ldots,m-1\}$, $p_i' \overset{a_i'}{\Longrightarrow}_{\mathcal{P}} p_{i+1}\gamma_{i+1}$,*
3. *$p_m \overset{\sigma_m'}{\Longrightarrow}_{\mathcal{P}} p'$,*
4. *$\sigma' = \sigma_0'a_0'\sigma_1'a_1'\cdots\sigma_{m-1}'a_{m-1}'\sigma_m'$,*
5. *$p_0 = p$.*

Notice that $p_i' \overset{a_i'}{\Longrightarrow}_{\mathcal{P}} p_{i+1}\gamma_{i+1}$ corresponds to a rule that pushes the stack symbol γ_{i+1} which will never be removed (i.e. $(\langle p_i',\varepsilon\rangle \overset{a_i'}{\Longrightarrow} \langle p_{i+1},\gamma_{i+1}\rangle) \in \Delta$).

Now, let σ be a word over Σ such that $\sigma \preceq \sigma'$. Then, $\exists \sigma_0,\ldots,\sigma_m' \in \Sigma^*$ and $\exists a_0,\ldots,a_{m-1} \in (\Sigma\cup\{\varepsilon\})$ such that: (1) $\forall i \in \{0,\ldots,m\}$, $\sigma_i \preceq \sigma_i'$ (i.e. $\sigma_i \in Traces_{\mathcal{P}}(p_i,p_i')\downarrow$), and (2) $\forall i \in \{0,\ldots,m-1\}$, $a_i \preceq a_i'$. Thus, we have:

$$(\gamma_m\cdots\gamma_1,\sigma^R) = (\varepsilon,\sigma_m^R)(\gamma_m,a_{m-1})(\varepsilon,\sigma_{m-1}^R)(\gamma_{m-1},a_{m-2})\cdots(\gamma_1,a_0)(\varepsilon,\sigma_0^R)$$

The 2-tape automaton T must recognize the pair $(\gamma_m\cdots\gamma_1,\sigma^R)$ given by the previous equation. Therefore to construct T we proceed as follows: (1) For each pair $(q,q') \in P\times P$, we construct the automaton $\mathcal{A}_{(q,q')} = (A_{(q,q')},\Sigma,\delta_{(q,q')},I_{(q,q')},F_{(q,q')})$ that recognizes the set $Traces_{\mathcal{P}}(q,q')\downarrow$. This automaton is effectively constructible due to

Proposition 2. (2) Then, we consider automata recognizing the mirror (reverse) language of each automata $\mathcal{A}_{(q,q')}$ and extend them to 2-tape words by adding ε on the first tape on all transitions. (3)Finally, we connect these automata using transitions of the forms $(p_{i+1},(\gamma_{i+1},a'_i),p'_i)$ and $(p_{i+1},(\gamma_{i+1},\varepsilon),p'_i)$, for every rule $\langle p'_i,\varepsilon\rangle \xRightarrow{a'_i} \langle p_{i+1},\gamma_{i+1}\rangle$ in Δ.

Formally, we define the 2-tape automaton T as follows: $T=(Q,\Gamma,\Sigma,\delta,I,F)$, where $Q=\bigcup_{q,q'\in P} A_{(q,q')}\cup P$ is a finite set of states, $I=P$, $F=\{p\}$, and δ is the following set of transitions:

- $(s',(\varepsilon,a),s))$ is in δ if $\exists(s,a,s')\in\delta_{(q,q')}$ for every $q,q'\in P$. (*these correspond to the transitions of the $\mathcal{A}_{(q,q')}$s read in the reverse direction).*
- $(q',(\gamma,\varepsilon),q)$ and $(q',(\gamma,a),q)$ are in δ if $\exists(\langle q,\varepsilon\rangle \overset{a}{\hookrightarrow} \langle q',\gamma\rangle)\in\Delta$.
- $(q',(\varepsilon,\varepsilon),s)$ is in δ for every $q'\in P$ and $s\in F_{(q,q')}$ with $q\in P$. (the initial states are related with ε-transitions to the final states of the $\mathcal{A}_{(q,q')}$s since we are interested in σ^R and so we read the $\mathcal{A}_{(q,q')}$s in the reverse direction),
- $(s,(\varepsilon,\varepsilon),q)$ for every $q\in P$ and $s\in I_{(q,q')}$ with $q'\in P$. (Similarly, the final states are related with ε-transitions to the initial states of the $\mathcal{A}_{(q,q')}$s).

We have the following facts:

Lemma 10. *If $\exists\sigma'\in\Sigma^*$ such that $q \xRightarrow{\sigma'}_{\mathcal{P}} q'$ with $q,q'\in P$, then $\forall\sigma\in\Sigma^*$ such that $\sigma\preceq\sigma'$ we have $(q',(\varepsilon,\sigma^R),q)\in\delta^*$. Moreover, if $\exists\sigma\in\Sigma^*$ such that $(q',(\varepsilon,\sigma^R),q)\in\delta^*$, then, $\exists\sigma'\in\Sigma^*$ such that $q \xRightarrow{\sigma'}_{\mathcal{P}} q'$ and $\sigma\preceq\sigma'$.*

Proof:

First, let us assume that $\exists\sigma'\in\Sigma^*$ such that $q \xRightarrow{\sigma'}_{\mathcal{P}} q'$. By definition of $\mathcal{A}_{(q,q')}$, we have that $\forall\sigma\in\Sigma^*$ such that $\sigma\preceq\sigma'$, there exist $s\in I$ and $s'\in F$ such that $(s,\sigma,s')\in\delta^*_{(q,q')}$ which implies that $(s',(\varepsilon,\sigma^R),s)\in\delta^*$. Therefore, $(q',(\varepsilon,\sigma^R),q)$ is in δ^* since $(q',(\varepsilon,\varepsilon),s')\in\delta$ and $(s,(\varepsilon,\varepsilon),q)\in\delta$.

On the other hand, suppose that there exist $\sigma\in\Sigma^*$ such that $(q',(\varepsilon,\sigma^R),q)$ is in δ^*. Then, there exists $q_0,\ldots,q_m\in P$ and $\forall i\in\{0,\ldots,m-1\}$ a pair $(s_i,s'_i)\in I_{(q_i,q_{i+1})}\times F_{(q_i,q_{i+1})}$ such that $(q',(\varepsilon,\sigma^R),q)\in\delta^*$ can be decomposed as follows: $(q_{i+1},(\varepsilon,\varepsilon),s'_i)\in\delta$, $(s'_i,(\varepsilon,\sigma_i^R),s_i)\in\delta^*$, and $(s_i,(\varepsilon,\varepsilon),q_i)\in\delta$ for every $i\in\{0,\ldots,m-1\}$ where $q_m=q'$, $q_0=q$, and $\sigma=\sigma_0\sigma_1\cdots\sigma_{m-1}$. Notice that $\sigma_i\in Traces_{\mathcal{P}}(q_i,q_{i+1})\!\downarrow$ for every $i\in\{0,\ldots,m-1\}$ since $(s_i,s'_i)\in I_{(q_i,q_{i+1})}\times F_{(q_i,q_{i+1})}$. Thus implies that there exists $\sigma'_i\in\Sigma^*$ such that $\sigma_i\preceq\sigma'_i$ and $q_i \xRightarrow{\sigma'_i}_{\mathcal{P}} q_{i+1}$ for every $i\in\{0,\ldots,m-1\}$. Hence, we obtain that $q_0 \xRightarrow{\sigma'}_{\mathcal{P}} q_m$ with $\sigma'=\sigma'_0\sigma'_1\cdots\sigma'_{m-1}$ and $\sigma\preceq\sigma'$.

□

Lemma 11. *If (w,σ^R) is accepted from p' by the 2-tape automaton T, then, there exists a word $\sigma'\in\Sigma^*$ such that $\sigma\preceq\sigma'$ and $p \xRightarrow{\sigma'}_{\mathcal{P}} p'w$. Moreover, if there exists a word $\sigma'\in\Sigma^*$ such that $p \xRightarrow{\sigma'}_{\mathcal{P}} p'w$, then, $\forall\sigma\in\Sigma^*$ such that $\sigma\preceq\sigma'$, the vector (w,σ^R) is accepted from p' by T.*

Proof:

Assume that there exists a word $\sigma' \in \Sigma^*$ such that $p \overset{\sigma'}{\Longrightarrow}_{\mathcal{P}} p'\gamma_m \cdots \gamma_1$, then, using lemma.9, there exist $p_0, p'_0, p_1, p'_1, \ldots, p_m \in P$, with $p_0 = p$, $\sigma'_0, \ldots, \sigma'_m \in \Sigma^*$, and $a'_0, \ldots, a'_{m-1} \in \Sigma \cup \{\varepsilon\}$ such that: (1) $\sigma' = \sigma'_0 a'_0 \cdots a'_{m-1}\sigma'_m$, and (2) $p \overset{\sigma'}{\Longrightarrow}_{\mathcal{P}} p'\gamma_m \cdots \gamma_1$ can be decomposed as follows: $p_i \overset{\sigma'_i}{\Longrightarrow}_{\mathcal{P}} p'_i$ and $p'_i \overset{a'_i}{\Longrightarrow}_{\mathcal{P}} p_{i+1}\gamma_{i+1}$ for every $i \in \{0, \ldots, m-1\}$, and $p_m \overset{\sigma'_m}{\Longrightarrow}_{\mathcal{P}} p'$.

Now, we can apply lemma.10 to each computation $p_m \overset{\sigma'_m}{\Longrightarrow}_{\mathcal{P}} p'$ and $p_i \overset{\sigma'_i}{\Longrightarrow}_{\mathcal{P}} p'_i$, with $i \in \{0, \ldots, m-1\}$, to prove that $\forall \sigma_i, \sigma_m \in \Sigma^*$ such that $\sigma_i \preceq \sigma'_i$ and $\sigma_m \preceq \sigma'_m$, we have $(p'_i, (\varepsilon, \sigma_i^R), p_i) \in \delta^*$ and $(p', (\varepsilon, \sigma_m^R), p_m) \in \delta^*$. Moreover, we remark that for every index $i \in \{0, \ldots, m-1\}$ and $\forall a_i \in \Sigma \cup \{\varepsilon\}$ such that $a_i \preceq a'_i$, we have $(p_{i+1}, (\gamma_{i+1}, a_i), p'_i)$ is in δ. Hence, $\forall \sigma \preceq \sigma'$, we obtain that $(p', (\gamma_m \cdots \gamma_1, \sigma^R), p) \in \delta^*$ which implies that $(\gamma_m \cdots \gamma_1, \sigma^R)$ is accepted from p'.

On the other hand, suppose that $(\gamma_m \cdots \gamma_1, \sigma^R)$ is accepted from the control state p' by the 2-tape automaton T. Then, there exist $p_0, p'_0, p_1, p'_1, \ldots, p_m \in P$, $\sigma_0, \ldots, \sigma_m \in \Sigma^*$, and $a_0, \ldots, a_{m-1} \in \Sigma \cup \{\varepsilon\}$ such that $(p', (\gamma_m \cdots \gamma_1, \sigma^R), p) \in \delta^*$ can be decomposed as follows: $(p', (\varepsilon, \sigma_m^R), p_m) \in \delta^*$ and $\forall i \in \{0, \ldots, m-1\}$. $(p_{i+1}, (\gamma_{i+1}, a_i), p'_i) \in \delta$ and $(p'_i, (\varepsilon, \sigma_i^R), p_i) \in \delta^*$ with $p_0 = p$ and $\sigma = \sigma_0 a_0 \cdots a_{m-1}\sigma_m$. Now, we can apply lemma.10 to $(p', (\varepsilon, \sigma_m^R), p_m) \in \delta^*$ and $(p'_i, (\varepsilon, \sigma_i^R), p_i) \in \delta^*$ for every index $i \in \{0, \ldots, m-1\}$ to show that there exist $\sigma'_m \in \Sigma^*$ and $\sigma'_i \in \Sigma^*$, for every $i \in \{0, \ldots, m-1\}$, such that $\sigma_m \preceq \sigma'_m$ and $\sigma_i \preceq \sigma'_i$ with $p_m \overset{\sigma'_m}{\Longrightarrow}_{\mathcal{P}} p'$ and $p_i \overset{\sigma'_i}{\Longrightarrow}_{\mathcal{P}} p'_i$. Moreover, we remark that, for every $i \in \{0, \ldots, m-1\}$, there exists $a'_i \in \Sigma$ such that $a_i \preceq a'_i$ with $p'_i \overset{a'_i}{\Longrightarrow}_{\mathcal{P}} p_{i+1}\gamma_{i+1}$. Combining the two previous results, we can prove that there exists $\sigma' = \sigma'_0 a'_0 \cdots a'_{m-1}\sigma'_m$ such that $\sigma \preceq \sigma'$ and $p \overset{\sigma'}{\Longrightarrow}_{\mathcal{P}} p'\gamma_m \cdots \gamma_1$.

$\square$

Hence, we have showed that the $D(\mathcal{P}, p)$ is rational and effectively constructible in exponential time in $|P| + |\Sigma| + |\Gamma|$. A similar proof can be used to $U(\mathcal{P}, p)$. $\square$

E Proof of Proposition 3

Proposition 3 *For every APN$_{\mathsf{obs}}$ $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$, the reachability problem can be reduced to the same problems for a linear APN$_{\mathsf{obs}}$ $N' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, R')$ such that $\forall i \in \{1, \ldots, n\}$, $P'_i = \prod_{j=i}^{n} P_j$.*

Proof (Sketch):

Let $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ be an APN$_{\mathsf{obs}}$. N can be *"simulated"* by the linear APN$_{\mathsf{obs}}$ N' constructed as follows: $N' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, R')$ where $R' = \{(i, i+1) \mid \forall i \in \{1, \ldots, n-1\}\}$, and for every $i \in \{1, \ldots, n\}$, $\mathcal{P}'_i = (P'_i, \Gamma_i, \Delta'_i)$ is a constrained pushdown system that simulates $\mathcal{P}_i$ where $P'_i = P_i \times \cdots \times P_n$ and Δ'_i is a finite set of rules defined as follows:

1. $\{(s_{i+1}, \ldots, s_n)\}$: $\langle (p_i, \ldots, p_n), \varepsilon \rangle \hookrightarrow \langle (p_i, s_{i+1}, \ldots, s_n), \varepsilon \rangle \in \Delta'_i$ for every $(s_{i+1}, \ldots, s_n) \in P_{i+1} \times \cdots \times P_n$. *($\mathcal{P}'_i$ updates its information about the observed processes).*

2. P'_{i+1} : $\langle (p, p_{i+1}, \ldots, p_n), u\rangle \hookrightarrow \langle (p', p_{i+1}, \ldots, p_n), u'\rangle$ is in Δ'_i iff $(\phi : \langle p,u\rangle \hookrightarrow \langle p', u'\rangle) \in \Delta_i$ such that $\forall j > i$, if $(i, j) \in R$, then $p_j \in \phi$. *(if the states corresponding to the processes $\mathcal{P}_{i+1}, \ldots, \mathcal{P}_n$ that are encoded in the state of $\mathcal{P}'_i$ satisfy the constraints of the rule of Δ_i, the transition is applied).*

We can show that:

Lemma 12. *There exists a computation of the network N: $(p_1 w_1, \ldots, p_n w_n) \Longrightarrow^*_N (p'_1 w'_1, \ldots, p'_n w'_n)$ if and only if $(q_1 w_1, \ldots, q_n w_n) \Longrightarrow^*_{N'} (q'_1 w'_1, \ldots, q'_n w'_n)$ is a run of N' where for every $i \in \{1, \ldots, n\}$, $q_i = (p_i, \ldots, p_n)$ and $q'_i = (p'_i, p'_{i+1}, \ldots, p'_n)$.*

$\square$

F Proof of Remark 1

Remark 1 *The extension of APN$_{\mathsf{obs}}$ by allowing cycles in communication relation leads to Turing powerful models.*

Proof (Sketch):

We show that the existence of a decision procedure to the reachability problem between two configurations for the extension of APN$_{\mathsf{obs}}$ by allowing cycles communication would implies a decision procedure for the intersection problem of two context-free languages.

Let L_1 and L_2 be two context free languages. Let $\mathcal{P}_1 = (P_1, \Sigma_1, \Gamma_1, \Delta_1)$ and $\mathcal{P}_2 = (P_2, \Sigma_2, \Gamma_2, \Delta_2)$ be two LPDSs such that $L_1 = Traces_{\mathcal{P}_1}(p_1, p'_1)$ and $L_2 = Traces_{\mathcal{P}_2}(p_2, p'_2)$.

Let $N = (\mathcal{P}'_1, \mathcal{P}'_2, R)$ be an Observation Relation Pushdown Network where $R = \{(1,2), (2,1)\}$ is a binary relation (that contains one cycle) and for every $i \in \{1,2\}$, we have:

1. $\mathcal{P}'_1 = (P'_1, \Gamma_1, \Delta'_1)$ is a constrained pushdown system where:
   - $P'_1 = P_1 \cup (P_1 \times \Sigma)$ is a finite set of states,
   - Γ_1 is a finite set of stack alphabet,
   - Δ'_1 is a finite set of transition rules defined as follows:
     (a) P_2 : $\langle p,u\rangle \hookrightarrow \langle p', u'\rangle \in \Delta'_1$ iff $\langle p,u\rangle \overset{\varepsilon}{\hookrightarrow} \langle p', u'\rangle \in \Delta_1$ which means that whenever $\mathcal{P}'_2$ in some state in P_2, process $\mathcal{P}'_1$ can execute its ε-transition rules,
     (b) P_2 : $\langle p, \varepsilon\rangle \hookrightarrow \langle (p,a), \varepsilon\rangle \in \Delta'_1$ for every $p \in P_1$ and $a \in \Sigma$ which means that the process $\mathcal{P}'_1$ chooses to execute a transition rule of the LPDS $\mathcal{P}_1$ labeled by a,
     (c) $P_2 \times \{a\}$: $\langle (p,a), u\rangle \hookrightarrow \langle p', u'\rangle \in \Delta'_1$ for every $p \in P_1$ and $a \in \Sigma$ iff $\langle p,u\rangle \overset{a}{\hookrightarrow} \langle p', u'\rangle \in \Delta_1$ which means that the process $\mathcal{P}'_1$ executes a transition rule of the LPDS $\mathcal{P}_1$ labeled by a if $\mathcal{P}'_2$ has chosen also to execute a transition labeled by a.
2. $\mathcal{P}'_2 = (P'_2, \Gamma_2, \Delta'_2)$ is a constrained pushdown system where:
   - $P'_2 = P_2 \cup (P_2 \times \Sigma)$ is a finite set of states,
   - Γ_2 is a finite set of stack alphabet,
   - Δ'_2 is a finite set of transition rules defined as follows:

(a) $P_1 : \langle q,u\rangle \hookrightarrow \langle q',u'\rangle \in \Delta_2'$ iff $\langle q,u\rangle \overset{\varepsilon}{\hookrightarrow} \langle q',u'\rangle \in \Delta_2$ which means that whenever $\mathcal{P}_1'$ in some state in P_1, process $\mathcal{P}_2'$ can execute its ε-transition rules,

(b) $P_1 \times \{a\} : \langle q,\varepsilon\rangle \hookrightarrow \langle (q,a),\varepsilon\rangle \in \Delta_2'$ for every $q \in P_2$ and $a \in \Sigma$ which means that if process $\mathcal{P}_1'$ has chosen to execute a transition rule of the LPDS $\mathcal{P}_1$ labeled by a, then $\mathcal{P}_2'$ must choose to execute a transition rule of $\mathcal{P}_2$ labeled by a,

(c) $P_1 : \langle (q,a),u\rangle \hookrightarrow \langle q',u'\rangle \in \Delta_2'$ for every $q \in P_2$ and $a \in \Sigma$ iff $\langle q,u\rangle \overset{a}{\hookrightarrow} \langle q',u'\rangle \in \Delta_1$ which means that the process $\mathcal{P}_2'$ executes a transition rule of the LPDS $\mathcal{P}_2$ labeled by a whenever $\mathcal{P}_1'$ has executed a transition rule labeled by a.

Hence, we have:

Lemma 13. *$(p_1', p_2') \in post_N^*((p_1,p_2))$ iff $L_1 \cap L_2 \neq \emptyset$.*

□

G Proof of Proposition 4

Proposition 4 *Given an APN$_{\mathsf{obs}}$ N of depth ≥ 1, and a configuration c of N, the set $post_N^*(c)$ is not recognizable in general.*

Proof:

Hereafter, we will give an example that if C is a recognizable set, then $post_N^*(C)$ is in general not recognizable.

Let $N = (\mathcal{P}_1, \mathcal{P}_2, R)$ be an APN$_{\mathsf{obs}}$ where $R = \{(1,2)\}$ and for every $i \in \{1,2\}$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ is a constrained pushdown system. The process $\mathcal{P}_2$ is defined as follows: (1) $P_2 = \{p_1, p_2\}$ is a finite set of states, (2) $\Gamma_2 = \{a\}$ is a finite stack alphabet, and (3) $\Delta_2 = \{\langle p_1,\varepsilon\rangle \hookrightarrow \langle p_2,a\rangle, \langle p_2,\varepsilon\rangle \hookrightarrow \langle p_1,\varepsilon\rangle\}$ is a finite set of transitions rules. On the other hand, the process $\mathcal{P}_1$ is defined as follows: (1) $P_1 = \{s_1, s_2\}$ is a finite set of states, (2) $\Gamma_1 = \{b\}$ is a finite stack alphabet, and (3) $\Delta_1 = \{\{p_1\}\langle s_1,\varepsilon\rangle \hookrightarrow \langle s_2,b\rangle, \{p_2\} : \langle s_2,\varepsilon\rangle \hookrightarrow \langle s_1,\varepsilon\rangle\}$ is a finite set of transitions rules.

We remark that $\mathcal{P}_2$ is cycling between two states, and that at each cycle, it pushes a symbol a in its stack. Then, process $\mathcal{P}_1$ tries to mimic $\mathcal{P}_2$ by pushing a symbol b in its stack at each cycle of $\mathcal{P}$ it can observe. Then, the intersection between the set of reachable configurations and the recognizable set $s_1\Gamma_1^* \times p_1\Gamma_2^*$, starting from (s_1, p_1), is $\{(s_1 b^m, p_1 a^n), : m \leq n\}$. This shows that even for networks of depth 1, the set of reachable configurations may be not recognizable. □

H Proof of Proposition 5

Proposition 5 *Given an APN$_{\mathsf{obs}}$ of depth 2, and a configuration c of N, the set $post_N^*(c)$ is not rational in general.*

Proof:

Hereafter, we will give an example that if C is a recognizable set, then $post_N^*(C)$ is in general not rational.

Let $N = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, R)$ where $R = \{(1,2),(2,3)\}$ ($\mathcal{P}_1$ observes $\mathcal{P}_2$, who observes $\mathcal{P}_3$), and for $i \in \{1,2,3\}$, processes $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ are defined as follows: $P_1 = \{p_1, p_2, p_3, p_4\}$, $P_2 = \{q_1, q_2, q_3, q_4\}$, $P_3 = \{s_1, s_2, s_3\}$, $\Gamma_1 = \{c, d\}$, $\Gamma_2 = \{b\}$, $\Gamma_3 = \{a\}$; $\Delta_3 = \{\langle s_1, \varepsilon\rangle \hookrightarrow \langle s_2, a\rangle, \langle s_2, \varepsilon\rangle \hookrightarrow \langle s_1, \varepsilon\rangle, \langle s_1, \varepsilon\rangle \hookrightarrow \langle s_3, \varepsilon\rangle)\}$, $\Delta_2 = \{\{s_1\} : \langle q_1, \varepsilon\rangle \hookrightarrow \langle q_2, b\rangle, \{s_2\} : \langle q_2, \varepsilon\rangle \hookrightarrow \langle q_1, \varepsilon\rangle, \{s_3\} : \langle q_1, \varepsilon\rangle \hookrightarrow \langle q_3, \varepsilon\rangle, \{s_3\} : \langle q_3, b\rangle \hookrightarrow \langle q_4, \varepsilon\rangle, \{s_3\} : \langle q_4, \varepsilon\rangle \hookrightarrow \langle q_3, \varepsilon\rangle\}$, and $\Delta_1 = \{\{q_1\} : \langle p_1, \varepsilon\rangle \hookrightarrow \langle p_2, c\rangle, \{q_2\} : \langle p_2, \varepsilon\rangle \hookrightarrow \langle p_1, \varepsilon\rangle, \{q_3\} : \langle p_1, \varepsilon\rangle \hookrightarrow \langle p_3, \varepsilon\rangle, \{q_3\} : \langle p_3, \varepsilon\rangle \hookrightarrow \langle p_4, d\rangle, \{q_4\} : \langle p_4, \varepsilon\rangle \hookrightarrow \langle p_3, \varepsilon\rangle\}$. Let $C = (p_1, q_1, s_1)$ be the starting language. We remark that the number of d's pushed by the process $\mathcal{P}_1$ is less or equal than the round that $\mathcal{P}_2$ can do between states q_3 and q_4 (i.e. the number of b's popped by process $\mathcal{P}_2$ will be greater or equal than the number of d's). Moreover, the number of c's that will be pushed by $\mathcal{P}_1$ is less or equal than the round that process $\mathcal{P}_2$ can do between states p_1 and p_2 (i.e. the number of c's pushed by process $\mathcal{P}_1$ will be less or equal than the number of b's pushed by process $\mathcal{P}_2$). Also, we have that the number of b's that will pushed by $\mathcal{P}_2$ is less or equal than the round that process $\mathcal{P}_3$ can do between states s_1 and s_2 (i.e. the number of b's pushed by process $\mathcal{P}_2$ will be less or equal than the number of a's pushed by process $\mathcal{P}_3$). In fact, we will have that the intersection of $post^*_N(C)$ with the recognizable set of configurations $p_3\Gamma_1^* \times q_3\{\varepsilon\} \times s_3\Gamma_3^*$ is the set of configurations $\{(p_3 d^\ell c^m, q_3, s_3 a^n) : \ell \le n, m \le n\}$ which is not rational. $\square$

I Proof of Theorem 2

Theorem 2 *Let $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ be a APN$_{\mathsf{obs}}$ of depth 1, and let $C = \{(p_1, \ldots, p_n)\}$ be a configuration. It is possible to construct a n-tape automaton T such that $L(T) = post^*_N(C)$ with double exponential states in $\sum_{i=1}^n |P_i| + |\Gamma_j|$.*

Proof:

For the sake of clarity, we consider hereafter the case where the network N contains two processes $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\mathcal{P}_1$ observes $\mathcal{P}_2$, i.e., $N = (\mathcal{P}_1, \mathcal{P}_2, \{(1,2)\})$, where $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ and $C = \{(p_1, p_2)\}$. We show later how our construction can be generalized.

To construct a 2-tape automaton T that accepts $post^*_N(C)$, we proceed as described in the paper: First, we compute a labeled pushdown system $\widehat{\mathcal{P}_1} = (P_1, P_2, \Gamma_1, \Delta_1')$ such that $p_1 \overset{\sigma}{\Longrightarrow}_{\widehat{\mathcal{P}_1}} pw$ means that the constrained pushdown system $\mathcal{P}_1$ can reach the configuration pw from p_1, provided that the sequence of control states of $\mathcal{P}_2$ that he observed is σ. $\widehat{\mathcal{P}_1}$ is defined as follows: Δ_1' is the set of transition rules such that $\langle p, u\rangle \overset{p_2}{\hookrightarrow} \langle p', u'\rangle$ is in Δ_1' iff $\exists(\phi : \langle p, u\rangle \hookrightarrow \langle p', u'\rangle) \in \Delta_1$ such that $p_2 \in \phi$. Then, we have that:

Lemma 14. *There exists a run $(pw, qv) \Longrightarrow_N^* (p'w', q'v')$ of the network N, iff there exist a computation $q_0v_0 \Longrightarrow_{\mathcal{P}_2} q_1v_1 \Longrightarrow_{\mathcal{P}_2} \cdots \Longrightarrow_{\mathcal{P}_2} q_mv_m$ of $\mathcal{P}_2$, and $\sigma, \sigma' \in P_2^*$ such that: (1) $qv = q_0v_0$ and $q'v' = q_mv_m$, (2) $\sigma' \in q_0^+ q_1^+ \cdots q_m^+$, (3) $\sigma \preceq \sigma'$, and (3) $pw \overset{\sigma}{\Longrightarrow}_{\widehat{\mathcal{P}_1}} p'w'$.*

Proof:

Let us assume that there exists a run $(p_0w_0, q_0v_0) \Longrightarrow_N (p_1w_1, q_1v_1) \Longrightarrow_N \cdots \Longrightarrow_N (p_mw_m, q_mv_m)$ of the network $N = (\mathcal{P}_1, \mathcal{P}_2, \{(1,2)\})$. Let $i_0, \ldots, i_k \in \{0, \ldots, m-1\}$ be the maximal sequence of indices such that $i_0 < \ldots < i_k$ and $\forall j \in \{0, \ldots, k\}$ there exists a transition rule $\langle s,u\rangle \hookrightarrow \langle s',u'\rangle$ in Δ_2 such that: (1) $s = q_{i_j}$, $s' = q_{i_j+1}$, and $p_{i_j} = p_{i_j+1}$, (2) $w_{i_j} = w_{i_j+1}$, $v_{i_j} = u\upsilon$, and $v_{i_j+1} = u'\upsilon$ for some $\upsilon \in \Gamma_2^*$. Notice that: $\forall j \notin \{i_0,\ldots,i_k\} \cup \{m\}$ we have that $q_j v_j = q_{j+1} v_{j+1}$. Clearly, the sequence of indices $i_0,\ldots,i_k \in \{0,\ldots,m-1\}$ indicates the set of places when the network N has applied a transition rule in Δ_2 in order to reach the configuration $(p_{i_j+1}w_{i_j+1}, q_{i_j+1}v_{i_j+1})$ from $(p_{i_j}w_{i_j}, q_{i_j}v_{i_j})$ for every $j \in \{0,\ldots,k\}$. Hence, $q_{i_0}v_{i_0} \Longrightarrow_{\mathcal{P}_2} \cdots \Longrightarrow_{\mathcal{P}_2} q_{i_k}v_{i_k} \Longrightarrow_{\mathcal{P}_2} q_{i_k+1}v_{i_k+1}$ is a run of $\mathcal{P}_2$ with $q_{i_k+1}v_{i_k+1} = q_m v_m$ and $q_{i_0}v_{i_0} = q_0 v_0$. Moreover, we remark $\forall j \notin \{i_0,\ldots,i_k\} \cup \{m\}$ there exists a transition rule $\phi : \langle t,u\rangle \hookrightarrow \langle t',u'\rangle \in \Delta_1$ such that: (1) $w_j = u\mu$ and $w_{j+1} = u'\mu$ for some $\mu \in \Gamma_1^*$, (2) $p_j = t$ and $p_{j+1} = t'$, (3) $q_j \in \phi$, and (4) $q_j v_j = q_{j+1} v_{j+1}$. Then, $p_j w_j \xRightarrow{q_j}_{\widehat{\mathcal{P}}_1} p_{j+1} w_{j+1}$ is a run of the pushdown system $\widehat{\mathcal{P}}_1$ for every $j \notin \{i_0,\ldots,i_k\} \cup \{m\}$. Hence, we can construct a run $p_0 w_0 \xRightarrow{\sigma}_{\widehat{\mathcal{P}}_1} p_m w_m$ of $\widehat{\mathcal{P}}_1$ such that there exists $\sigma' \in q_{i_0}^+ \ldots q_{i_k}^+ q_m^+$ and $\sigma \preceq \sigma'$.

On the other hand, suppose that there exist a run $q_0 v_0 \Longrightarrow_{\mathcal{P}_2} q_1 v_1 \Longrightarrow_{\mathcal{P}_2} \cdots \Longrightarrow_{\mathcal{P}_2} q_m v_m$ of $\mathcal{P}_2$, and $\sigma, \sigma' \in P_2^*$ such that: (1) $qv = q_0 v_0$ and $q'v' = q_m v_m$, (2) $\sigma' \in q_0^+ q_1^+ \cdots q_m^+$, (3) $\sigma \preceq \sigma'$, and (4) $pw \xRightarrow{\sigma}_{\widehat{\mathcal{P}}_1} p'w'$. In addition, we suppose that the computation $pw \xRightarrow{\sigma}_{\widehat{\mathcal{P}}_1} p'w'$ of $\widehat{\mathcal{P}}_1$ can be decomposed as follows: $p_0 w_0 \xRightarrow{s_1}_{\widehat{\mathcal{P}}_1} p_1 w_1 \xRightarrow{s_2}_{\widehat{\mathcal{P}}_1} \cdots \xRightarrow{s_k}_{\widehat{\mathcal{P}}_1} p_k w_k$ such that $p_0 w_0 = pw$, $p_k w_k = p'w'$, and $\sigma = s_1 \cdots s_k$ with $s_1,\ldots,s_k \in P_2$. Notice that $p_j w_j \xRightarrow{s_{j+1}} p_{j+1} w_{j+1}$, for every $j \in \{0,\ldots,k-1\}$, implies that there exists a transition rule $\phi : \langle p_j,u\rangle \hookrightarrow \langle p_{j+1},u'\rangle \in \Delta_1$ such that $s_{j+1} \in \phi$, $w_j = u\mu$, and $w_{j+1} = u'\mu$ for some $\mu \in \Gamma_1^*$. Then, let $i_0,\ldots,i_l \in \{0,\ldots,m\}$ and $j_0,\ldots,j_l \in \{1,\ldots,k\}$ two sequences of indices with $j_l = k$ such that: $\forall g \in \{1,\ldots,j_0\}$ we have that $s_g = q_{i_0}$ and $\forall t \in \{1,\ldots,l\}$ $\forall g$ s.t. $j_{t-1} < g \leq j_t$ we have that $s_g = q_{i_t}$. Hence a run of the network N can be constructed as follows:

- $(p_0 w_0, q_0 v_0) \Longrightarrow_N (p_0 w_0, q_1 v_1) \Longrightarrow_N \cdots \Longrightarrow_N (p_0 w_0, q_{i_0} v_{i_0})$ which means that the network N applies only transition rules of process $\mathcal{P}_2$ until he reaches $(p_0 w_0, q_{i_0} v_{i_0})$,
- $(p_0 w_0, q_{i_0} v_{i_0}) \Longrightarrow_N (p_1 w_1, q_{i_0} v_{i_0}) \Longrightarrow_N \cdots \Longrightarrow_N (p_{j_0} w_{j_0}, q_{i_0} v_{i_0})$ which means that the network N can apply all transition rules of $\mathcal{P}_1$ that are enabled when the process $\mathcal{P}_2$ is in the control state q_{i_0},
- For every $t \in \{0,\ldots,l-1\}$, we have that $(p_{j_t} w_{j_t}, q_{i_t} v_{i_t}) \Longrightarrow_N (p_{j_t} w_{j_t}, q_{i_t+1} v_{i_t+1}) \Longrightarrow_N \cdots \Longrightarrow_N (p_{j_t} w_{j_t}, q_{i_{t+1}} v_{i_{t+1}})$ which means that the network N applies only transition rules of $\mathcal{P}_2$ until this last process reaches the control state $q_{i_{t+1}}$,
- For every $t \in \{0,\ldots,l-1\}$, we have that $(p_{j_t} w_{j_t}, q_{i_{t+1}} v_{i_{t+1}}) \Longrightarrow_N (p_{j_t+1} w_{j_t+1}, q_{i_{t+1}} v_{i_{t+1}}) \Longrightarrow_N \cdots \Longrightarrow_N (p_{j_{t+1}} w_{j_{t+1}}, q_{i_{t+1}} v_{i_{t+1}})$ which means that the network N can apply all transition rules of $\mathcal{P}_1$ that are enabled when the process $\mathcal{P}_2$ is in the control state $q_{i_{t+1}}$,
- Finally, we have $(p_{j_l} w_{j_l}, q_{i_l} v_{i_l}) \Longrightarrow_N (p_{j_l} w_{j_l}, q_{i_l+1} v_{i_l+1}) \Longrightarrow_N \cdots \Longrightarrow_N (p_{j_l} w_{j_l}, q_{i_m} v_{i_m})$.

Notice that $p_{j_l} w_{j_l} = p_k w_k$ and this completes our proof. □

Then, we construct a labeled pushdown system $\widetilde{\mathcal{P}}_2 = (P_2, P_2, \Gamma_2, \Delta_2')$ such that $p_2 \xRightarrow{\sigma}_{\widetilde{\mathcal{P}}_2} p'w'$ means that the pushdown system $\mathcal{P}_2$ reaches the configuration $p'w'$ from p_2 while performing the state sequence σ, i.e., σ is the sequence of states of $\mathcal{P}_2$ observed by $\mathcal{P}_1$. Δ'_2 is a finite set of transition rules such that $\langle p,u\rangle \overset{p'}{\hookrightarrow} \langle p',u'\rangle$ is in Δ'_2 if there $\exists(\langle p,u\rangle \hookrightarrow \langle p',u'\rangle) \in \Delta_2$ or $p = p'$ and $u = u'$. We can easily show that:

Lemma 15. *If there exists a run $q_0w_0 \Longrightarrow_{\mathcal{P}_2} q_1w_1 \Longrightarrow_{\mathcal{P}_2} \cdots \Longrightarrow_{\mathcal{P}_2} q_mw_m$ of $\mathcal{P}_2$, then $\forall\sigma \in q_0^*q_1^+\cdots q_m^+$, we have $q_0w_0 \overset{\sigma}{\Longrightarrow}_{\widetilde{\mathcal{P}}_2} q_mw_m$. Moreover, if $q_0w_0 \overset{\sigma}{\Longrightarrow}_{\widetilde{\mathcal{P}}_2} q_mw_m$, then, there exists a run $q_0w_0 \Longrightarrow_{\mathcal{P}_2} q_1w_1 \Longrightarrow_{\mathcal{P}_2} \cdots \Longrightarrow_{\mathcal{P}_2} q_mw_m$ of $\mathcal{P}_2$ such that $\sigma \in q_0^*q_1^+\cdots q_m^+$.*

We want the 2-tape automaton T to accept a pair $(pw, p'w')$ iff there exists $\sigma_1,\sigma_2 \in P_2^*$ such that $\sigma_1 \preceq \sigma_2$, $p_1 \overset{\sigma_1}{\Longrightarrow}_{\widehat{\mathcal{P}}_1} pw$ and $p_2 \overset{\sigma_2}{\Longrightarrow}_{\widetilde{\mathcal{P}}_2} p'w'$ (σ_1 is the sequence of observations needed by $\mathcal{P}_1$ in order to perform its computation, and σ_2 is a sequence of potential observations guaranteed by $\mathcal{P}_2$).

Then, the 2-tape automaton T can be constructed as follows: (1) We construct a 2-tape automaton $\widehat{T}(\widehat{\mathcal{P}}_1, p_1)$ (resp. $\widetilde{T}(\widetilde{\mathcal{P}}_2, p_2)$) which accepts $U(\widehat{\mathcal{P}}_1, p_1)$ (resp. $D(\widetilde{\mathcal{P}}_2, p_2)$). These automata can be constructed as described in Theorem 1). (2) We compose these two 2-tape automata according to their second tape. This ensures that the observation sequences needed by $\widehat{\mathcal{P}}_1$ are provided by $\widetilde{\mathcal{P}}_2$. (3) Finally, we use projection to abstract away the second tape (used for composition). Formally, $T = \Pi_{(1,3)}(\widehat{T}(\widehat{\mathcal{P}}_1, p_1) \circ_{(2,2)} \widetilde{T}(\widetilde{\mathcal{P}}_2, p_2))$.

The following fact can be proved using Lemmas (14) and (15)

Lemma 16. $L(T) = post^*(C)$.

Proof:

The configuration (pw,qv) is accepted by T if and only there exists $\sigma \in P_2^*$ such that (pw,σ) is accepted by $\widehat{T}(\widehat{\mathcal{P}}_1, p_1)$ and (qv,σ) is accepted by $\widetilde{T}(\widetilde{\mathcal{P}}_2, p_2)$. Since (pw,σ) (resp. (qv,σ)) is in $U(\widehat{\mathcal{P}}_1, p_1)$ (resp. $D(\widetilde{\mathcal{P}}_2, p_2)$), there exists $\sigma_1 \in P_2^*$ (resp. $\sigma_2 \in P_2^*$) such that $\sigma_1 \preceq \sigma$ (resp. $\sigma \preceq \sigma_2$) and $p_1 \overset{\sigma_1}{\Longrightarrow}_{\widehat{\mathcal{P}}_1} pw$ (resp. $p_2 \overset{\sigma_2}{\Longrightarrow}_{\widetilde{\mathcal{P}}_2} qv$). By applying Lemma.15 to $p_2 \overset{\sigma_2}{\Longrightarrow}_{\widetilde{\mathcal{P}}_2} qv$, there exists a run $q_0v_0 \Longrightarrow_{\mathcal{P}_2} \cdots \Longrightarrow_{\mathcal{P}_2} q_mv_m$ of $\mathcal{P}_2$ such that $q_0v_0 = p_2$, $q_mv_m = qv$, and $\sigma_2 \in q_0^+\cdots q_m^+$. Notice that such σ_2 always exists. Now, we can apply Lemma.14 to $p_1 \overset{\sigma_1}{\Longrightarrow}_{\widehat{\mathcal{P}}_1} pw$, since there exists a run $q_0v_0 \Longrightarrow_{\mathcal{P}_2} \cdots \Longrightarrow_{\mathcal{P}_2} q_mv_m$ such that $\sigma_2 \in q_0^+\cdots q_m^+$ and $\sigma_1 \preceq \sigma_2$, to construct a run $(p_1, p_2) \Longrightarrow_N^* (pw,qv)$.

On the other hand, let $(p_1, p_2) \Longrightarrow_N^* (pw,qv)$ be a run of the network N, then, by applying lemma.14 there exists a run $q_0v_0 \Longrightarrow_{\mathcal{P}_2} \cdots \Longrightarrow_{\mathcal{P}_2} q_mv_m$ of $\mathcal{P}_2$, with $q_0v_0 = p_2$ and $q_mv_m = qv$. In addition, there exists $\sigma_1,\sigma_2 \in P_2^*$ such that $\sigma_2 \in q_0^+\cdots q_m^+$, $\sigma_1 \preceq \sigma_2$ and $p_1 \overset{\sigma_1}{\Longrightarrow}_{\widehat{\mathcal{P}}_1} pw$. Moreover, we have that $p_2 \overset{\sigma_2}{\Longrightarrow}_{\widetilde{\mathcal{P}}_2} qv$ due to Lemma.15. Hence, (pw,σ_1) is in $U((\widehat{\mathcal{P}}_1, p_1)$ and (qv,σ_2) is in $D((\widetilde{\mathcal{P}}_2, p_2)$. Moreover, we have that $\sigma_1 \preceq \sigma_2$ which implies that (pw,σ_2) is in $U((\widehat{\mathcal{P}}_1, p_1)$ too. As consequence, (pw,qv) will be accepted by T. □

The general case: The construction above can be extended to the general case of $\mathrm{APN_{obs}}$s of depth 1. For that, let $N = (\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$ be an $\mathrm{APN_{obs}}$ of depth 1 such that $R = \{(i, j) : 1 \leq i < k , k \leq j \leq n\}$. Notice that the processes of depth 0 are the processes of indices $\{k,\ldots,n\}$, and the processes of depth 1 are the ones of indices $\{1,\ldots,k-1\}$. Moreover, we remark that the processes of depth 1 can observe all the processes of depth 0. In fact, this is not a real restriction since if process $\mathcal{P}_i$ does not observe process $\mathcal{P}_j$, we can add to the constraints of its rules all the states of process $\mathcal{P}_j$ so that it becomes observable by him without introducing any new constraints on his behaviors. But before going into details, let $N_k = (\mathcal{P}_k, \ldots, \mathcal{P}_n, \emptyset)$ denote the restriction of N to the processes of depth 0.

To construct a n-tape automaton T that accepts $post_N^*(C)$, where $C = (p_1, \ldots, p_n)$ is the initial configuration, we proceed as follows: First, for every $i \in \{1, \ldots, k-1\}$, we compute a labeled pushdown system $\widehat{\mathcal{P}}_i = (P_i, \Sigma, \Gamma_i, \Delta_i')$, with $\Sigma = P_k \times \cdots \times P_n$, such that $p_i \overset{\sigma}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} p_i w_i$ means that the constrained pushdown system $\mathcal{P}_i$ can reach the configuration $p_i w_i$ from p_i, provided that the sequence of control states of the $\mathrm{APN}_{\mathsf{obs}}$ N_k that he observed is σ. $\widehat{\mathcal{P}}_i$ is defined as follows: Δ_i' is the set of transition rules such that $\langle p, u\rangle \overset{(s_k,\ldots,s_n)}{\hookrightarrow} \langle p', u'\rangle$ is in Δ_i' iff $\exists(\phi : \langle p, u\rangle \hookrightarrow \langle p', u'\rangle) \in \Delta_i$ such that $\forall j \in \{k, \ldots, n\}$ we have $s_j \in \phi$.

For each process $i \in \{1, \ldots, k-1\}$ of depth 1, we compute a 2-tape automaton $\widehat{T}(\widehat{\mathcal{P}}_i, p_i)$ that recognizes $U(\widehat{\mathcal{P}}_i, p_i)$ such that the first tape contains a reachable configuration $p_i w_i$ of $\mathcal{P}_i$, and the second one contains the sequence $\sigma \in (P_k \times \cdots \times P_n)^*$ of control states of processes $\mathcal{P}_k, \ldots, \mathcal{P}_n$ that are needed by process $\mathcal{P}_i$ to reach $p_i w_i$. Then, we compose all these automata by synchronizing them on their second tape to get a k-tape automaton $\widehat{T}' = \widehat{T}_{k-1}$ that recognizes a vector $(p_1 w_1, \sigma, \ldots, p_{k-1} w_{k-1})$ iff for $i \in \{1, \ldots, k-1\}$, $(p_i w_i, \sigma) \in U(\widehat{\mathcal{P}}_i, p_i)$. The automaton $\widehat{T}'$ can be computed inductively as follows:

1. $\widehat{T}_1 = \widehat{T}(\widehat{\mathcal{P}}_1, p_1)$,
2. for every $i \in \{2, \ldots, k-1\}$, we have that $\widehat{T}_i = \widehat{T}_{i-1} \circ_{(2,2)} \widehat{T}(\widehat{\mathcal{P}}_i, p_i)$.

Then, we need to see the k-tape finite state automaton $\widehat{T}'$ as a n-tape finite state automaton $\widehat{T}$ such that $(p_1 w_1, \sigma, p_2 w_2, p_3 w_3, \ldots, p_{k-1} w_{k-1})$ is accepted by $\widehat{T}'$ iff $(p_1 w_1, \ldots, p_{k-1} w_{k-1}, s_k, \ldots, s_n)$ is accepted by $\widehat{T}$ and $\sigma = (s_k, \ldots, s_n)$. Notice that the n-tape automaton $\widehat{T}$ can be easily constructed from the k-tape automaton $\widehat{T}'$

Next, for every processes $j \in \{k, \ldots, n\}$ of depth 0, we compute a labeled pushdown system $\widetilde{\mathcal{P}}_j = (P_j, P_j, \Gamma_j, \Delta_j')$ such that $p_j \overset{\alpha}{\Longrightarrow}_{\widetilde{\mathcal{P}}_j} p_j w_j$ means that the pushdown system $\mathcal{P}_j$ reaches the configuration $p_j w_j$ from p_j while performing the state sequence $\alpha \in P_j^*$, i.e., α is the sequence of states of $\mathcal{P}_j$ that can be observed. Δ_j' is a finite set of transition rules such that $\langle p, u\rangle \overset{p'}{\hookrightarrow} \langle p', u'\rangle$ is in Δ_j': (1) if there $\exists(\langle p, u\rangle \hookrightarrow \langle p', u'\rangle) \in \Delta_j$, or (2) $p = p'$ and $u = u'$.

Now, we can construct a 2-tape automaton $\widetilde{T}_j$, for every $j \in \{k, \ldots, n\}$, that recognizes $D(\widetilde{\mathcal{P}}_j, p_j)$. Then, we synchronize all these automata with $\widehat{T}$ (the second tape of $\widetilde{T}_j$ gets synchronized with the component of $\widehat{T}$ that corresponds to the states of P_j). We project then on the components corresponding to the configurations. The obtained n-tape automaton T accepts $(p_1 w_1, \ldots, p_n w_n)$ iff its is in the reachability set of N. Formally, the automaton $T = T_n$ can be computed inductively as follows:

1. $T_k = \Pi_{(1,\ldots,k-1,k+1,\ldots,n,n+1)}(\widehat{T} \circ_{(k,2)} \widetilde{T}_k)$
2. for every $j \in \{k+1, \ldots, n\}$, we have that $T_j = \Pi_{(1,\ldots,k-1,k+1,\ldots,n,n+1)}(T_{j-1} \circ_{(k,2)} \widetilde{T}_k)$.

□

J Proof of Theorem 3

Theorem 3 *Given a rational set C and a configuration c, the problem of checking whether C is reachable from c is undecidable for APN_{obs}.*

Proof:

We show that the existence of a decision procedure to the problem whether $post_N^*(c) \cap C = \emptyset$ would imply a decision procedure for the Post's Correspondance Problem. Let $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ be two sequences of words over an alphabet Σ.

Let $N = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, R)$ where $R = \{(1,2),(2,3)\}$ ($\mathcal{P}_1$ observes $\mathcal{P}_2$, who observes $\mathcal{P}_3$), and for $i \in \{1,2,3\}$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ is such that $P_1 = \{p, p_i \mid i \in \{1,\ldots,n\}\}$, $P_2 = \{q, q_i \mid i \in \{1,\ldots,n\}\}$, $P_3 = \{s, s_i \mid i \in \{1,\ldots,n\}\}$, $\Gamma_1 = \Gamma_2 = \Gamma_3 = \Sigma \cup \{\perp\} \cup \{\sharp_1, \ldots, \sharp_n\}$ (the union is disjoint); $\Delta_1 = \{\{q\} : \langle p_i, \perp\rangle \hookrightarrow \langle p, \perp\rangle\ ,\ \{q_i\} : \langle p, \perp\rangle \hookrightarrow \langle p_i, \perp\sharp_i\rangle, i \in \{1,\ldots,n\}\}$, $\Delta_2 = \{\{s\} : \langle q_i, \perp\rangle \hookrightarrow \langle q, \perp\rangle\ ,\ \{s_i\} : \langle q, \perp\rangle \hookrightarrow \langle q_i, \perp v_i\rangle, i \in \{1,\ldots,n\}\}$, and $\Delta_3 = \{\langle s_i, \perp\rangle \hookrightarrow \langle s, \perp\rangle\ ,\ \langle s, \perp\rangle \hookrightarrow \langle s_i, \perp u_i \sharp_i\rangle, i \in \{1,\ldots,n\}\}$.

$\perp$ is used as the top of the stack. $\mathcal{P}_3$ pushes the words $u_i\sharp_i$'s in its stack independently of $\mathcal{P}_1$ and $\mathcal{P}_2$. $\mathcal{P}_2$ pushes the word v_i in its stack if $\mathcal{P}_3$ is in state s_i, i.e., if the last word put by $\mathcal{P}_3$ is $u_i\sharp_i$. Hence, for $i \in \{1,\ldots,n\}$, the number of words v_i's in $\mathcal{P}_2$'s stack is bounded by the number of words $u_i\sharp_i$'s in $\mathcal{P}_3$'s stack. Finally, $\mathcal{P}_1$ puts $\sharp_i$ in its stack if the last word put by $\mathcal{P}_2$ is v_i (if $\mathcal{P}_2$ is in state q_i). Thus, for $i \in \{1,\ldots,n\}$, $\mathcal{P}_1$ can put on its stack at most as many $\sharp_i$'s as v_i's in $\mathcal{P}_2$'s stack.

Let $X = \{(\sharp_i, \varepsilon, \sharp_i) \mid i \in \{1,\ldots,n\}\} \cup \{(\varepsilon, x, x) \mid x \in \Sigma\}$, and let $C = X^*$. Then, we show below that $Post^*\big((p\perp, q\perp, s\perp)\big) \cap C$ is not empty iff there exist indices $i_1, i_2, \ldots, i_k (k > 0)$ such that $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$. Thus, iff the Post's Correspondance Problem has a solution.

Indeed, if $Post^*\big((p\perp, q\perp, s\perp)\big) \cap C$ is not empty, then $\mathcal{P}_3$, $\mathcal{P}_2$, and $\mathcal{P}_1$ can successively put in their stacks $u_{i_1}\sharp_{i_1} u_{i_2}\sharp_{i_2} \cdots u_{i_k}\sharp_{i_k}$, $v_{j_1} v_{j_2} \cdots v_{j_m}$, and $\sharp_{l_1}\sharp_{l_2} \cdots \sharp_{l_n}$ such that:

- $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{j_1} v_{j_2} \cdots v_{j_m}$ (because of the part $\{(\varepsilon, x, x) \mid x \in \Sigma\}$ of L).
- $\sharp_{i_1}\sharp_{i_2} \cdots \sharp_{i_k} = \sharp_{l_1}\sharp_{l_2} \cdots \sharp_{l_n}$. This is due to the loop $(\sharp_i, \varepsilon, \sharp_i)$ of C. This means that the indices $l_1, l_2, \ldots, l_n$ taken by $\mathcal{P}_1$ are exactly the same than those $i_1, i_2, \ldots, i_k$ taken by $\mathcal{P}_3$.

Since, as explained previously, the ordered sequence of indices $l_1, l_2, \ldots, l_n$ considered by $\mathcal{P}_1$ (resp., $j_1, \ldots, j_m$ considered by $\mathcal{P}_2$) is a subsequence of the sequence of indices $j_1, \ldots, j_m$ considered by $\mathcal{P}_2$ (resp., $i_1, i_2, \ldots, i_k$ considered by $\mathcal{P}_3$), it follows from the second item above that the indices taken by $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ are the same. Hence, $l_1, l_2, \ldots, l_n = j_1, \ldots, j_m = i_1, \ldots, i_k$. This means that there exist indices $i_1, i_2, \ldots, i_k$ such that $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$, which answers the Post's Correspondance Problem.

It is easy to see that if there exist indices $i_1, i_2, \ldots, i_k (k > 0)$ such that $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$, then $Post^*\big((p\perp, q\perp, s\perp)\big) \cap C$ is not empty.

□

K Proof of Theorem 4

Theorem 4 *Checking reachability between two configurations for a linear APN$_{\mathsf{obs}}$ $N = (\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$ can be decide in $2(n-1)$-exponential time in $\sum_{i=0}^{n}|P_i|+|\Gamma_i|$.*

Proof:

First, let $N_i = (\mathcal{P}_i,\ldots,\mathcal{P}_n,R_i)$ where $R_i = \{(j,j+1) : \forall j \in \{i,\ldots,n-1\}\}$ be a linear APN$_{\mathsf{obs}}$. Notice that the network N_i is the restriction of N to the $(n-i+1)$ last processes. We can show that:

Lemma 17. *There exists a run $(p_i^0 w_i^0,\ldots,p_n^0 w_n^0) \Longrightarrow_{N_i} \cdots \Longrightarrow_{N_i} (p_i^m w_i^m,\ldots,p_n^m w_n^m)$ of the network N_i, with $i<n$, iff:*

- *The set $\{0,\ldots,m-1\}$ can be partitioned into $\{j_0,\ldots,j_k\}$ and $\{j'_0,\ldots,j'_l\}$, i.e. $\forall j \in \{0,\ldots,m-1\}$ the index $j \in \{j_0,\ldots,j_k\}$ or $j \in \{j'_0,\ldots,j'_l\}$,*
- *$(p_{i+1}^{j_0} w_{i+1}^{j_0},\ldots,p_n^{j_0} w_n^{j_0}) \Longrightarrow_{N_{i+1}} \cdots \Longrightarrow_{N_{i+1}} (p_{i+1}^{j_k} w_{i+1}^{j_k},\ldots,p_n^{j_k} w_n^{j_k}) \Longrightarrow_{N_{i+1}} (p_{i+1}^{m} w_{i+1}^{m},\ldots,p_n^{m} w_n^{m})$ is a run of the network N_{i+1} with $(p_{i+1}^{j_0} w_{i+1}^{j_0},\ldots,p_n^{j_0} w_n^{j_0}) = (p_{i+1}^{0} w_{i+1}^{0},\ldots,p_n^{0} w_n^{0})$.*
- *$(p_i^{j'_0} w_i^{j'_0}) \overset{p_{i+1}^{j'_0}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} \cdots \overset{p_{i+1}^{j'_{l-1}}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} (p_i^{j'_l} w_i^{j'_l}) \overset{p_{i+1}^{j'_l}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} (p_i^m w_i^m)$ is a run of the pushdown system $\widehat{\mathcal{P}}_i$ with $(p_i^{j'_0} w_i^{j'_0}) = (p_i^0 w_i^0)$.*
- *$p_i^0 \cdots p_i^m \in (p_i^{j'_0})^+ \cdots (p_i^{j'_l})^+ (p_i^m)^+$.*
- *$p_{i+1}^0 \cdots p_{i+1}^m \in (p_{i+1}^{j_0})^+ \cdots (p_{i+1}^{j_k})^+ (p_{i+1}^m)^+$.*
- *$p_{i+1}^{j'_0} \cdots p_{i+1}^{j'_l} \preceq p_{i+1}^0 \cdots p_{i+1}^m$.*

The intuition behind is that the sequence of indices $j'_0,\ldots,j'_l$ is obtained by checking when we need to apply a rule from Δ_i in order that N_i is able to fire the transition relation $(p_i^{j'_t} w_i^{j'_t},\ldots,p_n^{j'_t} w_n^{j'_t}) \Longrightarrow_{N_i} (p_i^{j'_t+1} w_i^{j'_t+1},\ldots,p_n^{j'_t+1} w_n^{j'_t+1})$ for every $t \in \{0,\ldots,l\}$. In the same manner, the sequence of indices $j_0,\ldots,j_k$ is obtained by checking when we have applied a rule of the network N_{i+1}.

Next, we recall the definition of the product between a pushdown system and a finite automaton:

Definition 5. *Given a PDS $\mathcal{P} = (P,\Sigma,\Gamma,\Delta)$ and an automaton $\mathcal{A} = (A,P,\delta,I,F)$, let $\mathcal{P} \otimes \mathcal{A} = (P \times A,\Sigma,\Gamma,\Delta')$ be the pushdown system where Δ' is the set of transition rules such that:*

- *$\langle (p,s),u\rangle \overset{\varepsilon}{\hookrightarrow} \langle (p,s'),u'\rangle \in \Delta'$ iff $\exists((s,p,s')) \in \delta$.*
- *$\langle (p,s),u\rangle \overset{a}{\hookrightarrow} \langle (p',s'),u'\rangle$ is in Δ' iff $\exists(\langle p,u\rangle \overset{a}{\hookrightarrow} \langle p',u'\rangle) \in \Delta$ and $\exists((s,p',s')) \in \delta$.*

Clearly, we have that:

Lemma 18. *Let $\mathcal{P} = (P,\Sigma,\Gamma,\Delta)$ be an LPDS and $\mathcal{A} = (A,P,\delta,I,F)$ be a finite state automaton, then, there exists a run $(p,s)w \overset{\sigma}{\Longrightarrow}_{\mathcal{P}\otimes\mathcal{A}} (p',s')w'$ of the LPDS $\mathcal{P}\otimes\mathcal{A}$ iff there exist a run $p_0 w_0 \overset{a_1}{\Longrightarrow}_{\mathcal{P}} \cdots \overset{a_m}{\Longrightarrow}_{\mathcal{P}} p_m w_m$ of the LPDS $\mathcal{P}$, for some $a_1,\ldots,a_m \in \Sigma$, and $v \in P^*$ such that: (1) $p_0 w_0 = pw$, (2)$p_m w_m = p'w'$, (3) $\sigma = a_1 \cdots a_m$, (4) $(s,\alpha,s') \in \delta^*$, and (5) $\alpha \in p_0^* p_1^+ \cdots p_m^+$.*

Then, we prove that:

Lemma 19. *Given an index $i \in \{1,\ldots,n-1\}$, there exist a run $(p_i^0 w_i^0,\ldots,p_n^0 w_n^0) \Longrightarrow_{N_i} \cdots \Longrightarrow_{N_i} (p_i^m w_i^m,\ldots,p_n^m w_n^m)$ of the network N_i, where $(p_i^0 w_i^0,\ldots,p_n^0 w_n^0) = (p_i,\ldots,p_n)$, and $(p_i^m w_i^m,\ldots,p_n^m w_n^m) = (p_i',\ldots,p_n')$, and $\alpha \in L(\mathcal{A}_i)$ such that $\alpha \in (p_i^0)^+ \cdots (p_i^m)^+$ iff there exist a sequence $j_0,\ldots,j_k \in \{0,\ldots,m\}$ such that $(p_{i+1}^{j_0} w_{i+1}^{j_0},\ldots,p_n^{j_0} w_n^{j_0}) \Longrightarrow_{N_{i+1}} \cdots \Longrightarrow_{N_{i+1}} (p_{i+1}^{j_k} w_{i+1}^{j_k},\ldots,p_n^{j_k} w_n^{j_k})$ of the network N_{i+1}, and $\alpha' \in (p_{i+1}^{j_0})^+ \cdots (p_{i+1}^{j_k})^+$ such that α' is accepted by $\mathcal{A}_{i+1}$ with: (1) $(p_{i+1}^{j_0} w_{i+1}^{j_0},\ldots,p_n^{j_0} w_n^{j_0}) = (p_{i+1},\ldots,p_n)$, and (2) $(p_{i+1}^{j_k} w_{i+1}^{j_k},\ldots,p_n^{j_k} w_n^{j_k}) = (p_{i+1}',\ldots,p_n')$.*

Proof:

The intuition behind is that if there exists a run $(p_i^0 w_i^0,\ldots,p_n^0 w_n^0) \Longrightarrow_{N_i} \cdots \Longrightarrow_{N_i} (p_i^m w_i^m,\ldots,p_n^m w_n^m)$ of N_i, then, thanks to lemma.17, there exists $j_0,\ldots,j_k$ and $j_0',\ldots,j_l'$ such that:

- $(p_{i+1}^{j_0} w_{i+1}^{j_0},\ldots,p_n^{j_0} w_n^{j_0}) \Longrightarrow_{N_{i+1}} \cdots \Longrightarrow_{N_{i+1}} (p_{i+1}^{j_k} w_{i+1}^{j_k},\ldots,p_n^{j_k} w_n^{j_k}) \Longrightarrow_{N_{i+1}} (p_{i+1}^m w_{i+1}^m,\ldots,p_n^m w_n^m)$ is a run of the network N_{i+1} with $(p_{i+1}^{j_0} w_{i+1}^{j_0},\ldots,p_n^{j_0} w_n^{j_0}) = (p_{i+1}^0 w_{i+1}^0,\ldots,p_n^0 w_n^0)$.
- $(p_i^{j_0'} w_i^{j_0'}) \overset{p_{i+1}^{j_0'}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} \cdots \overset{p_{i+1}^{j_{l-1}'}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} (p_i^{j_l'} w_i^{j_l'}) \overset{p_{i+1}^{j_l'}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} (p_i^m w_i^m)$ is a run of the pushdown system $\widehat{\mathcal{P}}_i$ with $(p_i^{j_0'} w_i^{j_0'}) = (p_i^0 w_i^0)$.
- $p_i^0 \cdots p_i^m \in (p_i^{j_0'})^+ \cdots (p_i^{j_l'})^+ (p_i^m)^+$.
- $p_{i+1}^0 \cdots p_{i+1}^m \in (p_{i+1}^{j_0})^+ \cdots (p_{i+1}^{j_k})^+ (p_{i+1}^m)^+$.
- $p_{i+1}^{j_0'} \cdots p_{i+1}^{j_l'} \preceq p_{i+1}^0 \cdots p_{i+1}^m$.

Indeed, we know that $\exists \alpha \in L(\mathcal{A}_i)$ such that $\alpha \in (p_i^0)^+ \cdots (p_i^m)^+$ and $p_i^0 \cdots p_i^m \in (p_i^{j_0'})^+ \cdots (p_i^{j_l'})^+ (p_i^m)^+$. As consequence, we have that $\alpha \in (p_i^{j_0'})^+ \cdots (p_i^{j_l'})^+ (p_i^m)^+$ and $(p_i^{j_0'} w_i^{j_0'}) \overset{p_{i+1}^{j_0'}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} \cdots \overset{p_{i+1}^{j_{l-1}'}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} (p_i^{j_l'} w_i^{j_l'}) \overset{p_{i+1}^{j_l'}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} (p_i^m w_i^m)$ is a run of the LPDS $\widehat{\mathcal{P}}_i$ where $(p_i^{j_0'} w_i^{j_0'}) = (p_i^0 w_i^0)$. Hence, we can apply lemma.18, to prove that $p_{i+1}^{j_0'} \cdots p_{i+1}^{j_l'}$ is in $L_{\widehat{\mathcal{P}}_i \otimes \mathcal{A}_i}((p_i,s_i),(p_i',f_i))$ which implies that the sequence $p_{i+1}^0 \cdots p_{i+1}^m \in L(\mathcal{A}_{i+1})$, since $L(\mathcal{A}_{i+1})$ is upward closed, since $p_{i+1}^0 \cdots p_{i+1}^m \in (p_{i+1}^{j_0})^+ \cdots (p_{i+1}^{j_k})^+ (p_{i+1}^m)^+$.

Now, let $(p_{i+1}^0 w_{i+1}^0,\ldots,p_n^0 w_n^0) \Longrightarrow_{N_{i+1}} \cdots \Longrightarrow_{N_{i+1}} (p_{i+1}^m w_{i+1}^m,\ldots,p_n^m w_n^m)$ be a run of the network N_{i+1} such that $(p_{i+1}^0 w_{i+1}^0,\ldots,p_n^0 w_n^0) = (p_{i+1},\ldots,p_n)$, $(p_{i+1}^m w_{i+1}^m,\ldots,p_n^m w_n^m) = (p_{i+1}',\ldots,p_n')$, and $\exists \alpha \in (p_{i+1}^0)^+ \cdots (p_{i+1}^m)^+ \cap L(\mathcal{A}_{i+1})$. Notice that $\alpha \in L(\mathcal{A}_{i+1})$ implies (thanks to lemma.18) that there exist a run $p_i^0 w_i^0 \overset{s^0}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} \cdots \overset{s^{k-1}}{\Longrightarrow}_{\widehat{\mathcal{P}}_i} p_i^k w_i^k$ of the LPDS $\widehat{\mathcal{P}}_i$ such that $s^0 \cdots s^{k-1} \preceq \alpha$, and $\sigma \in L(\mathcal{A}_i) \cap ((p_i^0)^* (p_i^1)^+ \cdots (p_i^k)^+)$. Hence, we can apply lemma.17 to construct a run $(q_i^0 v_i^0,\ldots,q_n^0 v_n^0) \Longrightarrow_{N_i} \cdots \Longrightarrow_{N_i} (q_i^l v_i^l,\ldots,q_n^l v_n^l)$ of the network N_i such that $\{0,\ldots,l-1\}$ can be partitioned into $\{j_0,\ldots,j_{m-1}\}$ and $\{j_0',\ldots,j_{k-1}'\}$ such that $q_g^{j_t} w_g^{j_t} = p_g^t v_g^t$ for every $t \in \{0,\ldots,m-1\}$ and $g \in \{i+1,\ldots,n\}$, and $q_i^{j_t'} v_i^{j_t'} = p_i^t w_i^t$ for every $t \in \{0,\ldots,k-1\}$. Notice that $q_i^0 \cdots q_i^l \in (p_i^0)^+ \cdots (p_i^m)^+$ and that $\sigma \in L(\mathcal{A}_i) \cap ((p_i^0)^* (p_i^1)^+ \cdots (p_i^m)^+)$, implies that $\exists\sigma' \in L(\mathcal{A}_i) \cap (q_i^0)^+ \cdots (q_i^l)^+$ and such that $\sigma \preceq \sigma'$ since $L(\mathcal{A}_i)$ is upward closed.

$\square$

Finally, we obtain that:

Lemma 1 *The configuration $(p'_1, \ldots, p'_n)$ is reachable from $(p_1, \ldots, p_n)$ iff $Traces_{\widehat{\mathcal{P}_n} \otimes \mathcal{A}_n}((p_n, s_n), (p'_n, f_n)) \neq \emptyset$.*

Proof:

The idea is that if there exists a run $(p_1^0 w_1^0, \ldots, p_n^0 w_n^0) \Longrightarrow_N \cdots \Longrightarrow_N (p_1^m w_1^m, \ldots, p_n^m w_n^m)$ of the network N with $(p_1^0 w_1^0, \ldots, p_n^0 w_n^0) = (p_1, \ldots, p_n)$ and $(p_1^m w_1^m, \ldots, p_n^m w_n^m) = (p'_1, \ldots, p'_n)$, then, due to lemma.19, we know that there exist a sequence $j_0, \ldots, j_k \in \{0, \ldots, m\}$ such that $(p_2^{j_0} w_2^{j_0}, \ldots, p_n^{j_0} w_n^{j_0}) \Longrightarrow_{N_2} \cdots \Longrightarrow_{N_2} (p_2^{j_k} w_2^{j_k}, \ldots, p_n^{j_k} w_n^{j_k})$ of the network N_2 with $(p_2^{j_0} w_2^{j_0}, \ldots, p_n^{j_0} w_n^{j_0}) = (p_2, \ldots, p_n)$, $(p_2^{j_k} w_2^{j_k}, \ldots, p_n^{j_k} w_n^{j_k}) = (p'_2, \ldots, p'_n)$, and $\exists\alpha \in (p_2^{j_0})^+ \cdots (p_{i+1}^{j_k})^+$ such that $\alpha \in L(\mathcal{A}_2)$.

Inductively, lemma.19 proves that given an index $i \in \{2, \ldots, n-1\}$, and a run $(p_i^0 w_i^0, \ldots, p_n^0 w_n^0) \Longrightarrow_{N_i} \cdots \Longrightarrow_{N_i} (p_i^m w_i^m, \ldots, p_n^m w_n^m)$ of the network N_i, with $(p_i^0 w_i^0, \ldots, p_n^0 w_n^0) = (p_i, \ldots, p_n)$, and $(p_i^m w_i^m, \ldots, p_n^m w_n^m) = (p'_i, \ldots, p'_n)$, and there exists $\alpha' \in L(\mathcal{A}_i)$ such that $\alpha' \in (p_i^0)^+ \cdots (p_i^m)^+$ iff there exist a sequence $j_0, \ldots, j_k \in \{0, \ldots, m\}$ such that $(p_{i+1}^{j_0} w_{i+1}^{j_0}, \ldots, p_n^{j_0} w_n^{j_0}) \Longrightarrow_{N_{i+1}} \cdots \Longrightarrow_{N_{i+1}} (p_{i+1}^{j_k} w_{i+1}^{j_k}, \ldots, p_n^{j_k} w_n^{j_k})$ of the network N_{i+1} with $(p_{i+1}^{j_0} w_{i+1}^{j_0}, \ldots, p_n^{j_0} w_n^{j_0}) = (p_{i+1}, \ldots, p_n)$, $(p_{i+1}^{j_k} w_{i+1}^{j_k}, \ldots, p_n^{j_k} w_n^{j_k}) = (p'_{i+1}, \ldots, p'_n)$, and $\exists\alpha' \in (p_{i+1}^{j_0})^+ \cdots (p_{i+1}^{j_k})^+$ such that $\alpha' \in L(\mathcal{A}_{i+1})$.

In particular, we have that there exists a run $q_n^0 v_n^0 \Longrightarrow_{N_n} \cdots \Longrightarrow_{N_n} q_n^k v_n^k$, of the network $N_n = (\mathcal{P}_n, \emptyset)$, with $q_n^0 v_n^0 = p_n$ and $q_n^k v_n^k = p'_n$, such that there exists $\alpha_0 \in (q_n^0)^+ \cdots (q_n^k)^+$ accepted by $\mathcal{A}_n$, then using lemma.18, we have $Traces_{\widehat{\mathcal{P}_n} \otimes \mathcal{A}_n}((p_n, s_n), (p'_n, f_n)) \neq \emptyset$.

On the other hand, if $Traces_{\widehat{\mathcal{P}_n} \otimes \mathcal{A}_n}((p_n, s_n), (p'_n, f_n)) \neq \emptyset$, then we can construct a run of $\widehat{\mathcal{P}_n} \otimes \mathcal{A}_n$ from the configuration (p_n, s_n) to (p'_n, f_n). By applying lemma.18, there exists a run $p_n^0 w_n^0 \Longrightarrow_{\mathcal{P}_n} \cdots \Longrightarrow_{\mathcal{P}_n} p_n^m w_n^m$ of $\mathcal{P}_n$, with $p_n^0 w_n^0 = p_n$ and $p_n^m w_n^m = p'_n$, and $\alpha_n \in P_n^*$ such that $(s_n, \alpha_n, f_n) \in \delta_n^*$ and $\alpha_n \in (p_n^0)^+ \cdots (p_n^m)^+$. Notice that such α_n exists since $L(\mathcal{A}_n)$ is upward closed.

Inductively, lemma.19 shows that given a run $(q_i^0 v_i^0, \ldots, q_n^0 v_n^0) \Longrightarrow_{N_i} \cdots \Longrightarrow_{N_i} (q_i^k v_i^k, \ldots, q_n^k v_n^k)$ of the network N_i, with $1 < i$, $(q_i^k v_i^k, \ldots, q_n^k v_n^k) = (p'_i, \ldots, p'_n)$, $(q_i^0 v_i^0, \ldots, q_n^0 v_n^0) = (p_i, \ldots, p_n)$, and there exists $\alpha_i \in L(\mathcal{A}_i)$ such that $\alpha_i \in (q_i^0)^+ \cdots (q_i^k)^+$, we can construct a run $(s_{i-1}^0 w_{i-1}^0, \ldots, s_n^0 w_n^0) \Longrightarrow_{N_{i-1}} \cdots \Longrightarrow_{N_{i-1}} (s_{i-1}^m w_{i-1}^m, \ldots, s_n^m w_n^m)$ of the network N_{i-1} such that $(s_{i-1}^m w_{i-1}^m, \ldots, s_n^m w_n^m) = (p'_{i-1}, \ldots, p'{}_n)$, $(s_{i-1}^0 w_{i-1}^0, \ldots, s_n^0 w_n^0) = (p_{i-1}, \ldots, p_n)$, and there exists $\alpha_{i-1} \in L(\mathcal{A}_{i-1})$ such that $\alpha_{i-1} \in (s_{i-1}^0)^+ \cdots (s_{i-1}^m)^+$.

Hence, we can construct a run of the network $N_1 = N$ that reaches $(p'_1, \ldots, p'_n)$ from the configuration $(p_1, \ldots, p_n)$. $\square$

$\square$

L Bottom-Up Algorithm to solve the reachability problem for APN$_{\mathsf{obs}}$

We can also check whether $(p'_1,\ldots,p'_n)$ is reachable from $(p_1,\ldots,p_n)$ for a linear APN$_{\mathsf{obs}}$ $N=(\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$ in a bottom-up manner as follows: For every index i, we compute the set $\mathcal{A}'_i$ of state sequences that $\mathcal{P}_{i+1}$ can perform to go from p_{i+1} to configuration p'_{i+1}. Then, we compute a "standard" pushdown system that performs the same transitions as $\mathcal{P}_i$ when the sequences of states performed by $\mathcal{P}_{i+1}$ are in $\mathcal{A}'_i$ (remember that $\mathcal{P}_i$ observes $\mathcal{P}_{i+1}$). This standard pushdown system is computed by a kind of product ($\odot$ defined below) between $\mathcal{A}'_i$ and $\mathcal{P}_i$. The finite state automaton $\mathcal{A}'_{i-1}$ can be computed as the downward closure of the language of the product between $\mathcal{P}_i$ and $\mathcal{A}'_i$. We start by computing $\mathcal{A}'_{n-1}$. $(p'_1,\ldots,p'_n)$ is reachable from $(p_1,\ldots,p_n)$ iff the pushdown process corresponding to the restriction of $\mathcal{P}_1$ can go from p_1 to p'_1.

Formally, we define the product as follows:

Definition 6. *Given the constrained pushdown $\mathcal{P}_i=(P_i,\Gamma_i,\Delta_i)$ with $i\in\{1,\ldots,n-1\}$ and a finite state automaton $\mathcal{A}=(A,P_{i+1},\delta,I,F)$, let $\mathcal{P}_i\odot\mathcal{A}=(P_i\times A,P_{i+1},\Gamma_i,\Delta'_i)$ be an LPDS where Δ'_i is the set of transition rules such that $\langle(p,s),u\rangle\overset{q}{\hookrightarrow}\langle(p',s'),u'\rangle$ is in Δ' iff $\exists(\phi\,:\,\langle p,u\rangle\hookrightarrow\langle p',u'\rangle)\in\Delta_i$ and $\exists((s,q,s'))\in\delta$ such that $q\in\phi$.*

We can easily show:

Lemma 20. *Given the constrained pushdown $\mathcal{P}_i=(P_i,\Gamma_i,\Delta_i)$ with $i\in\{1,\ldots,n-1\}$ and a finite state automaton $\mathcal{A}=(A,P_{i+1},\delta,I,F)$, then, there exist a run $(p,s)w\overset{\sigma}{\Longrightarrow}_{\mathcal{P}_i\odot\mathcal{A}}(p',s')w'$ iff $pw\overset{\sigma}{\Longrightarrow}_{\widehat{\mathcal{P}_i}}p'w'$ and $(s,\sigma,s')\in\delta^*$.*

Moreover, we can define the LPDS $\widetilde{\mathcal{P}_i\odot\mathcal{A}}$ from $\mathcal{P}_i\odot\mathcal{A}$ as follows:

Definition 7. *Given the constrained pushdown $\mathcal{P}_i=(P_i,\Gamma_i,\Delta_i)$ with $i\in\{1,\ldots,n-1\}$ and a finite state automaton $\mathcal{A}=(A,P_{i+1},\delta,I,F)$, let $\widetilde{\mathcal{P}_i\odot\mathcal{A}}=(P_i\times A,P_i,\Gamma_i,\Delta''_i)$ be an LPDS where Δ''_i is a finite set of transition rules such that $\langle(p,s),u\rangle\overset{p'}{\hookrightarrow}\langle(p',s'),u'\rangle$ is in Δ''_i if: (1) $\exists(\langle(p,s),u\rangle\overset{a}{\hookrightarrow}\langle(p',s'),u'\rangle)\in\Delta'_i$ or (2) $(p,s)=(p',s')$ and $u=u'=\varepsilon$.*

Notice that we can prove the following fact:

Lemma 21. *Given the constrained pushdown $\mathcal{P}_i=(P_i,\Gamma_i,\Delta_i)$ with $i\in\{1,\ldots,n-1\}$ and a finite state automaton $\mathcal{A}=(A,P_{i+1},\delta,I,F)$, then, there exists a run $(p,s)w\overset{\sigma}{\Longrightarrow}_{\widetilde{\mathcal{P}_i\odot\mathcal{A}}}(p',s')w'$ of the LPDS $\widetilde{\mathcal{P}_i\odot\mathcal{A}}$ iff there exists a run $(p_0,s_0)w_0\overset{q_1}{\Longrightarrow}_{\mathcal{P}_i\odot\mathcal{A}}\cdots\overset{q_m}{\Longrightarrow}_{\mathcal{P}_i\odot\mathcal{A}}(p_m,s_m)w_m$ of $\mathcal{P}_i\odot\mathcal{A}$ for some $q_1,\ldots,q_m\in P_{i+1}$ such that: (1) $(p_0,s_0)w_0=(p,s)w$, (2) $(p_m,s_m)w_m=(p',s')w'$, (3) $(s,q_1\cdots q_m,s')\in\delta^*$, and (4) $\sigma\in p_0^*p_1^+\cdots p_m^+$.*

Then, our bottom-up algorithm is given as follows: we construct a sequence of finite state automata $\mathcal{A}'_i=(A'_i,P_i,\delta'_i,s'_i,f'_i)$ such that:

1. $\mathcal{A}'_{n-1}$ accepts the regular language $Traces_{\widetilde{\mathcal{P}}_n}(p_n,p'_n)\downarrow$, where $\widetilde{\mathcal{P}}_n=(P_n,P_n,\Gamma_n,\Delta'_n)$ is a pushdown system constructed from $\mathcal{P}_n$ such that $\langle p,u\rangle\overset{p'}{\hookrightarrow}\langle p',u'\rangle\in\Delta'_n$ iff (1) $\exists(\langle p,u\rangle\hookrightarrow\langle p',u'\rangle)\in\Delta_n$ or (2) $p=p'$ and $u=u'=\varepsilon$.

2. For every $i \in \{2,\ldots,n-1\}$, $\mathcal{A}'_{i-1}$ accepts the regular language $Traces_{\widetilde{\mathcal{P}_i \odot \mathcal{A}'_i}}((p_i,s'_i),(p'_i,f'_i)) \downarrow$

Similarly, we can show that:

Lemma 22. *The linear APN*$_{\mathsf{obs}}$ *N reaches $(p'_1,\ldots,p'_n)$ from $(p_1,\ldots,p_n)$ iff $Traces_{\mathcal{P}_1 \odot \mathcal{A}'_1}((p_1,s'_1),(p'_1 f'_1)) \neq \emptyset$.*

Hence, we have:

Theorem 8. *Checking reachability problem between two configurations for a linear APN*$_{\mathsf{obs}}$ *$N = (\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$ can be decide in $(n-1)$-exponential time in $\sum_{i=0}^{n}|P_i| + |\Gamma_i|$.*

M Computing the projection of the reachability set for APN$_{\mathsf{obs}}$

We show in this section that we can compute a tape automaton that recognizes the projection of the reachability set of a linear APN$_{\mathsf{obs}}$ N on the sets of configurations of two consecutive processes $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ (notice that the linearity is not a restriction thanks to Proposition 3).

Theorem 9. *Let $\mathcal{N} = (\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$ be a linear APN*$_{\mathsf{obs}}$*, $(p_1,\ldots,p_n)$ be an initial configuration, and $\forall i \in \{1,\ldots,n\}$. $p'_i \in P_i$ be a control state. Then, for every index $i < n$, we can compute a 2-tape automaton T_i such that $(p'_i w_i, p'_{i+1} w_{i+1})$ is accepted by T_i iff there exist sequences $w_j \in \Gamma_j^*$ for $j \notin \{i,i+1\}$ such that $(p'_1 w_1,\ldots,p'_n w_n) \in post^*_N((p_1,\ldots,p_n))$.*

The idea of the construction is as follows: Let us fix an index i. First we compute the set $\mathcal{B}_{i+1}$ of state sequences that $\mathcal{P}_{i+2}$ can perform when the network composed of the processes $\mathcal{P}_{i+2},\ldots,\mathcal{P}_n$ goes from $(p_{i+2},\ldots,p_n)$ to a configuration in $p'_{i+2}\Gamma^*_{i+2}\cdots\times\cdots p'_n\Gamma^*_n$. This set can be computed in a bottom-up manner as described in the previous section. Then we compute the new pushdown system $\mathcal{P}'_{i+1} = \mathcal{P}_{i+1} \odot \mathcal{B}_{i+1}$. This PDS has the same behavior as $\mathcal{P}_{i+1}$ while taking into account the executions of the observed processes $\mathcal{P}_{i+2},\ldots,\mathcal{P}_n$. Second, we compute the automaton $\mathcal{A}_i$ which accepts the sequences of states of $\mathcal{P}_i$ that the network composed of the processes $\mathcal{P}_1,\ldots,\mathcal{P}_{i-1}$ needs to observe in order to go from $(p_1,\ldots,p_{i-1})$ to a configuration in $p'_1\Gamma^*_1 \times \cdots \times p'_{i-1}\Gamma^*_{i-1}$ (this automaton can be computed in top-down manner). Let $\mathcal{P}'_i = \mathcal{P}_i \otimes \mathcal{A}_i$. $\mathcal{P}'_i$ has the same behavior as $\mathcal{P}_i$ while ensuring the sequence of states needed by the processes $\mathcal{P}_1,\ldots,\mathcal{P}_{i-1}$. Then, the network composed of $(\mathcal{P}'_i,\mathcal{P}'_{i+1})$ is a APN$_{\mathsf{obs}}$ of depth 1. Therefore, we can apply the construction described in Section 5 to compute a 2-dim transducer representing its reachability set. This computed set is the projection of the reachable configurations of the APN$_{\mathsf{obs}}$ N on the components i and $i+1$. But before going into details , we need to extend some definitions.

Definition 8. *Given the constrained pushdown system $\mathcal{P}_i = (P_i,\Gamma_i,\Delta_i)$, with $i \in \{1,\ldots,n-1\}$, and an automaton $\mathcal{A} = (A,P_i,\delta,I,F)$, let $\mathcal{P}_i \otimes \mathcal{A} = (P_i \times A,\Gamma_i,\Delta'_i)$ be the constrained pushdown system where Δ'_i is the set of transition rules such that:*

- $P_{i+1} : \langle (p,s),\varepsilon\rangle \hookrightarrow \langle (p,s'),\varepsilon\rangle \in \Delta'_i$ *iff there* $(s,p,s') \in \delta$*,*
- $\phi : \langle (p,s),u\rangle \hookrightarrow \langle (p',s'),u'\rangle \in \Delta'_i$ *iff* $\phi : \langle p,u\rangle \hookrightarrow \langle p',u'\rangle \in \Delta_i$ *and* $(s,p',s') \in \delta$*.*

Now, we are ready to describe formally our decision procedure: In the first step, we compute inductively a sequence of finite state automata $\mathcal{A}_j = (A_j, P_j, \delta_j, s_j, f_j)$, where s_j is the initial state and f_j is the acceptor state, defined as follows:

1. $\mathcal{A}_1$ recognizes P_1^* *(since there is no constraint on the executions of process $\mathcal{P}_1$).*
2. $\forall j \in \{1, \ldots, i-1\}$, $\mathcal{A}_{j+1}$ recognizes the regular language $Traces_{\widehat{\mathcal{P}_j \otimes \mathcal{A}_j}}((p_j, s_j), (p'_j, f_j)\Gamma_j^*) \uparrow$.

In the second step, we compute inductively a sequence of finite state automata $\mathcal{B}_k = (B_k, P_k, \delta_k, s_k, f_k)$, where s_k is the initial state and f_k is the acceptor state, defined as follows:

1. $\mathcal{B}_{n-1}$ recognizes $Traces_{\mathcal{P}_n}(p_n, p'_n\Gamma_n^*) \downarrow$.
2. $\forall k \in \{i+2, \ldots, n-1\}$, $\mathcal{B}_{k-1}$ recognizes the regular language $L_{\widetilde{\mathcal{P}_k \odot \mathcal{B}_k}}((p_k, s_k), (p'_k, f_k)\Gamma_k^*) \downarrow$.

Now, Let us consider the pushdown system $\mathcal{P}_{i+1} \odot \mathcal{B}_{i+1} = (P_{i+1} \times B_{i+1}, P_{i+2}, \Gamma_{i+1}, \Delta'_{i+1})$, if $i < n-1$, and the constrained pushdown system $\mathcal{P}_i \otimes \mathcal{A}_i = (P_i \times A_i, \Gamma_i, \Delta'_i)$. Then, we can compute an $\mathrm{APN_{obs}}$ $N_{i,i+1} = (\mathcal{P}'_i, \mathcal{P}'_{i+1}, \{(1,2)\})$ as follows:

1. if $i < n-1$, then we have $\mathcal{P}'_{i+1} = (P_{i+1} \times B_{i+1}, \Gamma_{i+1}, \Delta"_{i+1})$, where $\langle s, u\rangle \hookrightarrow \langle s', u'\rangle \in \Delta"_{i+1}$ if and only if $\exists(\langle s, u\rangle \overset{a}{\hookrightarrow} \langle s', u'\rangle) \in \Delta'_{i+1}$ for some $a \in P_{i+2}$.
2. if $i = n-1$, then we have that $\mathcal{P}'_n = \mathcal{P}_n$.
3. $\mathcal{P}'_i = (P_i \times A_i, \Gamma_i, \Delta"_i)$ is a constrained pushdown system with $\Delta"_i$ is a finite set of transition rules such that: $\phi : \langle (p, s), u\rangle \hookrightarrow \langle (p', s'), u'\rangle$ is in Δ'_i iff $\phi \times B_{i+1} : \langle (p, s), u\rangle \hookrightarrow \langle (p', s'), u'\rangle$ is in $\Delta"_i$.

Then, we obtain that:

Theorem 10. *$((p'_i, f_i)w'_i, (p'_{i+1}, f_{i+1})w'_{i+1}) \in post^*_{N_{i,i+1}}(((p_i, s_i), (p_{i+1}, s_{i+2})))$ if and only $(p'_1w'_1, \ldots, p'_nw'_n) \in post^*_N((p_1, \ldots, p_n))$ for some $w'_j \in \Gamma_j^*$ with $j \notin \{i, i+1\}$.*

Hence, given an index $i \in \{1, \ldots, n-1\}$, we can construct a 2-tape automaton $T_{i,i+1}$ that recognizes $((p'_i, f_i)w'_i, (p'_{i+1}, f_{i+2})w'_{i+1})$. Notice that $T_{i,i+1} \circ_{2,1} T_{i+1,i+1}$ recognizes an over approximation of the projection of the set reachable configurations of processes $\mathcal{P}_i$, $\mathcal{P}_{i+1}$, and $\mathcal{P}_{i+2}$.

N Proof of Proposition 6

Proposition 6 *For every APN_{obs} $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$, the reachability problem can be reduced to the same problem for an APN_{lc} $H = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, M, C)$. Moreover, for every linear APN_{lc} $H = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, M, C)$, the reachability problem between two configurations can be reduced to the same problem for a linear APN_{obs} $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ such that $\forall i \in \{1, \ldots, n\}$, $P_i = P'_i \times M' \times M'$ where $M' = M \cup \{\#\}$ (with $\# \notin M$).*

Proof:

It is easy to see that a linear $\mathrm{APN_{obs}}$ $N = (\mathcal{P}_1, \ldots, \mathcal{P}_n, R)$ can be simulated by a linear $\mathrm{APN_{lc}}$ $H = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, C, M)$ where the observed process $\mathcal{P}_i$ sends its state in the channel $(i, i-1)$ so that process $\mathcal{P}_{i-1}$ can access to it. In addition, we need to add loops to ensure that each state can be sent an arbitrary number of times in the channel of H, because in N each state can be observed an arbitrary number of times. Hence, For every linear APN$_{\mathsf{obs}}$ $N = (\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$, the reachability problem can be reduced to the same problem for a linear APN$_{\mathsf{lc}}$ $H = (\mathcal{P}'_1,\ldots,\mathcal{P}'_n,M,C)$ such that $\forall i \in \{1,\ldots,n\}$, $P'_i = P_i$, $M = \bigcup_{i=1}^{n} P_i$, and $(i,j) \in C$ iff $(j,i) \in R$. Thanks to Proposition 3, we get that every APN$_{\mathsf{obs}}$ $N = (\mathcal{P}_1,\ldots,\mathcal{P}_n,R)$, the reachability problem can be reduced to the same problem for a linear APN$_{\mathsf{lc}}$ $H = (\mathcal{P}'_1,\ldots,\mathcal{P}'_n,M,C)$.

For the other direction, let $H = (\mathcal{P}_1,\ldots,\mathcal{P}_n,C,M)$ be a linear APN$_{\mathsf{lc}}$. H can be simulated by the linear APN$_{\mathsf{obs}}$ $N = (\mathcal{P}'_1,\ldots,\mathcal{P}'_n,R)$ where $R = \{(i,i+1) \ : \ \forall i \in \{1,\ldots,n-1\}\}$ such that for every i, the process $\mathcal{P}'_i$ behaves like the process $\mathcal{P}_i$ and its states are composed from tuples of the form (p,m,m') where p is the current control state of $\mathcal{P}_i$, m is the last message sent to $\mathcal{P}_{i-1}$, and m' is the last message received from $\mathcal{P}_{i+1}$. The action where $\mathcal{P}_i$ receives a message is simulated by an action where $\mathcal{P}'_i$ observes the control state of $\mathcal{P}'_{i+1}$. In addition, we need to make sure that the number of messages observed by $\mathcal{P}'_i$ does not exceed the ones that can be received by $\mathcal{P}_i$. This is ensured by imposing that if $\mathcal{P}_i$ receive a message a in H, then $\mathcal{P}'_i$ moves to a state where its third dimension is labeled by a, i.e. $m' = a$, and from this configuration it will ignore all observed a's. $\mathcal{P}'_i$ will resume its execution when it observes $\mathcal{P}'_{i+1}$ sending a new message different from the last message saved in m'. Now, if process $\mathcal{P}_i$ send a message b to $\mathcal{P}_{i-1}$, then $\mathcal{P}'_i$ goes to a control state where $m = b$ to allow $\mathcal{P}'_{i-1}$ to read this message and has a nondeterministic move to a special state where $m = \sharp$ to express that it has already sent b and that it is waiting to send a new message. Formally, let $N = (\mathcal{P}'_1,\ldots,\mathcal{P}'_n,R)$ be a linear APN$_{\mathsf{obs}}$ where $\forall i \in \{1,\ldots,n\}$, $\mathcal{P}'{}_i = (P'_i,\Gamma_i,\Delta'_i)$ is a constrained pushdown system such that $P'_i = P_i \times M' \times M'$, with $M' = M \cup \{\sharp\}$, and Δ'_i is the smallest set of transition rules satisfying the following conditions:

- $P_{i+1} \times M' \times M'$: $\langle (p,m_1,m_2),u\rangle \hookrightarrow \langle (p',m'_1,m'_2),u'\rangle$ is in Δ'_i if $\exists ((p,u) \overset{(!,a,i-1)}{\hookrightarrow} (p',u')) \in \Delta_i$, $m'_1 = a$ and $m'_2 = m'_1$,
- $P_{i+1} \times M' \times M'$: $\langle (p,m_1,m_2),\varepsilon\rangle \hookrightarrow \langle (p,m'_1,m'_2),\varepsilon\rangle$ is in Δ'_i if $m'_1 = \sharp$ and $m'_2 = m'_1$,
- $P_{i+1} \times \{a\} \times M'$: $\langle (p,m_1,m_2),u\rangle \hookrightarrow \langle (p',m'_1,m'_2),u'\rangle$ is in Δ'_i if $\exists ((p,u) \overset{(?,a,i+1)}{\hookrightarrow} (p',u')) \in \Delta_i$, $m'_1 = m_1$, $m_2 \neq a$, and $m'_2 = a$,
- $P_{i+1} \times \{\sharp\} \times M'$: $\langle (p,m_1,m_2),\varepsilon\rangle \hookrightarrow \langle (p,m'_1,m'_2),\varepsilon\rangle$ is in Δ'_i if $m'_1 = m_1$, $m_2 \neq \sharp$, and $m'_2 = \sharp$,
- $P_{i+1} \times M' \times M'$: $\langle (p,m_1,m_2),\varepsilon\rangle \hookrightarrow \langle (p',m_1,m_2),\varepsilon\rangle$ is in Δ'_i if $\exists ((p,u) \overset{nop}{\hookrightarrow} (p',u')) \in \Delta_i$.

Then, we can show easily that:

Proposition 9. *$\langle p_1w_1,\ldots,p_nw_n,\mathcal{V}_0\rangle \Longrightarrow_H^* \langle p'_1w'_1,\ldots,p'_nw'_n,\mathcal{V}\rangle$ for some $\mathcal{V}$, with $\forall i \in \{1,\ldots,n-1\}$ $\mathcal{V}_0(i+1,i) = \varepsilon$, iff $((p_1,\sharp,\sharp)w_1,\ldots,(p_n,\sharp,\sharp)w_n) \Longrightarrow_N^* ((p'_1,\sharp,\sharp)w'_1,\ldots,(p'_n,\sharp,\sharp)w'_n)$*

Finally, note that the reachability problem between two configuration for a linear APN$_{\mathsf{lc}}$ can be always reduced to checking whether $\langle p_1w_1,\ldots,p_nw_n,\mathcal{V}_0\rangle \Longrightarrow_H^* \langle p'_1w'_1,\ldots,p'_nw'_n,\mathcal{V}\rangle$ for some $\mathcal{V}$, with $\forall i \in \{1,\ldots,n-1\}$ $\mathcal{V}_0(i+1,i) = \varepsilon$. $\square$

O Dynamic Acyclic Observation Relation Pushdown Network

In this section, we show that when the observation relation of the network of $\text{APN}_{\mathsf{obs}}$ changes dynamically, then, the reachability problem between two configurations becomes undecidable. The undecidability result holds even after only one dynamic change.

To see that, consider the pair of networks (N_1, N_2) where for every $j \in \{1,2\}$, $N_j = (\mathcal{P}_1^j, \ldots, \mathcal{P}_n^j, R_j)$ is an $\text{APN}_{\mathsf{obs}}$ where for $i \in \{1,\ldots,n\}$, $\mathcal{P}_i^j = (P_i, \Gamma_i, \Delta_i^j)$ and R_1, R_2 are acyclic binary relations (not necessarily antisymmetric). Notice that, for every $i \in \{1,\ldots,n\}$, $\mathcal{P}_i^1$ and $\mathcal{P}_i^2$ have the same set of states and stack alphabet.

We can think of the network N_j, for every $j \in \{1,2\}$ 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 rules Δ_i^j (behaves like process $\mathcal{P}_i^j$), and where these processes observe each other according to the structure R_j.

Theorem 11. *Given two configurations $c, c' \in P_i\Gamma_i^* \times \cdots \times P_n\Gamma_n^*$, checking whether $c' \in post^+_{N_2}\left(post^+_{N_1}(c)\right)$ is undecidable. This holds even if the network contains 3 processes.*

Proof (Sketch):

We show that solving this problem would imply a decision procedure for the Post's Correspondence Problem. 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 will construct a couple of networks (N_1, N_2) such that:

- for every $j \in \{1,2\}$, $N_j = (\mathcal{P}_1^j, \mathcal{P}_2^j, \mathcal{P}_3^j, R_j)$ is an $\text{APN}_{\mathsf{obs}}$;
- for every $j \in \{1,2\}$, for $i \in \{1,2,3\}$, $\mathcal{P}_i^j = (P_i, \Gamma_i, \Delta_i^j)$ is a constrained pushdown system;
- we consider $R_1 = \{(1,2),(2,3)\}$ and $R_2 = \{(2,3),(3,1)\}$ be two acyclic relations.

Then, deciding whether

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

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 can be done by considering rules as the ones described in the proof of Theorem 3. 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 $i_1 i_2 \cdots i_k$ is a subsequence of $l_1 l_2 \cdots l_n$ which is a subsequence of $j_1 j_2 \cdots j_m$.
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 $f_1 f_2 \cdots f_z$ is a subsequence of $g_1 g_2 \cdots g_r$ which is a subsequence of $h_1 h_2 \cdots h_s$.

The two items above infer that from a configuration where the three processes have empty stacks, we can reach a configuration where the four processes have empty stacks by first executing $\mathcal{N}_1$ and then $\mathcal{N}_2$ iff the sequences of indices $h_1 h_2 \cdots h_s$, $g_1 g_2 \cdots g_r$, $f_1 f_2 \cdots f_z$, $i_1 i_2 \cdots i_k$, $l_1 l_2 \cdots l_n$, and $j_1 j_2 \cdots j_m$ are the same, i.e., iff $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$. $\square$