Discrete Parameters in Petri Nets

Nicolas David, Claude Jard, Didier Lime, Olivier Henri Roux

Discrete Parameters in Petri Nets*

Nicolas David[1], Claude Jard[1], Didier Lime[2], and Olivier H. Roux[2]

[1] University of Nantes, LINA
nicolas.david1@univ-nantes.fr
claude.jard@univ-nantes.fr
[2] École Centrale de Nantes, IRCCyN
didier.lime@ec-nantes.fr
olivier-h.roux@irccyn.ec-nantes.fr

Abstract. With the aim of significantly increasing the modeling capability of Petri nets, we suggest that models involve parameters to represent the weights of arcs, or the number of tokens in places. We consider the property of coverability of markings. Two general questions arise: "Is there a parameter value for which the property is satisfied?" and "Does the property hold for all possible values of the parameters?". We show that these issues are undecidable in the general case. Therefore, we also define subclasses of parameterised networks, depending on whether the parameters are used on places, input or output arcs of transitions. For some subclasses, we prove that certain problems become decidable, making these subclasses more usable in practice.

Keywords: Petri net, Parameters, Coverability

1 Introduction

The introduction of parameters in models aims to improve genericity. It also allows the designer to leave unspecified aspects, such as those related to the modeling of the environment. This increase in modeling power usually results in greater complexity in the analysis and verification of the model. Beyond verification of properties, the existence of parameters opens the way to very relevant issues in design, such as the computation of the parameters values ensuring satisfaction of the expected properties.

We chose to explore the subject on concurrent models whose archetype is that of Petri nets. We consider discrete parameterisation of markings (the number of tokens in the places of the net) or weight of arcs connecting the input or output places to transitions. We call these Petri nets parameterised nets or PPNs.

We consider the general properties of coverability and, to a lesser extent, reachability (that are often the basis for the verification of more specific properties).

First issues are:

*Work partially supported by ANR project PACS (ANR-14-CE28-0002) and Pays de la Loire research project AFSEC.

– Is there a value of the parameters such that the property is satisfied?
– Is the property satisfied for all possible values of the parameters?

Given the modeling power offered by PPNs, we first study the decidability of these issues. Since in the general case, they are undecidable, we then examine decidable subclasses.

Related work. There is not much work on Petri nets with parameters. One example is regular model checking [3] for algorithmic verification of several classes of infinite-state systems whose configurations can be modeled as words over a finite alphabet. The main idea is to use regular languages as the representation of sets of configurations, and finite-state transducers to describe transition relations. This is only possible for particular examples including parameterised systems consisting of an arbitrary number of homogeneous finite-state processes connected in a regular topology, and systems that operate on linear data structures. Parameters are also introduced in models such as predicate Petri nets [8], in the aim to have more concise models, in particular to take into account symmetries in the model [4]. Domains of values are generally finite. Parameterised verification on timed systems has also been studied in several papers since its introduction by Alur et al. in [1]. Parameterisation of time uses continuous parameters. In this paper, we focus on discrete parameters on untimed Petri nets.

The remainder of the paper is structured as follows: Section 2 re-visites the semantics of Petri Nets and introduces discrete parameters in Petri Nets. Section 3 presents the undecidability results. Section 4 introduces subclasses of our parameterised models. Section 5 answers decidability results over those subclasses and underlines issues encountered with reachability. Section 6 concludes and points to future work.

2 Definitions

Notations

$\mathbb{N}$ is the set of natural numbers. $\mathbb{N}^*$ is the set of positive natural numbers and $\mathbb{N}_\omega$ is the classic union $\mathbb{N} \cup \{\omega\}$ where for each $n \in \mathbb{N}$, $n + \omega = \omega$, $\omega - n = \omega$, $n < \omega$ and $\omega \leq \omega$. $\mathbb{Z}$ is the set of integers. Let X be a finite set. 2^X denotes the powerset of X and $|X|$ the size of X. Let $V \subseteq \mathbb{N}$, a V-valuation for X is a function from X to V. We therefore denote $V^X$the set of V-valuations on X. Given an alphabet Σ, we denote as Σ_ϵ the union $\Sigma \cup \{\epsilon\}$ where ϵ is the silent action. Given a set X, let $k \in \mathbb{Z}$ and $x \in X$, we define a linear expression on X by the following grammar: $\lambda ::= k \mid k * x \mid \lambda + \lambda$. Given a linear expression λ on X and a $\mathbb{N}$-valuation ν for X, $\nu(\lambda)$ is the integer obtained when replacing each element x in X from λ, by the corresponding value $\nu(x)$.

2.1 Petri Nets and Marked Petri Nets

Definition 1 (Petri Net) *A Petri Net is a 4-tuple* $\mathcal{N} = (P, T, Pre, Post)$ *where* P *is a finite set of places,* T *is a finite set of transitions,* Pre *and*

$Post \in \mathbb{N}^{|P|\times|T|}$ are the backward and forward incidence matrices, such that $Pre(p,t) = n$ with $n > 0$ when there is an arc from place p to transition t with weight n and $Post(p,t) = n$ with $n > 0$ when there is an arc from transition t to place p with weight n.

Given a Petri Net $\mathcal{N} = (P, T, Pre, Post)$, we denote $Pre(\bullet, t)$ (also written ${}^\bullet t$) as the vector $(Pre(p_1,t), Pre(p_2,t), ..., Pre(p_{|P|},t))$ *i.e.* the t^{th} column of the matrix Pre. The same notation is used for $Post(\bullet, t)$ (or $t^\bullet$).

Definition 2 (Marking) *A marking of a Petri Net $\mathcal{N} = (P, T, Pre, Post)$ is a vector $m \in \mathbb{N}^{|P|}$.*

If $m \in \mathbb{N}^{|P|}$ is a marking, $m(p_i)$ is the number of tokens in place p_i. We can define a partial order over markings.

Definition 3 (Partial Order) *Let $\mathcal{N}$ be a Petri Net such that $\mathcal{N} = (P, T, Pre, Post)$, let m and m' be two markings of $\mathcal{N}$. We define $\leq$ as a binary relation such that $\leq$ is a subset of $\mathbb{N}^{|P|} \times \mathbb{N}^{|P|}$ defined by:*

$$m \leq m' \Leftrightarrow \forall p \in P, m(p) \leq m'(p) \tag{1}$$

Definition 4 (Marked Petri Net) *A marked Petri Net (PN) is a couple $\mathcal{S} = (\mathcal{N}, m_0)$ where $\mathcal{N}$ is a Petri Net and m_0 is a marking of $\mathcal{N}$ called the initial marking of the system.*

An example of *marked Petri Net* is given in Figure 1.

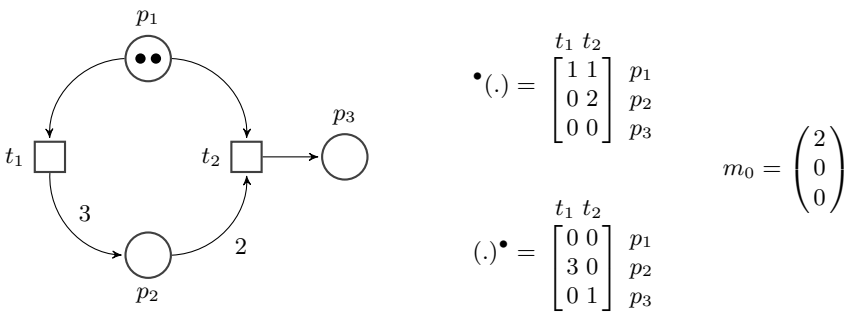


Fig. 1. A Marked Petri Net

2.2 Operational Semantics

Augmenting Petri Nets with markings leads to the notion of enabled-transitions and firing of transitions. Given a marked Petri Net $\mathcal{S}$, a transition $t \in T$ is said *enabled by a marking m* when $m \geq Pre(\bullet, t)$.

Definition 5 (PN Semantics) *The semantics of PN is a transition system* $\mathcal{S}_{\mathcal{T}} = (Q, q_0, \rightarrow)$ *where,* $Q = \mathbb{N}^{|P|}$, $q_0 = m_0$, $\rightarrow \in Q \times T \times Q$ *such that,*

$$m \xrightarrow{t_i} m' \Leftrightarrow \begin{cases} m \geq {}^\bullet t_i \\ m' = m - {}^\bullet t_i + t_i^\bullet \end{cases} \tag{2}$$

This relation holds for sequences of transitions:

- $m \xrightarrow{w} m'$ if w is the empty word and $m = m'$
- $m \xrightarrow{wt} m'$ if $\exists m'', m \xrightarrow{w} m'' \wedge m'' \xrightarrow{t} m'$ where $w \in T^*$ and $t \in T$.

Definition 6 (Reachability set) *Given a PN,* $\mathcal{S} = (\mathcal{N}, m_0)$*, the reachability set of* $\mathcal{S}$*,* $RS(\mathcal{S})$ *is the set of all reachable markings of* $\mathcal{S}$ *i.e.* $RS(\mathcal{S}) = \{m \mid \exists w \in T^*, m_0 \xrightarrow{w} m\}$

2.3 Parametric Petri Nets

We would like to use less rigid modeling in order to model systems where some data are not known *a priori*. Therefore, in this subsection, we extend the previous definitions by adding a set of parameters Par. Working with Petri nets and discrete parameters leads to consider two main situations: the first one involves parameters on markings, by replacing the number of tokens in some places by parameters, the second one involves parameters as weights. The same parameter can be used in both situations. Using parameters on markings can be easily understood as modeling an unfixed amount of resources that one may want to optimise. Let us consider a concrete example to illustrate parameterised weights. In a production line, we consider two operations: first, to supply raw material, we need to unpack some boxes containing an amount λ_1 of resources, as depicted in Figure 2, and at the end, we need to pack end products in boxes of capacity λ_2, as in Figure 3. This is part of a whole packaging process that one may want to optimise. The level of abstraction induced by parameters permits to leave those values unspecified in order to perform an early analysis.

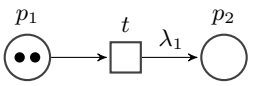


Fig. 2. Unpacking raw material

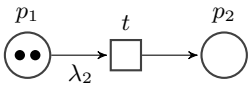
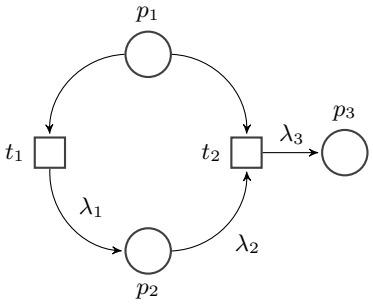


Fig. 3. Packing end products

Definition 7 (Parametric Petri Net) *A parametric Petri Net,* $\mathcal{NP}$ *is a* 5*-tuple* $\mathcal{NP} = (P, T, Pre, Post, Par)$ *such that* P *is a finite set of places of* $\mathcal{NP}$*,* T *is a finite set of transitions of* $\mathcal{NP}$*,* Par *is a finite set of parameters of* $\mathcal{NP}$*,* Pre *and* $Post \in (\mathbb{N} \cup Par)^{|P| \times |T|}$



$$\bullet(.) = \begin{array}{c} \begin{matrix} t_1 & t_2 \end{matrix} \\ \begin{bmatrix} 1 & 1 \\ 0 & \lambda_2 \\ 0 & 0 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix}$$

$$(.)^\bullet = \begin{array}{c} \begin{matrix} t_1 & t_2 \end{matrix} \\ \begin{bmatrix} 0 & 0 \\ \lambda_1 & 0 \\ 0 & \lambda_3 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix}$$

Fig. 4. A Parametric Petri Net

Intuitively, a parametric Petri net is a Petri net where the number of tokens involved in a transition is parameterised as depicted in Figure 4.

Definition 8 (Parametric marking) *Given a parametric Petri Net* $\mathcal{NP} = (P, T, Pre, Post, Par)$*, a parametric marking is a* $|P|$*-dimensional vector* μ *of linear expressions on* $\mathbb{N} \cup Par$*.*

Modeling with parameters means using parameters over weights and markings rather than setting numeric values everywhere. Therefore we may also use a parametric initial marking.

Definition 9 (parametric PN or PPN) *A parametric marked Petri Net (PPN) is a couple,* $\mathcal{SP} = (\mathcal{NP}, \mu_0)$ *where* $\mathcal{NP}$ *is a Parametric Petri Net and* μ_0 *is the parametric initial marking of* $\mathcal{NP}$*.*

PPNs can be used to design systems where some parts have not been analysed or where we need to keep flexibility. We now need to define a way to instantiate classic Petri nets from our parametric marked Petri Nets, in order to define a semantics.

Definition 10 (Parametric Semantics) *Let* $\mathcal{SP} = (P, T, Pre, Post, Par, \mu_0)$ *be a PPN, we consider the set of valuations* $\mathbb{N}^{Par}$*. Let* $\nu \in \mathbb{N}^{Par}$*, we define* $\nu(\mathcal{SP})$ *as the PN obtained from* $\mathcal{SP}$ *by replacing each parameter* $\lambda \in Par$ *by* $\nu(\lambda)$*, its valuation by* ν*, i.e.* $\nu(\mathcal{SP}) = (P, T, Pre', Post', m_0)$ *where* $\forall i \in [\![1, |P|]\!], \forall j \in [\![1, |T|]\!]$*,*

$$Pre'(i,j) = \begin{cases} Pre(i,j) & \text{if } Pre(i,j) \in \mathbb{N} \\ \nu(Pre(i,j)) & \text{if } Pre(i,j) \in Par \end{cases} \tag{3}$$

$$Post'(i,j) = \begin{cases} Post(i,j) & \text{if } Post(i,j) \in \mathbb{N} \\ \nu(Post(i,j)) & \text{if } Post(i,j) \in Par \end{cases} \tag{4}$$

$$m_0(i) = \begin{cases} \mu_0(i) & \text{if } \mu_0(i) \in \mathbb{N} \\ \nu(\mu_0(i)) & \text{if } \mu_0(i) \text{ is a linear expression on } Par \end{cases} \tag{5}$$

A marked Petri Net is an *instance* of a parametric marked Petri net.

2.4 Parametric problems

We can define several interesting parametric problems on PPNs. In fact, the behaviour of a PPN is described by the behaviours of all the PNs obtained by considering all possible valuations of the parameters. It seems therefore obvious to ask, in a first time, if *there exists valuations for the parameters such that a property holds for the corresponding instance* and its dual, *i.e.*, if *every instance of the parametric marked Petri Net satisfies the property.* Given a class of problem $\mathcal{P}$ (coverability, reachability,...), $\mathcal{SP}$ a PPN and ϕ is an instance of $\mathcal{P}$, parameterised problems are written as follows:

Definition 11 ($\mathcal{P}$-Existence problem) *($\mathscr{E}$-$\mathcal{P}$): Is there a valuation $\nu \in \mathbb{N}^{Par}$ s.t. $\nu(\mathcal{SP})$ satisfies the property ϕ ?*

Definition 12 ($\mathcal{P}$-Universality problem) *($\mathscr{U}$-$\mathcal{P}$): Does $\nu(\mathcal{SP})$ satisfies the property ϕ for each $\nu \in \mathbb{N}^{Par}$?*

This paper focuses on *reachability* and *coverability* issues.

Definition 13 (Reachability) *Let $\mathcal{S} = (\mathcal{N}, m_0) = (P, T, Pre, Post, m_0)$ and m a marking of $\mathcal{S}$, $\mathcal{S}$ reaches m iff $m \in RS(\mathcal{S})$.*

Definition 14 (Coverability) *Let $\mathcal{S} = (\mathcal{N}, m_0) = (P, T, Pre, Post, m_0)$ and m a marking of $\mathcal{S}$, $\mathcal{S}$ covers m if there exists a reachable marking m' of $\mathcal{S}$ such that m' is greater or equal to m i.e.*

$$\exists m' \in \mathcal{RS}(\mathcal{S}) s.t.\ \forall p \in P, m'(p) \geq m(p) \tag{6}$$

We recall that reachability [9] and coverability [7] are decidable on classic Petri nets. In the context of parametric Petri nets, coverability leads to two main problems presented previously, that is to say: the *existence problem*, written ($\mathscr{E}$-cov) and the *universal problem*, written ($\mathscr{U}$-cov). For instance, ($\mathscr{U}$-cov) asks: *"Does each valuation of the parameters implies that the valuation of the parametric P/T net system covers m ?" i.e.*

$$m \text{ is } \mathscr{U}\text{-coverable in } \mathcal{SP} \Leftrightarrow \begin{cases} \forall \nu \in \mathbb{N}^{Par}, \ \exists m' \in \mathcal{RS}(\nu(\mathcal{SP})) \\ \qquad\qquad \text{s.t. } m' \geq m \end{cases} \tag{7}$$

We can similarly define $\mathscr{E}$-reach and $\mathscr{U}$-reach for parameterised reachability.

3 Undecidability Results for the General Case

In their paper suming-up results of decidability for reset-nets and corresponding subclasses, Dufourd, Finkel and Schnoebelen noticed that *"Reachability is known to become undecidable as soon as the power of Petri nets is increased"* [5], for instance, adding *reset arcs* [2] or *inhibitor arcs* [6] makes reachability undecidable. In this section, we focus on showing that adding parameters to PN leads to undecidability. More specifically, ($\mathscr{U}$-cov) and ($\mathscr{E}$-cov) are undecidable on PPNs.

As we will proceed by reduction to the halting problem (and counter boundedness problem) for counter machines to answer our problem, we first recall some definitions. A 2-Counters Machine has a pointer and a tape which contains finite number of instructions in three types: *increment*, *decrement* and *zero-test*. The pointer reads the tape to execute increment or decrement instructions sequentially. When the pointer reaches a *zero-test* instruction, then it will jump to a certain position on the tape and continue. Formally, it consists of two counters c_1, c_2, a set of states $P = \{p_0, ...p_m\}$, a terminal state labelled *halt* and a finite list of instructions $l_1, ..., l_s$ among the following list:

- increment: increase c_k by one and go to next state, where $k \in \{1, 2\}$
- decrement: decrease c_k by one and go to next state, where $k \in \{1, 2\}$
- zero-test: if $c_k = 0$ go to state p_j else go to state p_l, where $p_j, p_l \in P \cup \{halt\}$ and $k \in \{1, 2\}$

We can assume without restriction that the counters are non negative integers *i.e.* that the machine is well-formed in the sense that a *decrement instruction* is guarded by a *zero-test* and that the counters are initialised to zero. It is well known that the *halting problem* (whether state *halt* is reachable) and the *counters boundedness problem* (whether the counters values stay in a finite set) are both *undecidable* as proved by Minksy [10].

Theorem 1 (Undecidability of $\mathscr{E}$-cov on PPN) *The $\mathscr{E}$-coverability problem for PPN is undecidable* [1].

Proof. We proceed by *reduction from a 2-counters machine*. Given a Minksy 2-counters machine $\mathcal{M}$, we construct a PPN that simulates it, $\mathcal{SP}_{\mathcal{M}}$, as follows.

- Each counter c_i is modeled by two places C_i and $\neg C_i$. The value of the counter is encoded by the number of tokens in C_i.
- For each state p of $P \cup \{halt\}$ a 1-bounded place p is created in the net.
- The instructions of the previous definition are modeled by the transitions and arcs depicted in Figure 5.
- A unique additional place π with an additional transition θ serves to initialise the net. The initial marking is composed of one token in π and one token in the place p corresponding to the initial state p of $\mathcal{M}$.

Initially, only θ can be fired, which leads to the initial configuration of the machine (state p_0 and counters values null), with one token in p_0, no tokens in C_1 and C_2 and a parameterised number of tokens in $\neg C_i$. The value of this parameter will therefore represent the upper bound of the counter over the instructions sequence. We have to verify that each time $m(C_i) + m(\neg C_i) = \lambda$. First we show that $\mathcal{SP}_{\mathcal{M}}$ simulates $\mathcal{M}$ by verifying the behaviour of each instruction:

[1] We can be more accurate by specifying that we need at least 1 parameter used on 6 distinct arcs. The question remains opened for fewer parameterised arcs

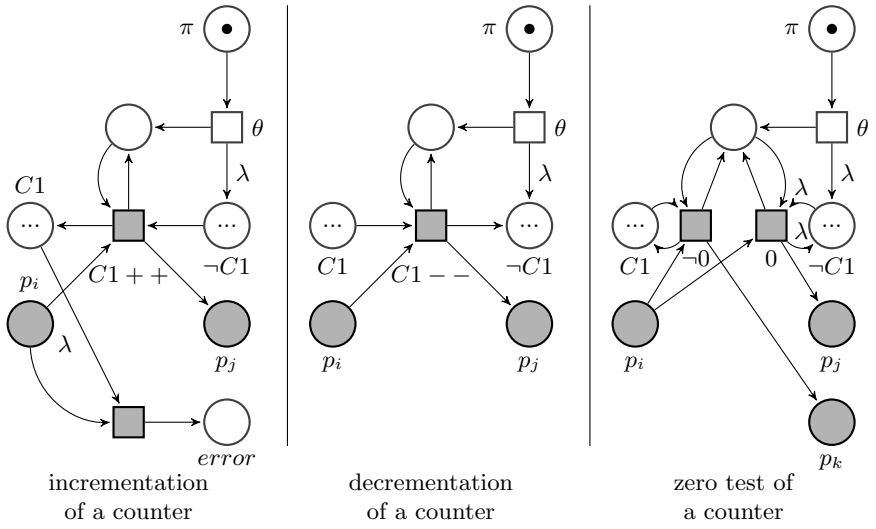


Fig. 5. Modeling a counter with PPN

- *Increment instruction:* As C_i models the counter, the transition $C_i + +$ adds one token in C_i, removes one token from $\neg C_i$ and changes the *current state* by removing the token from p_i and adding a token in p_j. The *error* states is marked iff the incrementation instruction is performed whereas we have already reached the upper bound over the execution. This state will be useful for the second proof.
- *Decrement instruction:* As C_i models the counter, the transition C_i - - removes one token from C_i, adds one token in $\neg C_i$ and changes the *current state* by removing the token from p_i and adding a token in p_j. We recall the machine is well-formed.
- *Zero Test:* As C_i models the counter, and as we know the sum of tokens available in C_i and $\neg C_i$, there is no token in C_i iff there are λ tokens in $\neg C_i$. According to this test the *current state* is updated by removing the token from p_i and adding a token in p_j or p_k. The value of the counter is left unchanged.

℘ Coverability is undecidable :
We will show that given a 2-counters machine $\mathcal{M}$, *(a)* $\mathcal{M}$ halts (it reaches the *halt* state) iff *(b)* there exists a valuation ν such that $\nu(\mathcal{SP}_{\mathcal{M}})$ covers the corresponding p_{halt} place.

- $(a) \Rightarrow (b)$ First, let us assume that $\mathcal{M}$ halts. As $\mathcal{M}$ halts, the execution of the machine is finite. On this execution the two counters are bounded by c_{lim1} and c_{lim2}. Let c_{lim} be the maximum of those two values. Let ν be the valuation such that $\nu(\lambda) = c_{lim}$. By the previous explanation, $\mathcal{SP}_{\mathcal{M}}$ simulates $\mathcal{M}$. Moreover, the valuation ν ensures that $\mathcal{SP}_{\mathcal{M}}$ does not reach

a deadlock state where p_{error} is marked. Therefore, when $\mathcal{M}$ reaches *halt*, $\mathcal{SP}_{\mathcal{M}}$ will add 1 token in p_{halt}. So, a marking where there is one token in p_{halt} is coverable.

- $(b) \Rightarrow (a)$ We proceed by contrapositive. Let us assume that $\mathcal{M}$ does not halt. We want to show that there is no valuation ν such that $\nu(\mathcal{SP}_{\mathcal{M}})$ adds a token in p_{halt}. Let us consider the two following distinct alternatives:
  - If the counters are bounded along the execution, either the value of λ is less than the maximum value of the counters and error will be reached during some increment resulting in a deadlock, or the value of λ is big enough so that error is never marked, but, in this case, then, as the machine does not halt, it means that it does not reach *halt*. So there is no instruction that leads to *halt* in $\mathcal{M}$. Therefore, according to the previous explanation, there is no transition that adds a token in p_{halt}.
  - If at least one counter is not bounded, then for any given valuation ν, we will reach an instruction *inc*(c_i), where i is 1 or 2, and $c_i = \nu(\lambda)$. Therefore, a token will be added in p_{error} leading to a deadlock. So $\mathcal{SP}_{\mathcal{M}}$ will not cover a terminal state.

The undecidability of the halting problem on the 2-counters machine gives the undecidability of the $\mathscr{E}$-coverability problem.

Theorem 2 (Undecidability of $\mathscr{U}$-cov on PPN) *The $\mathscr{U}$-coverability problem for PPN is undecidable.*

Proof. $\mathscr{U}$ Coverability is undecidable:

We proceed by *reduction from a 2-counters machine*. We use the same construction as in the previous proof. We denote m_{error} the marking were $m_{error}(p) = 0$ for each $p \in P$ except $m_{error}(p_{error}) = 1$. We will show that given a 2-counter machine $\mathcal{M}$, *(a)* the counters are unbounded along the instructions sequence of $\mathcal{M}$ (counters boundedness problem) iff *(b)* for each valuation ν, $\nu(\mathcal{SP}_{\mathcal{M}})$ covers the m_{error}.

- $(a) \Rightarrow (b)$ First, let us assume that on a given instruction sequence, one counter of $\mathcal{M}$ is unbounded. By the second alternative considered in the proof for $\mathscr{E}$-cov we proved that for any valuation, a token will be added in p_{error}.
- $(b) \Rightarrow (a)$ Reciprocally, by contrapositive, we want to show that if the counters are bounded, there exists a valuation ν such that $\nu(\mathcal{SP}_{\mathcal{M}})$ does not cover m_{error}. This comes directly from the previous proof. As the counters are bounded along the instructions sequence, we consider a valuation ν such that $\nu(\lambda) = c_{lim}$ where c_{lim} is an upper bound of the values of the counters. By construction, there is no possibilities to add a token in p_{error}, otherwise, it means that $\mathcal{SP}_{\mathcal{M}}$ took an incrementation transition meaning that c_{lim} is not an upper bound.

The undecidability of the counters boundedness problem on the 2-counters machine gives the undecidability of the $\mathscr{U}$-coverability problem.

4 Subclasses of Parametric Petri Nets

4.1 Introducing Subclasses

On the one hand, our parametric model increases the modeling power of Petri nets but on the other hand, using parameters leads to complex models where properties become undecidable. In order to obtain parameterised models that are easier to analyse and therefore can be used in practice, we should reduce the power of modeling. We will therefore introduce some subclasses of the PPN in which we restrict the use of parameters to only markings, which could be used to model arbitrary number of identical processes, to only output arcs, which, we will see, is a bit more general or to only input arcs, which could model synchronizations among arbitrary numbers of identical process, and finally some combinations of those.

The following subclasses have therefore a dual interest. From a modeling point of view, restrict the use of parameters to tokens, output or input can be used to model concrete examples such as respectively processes or synchronisation of a given number of processes. From a theoretical point of view, it is interesting to introduce those subclasses of PPN in a concern of completeness of the study.

Definition 15 (P-parametric PN) *A P-parametric marked Petri Net (P-PPN),* $\mathcal{SP} = (\mathcal{NP}, \mu_0)$ *where* $\mathcal{NP}$ *is a Parametric Petri Net such that* Pre *and* $Post \in \mathbb{N}^{|P|\times|T|}$ *and* μ_0 *is a parametric marking of* $\mathcal{NP}$*.*

A P-PPN is a classic Petri net with a parametric initial marking.

Definition 16 (T-parametric PN) *A T-parametric Petri Net (T-PPN),* $\mathcal{SP} = (\mathcal{NP}, m_0)$ *where* $\mathcal{NP}$ *is a Parametric Petri Net and* m_0 *is a marking of* $\mathcal{NP}$

Intuitively, using parameters on outputs means we will create parametric markings. To complete this study, we can extract a subclass in which parameters involved in the Pre matrix and parameters involved in the Post matrix correspond to disjoints subsets of parameters. *i.e.* $par(Pre) \cap par(Post) = \emptyset$ where *par* is the application that maps to the set of parameters involved in a matrix (or a vector). We call this subclass *distinctT-PPN*[2]. We can even refine the subclass of distinctT-PPN by considering the two distinct classes of *Pre-T-parametric PPN* (preT-PPN), where $Post \in \mathbb{N}^{|P|}$ and *Post-T-parametric PPN* (postT-PPN), where $Pre \in \mathbb{N}^{|P|}$.

As we introduced several subclasses, it is interesting to study whether one of this subclass is more expressive than the other. We will show that P-PPN and postT-PPN are related. Therefore, we introduce here some useful definitions. Our translations add *silent actions* that detail the beahviour of the Petri nets. Therefore, we introduce a labelling function Λ from the set of transitions T to

[2]Studying the undecidability proof, it is relevant to think that using different parameters for the input and the output would reduce the modeling power.

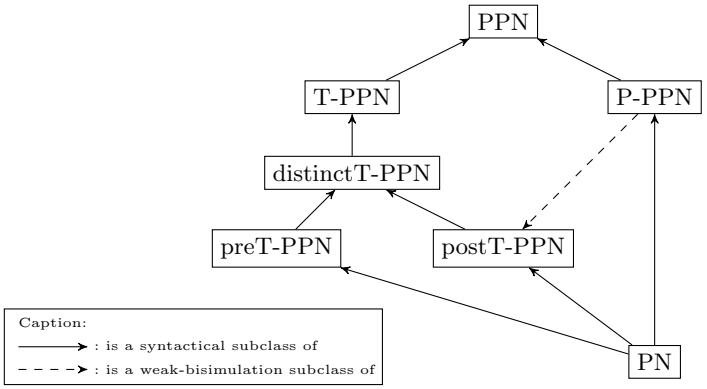


Fig. 6. Subclasses of PPN

Σ_ϵ, $\Lambda : T \to \Sigma_\epsilon$, such that $\Sigma_\epsilon \subseteq T \cup \{\epsilon\}$ and $\Lambda(t_i)$ equals either t_i or ϵ. We extend the previous definitions by using $m \xrightarrow{t} m'$ or $m \xrightarrow{\Lambda(t)} m'$ depending on the context[3]. For instance, $m \xrightarrow{\epsilon^*} m'$ means that m leads to m' by using zero or more internal ϵ-transitions. Given two markings m and m' we write:

$$m \xrightarrow{\alpha}_\epsilon m' \Leftrightarrow m \xrightarrow{\epsilon^*}\xrightarrow{\alpha}\xrightarrow{\epsilon^*} m' \text{ with } \alpha \neq \epsilon \tag{8}$$

Definition 17 (Weak-Simulation) *Given two labelled marked Petri nets,* $\mathcal{S}_1 = (P_1, T_1, F_1, \Lambda_1, \Sigma_\epsilon, m_1^0)$ *and* $\mathcal{S}_2 = (P_2, T_2, F_2, \Lambda_2, \Sigma_\epsilon, m_2^0)$*, a binary relation* $\mathcal{R} \subseteq \mathbb{N}^{|P_1|} \times \mathbb{N}^{|P_2|}$ *is a simulation if*

$$\forall (m_1, m_2) \in \mathcal{R} \Leftrightarrow \begin{cases} \forall \alpha \in \Sigma \text{ and } m_1' \text{ s.t. } m_1 \xrightarrow{\alpha}_\epsilon m_1', \\ \exists m_2' s.t.\ m_2 \xrightarrow{\alpha}_\epsilon m_2' \text{ and } (m_1', m_2') \in \mathcal{R} \end{cases} \tag{9}$$

If we can find a weak-simulation $\mathcal{R} \subseteq \mathbb{N}^{|P_1|} \times \mathbb{N}^{|P_2|}$ such that $(m_1^0, m_2^0) \in \mathcal{R}$ we say that $\mathcal{S}_2$ weakly simulates $\mathcal{S}_1$, which means intuitively that $\mathcal{S}_2$ can match all the moves of $\mathcal{S}_1$. Moreover if we can find another weak-simulation $\mathcal{R}' \subseteq \mathbb{N}^{|P_1|} \times \mathbb{N}^{|P_2|}$ such that $\mathcal{S}_1$ weakly simulates $\mathcal{S}_2$, we say that $\mathcal{S}_1$ and $\mathcal{S}_2$ are *weakly co-similar.*

Definition 18 (Weak-Bisimulation) *Given two labelled PN,* $\mathcal{S}_1 = (P_1, T_1, F_1, \Lambda_1, \Sigma_\epsilon, m_1^0)$ *and* $\mathcal{S}_2 = (P_2, T_2, F_2, \Lambda_2, \Sigma_\epsilon, m_2^0)$*, a binary relation* $\mathcal{R} \subseteq \mathbb{N}^{|P_1|} \times \mathbb{N}^{|P_2|}$

[3]Indeed, if we consider the alphabet A equals to the set of the transition T of the Petri Net, and L as the identity function, the two definitions are equivalent. Using labelling is more general and allows to introduce non deterministic behaviours.

is a weak-bisimulation [4] *if*

$$\forall (m_1, m_2) \in \mathcal{R} \Leftrightarrow \begin{cases} - \ \forall \alpha \in \Sigma \ and \ m_1' \ s.t. \ m_1 \xrightarrow{\alpha}_{\epsilon} m_1' \\ \ \ there \ is \ m_2' s.t. \ m_2 \xrightarrow{\alpha}_{\epsilon} m_2' \ and \ (m_1', m_2') \in \mathcal{R} \\ - \ \forall \alpha \in \Sigma \ and \ m_2' \ s.t. \ m_2 \xrightarrow{\alpha}_{\epsilon} m_2' \\ \ \ there \ is \ m_1' s.t. \ m_1 \xrightarrow{\alpha}_{\epsilon} m_1' \ and \ (m_1', m_2') \in \mathcal{R} \end{cases} \tag{10}$$

Two labelled Petri Nets $\mathcal{S}_1$ and $\mathcal{S}_2$ are *weakly bisimilar* if there is a weak bisimulation relating their initial markings. In the sequel, every transition called θ is mapped to ϵ by Λ whereas for a transition called t, $\Lambda(t) = t$. If the original PPN, $\mathcal{SP} = (P, T, Pre, Post, Par, \Lambda, \mu_0)$, has a set of transition T and T' denotes the set of transition of the constructed PPN, $\mathcal{SP}' = (P', T', Pre', Post', Par, \Lambda', \mu_0')$ then $T' = T \cup \Theta$ with $T \cap \Theta = \emptyset$. For each $t \in T$, $\Lambda(t) = t$ and for each θ in Θ, $\Lambda(\theta) = \epsilon$.

4.2 Translating P-PPN to postT-PPN

In order to simulate the behaviour of parameterised places, we translate those places in a parameterised initialisation process that needs to be fired before firing any other transitions in the net. The idea relies on using a new place π and a new transition θ enabled by this place, such that $\theta^\bullet$ initializes a P-PPN, as showed in Figure 7. We define the initial marking $m_0 = (0, ..., 0, 1)$ *i.e.* $\forall p \in P$, $m_0(p) = 0$ and $m_0(\pi) = 1$. We will show that $\mathcal{SP}'$ and $\mathcal{SP}$ are weakly-bisimilar by showing that each behaviour of $\mathcal{SP}$ can be done in $\mathcal{SP}'$ if we begin by firing θ and reciprocally.

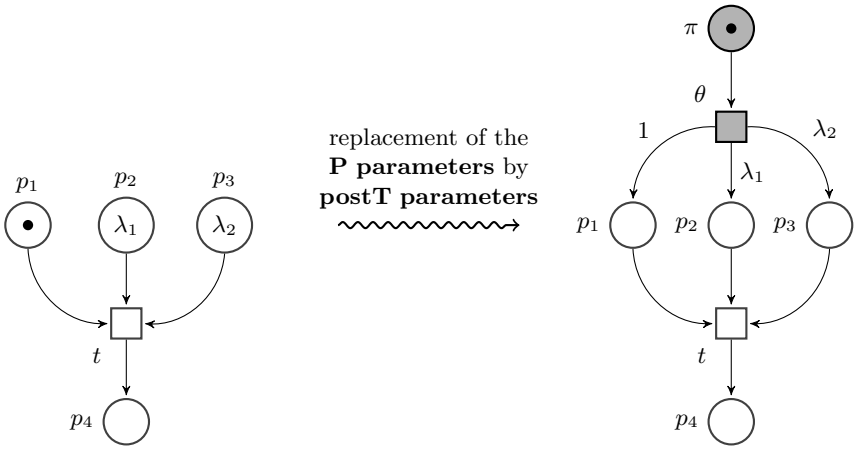


Fig. 7. From P-PPN to postT-PPN

[4] There exists several definitions of bisimulation, for instance preserving deadlocks or epsilon-branching, but the one we use is sufficient for our purpose.

Lemma 3 *$\forall \nu \in \mathbb{N}^{Par}$, $\nu(\mathcal{SP})$ and $\nu(\mathcal{SP}')$ are weakly bisimilar.*

Note that each path in $\mathcal{SP} = (P, T, Pre, Post, Par, \Lambda, \mu_0)$ can be done in $\mathcal{SP}' = (P', T', Pre', Post', Par, \Lambda', m'_0)$ by adding θ at the beginning. And reciprocally, each path in $\mathcal{SP}'$ begins by θ so is written $\theta.w$ where w is a path in $\mathcal{SP}$.

Proof. Let $\nu \in \mathbb{N}^{Par}$ a valuation of the parameters. We want to show that $\nu(\mathcal{SP})$ and $\nu(\mathcal{SP}')$ are weakly bisimilar. Let $\nu(\mu_0)$ be the parametric initial marking of $\nu(\mathcal{SP})$ and $\nu(m'_0) = m'_0$ the initial marking of $\nu(\mathcal{SP}')$. The only transition firable from m'_0 is θ and $m'_0 \xrightarrow{\theta} \nu(\mu_0)$ as shown in Figure 7. From $\nu(\mu_0)$, $\mathcal{SP}$ and $\mathcal{SP}'$ are isomorphic. So $\nu(\mathcal{SP}_1)$ and $\nu(\mathcal{SP}_2)$ are weakly-bisimilar.

Those results underline that using parameters on outputs is more powerfull than using parameters on markings. We can conclude that T-PPN are more expressive than PPN.

4.3 Translating postT-PPN to P-PPN

We will show that from a postT-PPN, $\mathcal{SP}_1 = (P_1, T_1, Pre_1, Post_1, Par_1, \Lambda_1, m_1^0)$ we can construct a P-PPN, $\mathcal{SP}_2 = (P_2, T_2, Pre_2, Post_2, Par_2, \Lambda_2, \mu_2^0)$ that weakly-simulates the behaviours of the postT-PPN. Reciprocally, the postT-PPN also weakly-simulates the behaviours of the P-PPN built.

For each transition t and place p such that the arc (t, p) is weighted by a parameter, we construct the net depicted in Figure 8 which replace this arc [5]. Therefore, $T_1 \subseteq T_2$. As previously, we introduce two labelling functions Λ_1 and Λ_2 from T_1 (resp. T_2) to T_ϵ such that, for each $t \in T_1$, $\Lambda_1(t) = \Lambda_2(t) = t$ and $\Lambda_2(t) = \epsilon$ otherwise (*i.e.* for each $t \in T_2 \backslash T_1$).

Lemma 4 *$\forall \nu \in \mathbb{N}^{Par}$, $\nu(\mathcal{SP}_1)$ and $\nu(\mathcal{SP}_2)$ are weakly cosimilar.*

Proof. We will prove the 2 weak-simulations.

- $\forall \nu \in \mathbb{N}^{Par}$, $\nu(\mathcal{SP}_2)$ simulates $\nu(\mathcal{SP}_1)$. Let us consider $\nu \in \mathbb{N}^{Par}$, $\nu(\mathcal{SP}_1)$ has the following behaviour: each time t is fired, $\nu(\lambda)$ tokens are created in p. In $\mathcal{SP}_2$, it is possible to generates $\nu(\lambda)$ tokens in p after firing the sequence $t\ \theta_{t,p,1}^{\nu(\lambda)}\ \theta_t\ \theta_{t,p,2}^{\nu(\lambda)}$, labeled $t\epsilon^*$. Moreover, this sequence resets the sub-net constructed for the weak-simulation. As the other transitions of the network are not affected, monotony gives directly the weak-simulation.

[5] Notice that if several labeled arcs come from the same transition, some places and transitions of the Figure 8 should be duplicated according to indices

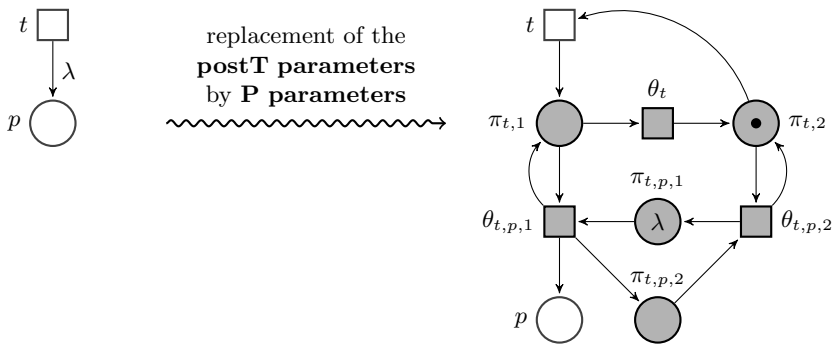


Fig. 8. From postT-PPN to P-PPN

– $\forall \nu \in \mathbb{N}^{Par}$, $\nu(\mathcal{SP}_1)$ simulates $\nu(\mathcal{SP}_2)$. Reciprocally, a marking with $\nu(\lambda)$ tokens in p allows to simulate the behaviours of every marking such that $m(p) \leq \nu(\lambda)$ according to monotony therefore, the reachable markings induced by creating less than $\nu(\lambda)$ tokens in $\mathcal{SP}_2$ are simulated by the one with $\nu(\lambda)$ tokens, and therefore by $\mathcal{SP}_1$. As the other transitions of the network are not affected, monotony gives directly the weak-simulation.

Therefore, $\mathcal{SP}_1$ and $\mathcal{SP}_2$ are *weakly co-similar*.

Remark 1. This is not a weak bisimulation. Indeed, if $\mathcal{SP}_2$ adds 3 tokens in p (leading to a marking m_2) whereas $\mathcal{SP}_1$ adds $\nu(\lambda) = 4$ tokens in p (leading to a marking m_1). Then any transitions needing more than 3 tokens could only be fired from m_1 in $\mathcal{SP}_1$ only. Here the two simulations relations are not reciprocal: m_1 would simulates m_2 but m_2 would not.

5 Decidability Results

We will now consider the parameterised properties defined in Section 2 and the different subclasses of parameterised models of Section 4. Table 1 sums up the results that we present in this section.

	$\mathscr{U}$-problem		$\mathscr{E}$-problem	
	Reachability	Coverability	Reachability	Coverability
preT-PPN	?	?	?	D
postT-PPN	?	D	?	D
PPN	U	U	U	U
distinctT-PPN	?	?	?	D
P-PPN	?	D	D	D

Table 1. Decidability results for parametric coverability and reachability

5.1 Study of Parameterised Coverability

The easiest proofs rely on monotony. Indeed, some instances simulate other instances. We recall that the *zero valuation* (written 0) is the valuation that maps every parameter to zero.

Lemma 5 *Decidability of $\mathscr{U}$-coverability on postT-PPN (resp. P-PPN) can be reduced to a test with the zero valuation.*

Proof. For postT-PPN and P-PPN, the zero valuation is the one allowing the lowest amount of behaviours for coverability *i.e.* it is the most restrictive valuation for coverability. Indeed, considering a marking m that we try to cover, m is $\mathscr{U}$-coverable if and only if there is a firing sequence w such that $m_0 \xrightarrow{w} m1 \geq m$ in the 0-instanced postT-PPN (or P-PPN). Formally, given a postT-PPN or a P-PPN $\mathcal{SP}$ and a marking m, we have:

$$\exists \nu \text{ s.t. } m \text{ is not coverable in } \nu(\mathcal{SP}) \text{ iff } m \text{ is not coverable in } 0(\mathcal{SP})$$

Indeed, for any valuation ν we can fire w in the ν-instanced PPN, leading to a marking $m2 \geq m1$ by monotony. Moreover, on the instance of a PPN (*i.e.* on a PN), the coverability is known decidable, so we can answer to the problem on the zero instanced postT-PPN (or P-PPN). If the answer is *no*, then we have found a counter example. Else, monotony directly implies that using a greater valuation ν will provide at least behaviours covering the current ones. The *winning behaviour* that allowed to answer yes for the zero-instance will still works on this *ν-instance*. So every instance will satisfy the coverability.

Therefore we can claim that $\mathscr{U}$-cov is decidable on postT-PPN and P-PPN. Let us consider $\mathscr{E}$-cov for the same subclasses.

Theorem 6 *$\mathscr{E}$-cov is decidable on P-PPN.*

Proof. Decidability of $\mathscr{E}$-cov on P-PPN:
We consider a P-PPN, $\mathcal{SP}_1$. We will now build a PN $\mathcal{S}_2$ with *token-canons* that will supply the parameterised places of $\mathcal{SP}_1$ as depicted in Figure 9. Each *token-canon* consists in two places π_p, π'_p and two transitions θ_p, θ'_p. θ_p supplies p of $\mathcal{S}_2$. Moreover, each transition of $\mathcal{SP}_1$ is added as an input and an output of π'_p, meaning that the net is blocked as long as every θ'_p has not been fired. This is repeated for each place initially marked by a parameter. We initialize $\mathcal{S}_2$ with 1 token in each π_p. So for each valuation ν of $\mathcal{SP}_1$, firing the sequence $\theta_p^{\nu(\lambda_p)}\ \theta'_p$ for each parameterised place p leads to a marking m_2 equals to the valuation of the initial marking of $\mathcal{SP}_1$. Moreover, the θ-transitions added have been fired, so every π'_p is marked. The two nets have now the same behaviour. This shows that $\mathcal{S}_2$ simulates any valuation of $\mathcal{SP}_1$. Therefore, the existence of a valuation such that a given marking is covered can be reduced to the coverability of the same marking (completed with 0 for each π_p and 1 for each π'_p added) which is known decidable as a classic coverability problem on an unbounded Petri net.

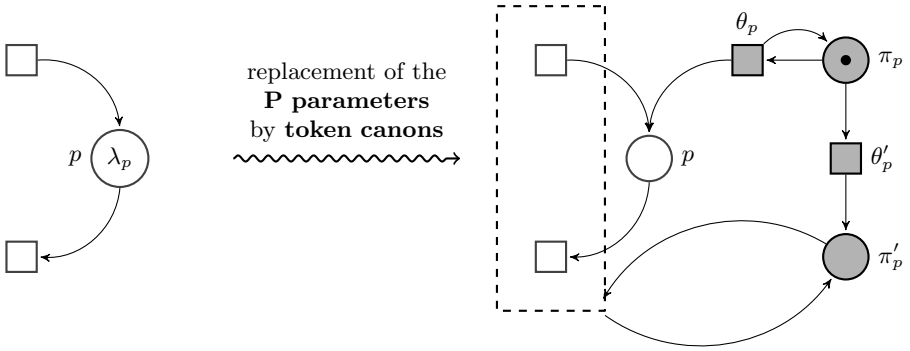


Fig. 9. From PPN to PN

Corollary 7 *$\mathscr{E}$-cov is decidable on postT-PPN.*

Proof. Decidability of $\mathscr{E}$-cov on postT-PPN:
We proved in previous section that postT-PPN and P-PPN are weakly-cosimilar. Therefore, given a postT-PPN we can built a P-PPN which is weakly-co-similar. Moreover, as coverability can be reduced to firing transition (by adding an observer transition), weak-simulation holds coverability. Theorem 6 gives us the decidability.

Theorem 8 *$\mathscr{E}$-cov is decidable for preT-PPN.*

Proof. $\mathscr{E}$-cov for the preT-PPN is decidable:
Let us consider a preT-PPN and a marking m that we try to cover. For an input transition with a weight of zero, we do not require the input place to be marked. Therefore, in terms of *input parameters*, by monotony, the zero valuation is the most permissive one for firing. Thus, there is at some valuation a firing sequence w such that $m_0 \xrightarrow{w} m_1 \geq m$ if and only if we can fire w in the *0-instanced* one, leading to a marking $m_2 \geq m_1$. Formally, given a preT-PPN $\mathcal{SP}$, we have:

$$m_0 \xrightarrow{w} m_1 \geq m \text{ in } \nu(\mathcal{SP}) \text{ iff } m_0 \xrightarrow{w} m_2 \geq m \text{ in } 0(\mathcal{SP}) \text{ with } m_2 \geq m_1$$

Informally, it means that the zero instance of the preT-PPN has the greatest amount of behaviours (in terms of coverability). Therefore it is the one which is necessary and sufficient to satisfy the $\mathscr{E}$-cov of m, meaning that if it does not satisfy the property, monotony implies that any instance of the preT-PPN will not satisfy either. If the *0-instanced* net covers m, we have a witness for the $\mathscr{E}$-cov.

Corollary 9 *$\mathscr{E}$-cov is decidable for distinctT-PPN.*

Proof. $\mathscr{E}$-cov for the distinctT-PPN is decidable:
As we can create a partition over Par between Par_{Pre} and Par_{Post}, respectively

sets of parameters involved on inputs and outputs which are disjoint. We can consider the partial valuation $0_{|Par_{Pre}}$, which maps every parameter of Par_{Pre} to 0. We therefore get a postT-PPN on which the problem is decidable. Moreover, the post-PPN built is the one with the greatest amount of behaviours for coverability as explained previously. Considering that, if we cannot find any instance of this postT-PPN satisfying the property, we cannot find any instance of this distinctT-PPN satisfying it either.

5.2 Study of Parameterised Reachability

In classic Petri nets *decidability of reachability* certainly implies *decidability of coverability*. Indeed, given a marked Petri Net and a coverability problem, we can construct another marked Petri Net over which the previous coverability problem is equivalent to a reachability problem. Actually, with notations of Fig-

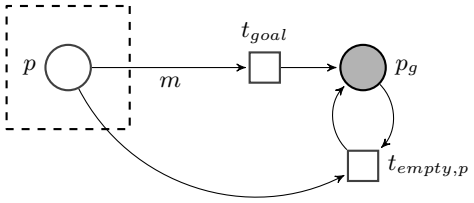


Fig. 10. Reducing Coverability to Reachability

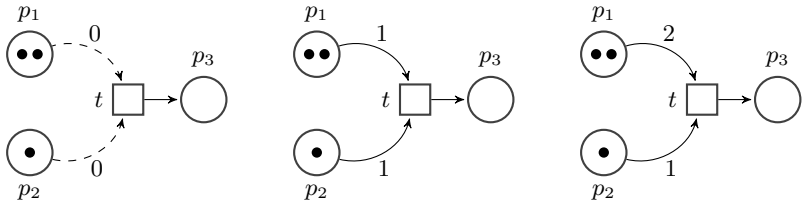
ure 10 covering a marking m is equivalent to reach the marking with only one token in place p_g in the net augmented with this new place p_g, a new transition t_{goal} such that ${}^\bullet t_{goal} = m$, with $Post(p_g, t_{goal}) = 1$ and, for each place p, a transition $t_{empty,p}$ such that: if p is not equal to p_g, $Pre(p, t_{empty,p}) = 1$, $Pre(p_g, t_{empty,p}) = 1$ and $Post(p_g, t_{empty,p}) = 1$. It is clear that the same can be done for PPN, which implies that *decidability of $\mathscr{E}$-reachability* implies *decidability of $\mathscr{E}$-coverability* and *decidability of $\mathscr{U}$-reachability* implies *decidability of $\mathscr{U}$-coverability*. Section 3 provides therefore the undecidability of ($\mathscr{E}$-reach) and ($\mathscr{U}$-reach) in the general case of PPN.

Theorem 10 *$\mathscr{E}$-reach is decidable on P-PPN.*

Proof. We can trivially adapt the conclusion of the proof of Theorem 6. We keep the same construction: the existence of a valuation such that a given marking is reached can be reduced to the reachability of the same marking (completed with 0 for each π_p and 1 for each π'_p added) which is known decidable as a classic reachability problem on an unbounded Petri net.

Nevertheless, for the other subclasses, the decidability of reachability is more complex. Intuitively, increasing the valuation used to instanciate a preT-PPN

(resp. a postT-PPN) leads to disable (resp. enable) transitions, *i.e.* the coverability of a marking, but this is not sufifcient to deduce the exact number of tokens involved, *i.e.* reachability.



(a) 0-instance (b) 1-instance (c) ν-instance

Fig. 11. Several instances of a preT-PPN

Figure 11 presents a preT-PPN. It is obvious that using the 0-valuation leads to enable the firing of t in any case, so it allows to cover any amount of tokens in p_2. In Figure 11(a), the coverability set is $CS_0 = \{m | m \leq (2, 1, \omega)\}$. On the other hand, increasing the valuation leads to potentially disable t. We will therefore reduce the coverability set as we strengthen the pre-condition to fire t: in Figure 11(b), the coverability set is $CS_1 = \{m | m \leq (2, 1, 0) \vee m \leq (1, 0, 1)\} \subseteq CS_0$, and in Figure 11(c), we have $CS_2 = \{m | m \leq (2, 1, 0) \vee m \leq (0, 0, 1)\} \subseteq CS_1$. Nevertheless, this strengthening of the pre-condition, does not imply general consequences in terms of reachability sets. Indeed, in Figure 11(a), the reachability set is $\{(2, 1, n) | n \in \mathbb{N}\}$, whereas in Figure 11(b) we can reach $\{(2, 1, 0), (1, 0, 1)\}$ and in Figure 11(c), we can reach $\{(2, 1, 0), (0, 0, 1)\}$.

(a) 0-instance (b) 1-instance

Fig. 12. Several instances of a postT-PPN

Equivalent observations rise from the study of Figure 12. When increasing the valuation, we may fire at least the same transitions, therefore, the coverability set is increasing: Figure 12(a) can cover any markings lower or equal to $(2, 0)$ and can reach the set $\{(2, 0), (1, 0), (0, 0)\}$ whereas Figure 12(b) can cover markings lower or equal to $(2, 0)$,$(1, 1)$ or $(0, 2)$ but can reach the set $\{(2, 0)(1, 1)(0, 2)\}$.

6 Conclusion

6.1 Main results

In this paper, we have introduced the use of discrete parameters and suggested parametric versions of the well known reachability and coverability problems. The study of the decidability of those problems leads to the results summed up in Figure 13 for coverability (Classes inside a dashed outline are decidable for the two corresponding parametric coverability problems). We recall that the

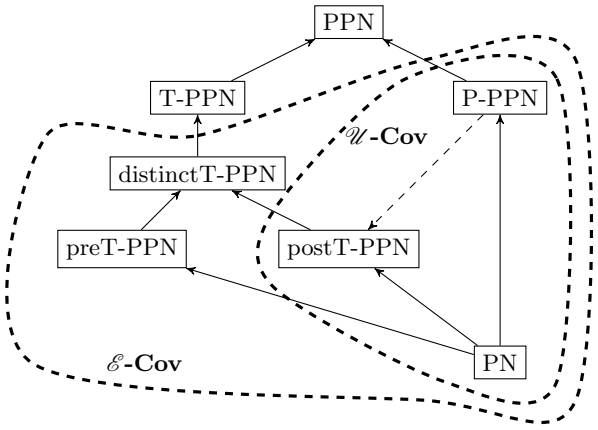


Fig. 13. What is decidable among the subclasses ? *(for coverability)*

other results are presented in Tab 1.

6.2 Future work

If we have strong intuitions for several empty cases such as decidability of $\mathscr{U}$-Coverability on preT-PPN and distinctT-PPN which would join the intuition that using the same parameters on inputs and outputs considerably increases the power of modeling of classic Petri nets, a deeper study should be carried to answer the decidability of Parametric-Reachability for instance. Being able to treat these parameterised models constitutes a scientific breakthrough in two ways:

- It significantly increases the level of abstraction in models. We will therefore be able to handle a much larger and therefore more *realistic class of models.*
- The existence of parameters can also address more relevant and *realistic verification issues.* Instead of just providing a binary response on the satisfaction or not of an expected property, we can aim to*synthetise* constraints

on the parameters ensuring that if these constraints are satisfied, the property is satisfied. Such conditions for the proper functioning of the system are essential information for the designer.

Acknowledgement

We wish to thank the anonymous reviewers, who helped us to improve the paper by their suggestions.

References

1. Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 592–601, New York, NY, USA, 1993. ACM.
2. Toshiro Araki and Tadao Kasami. Some decision problems related to the reachability problem for Petri nets. *Theoretical Computer Science*, 3(1):85–104, October 1976.
3. Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili. Regular model checking. In CAV, 2000.
4. G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. A symbolic reachability graph for coloured petri nets. *Theor. Comput. Sci.*, 176(1-2):39–65, April 1997.
5. C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In Kim G. Larsen, Sven Skyum, and Glynn Winskel, editors, *Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115. Springer Berlin Heidelberg, 1998.
6. N. D. Jones, L. H. Landweber, and Y. E. Lien. Complexity of some problems in Petri nets. *Theoretical Computer Science 4*, pages 277–299, 1977.
7. Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147 – 195, 1969.
8. Markus Lindqvist. Parameterized reachability trees for predicate/transition nets. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes in Computer Science*, pages 301–324. Springer Berlin Heidelberg, 1993.
9. Ernst W. Mayr. An algorithm for the general petri net reachability problem. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, pages 238–246, New York, NY, USA, 1981. ACM.
10. Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.