Boundedness Undecidability for Synchronized Nets

Raymond Devillers and Laurent Van Begin

Université Libre de Bruxelles

Keywords. Synchronized nets, concurrency, boundedness, decidability.

1 Introduction and Motivation

Synchronized Petri nets are a well-known class of non-autonomous models derived from usual Petri nets, devised to fit the needs of the logic controllers modelling. In those models, a transition may only fire if the needed resources are available AND if some external event associated with the transition occurs, and in that case the transition must fire, unless there is a true conflict with other transitions in the same situation. Those models are detailed for instance in [1].

In the same book, the authors use a Karp and Miller coverability tree to visualize in a finite way the evolutions of a possibly unbounded synchronized Petri net. However, as it occurs, the algorithm proposed in [1] to build coverability trees may compute trees that do not faithfully represent the possible evolutions of the synchronized Petri nets, for various reasons; for instance, synchronized nets are not monotonic in the sense that if a marking allows an evolution, a larger marking does not always allow the very same evolution (sometimes, a larger evolution is mandatory); also, the successors of a node with some ω-marker do not always have this marker (an arbitrarily large marking may be emptied when some event occurs, whatever large it was).

The problem then arises to know if the coverability tree construction may be modified in order to yield a faithful representation. That problem is very important since coverability trees are very useful tools to analyze nets. More precisely, coverability trees are aimed to answer questions (among others) like: "*is a net bounded?*", "*which places are unbounded?*" or "*which places may be simultaneously unbounded?*" Unfortunately, we shall see that this hope is deceived. Indeed, we shall show that the boundedness problem is undecidable for this class of systems. To do so, we shall proceed by a reduction from the similar problem for 2-counters systems. That reduction is inspired from reductions proposed in [5, 4] to obtain results for another kind of extension of Petri nets.

We also show how to adapt our reduction for timed versions of Petri nets, for which a coverability tree-based approach is also proposed in [1].

2 Synchronized Petri nets

We present in this section the model of *synchronized Petri nets*, a popular extension of the Petri nets model that allow to model concurrent systems embedded into an environment. Then, we present the coverability sets, trees and graphs, which are powerful tools to analyze nets.

Definition 1 A (finite) *Synchronized Petri net*, SPN for short, is a tuple $\langle P, T, \mathsf{Pre}, \mathsf{Post}, E, \mathsf{sync}\rangle$ where P is a finite set of places, T is a finite set of transitions, $\mathsf{Pre}, \mathsf{Post} : T \mapsto 2^P$, E is a finite set of events, $\mathsf{sync} : T \mapsto E$ is a function that labels each transition with an event.

As usual, a *marking* m is a function that assigns tokens to places, i.e. $\mathsf{m} : P \mapsto \mathbb{N}$. In the following, markings will be sometime seen as vectors in $\mathbb{N}^{|P|}$ where the *ith* component gives the tokens inside the *ith* place. $\mathsf{tokens}(.)$ is a function from markings to $\mathbb{N}$ such that $\mathsf{tokens}(\mathsf{m}) = \Sigma_{p \in P}\mathsf{m}(p)$.

An elementary firing sequence of the SPN $\langle P, T, \mathsf{Pre}, \mathsf{Post}, E, \mathsf{sync}\rangle$ for a marking m, EFS for short, is a nonempty finite set of transitions $Elem \subseteq T$ that satisfies the following three conditions: (**i**) $\exists E_i \in E : \forall tr \in Elem : \mathsf{sync}(tr) = E_i$ (transitions are labelled by the same event); (**ii**) $\forall p \in P$: $\mathsf{m}(p) \geq \Sigma_{tr \in Elem}\mathsf{Pre}(tr, p)$ (there are enough tokens in places to fire all the transitions in $Elem$ at the same time); (**iii**) there is no set of transitions $Elem'$ with $Elem \subset Elem'$ that satisfies the two first conditions ($Elem$ is a maximal set that satisfies the conditions **(i)** and **(ii)**).

An EFS $Elem$ is a E_i-EFS if and only if $\forall tr \in Elem : \mathsf{sync}(tr) = E_i$. When event E_i occurs, an E_i-EFS is applied from the current marking m and leads to a new current marking m', noted $\mathsf{m} \xrightarrow{E_i} \mathsf{m}'$. The marking m' is computed as follows: $\forall p \in P : \mathsf{m}'(p) = \mathsf{m}(p) - \Big(\Sigma_{tr \in Elem}\mathsf{Pre}(tr, p)\Big) + \Big(\Sigma_{tr \in Elem}\mathsf{Post}(tr, p)\Big)$. If there is no E_i-EFS from m, i.e., if no E_i-labelled transition may be fired from m, then the event E_i is lost and the net stays in m.

The semantics presented here is the *single server* semantics, i.e., there is at most one instance for each transition in EFS. In [1], a *infinite server* semantics is also presented where EFS are multi-set instead of set, i.e. there is no bound on the number of instances for each transition in an EFS.

Given a SPN $\mathcal{N} = \langle P, T, \mathsf{Pre}, \mathsf{Post}, E, \mathsf{sync}\rangle$ and an initial (stable) marking m_0, $\mathsf{Reach}(\mathcal{N}, \mathsf{m}_0) = \{\mathsf{m} \mid \exists E_1, \ldots, E_k \in E : \mathsf{m}_0 \xrightarrow{E_1} \mathsf{m}_1 \ldots \xrightarrow{E_k} \mathsf{m}\}$.

The *boundedness problem for* SPN is defined as follows: "*Given a* SPN $\mathcal{N}$ *with the initial marking* m_0*, is* $\mathsf{Reach}(\mathcal{N}, \mathsf{m}_0)$ *finite?*".

In the rest of the paper, we prove that we cannot build algorithmically a coverability set, and hence a coverability tree or graph, for an SPN in all generality.

3 Two-counter machines

A *two-counter machine*, 2CM for short, is a tuple $C = \langle c_1, c_2, L, \mathsf{Instr}\rangle$ where c_1, c_2 are two counters taking their values in $\mathbb{N}$; L is a finite non-empty set of locations; Instr is a function that labels each location $l \in L$ with an instruction that has one of the three following forms: (**i**) $l : c_j := c_j + 1;$ `goto` l';, where $j \in \{1, 2\}$ and $l' \in L$, this is called an increment; (**ii**) $l : c_j := c_j - 1;$ `goto` l';, where $j \in \{1, 2\}$ and $l' \in L$, this is called a decrement; (**iii**) $l :$ `if` $c_j = 0$ `then goto` l' `else goto` l'';, where $j \in \{1, 2\}$ and $l', l'' \in L$, this is called a zero-test. Those instructions have their usual obvious semantics; in particular, decrement can only be done if the value of the counter is strictly greater than zero.

A *configuration* of a 2CM C is a tuple $\langle loc, v^1, v^2\rangle$ where $loc \in L$ is the value of the program counter and v^1, respectively v^2, is a natural number that gives the valuation of the counter c_1, respectively c_2. A *computation* γ of a 2CM C is either a finite sequence of configurations $\langle loc_1, v^1_1, v^2_1\rangle, \langle loc_2, v^1_2, v^2_2\rangle, \ldots, \langle loc_r, v^1_r, v^2_r\rangle$, in which case $|\gamma| = r$, or an infinite sequence of configurations $\langle loc_1, v^1_1, v^2_1\rangle$, $\langle loc_2, v^1_2, v^2_2\rangle, \ldots, \langle loc_r, v^1_r, v^2_r\rangle, \ldots$, in which case $|\gamma| = \infty$, such that : (**i**) "Initialization": $loc_1 = l_1$, $v^1_1 = 0$, and $v^2_1 = 0$, i.e. a computation starts in l_1 and the value of counters is zero; (**ii**) "Consecution": for each $i \in \mathbb{N}$ such that $1 \le i < |\gamma|$ we have that $\langle loc_{i+1}, v^1_{i+1}, v^2_{i+1}\rangle$ is the configuration obtained from $\langle loc_i, v^1_i, v^2_i\rangle$ by applying the instruction $\mathsf{Instr}(loc_i)$.

The *boundedness problem for* 2CM is defined as follows: "*Given a* 2CM $C = \langle c_1, c_2, L, \mathsf{Instr}\rangle$, *is there* $c \in \mathbb{N}$ *such that for all reachable configurations* $\langle loc, v_1, v_2\rangle$ *in* C *we have* $v_1 + v_2 \le c$ *?*"

It is well-known that the boundedness problem cannot be answered completely with an algorithm [3]:

Theorem 1 (from [3]) *The boundedness problem for* 2CM *is undecidable.*

4 Undecidability of the boundedness problem for SPN

In this section we present a reduction of the boundedness problem for 2CM to the boundedness problem of SPN. Since the boundedness problem for 2CM is undecidable (Theorem 1), that reduction allows us to conclude the undecidability of the boundedness problem for SPN. First, we show how to build a SPN that *weakly* simulates a 2CM (i.e., for infinite evolutions). Then, we prove that the SPN is bounded if and only if the 2CM is bounded.

Construction. For any 2CM $C = \langle c_1, c_2, L = \{l_1, l_2, \ldots, l_u\}, \mathsf{Instr}\rangle$, we construct a ($e$-free) SPN $\mathcal{N}_C = \langle P, T, \mathsf{Pre}, \mathsf{Post}, E, \mathsf{sync}\rangle$, called the *simulation artefact*, defined as follows. The set of places P is equal to $\{c_1, c_2, l_1, l_2, \ldots, l_u, ok\} \cup \{l^T_i, l'_i, l''_i \mid \mathsf{Instr}(l_i) \text{ is a zero-test}\}$. The places c_1 and c_2 will be used to keep track of the values of the two counters of C; $l_1, l_2, \cdots, l_u$ (called the *control*

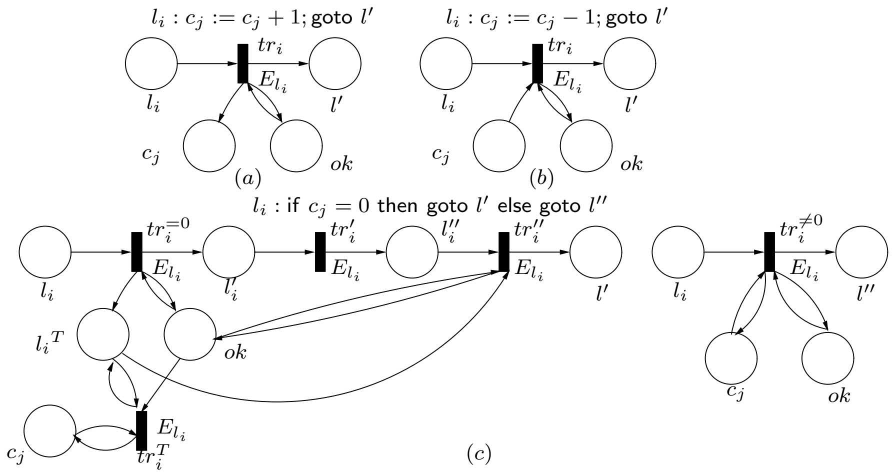

Figure 1: (a) simulation of an increment, (b) simulation of a decrement, (c) simulation of a test for zero in the case where $c_j \neq 0$ (transition $t_i^{\neq 0}$) and when $c_j = 0$.

places) will be used to keep track of the program counter of C; the places l_i^T, l_i', l_i'' will be used to test if the place c_j is empty; ok will be used to detect mistakes in a simulation of 2CM. The set E of events is the set $\{E_{l_1}, \ldots, E_{l_u}\}$. The set of transitions T is the smallest set of transitions such that for each $l_i \in L$:

- **(i)** if $\mathsf{Instr}(l_i)$ is of the form $c_j := c_j - 1; \texttt{goto}\ l'$, then T contains the transition tr_i such that $\mathsf{sync}(tr_i) = E_{l_i}$, $\mathsf{Pre}(tr_i) = \{l_i, c_j, ok\}$; $\mathsf{Post}(tr_i) = \{l', ok\}$;
- **(ii)** if $\mathsf{Instr}(l_i)$ is of the form $c_j := c_j + 1; \texttt{goto}\ l'$, then T contains the transition tr_i such that $\mathsf{sync}(tr_i) = E_{l_i}$, $\mathsf{Pre}(tr_i) = \{l_i, ok\}$; $\mathsf{Post}(tr_i) = \{l', c_j, ok\}$;
- **(iii)** if $\mathsf{Instr}(l_i)$ is of the form $\texttt{if}\ c_j = 0\ \texttt{then goto}\ l'\ \texttt{else goto}\ l''$ then T contains five transitions $tr_i^{=0}$, $tr_i^{\neq 0}$, tr_i^T, tr_i' and tr_i'' defined as follows:
  - **(a)** $tr_i^{=0}$ is such that $\mathsf{sync}(tr_i^{=0}) = E_{l_i}$ and $\mathsf{Pre}(tr_i^{=0}) = \{l_i, ok\}$; $\mathsf{Post}(tr_i^{=0}) = \{l_i', ok, l_i^T\}$;
  - **(b)** tr_i^T is such that $\mathsf{sync}(tr_i^T) = E_{l_i}$ and $\mathsf{Pre}(tr_i^T) = \{c_j, ok, l_i^T\}$; $\mathsf{Post}(tr_i^T) = \{l_i^T, c_j\}$;
  - **(c)** tr_i' is such that $\mathsf{sync}(tr_i') = E_{l_i}$ and $\mathsf{Pre}(tr_i') = \{l_i'\}$; $\mathsf{Post}(tr_i') = \{l_i''\}$;
  - **(d)** tr_i'' is such that $\mathsf{sync}(tr_i'') = E_{l_i}$ and $\mathsf{Pre}(tr_i'') = \{l_i'', l^T{}_i, ok\}$; $\mathsf{Post}(tr_i'') = \{l', ok\}$;
  - **(e)** $tr_i^{\neq 0}$ is such that $\mathsf{sync}(tr_i^{\neq 0}) = E_{l_i}$ and $\mathsf{Pre}(tr_i^{\neq 0}) = \{l_i, c_j, ok\}$; $\mathsf{Post}(tr_i^{\neq 0}) = \{l'', c_j, ok\}$.

This construction is schematized in Figure 1.

The marking m_0 is such that $\mathsf{m}_0(l_1) = 1$,$\mathsf{m}_0(ok) = 1$ and for all $p \in P \setminus \{l_1, ok\} : \mathsf{m}_0(p) = 0$. We define $\alpha(\mathsf{m}) = \mathsf{m}$ if $\exists \ell \in L : \mathsf{m}(\ell) \neq 0$, $\alpha(\mathsf{m}) = \varepsilon$ otherwise. α is extended to sequences of markings as follows: $\alpha(\mathsf{m}_0\mathsf{m}_1\ldots\mathsf{m}_k) = \alpha(\mathsf{m}_0)\alpha(\mathsf{m}_1\ldots\mathsf{m}_k)$. $\mathsf{m}\in\alpha(\mathsf{m}_0\ldots\mathsf{m}_k)$ denotes that m appears in $\alpha(\mathsf{m}_0\ldots\mathsf{m}_k)$.

We say that $\mathsf{m}_1 \xrightarrow{E_1} \ldots \xrightarrow{E_{k-1}} \mathsf{m}_k$ simulates a computation of C if and only if $\alpha(\mathsf{m}_1\ldots\mathsf{m}_k) = \mathsf{m}_{i_1}\ldots\mathsf{m}_{i_v}$ and there exists a finite computation $\langle loc_1, v_1^1, v_1^2\rangle, \ldots, \langle loc_{i_v}, v_{i_v}^1, v_{i_v}^2\rangle$ of the 2CM C such that for all $1 \le j \le i_v : \mathsf{m}_{i_j}(loc_j) = 1, \mathsf{m}_{i_j}(c_1) = v_j^1, \mathsf{m}_{i_j}(c_2) = v_j^2, \mathsf{m}_{i_j}(ok) = 1$ and $\forall p \in P\setminus\{loc_j, c_1, c_2, ok\} : \mathsf{m}_{i_j}(p) = 0$. Notice that if $\mathsf{m}_1 \xrightarrow{E_1} \ldots \xrightarrow{E_{k-1}} \mathsf{m}_k$ simulates a computation of C and $\mathsf{m}'_1 \xrightarrow{E'_1} \ldots \xrightarrow{E'_{k'-1}} \mathsf{m}'_{k'}$ such that $\alpha(\mathsf{m}_1\ldots\mathsf{m}_k) = \alpha(\mathsf{m}'_1\ldots\mathsf{m}'_{k'})$ then $\mathsf{m}'_1 \xrightarrow{E'_1} \ldots \xrightarrow{E'_{k'-1}} \mathsf{m}'_{k'}$ simulates the same computation. In particular, if $\mathsf{m}_1 \xrightarrow{E_1} \ldots \xrightarrow{E_{k-1}} \mathsf{m}_k$ simulates a computation of C with $\alpha(\mathsf{m}_1\ldots\mathsf{m}_k) = \mathsf{m}_{i_1}\ldots\mathsf{m}_{i_v}$ and $\mathsf{m}_k \ne \mathsf{m}_{i_v}$ there is a shortest prefix $\mathsf{m}_1 \xrightarrow{E_1} \ldots \xrightarrow{E_{k-1}} \mathsf{m}_{i_v}$ with the same property.

We now prove that C is bounded if and only if $\mathcal{N}_C$ is bounded. In order to prove that result, we need the three following lemmas.

Lemma 1 *Let a* 2CM C*, its simulation artefact* $\mathcal{N}_C$ *and* $\mathsf{m}_0 \xrightarrow{E_0} \ldots \xrightarrow{E_{k-1}} \mathsf{m}_k$ *in* $\mathcal{N}_C$*. If* $\mathsf{m}_k(ok) = 1$*, then* $\mathsf{m}_0 \xrightarrow{E_0} \ldots \xrightarrow{E_{k-1}} \mathsf{m}_k$ *simulates a computation of* C*.*

Proof. Let $\alpha(\mathsf{m}_0\ldots\mathsf{m}_k) = \mathsf{m}_{i_1}\ldots\mathsf{m}_{i_j}$. We exhibit a proof by induction on j, about the shortest prefix of $\mathsf{m}_0\ldots\mathsf{m}_k$ with that property.

$j = 1$. $\mathsf{m}_{i_1} = \mathsf{m}_0$ and $\mathsf{m}_0(l_1) = 1, \mathsf{m}_0(c_1) = 0, \mathsf{m}_0(c_2) = 0, \mathsf{m}_0(ok) = 1$ and $\forall p \in P \setminus \{l_1, c_1, c_2, ok\} : \mathsf{m}_0(p) = 0$. Hence, m_{i_1} simulates the computation $\langle l_1, 0, 0\rangle$ composed of the initial configuration of C.

$j = v$. Suppose that the property holds for $j \le v-1$. Let us prove that the property holds for $j = v$. By induction hypothesis and the remarks above, we know that $\mathsf{m}_0 \xrightarrow{E_1} \mathsf{m}_1 \ldots \xrightarrow{E_{i_{v-2}}} \mathsf{m}_{i_{v-1}}$ simulates a computation $\langle loc_1, v_1^1, v_1^2\rangle, \ldots, \langle loc_{i_{v-1}}, v_{i_{v-1}}^1, v_{i_{v-1}}\rangle$. In particular, $\mathsf{m}_{i_{v-1}}(loc_{i_{v-1}}) = 1, \mathsf{m}_{i_{v-1}}(c_1) = v_{i_{v-1}}^1, \mathsf{m}_{i_{v-1}}(c_2) = v_{i_{v-1}}^2, \mathsf{m}_{i_{v-1}}(ok) = 1$ and $\forall p \in P\setminus\{loc_{i_{v-1}}, c_1, c_2, ok\} : \mathsf{m}_{i_{v-1}}(p) = 0$. We consider three cases: **(i)** $\mathsf{Instr}(loc_{i_{v-1}}) = loc_{i_{v-1}} : c_j := c_j + 1; \texttt{goto}\ l;$ The only transition that can be fired from $\mathsf{m}_{i_{v-1}}$ is $tr_{i_{v-1}}$ and it is labelled by $E_{i_{v-1}}$. When that event occurs, $\mathcal{N}_C$ reaches a marking m such that $\mathsf{m}(l) = 1, \mathsf{m}(c_j) = v_{i_{v-1}}^j + 1, \mathsf{m}(c_{3-j}) = v_{i_{v-1}}^{3-j}, \mathsf{m}(ok) = 1$ and $\forall p \in \{l, c_1, c_2, ok\} : \mathsf{m}(p) = 0$. Hence, $\mathsf{m}_{i_j} = \mathsf{m}$. Moreover, the successor of $\langle loc_{i_{v-1}}, v_{i_{v-1}}^1, v_{i_{v-1}}^2\rangle$ is $\langle l, v^1, v^2\rangle$ where $v^j = v_{i_{v-1}}^j + 1$ and $v^{3-j} = v_{i_{v-1}}^{3-j}$; **(ii)** $\mathsf{Instr}(loc_{i_{v-1}}) = loc_{i_{v-1}} : c_j := c_j - 1; \texttt{goto}\ l;$ The only transition that can be fired from $\mathsf{m}_{i_{v-1}}$ is $tr_{i_{v-1}}$ and it is labelled by $E_{i_{v-1}}$. When that event occurs, $\mathcal{N}_C$ reaches a marking m such that $\mathsf{m}(l) = 1, \mathsf{m}(c_j) = v_{i_{v-1}}^j - 1, \mathsf{m}(c_{3-j}) = v_{i_{v-1}}^{3-j}, \mathsf{m}(ok) = 1$ and $\forall p \in \{l, c_1, c_2, ok\} : \mathsf{m}(p) = 0$. Hence, $\mathsf{m}_{i_j} = \mathsf{m}$. Moreover, the successor of $\langle loc_{i_{v-1}}, v_{i_{v-1}}^1, v_{i_{v-1}}^2\rangle$ is $\langle l, v^1, v^2\rangle$ where $v^j = v_{i_{v-1}}^j - 1$ and $v^{3-j} = v_{i_{v-1}}^{3-j}$; **(iii)** $\mathsf{Instr}(loc_{i_{v-1}}) =$ $\texttt{if}\ c_j = 0\ \texttt{then goto}\ l\ \texttt{else goto}\ l';$ We consider two cases: **(a)** $\mathsf{m}_{i_{v-1}}(c_j) = 0$. In that case, the only transition that can be fired from $\mathsf{m}_{i_{v-1}}$ is $tr_{l_{i_{v-1}}}^{=0}$. That transition is labelled by $E_{i_{v-1}}$. When that event occurs, $\mathcal{N}_C$ reaches a marking m such that $\mathsf{m}(l'_{i_{v-1}}) = 1, \mathsf{m}(c_1) = v_{i_{v-1}}^1$, $\mathsf{m}(c_2) =$

$v^2_{i_{v-1}}, \mathsf{m}(l^T_{i_{v-1}}) = 1, \mathsf{m}(ok) = 1$ and $\forall p \in P \setminus \{l'_{i_{v-1}}, l^T_{i_{v-1}}, c_1, c_2, ok\} : \mathsf{m}(p) = 0$. ¿From m the only transition that may be fired is $tr'_{i_{v-1}}$ and that transition is also labelled by event $E_{i_{v-1}}$. When that event occurs, $\mathcal{N}_C$ reaches a marking m' built from m by moving one token from $l'_{i_{v-1}}$ to $l''_{i_{v-1}}$. Finally, the only transition that can be fired from m' is $tr''_{i_{v-1}}$ and it is labelled by event $E_{i_{v-1}}$. When $E_{i_{v-1}}$ occurs, $\mathcal{N}_C$ reaches the marking m'' such that $\mathsf{m}''(l) = 1, \mathsf{m}''(c_1) = v^1_{i_{v-1}}, \mathsf{m}''(c_2) = v^2_{i_{v-1}}, \mathsf{m}''(ok) = 1$ and $\forall p \in P \setminus \{l, c_1, c_2, ok\} : \mathsf{m}''(p) = 0$. Hence, $\mathsf{m}_{i_j} = \mathsf{m}''$. Moreover, since $v^j_{i_{v-1}} = 0$, the successor of $\langle loc_{i_{v-1}}, v^1_{i_{v-1}}, v_{i_{v-1}}\rangle$ is $\langle l, v^1_{i_{v-1}}, v^2_{i_{v-1}}\rangle$; **(b)** $\mathsf{m}_{i_{v-1}}(c_j) > 0$. Two transitions may be fired from $\mathsf{m}_{i_{v-1}}$: $tr^{=0}_{i_{v-1}}$ and $tr^{\neq 0}_{i_{v-1}}$, and they are in conflict. Both are labelled by event $E_{i_{v-1}}$. But, when $E_{i_{v-1}}$ occurs, if $tr^{=0}_{i_{v-1}}$ is fired then $\mathcal{N}_C$ reaches a marking m such that $\mathsf{m}(l'_{i_{v-1}}) = 1, \mathsf{m}(c_1) = v^1_{i_{v-1}}, \mathsf{m}(c_2) = v^2_{i_{v-1}}, \mathsf{m}(l^T_{i_{v-1}}) = 1, \mathsf{m}(ok) = 1$ and $\forall p \in P \setminus \{l'_{i_{v-1}}, l^T_{i_{v-1}}, c_1, c_2, ok\} : \mathsf{m}(p) = 0$. The only event that allows transitions to be fired from m is $E_{l_{i_{v-1}}}$. When that event occurs, $tr^T_{i_{v-1}}$ and $tr''_{i_{v-1}}$ are fired together and a marking m' with $\mathsf{m}'(ok) = 0$ is reached. Since no transition increases the number of tokens in ok and $\mathsf{m}_k(ok) = 1$, it is not possible to reach m_k from $\mathsf{m}_{i_{v-1}}$ by firing $tr^{=0}_{i_{v-1}}$. Hence, $tr^{\neq 0}_{i_{v-1}}$ must be fired from $\mathsf{m}_{i_{v-1}}$ to reach m_k. When that transition is fired, a marking m is reached such that $\mathsf{m}(l'') = 1, \mathsf{m}(c_1) = v^1_{i_{v-1}}, \mathsf{m}(c_2) = v^2_{i_{v-1}}, \mathsf{m}(ok) = 1$ and $\forall p \in P \setminus \{l'', c_1, c_2, ok\} : \mathsf{m}(p) = 0$. Hence, $\mathsf{m}_{i_j} = \mathsf{m}$. Moreover, the successor of $\langle loc_{i_{v-1}}, v^1_{i_{v-1}}, v^2_{i_{v-1}}\rangle$ is $\langle l'', v^1_{i_{v-1}}, v^2_{i_{v-1}}\rangle$. In any case, we conclude that $\mathsf{m}_0 \xrightarrow{E_0} \mathsf{m}_1 \ldots \xrightarrow{E_{k-1}} \mathsf{m}_k$ simulates a computation of C. □

Lemma 2 *Let a* 2CM *C and its simulation artefact $\mathcal{N}_C$. For any marking m of $\mathcal{N}_C$ such that $\mathsf{m}(ok) = 0$, there does not exist an infinite sequence of markings $\mathsf{m}_1\mathsf{m}_2\ldots$ and an infinite sequence of events $E_1E_2\ldots$ such that $\mathsf{m} \xrightarrow{E_1} \mathsf{m}_1 \xrightarrow{E_2} \mathsf{m}_2 \ldots$*

Proof. If $\mathsf{m}(ok) = 0$ then the only transitions that may be fired from m are transitions $\{tr'_i \mid \mathsf{Instr}(l_i)$ is a zero-test$\}$. Those transitions move one token from l'_i to l''_i, hence leaving ok empty. Since the number of tokens assigned by m to each place is finite, the lemma holds. □

Lemma 3 *Let $\mathcal{N}_C$ be the simulation artefact associated to $C = \langle c_1, c_2, L, \mathsf{Instr}\rangle$. For any marking m reachable from m_0 in $\mathcal{N}_C$, we have that $\mathsf{m}(L \cup \{l'_i, l''_i \mid \mathsf{Instr}(l_i)$ is a zero-test$\}) = 1$.*

Proof. Immediate from the definitions of the various transitions in $\mathcal{N}_C$ and m_0. □

We are now ready to prove the following theorem:

Theorem 2 *Given a* 2CM *C and its simulation artefact $\mathcal{N}_C$, C is unbounded if and only if $\mathcal{N}_C$ with the initial marking m_0 is unbounded.*

Proof. $\Rightarrow$ For any computation of C $\langle loc_1, v^1_1, v^2_1\rangle, \ldots, \langle loc_k, v^1_k, v^2_k\rangle$ we can construct by induction on k an execution of $\mathcal{N}_C$ $\mathsf{m}_0 \xrightarrow{E_0} \ldots \xrightarrow{E_{n-1}} \mathsf{m}_n$ such that $\alpha(\mathsf{m}_0 \ldots \mathsf{m}_n) = \mathsf{m}_{i_1} \ldots \mathsf{m}_{i_k}$ and for all $1 \leq j \leq k : \mathsf{m}_{i_j}(loc_j) = 1, \mathsf{m}_{i_j}(c_1) = v^1_j, \mathsf{m}_{i_j}(c_2) = v^2_j, \mathsf{m}_{i_j}(ok) = 1$ and $\forall p \in P \setminus \{loc_j, c_1, c_2, ok\} :$ $\mathsf{m}_{i_j}(p) = 0$. We conclude that $\mathcal{N}_C$ is unbounded.

$\Leftarrow$ If $\mathcal{N}_C$ is unbounded, it has infinitely many reachable markings. Let us consider the reachability tree of $\mathcal{N}_C$ where the branches are stopped when they pass through a marking previously visited in the branch. That tree is infinite and each node has a finite number of successors (in fact in this case 1 or 2). By König's Lemma, there exists an infinite branch in that tree. Moreover, that branch passes through infinitely many reachable markings. Let that branch corresponds to $\mathsf{m}_0 \xrightarrow{E_1} \mathsf{m}_1 \xrightarrow{E_2} \ldots$ From Lemma 2, we know that for all $i \geq 0 : \mathsf{m}_i(ok) = 1$. Hence, following Lemma 1, $\forall i \geq 0 : \mathsf{m}_0 \xrightarrow{E_0} \ldots \xrightarrow{E_{i-1}} \mathsf{m}_i$ simulates a computation of C. Moreover, following Lemma 3, $\forall i \geq 0 : \mathsf{m}_i(\{l_1, \ldots, l_u\} \cup \{l'_i, l''_i \mid \mathsf{Instr}(l_i) \text{ is a zero-test}\}) = 1$. In particular, $\mathsf{m}_0(l_1) = 1$ and hence $\alpha(\mathsf{m}_0) = \mathsf{m}_0$. Suppose that $p \in \{l_1, \ldots, l_u\}$ and $\mathsf{m}_i(p) = 1$ for some $i \geq 0$, hence $\alpha(\mathsf{m}_0 \ldots \mathsf{m}_i) = \mathsf{m}_{i_1} \ldots \mathsf{m}_{i_k}$ with $\mathsf{m}_i = \mathsf{m}_{i_k}$. If $\mathsf{Instr}(p)$ is of the form $p\ :\ c_j := c_j + 1;$ $\texttt{goto}\ l;$, then it is easy to see that $\mathsf{m}_{i+1}(l) = 1$, i.e. $\alpha(\mathsf{m}_0 \ldots \mathsf{m}_{i+1}) = \mathsf{m}_{i_1} \ldots \mathsf{m}_{i_k}\mathsf{m}_{i+1}$; If $\mathsf{Instr}(p)$ is of the form $p\ :\ c_j := c_j - 1;$ $\texttt{goto}\ l;$, then it is easy to see that $\mathsf{m}_{i+1}(l) = 1$, i.e. $\alpha(\mathsf{m}_0 \ldots \mathsf{m}_{i+1}) = \mathsf{m}_{i_1} \ldots \mathsf{m}_{i_k}\mathsf{m}_{i+1}$; If $\mathsf{Instr}(p)$ is of the form $p\ :\ \texttt{if}\ c_j = 0\ \texttt{then goto}\ l\ \texttt{else goto}\ l';$, then it is easy to see that either $\mathsf{m}_{i+1}(l') = 1$, i.e. $\alpha(\mathsf{m}_0 \ldots \mathsf{m}_{i+1}) = \mathsf{m}_{i_1} \ldots \mathsf{m}_{i_k}\mathsf{m}_{i+1}$ or $\mathsf{m}_{i+3}(l) = 1$, i.e. $\alpha(\mathsf{m}_0 \ldots \mathsf{m}_{i+1}\mathsf{m}_{i+2}\mathsf{m}_{i+3}) = \mathsf{m}_{i_1} \ldots \mathsf{m}_{i_k}\mathsf{m}_{i+3}$, and $\mathsf{tokens}(\mathsf{m}_{i+1}) = \mathsf{tokens}(\mathsf{m}_{i+2}) = \mathsf{tokens}(\mathsf{m}_i) + 1$. It follows that, since for all $c \in \mathbb{N}$ there exists $i \in \mathbb{N} : \mathsf{tokens}(\mathsf{m}_i) > c + 3$, we have $\forall c \in \mathbb{N} : \exists i \in \mathbb{N} : \alpha(\mathsf{m}_0 \ldots \mathsf{m}_i) = \mathsf{m}_{i_1} \ldots \mathsf{m}_{i_k}$ with $\mathsf{tokens}(\mathsf{m}_{i_k}) > c+2$. Since $\mathsf{m}_{i_k}(ok) = 1$, there exists a computation of C $\langle loc_1, v_1^1, v_1^2\rangle, \ldots, \langle loc_k, v_k^1, v_k^2\rangle$ such that $\mathsf{m}_{i_k}(loc_j) = 1, \mathsf{m}_{i_k}(c_1) = v_k^1, \mathsf{m}_{i_k}(c_2) = v_k^2, \mathsf{m}_{i_k}(ok) = 1$ and $\forall p \in P \setminus \{loc_j, c_1, c_2, ok\} : \mathsf{m}_{i_k}(p) = 0$. Hence, $v_k^1 + v_k^2 > c$ and we conclude that C is unbounded. $\square$

Corollary 1 *The boundedness problem is undecidable for* SPN.

Remark 1 *Our definition of* SPN *is actually is much more restrictive than the definition given in [1] where* SPN *are build on an underlying weighted Petri nets, and transitions may be associated with a special event e with a* high *priority. It is clear that our proof may be directly extended to the full class presented in [1].*

Remark 2 *It is routine to see that $\mathcal{N}_C$ has the same behaviour if we consider an* infinite server semantics*. Indeed,* EFS *remain sets during the executions of the net in the particular case of $\mathcal{N}_C$. Hence, our proof may be directly adapted to this case.*

Remark 3 *Similarly, we may observe that the behavior of $\mathcal{N}_C$ is unchanged if we replace the events by a temporisation either on places or transitions (or both) in such a way that, when $c_j > 0$ and $tr_i^{=0}$ is fired, tr_i^T is forced to fire before tr''_i. Hence the property also holds for the class of timed nets and for the class of temporized nets. It also holds for stochastic Petri nets, provided the timing distributions are concentrated on domains excluding the vicinity of 0 and ∞ (in particular they must not have infinite queues), so as to allow to get the forced behavior mentioned above.*

5 Conclusion

Petri nets is a very delicate kind of theory. Very often, when one tries to extend their expressive power, one faces new undecidability issues, see for instance [2, 4]. As we saw, this is also the case when going to synchronized nets. Fortunately for specialists of logic controllers, this does not mean that synchronized nets are useless: there is a life beyond undecidability. And indeed we still continue to use, study and develop ordinary arithmetic, while we know since Gödel that it is undecidable. While the general problem of boundedness is undecidable for SPNs, there are many cases where boundedness may be asserted (for instance, if the usual Petri net underlying a SPN is bounded, so is the SPN). Also, it is likely that it is possible to define interesting synthesis rules allowing to construct new bounded SPNs from known ones. And finally, boundedness is not always the needed property: often we prefer to work with safe systems, and safeness, .i.e. "*is the number of tokens in each place bounded by 1 in all the reachable markings?*", is trivially decidable (by constructing the reachability graph for instance).

References

[1] R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets.* Springer, 2004.

[2] C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset Nets Between Decidability and Undecidability. In *Proceedings of ICALP'98*, LNCS 1443, pages 103–115. Springer, 1998.

[3] N.M. Minsky. *Finite and Infinite Machines.* Englewood Cliffs, N.J., Prentice-Hall, 1967.

[4] J.-F. Raskin and L. Van Begin. Petri Nets with Non-blocking Arcs are Difficult to Analyse. In volume 98 of *ENTCS*, pages 35–55. Elsevier, 2003.

[5] L. Van Begin. *Efficient Verification of Counting Abstractions for Parametric systems.* PhD thesis, Université Libre de Bruxelles, Belgium, 2003.