Discrete Parameters in Petri Nets

Nicolas David

UNIVERSITE BRETAGNE LOIRE

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Nicolas DAVID

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Nantes
Label européen
sous le sceau de l'Université Bretagne Loire

École doctorale : Sciences et technologies de l'information, et mathématiques

Discipline : Informatique et applications, section CNU 27
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N)

Soutenue le 20 Octobre 2017

Discrete Parameters in Petri Nets

JURY

Président :	**M. Serge HADDAD**, Professeur, ENS de Cachan, LSV
Rapporteurs :	**M. Gilles GEERAERTS**, Chargé de cours, autorisé à porter le titre de professeur, Université Libre de Bruxelles
	M. Jérôme LEROUX, Directeur de Recherche, CNRS, LaBRI, Bordeaux
Examinateurs :	**M^{me} Nathalie BERTRAND**, Chargée de Recherche, INRIA, Rennes
	M. Serge HADDAD, Professeur, ENS de Cachan, LSV
Directeur de thèse :	**M. Claude JARD**, Professeur, Université de Nantes, LS2N
Co-directeur de thèse :	**M. Didier LIME**, Maître de conférences, École Centrale de Nantes, LS2N

Discrete Parameters in Petri Nets

Mémoire présenté en vue de l'obtention du grade de

Docteur de l'Université de Nantes - Label européen

sous le sceau de l'Université Bretagne Loire

Nicolas David

École doctorale : Sciences et technologies de l'information, et mathématiques
Discipline : Informatique et applications, section CNU 27
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N - UMR 6004)

Directeur de thèse : Pr. Claude Jard, professeur de l'Université de Nantes
Co-directeur : Dr. Didier Lime, maître de conférences de l'École Centrale de Nantes

Soumis en Août 2017

Jury

Président du jury :

M. Serge Haddad, *Professeur*, ENS de Cachan, LSV

Directeur de thèse :

M. Claude Jard, *Professeur*, Université de Nantes, LS2N

Co-directeur :

M. Didier Lime, *Maître de conférences*, École Centrale de Nantes, LS2N

Rapporteurs :

M. Gilles Geeraerts, *Chargé de cours*, Université Libre de Bruxelles
M. Jérôme Leroux, *Directeur de Recherche*, LaBRI, Bordeaux

Examinateurs :

Mme Nathalie Bertrand, *Chargée de Recherche*, INRIA, Rennes
M. Serge Haddad, *Professeur*, ENS de Cachan, LSV

Discrete Parameters in Petri Nets

Short abstract: With the aim of increasing the modelling 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, the universal and the existential one: "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 nets, depending on whether the parameters are used on places, input or output arcs of transitions. For some classes, we prove that universal and existential coverability become decidable, making these classes more usable in practice. To complete this study, we prove that those problems are ExpSpace-complete. We also address a problem of parameter synthesis, that is computing the set of values for the parameters such that a given marking is coverable in the instantiated net. Restricting parameters to only input weights (preT-PPNs) provides a downward-closed structure to the solution set. We therefore invoke a result for the representation of upward closed set from Valk and Jantzen. The condition to use this procedure is equivalent to decide the universal coverability. We also propose an adaptation of this reasoning to the case of parameters used only as output weights (postT-PPNs). In this case, the condition to use this procedure can be reduced to the decidability of the existential coverability. Finally, we broaden this study by establishing decision frontiers through the study of existential and universal reachability.

Keywords: Petri nets - Parameters - Synthesis - Decidability - Complexity - Coverability - Reachability

Réseaux de Petri à Paramètres Discrets

Résumé court : Afin de permettre une modélisation plus souple des systèmes, nous proposons d'étendre les réseaux de Petri par des paramètres discrets représentant le poids des arcs ou le nombre de jetons présents dans les places. Dans ce modèle, tout problème de décision peut être décliné sous deux versions, une universelle, demandant si la propriété considérée est vraie quelles que soient les valeurs que prennent les paramètres et une existentielle, qui s'interroge sur l'existence d'une valeur pour les paramètres telle que la propriété soit satisfaite. Concernant la couverture, nous montrons que ces deux problèmes sont indécidables dans le cas général. Nous introduisons donc des sous classes syntaxiques basées sur la restriction des paramètres aux places, aux arcs en sortie ou aux arcs en entrée des transitions. Dans ces différents cas, nous montrons que la couverture existentielle et universelle sont décidables et EXPSPACE-complètes. Nous étudions alors le problème de la synthèse de paramètres qui s'intéresse à calculer l'ensemble des valeurs de paramètres telles que la propriété considérée soit vraie. Sur les sous classes introduites, concernant la couverture, nous montrons que les ensembles solutions à la synthèse ont des structures fermée supérieurement (cas des arcs de sortie) et fermée inférieurement (cas des arcs d'entrée). Nous prouvons alors que ces ensembles se calculent par un algorithme de la littérature, proposé par Valk et Jantzen, dont les conditions d'application se réduisent aux problèmes de décision étudiés précédemment. Enfin nous étudions les frontières de décision en nous intéressant aux versions paramétrées de l'accessibilité pour ces sous classes.

Mots clés : Réseaux de Petri - Paramètres - Synthèse - Décidabilité - Complexité - Couverture - Accessibilité

Acknowledgements

I would like to thank my advisors, Professor Claude Jard and Dr Didier Lime, for their support, their trust and their knowledge. I am especially grateful to Dr Didier Lime who offered me his guidance from my Master to this PhD. I also present special thanks to Professor Olivier (H.) Roux whose informed discussions helped me a lot. I am grateful to the rest of my thesis committee, Dr Étienne André and Dr Sébastien Faucou to whom it has always been a pleasure to present the progress of this thesis.

I would like to thank the University of Nantes who hired me during those 3 years through the funds of Pays de la Loire research project AFSEC. This work was also partially supported by ANR project PACS (ANR-14-CE28-0002). I am also grateful to the University staff and more specifically to the LS2N staff. I also thank the staff from the IUT of Nantes: it has been a pleasure to teach there.

I thank my colleagues from the AeLoS and the STR teams with whom I had numerous interesting discussions. I also thank those who had the habits of having lunch with me for their pleasant company. Of course, my appreciation also goes to the doctoral students for their cooperation and for their motivation.

My friends also deserve my appreciation and gratitude, from those I have known for a long time to those I met more recently. I would like to give heartfelt thanks to my parents Gilles and Isabelle, and my sister Marion. Even from the other side of France, they always supported me. Last but not least, thank you Soizic, for all your love, your caring understanding and your precious support.

Nicolas David

Contents

Part I
Introduction and State of the Art

CHAPTER 1

Introduction

"When you are curious, you find lots of interesting things to do."

— Walter Elias Disney

Contents

1.1 Context and Motivations

Of late, the growing impact and press coverage of security bugs (*Heartbleed*, or the exploit *EternalBlue* for instance) and of cyber attacks (such as the recent ransomware attack *WannaCry*) show how our society has become, during the last decades, a computer based society. Software and computers are involved while a majority of the population is not aware of this fundamental dependency. "We do not understand what the software does, regardless of how well educated or smart we are" explained Holger Hermanns, Professor of Dependable Systems and Software at Saarland University in the description of a recent European project [Hermanns, 2016]. Wireless networks, connected devices, self driving cars or computer assisted medical interventions have become ubiquitous and therefore more and more safety-critical. It is not only about money: lives and environment are directly involved. In parallel, complexity and interactions of systems are also spreading. A great importance must be given to asserting, with certainty, the safety of a system despite this growing complexity. That is what *formal methods* and *model checking* are all about: proving that the design of a system is correct with respect to some meaningful properties. In their book Principle of Model Checking [Baier and Katoen, 2008], Christel Baier and Joost-Pieter Katoen define model checking as follow:

> "Model checking requires a model of the system under consideration and a desired property and systematically checks whether or not the given model satisfies the property."

Model checking tools have proved their efficiency. A well-known example through the community is the modelling of the embedded software which controlled the Ariane-5 launcher during its flight (this system is detailed in [Bozga et al., 2001]). In the last few years, the European Research Council has funded many grants to formal methods related project. Finally, to demonstrate this growing interest toward verification, if there were still a need to do so, we can recall to the reader that the Turing award of 2007 has been awarded to Sifakis, Clarke and Emerson for their work in model checking and their concern to develop it into an effective verification technology. More recently, Lamport has been the recipient of the Turing award of 2013, notably for his work on the theory and practice of distributed and concurrent systems. In particular, he has worked on notions such as causality and logical clocks, safety and liveness. Therefore, maybe the conclusion of FAA (Federal Aviation Authority) and NASA (National Aeronautics and Space Administration) stated in the book Principle of Model Checking has never been this true:

> "Formal methods should be part of the education of every computer scientist and software engineer, just as the appropriate branch of applied maths is a necessary part of the education of all other engineers."

Nevertheless if this enthusiasm is founded and legitimate, it must be qualified. Indeed, formal methods are, in the best case, resources and time consuming and in the worst case inefficient. Model checking typically requires a complete knowledge of the system and its environment. Therefore, the verification step can be performed only once the design stage is fully accomplished which can be complex or impossible if its environment has an unpredictable behaviour, or if some variables may be bounded to a given range of values without clue to determine the ideal one, etc. Therefore, designing the system toward formal verification leads to an increase of the complexity of the model, and thus of the verification steps, the procedures being generally resource-hungry [1]. Moreover, this verification step only provides a Boolean answer. This implies that if the model of the system is proved wrong or if the environment changes, this complex verification process must be carried out again after changing the design accordingly. It would thus be interesting to provide the possibility of performing verification in an early design step, once a first abstract model has been defined but not necessarily entirely specified. This *less rigid modelling* and *early verification* would be more convenient since they would reduce the danger of losing time and resources through the complex phase of exhaustive but not necessarily correct specification. This would also accelerate the convergence toward the safe model by reducing the number of iterations between design and verification.

This might seems a bit paradoxical for a casual reader: what we assert and will underline in this thesis dissertation is how abstracting the model, by leaving some *holes* in it, will help to provide a more precise knowledge of the system. Let us first clarify what those *holes* exactly are by introducing the notion of parameters in this context.

1.2 What are Parameters?

Parameter comes from the Greek $\pi\alpha\rho\alpha$ meaning *beside* and $\mu\epsilon\tau\rho\sigma\nu$ meaning *measure*. Thus, following generic definitions, it is commonly used to refer to a feature or a measurable characteristic that is used to define a system with no restriction on the system considered (concrete or abstract, from a scientific field or not). From this wide definition, we target more specific interpretations: from an abstract point of view, in mathematics, it can be defined [2] as "a quantity whose value is selected for the particular circumstances and in relation to which other variable

1. This is a direct consequence of the state space explosion occurring when one wants to explore exhaustively the whole state space of a system. The state space can be very large or even infinite. In this latter case, it is impossible to explore the entire state space with limited resources of time and memory. The search must then be performed efficiently by using different methods of abstraction.

2. This definition is taken from the Oxford online dictionary. (see `https://en.oxforddictionaries.com/definition/parameter`)

quantities may be expressed". From a more concrete point of view, in experimental/technical field, it can be defined [3] as "a numerical or other measurable factor forming one of a set that defines a system or sets the conditions of its operation". Parameterisation consists in leaving holes represented by abstract entities. Those abstract entities can be then filled with concrete ones by instantiation. Let us consider some concrete examples of the use of parameters :

- In computer programming, a parameter used in the implementation of a function refers to one of the pieces of data provided as input to the function. When the function is effectively called, the actual pieces of data used are called arguments. The argument is thus the actual input passed to a function, that is to say an instance of the parameter inside its implementation. For example, we could define a procedure to concatenate two strings together. This would need two parameters, one for each string occurring in the effective call.
- In computer programming, it is also possible to encode algorithms or structures without mentioning the type of the data. Types can thus be specified latter. This is what we call generic programming or parametric polymorphism. Concrete examples are for instance definition of functions in Haskell where no types need to be specified or templates in C++.
- In automatic control, it is classic to experiment in order to adjust the model thanks to observations. Models can be directed by differential equations in which parameters are involved. The feedback of the experience permits then to adjust the instance of the model by adapting the value of those parameters. In fact, in automatic control, the comparison of a measured value of a process with a desired value leads to sharpen the model.

As stated previously, the introduction of parameters in the context of formal methods aims to improve genericity. It also allows the designer to leave unspecified aspects, such as those related to the modelling of the environment. This increase in modelling power usually results in greater complexity in the analysis and verification of the model, however.

1.3 Research Questions

Parameterised systems are of particular interest both in allowing the handling of more realistic classes of models and addressing more realistic verification issues. It can be challenging to find meaningful parametric infinite state systems with decidable decision problems. This thesis dissertation addresses three research questions:

RQ_1 *How to extend Petri nets with parameters in order to represent a whole family of nets through one given model?* We choose 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 *parametric Petri nets* or PPNs. The goal is here to find meaningful parametric models based on the paradigm of Petri nets. This includes finding subclasses that can be analysed more easily.

RQ_2 *How can we adapt decision problems to this parametric context? What problems are decidable? What are their complexities?* The introduction of parameters induces the need of quantifying the valuation of these parameters in decision problems. Basically, two main type of decision problems can be emphasised: on the one hand *existential problems*, asking whether there exists an integer valuation v on the set of parameters such that the instance where parameters are replaced by the value given by v satisfies some property, and on the other hand *universal problems*, asking if that property is satisfied in every possible instance of the parametric model.

3. See previous note.

RQ_3 *Can we answer the problem of the synthesis of parameters?* Beyond verification of properties, the use 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. This is the *synthesis problem*: given a property, compute the exact set of all the values of the parameters such that, instantiated with these values, the system satisfies this property. This notably permits an estimation of the *robustness* of a given instance of a model. Indeed, in full knowledge of "good values" for the parameters, we may be able to quantify the distance from a "bad value" providing an idea of how reliable is the system.

We will now focus on the position of our work in the current literature.

1.4 Position in the Current Literature

In terms of modelling, the first intuition behind parameters is to model an arbitrary number of identical resources.

1.4.1 Arbitrary Number of Identical Resources

In [Bouajjani et al., 2000], a framework for the verification of infinite-state systems called *regular model checking* is presented. States are represented by strings over a finite alphabet and the transition relation by a regular length-preserving relation on strings. This analysis relies on the use of regular sets of words over a finite alphabet to represent (symbolically) sets of states, and finite-state transducers (in fact regular length-preserving relations between words) to represent transition relations. The verification procedure is based on automata techniques. Nevertheless, states of the system need to be representable as finite strings (of arbitrary length) over a finite alphabet.

Many concurrent systems consist of an arbitrary number of identical processes running in parallel. The literature on this subject is therefore well developed. In [Bertrand et al., 2015] systems with an arbitrary amount of processes which communicate by broadcast are studied. Note that here the decision of the processes are constrained: two processes with same past behave similarly. This is called *local strategies*.

A fundamental intuition behind systems with arbitrary identical processes is to begin the analysis by considering a reduced system with one or two processes. This informal reasoning is addressed in [Browne et al., 1989] in order to provide a theoretical basis for this kind of analysis.

In [Abdulla et al., 2013], the author presents a simple framework for the verification of safety problems on systems with a parameterised number of processes. The technique relies on the detection of cut-off points beyond which the search of the state space does not need to continue, allowing to exploit small model property. Informally, the idea is that the reachability of bad configurations can be detected with only a small number of processes. In [Aminof et al., 2014] a similar analysis technique is notably addressed on concurrent systems in which processes communicate via pairwise rendezvous.

In [Esparza et al., 2016], systems with a leader process and arbitrarily many anonymous identical contributors are considered. It provides an analysis of the complexity of the safety verification problem on such systems.

This short overview of the literature is of course not exhaustive. A much more complete survey of different parameterised models and verification methods that can be applied to concurrent and distributed systems with broadcast communication based on transition systems is presented in [Delzanno, 2016].

1.4.2 Parameters Synthesis

The study of parameterised models and more specifically the question of the synthesis has been studied in different parametric settings. Parameters representing delays in timed systems

modelled as timed automata have been particularly studied, but with very few decidability results [Alur et al., 1993]. Synthesis for such system is only possible in very particular settings, such as bounded integer parameters computed symbolically in timed automata [Jovanović et al., 2015] or parameters in timed automata with parameters used only as upper bounds, or only as lower bounds, in timing constraints [Bozzelli and Torre, 2009].

Note that in the case of parameterising time, the domain of parameters is often the entire set of reals number. This is out of the scope of this thesis dissertation since we will only address discrete parameters but this would be an interesting way to extend this work for instance by considering continuous Petri nets [David and Alla, 1987] (or fluid Petri nets) where the marking is a vector of positive real numbers.

1.4.3 Parameters in Petri Nets

As noticed in [Biel et al., 2011], the literature dealing with parameters in Petri nets is a bit fragmented: parametric, parameterised or parameterized Petri nets are introduced with different viewpoints. The notion of parameter is indeed subject to different interpretations depending on the context and the problems addressed.

Parameters to Simplify the Modelling

Most articles introduce parameters in order to dynamically change the network structure when the parameter is valued. This is for example used to describe in the same network several levels of abstraction and allow the implementation of refinements. The appropriate level can be chosen in order to get the best representation for the specific problem. Depending on the context, places or transitions can themselves be parameters that can be replaced with more complex subnets, for instance in [Gracanin et al., 1993] or [Christensen and Mortensen, 1997]. In this latter paper, parameterisation on values on the arcs with quantities or functions is also considered but no verification is performed on those parameterised models. When parameters are used as place-holders for quantities, and that the set of possible valuations of those parameters is finite, parameterisation can be used primarily to shortcut the writing of the model. Parameters are also introduced in models such as predicate Petri nets [Lindqvist, 1993], directly on the markings, as a mean to fold reachability trees of Predicate/Transition nets into more concise parameterised reachability trees. Considering a similar idea, [Chiola et al., 1997] studies some symbolic reachability graph for coloured Petri net taking into account symmetries.

Verification Involving Parameters

When value parameterisation is considered, with a finite valuation domains, an expansion on all possible values is imaginable, although it is sometimes avoided by the use of symbolic techniques to conduct a formal analysis. For example, the model of Automaton Controlled Petri nets (ACPN), introduced in [Badouel and Oliver, 1999] uses parameters to handle dynamics change in a system. The possible valuations of the parameters (described in the weights of the arcs) are provided by the state of a finite automaton. On the other hand, some papers consider quantitative parameterisation with infinite valuation domains of the parameters. The analysis of such models can be based on abstract interpretation, as in [Abdulla et al., 2013]. The goal is indeed to reduce the verification problem to finite-state verification. This classic technique has been mentioned in Section 1.4.1. Valk's self-modifying Petri Nets [Valk, 1978] can also be considered in this category: those nets are able to modify their own firing rules by using linear function of the place markings as flow relations, but they are Turing powerful.

1.4.4 Position of our Work

In this manuscript, we only consider genericity on a unique level of abstraction since we would like to provide a quantitative analysis on parameters which would rely on a perennial

structure. We would like to provide a model allowing to model identical resources but also their synchronisation thanks to the paradigm of Petri nets. The formalism introduced in this thesis is not introduced in the aim to depict a precise case study. In this sense, we want to introduce a general purpose formalism. Nevertheless, we can guess that this increase in generecity will be gained at the cost of the complexity of the analysis. Moreover, we would like that the structure of the difference instances of the parameterised models keep the same structure in order to provide a quantitative analysis upon the valuations. Typically, a parameter can represent a number of processes but not only since we later provide an illustrative example by modelling a financial loan. In this context, we do not consider finite sets of valuation but infinite ones in order to study classic properties of Petri nets.

Close to our work, [Chiola et al., 1991] and [Marsan et al., 1994] present Petri Nets where the initial marking of the net can be parameterised. An initial marking with one or more parameters is an abstraction of a set of markings obtained by assigning different legal values to the parameters. We will prove in the sequel of this thesis that parameterising the markings can be simulated by parameterising the output weights. In a different manner, [Geeraerts et al., 2015] introduces ω-Petri nets (ωPN). This model augments classic Petri nets by allowing ω-labelled input or output arcs. We will provide more details on this latest model in Section 3.5 since we will elaborate on some links with our model of parametric Petri nets and reuse some of the results proved for ωPNs. The fundamental difference with our model is the non-determinism introduced by those labels. Indeed, an ω-label input (resp. output) arc consumes (resp. produces), non-deterministically, any number of tokens in its input place (resp. output place). This model is said to be mainly used to analyse parametric concurrent systems with dynamic threads creation.

1.5 Contributions of this Thesis

As briefly mentioned earlier, RQ_1 is addressed by the introduction of the model of parametric Petri nets (PPNs), aiming to represent families of concurrent systems. Discrete parameters are involved on the weight of arcs of the nets or in the initial marking. Moreover, syntactical subclasses are introduced in a matter of decidability and tractability. Parameters are restricted to initial markings only or weights of transitions only. In the latter case, we can impose that parameters on input arcs and output are distinct. Finally, we consider the case where parameters are used only as input weights or only as output weights.

To address RQ_2, we consider the general property of coverability to which many safety properties can be reduced and, to a lesser extent, boundedness and reachability (that are often the basis for the verification of more specific properties). This thesis dissertation deals with two decision problems induced by the use of parameters: The *existential coverability*: does there exists an integer valuation v on the set of parameters such that m is coverable in the marked Petri net where parameters are replaced by the value given by v? And the *universal coverability*: is m is coverable in such a net for every possible valuation v? Those problems are both undecidable in the most general case. Syntactic subclasses restricting the use of parameters have been introduced, for which the different problems are decidable and ExpSpace-complete. These results interestingly allows us to carry over a Rackoff upper bound into this parametric setting. Considering boundedness, we provide the result of decidability using some adaptations of the procedure of Karp and Miller for all those subclasses. We finally elaborate on decision frontiers by showing that the parametric reachability problems are undecidable for the subclasses of parametric Petri nets where parameters are used only on input arcs or only on output arcs, whereas it becomes decidable (for the existential problem) when parameters are used only on the initial marking. Note that the universal problem for this subclass is left open.

In order to address RQ_3, we then focus on computing the exact solution set to the synthesis problem for coverability in parametric Petri nets, i.e., the set of all parameter values such that in the net instantiated with these values, a given marking is coverable. The emptiness and universality of the solution set being undecidable in general, computing this set can only be

done in a restricted setting. We thus focus on the case where parameters are used only as input weights (preT-PPNs) or only as output weights (postT-PPNs). These assumptions give some structure to the solution set: we prove that it is then downward-closed wrt. the usual order on integer vectors for preT-PPNs, and upward-closed for postT-PPNs. We show how a procedure by Valk and Jantzen from [Valk and Jantzen, 1985] can be used for computing a finite minimal basis of the solution set for postT-PPNs or its complement for preT-PPNs. This requires deciding universal coverability in preT-PPNs and existential coverability in postT-PPNs. Finally, we prove that in what will be called distinctT-PPNs, *i.e.*, when the set of parameters appearing as input weights, and the set of parameters appearing as output weights are disjoint, the solution set cannot be represented using any formalism for which the emptiness of the intersection with equality constraints is decidable.

Those different results have been published in international conferences. The introduction of the model and some proofs of decidability have been first presented in a workshop [David et al., 2015b] and then have been published in [David et al., 2015a]. The work on complexities and synthesis has been accepted and will be published in [David et al., 2017].

1.6 Outline

All those results are presented in this thesis dissertation according to the following outline.

Chapter 2 gives basic notations and recalls useful mathematical results from different theories.

Chapter 3 reviews some elements of the state of the art regarding Petri nets, related decision problems and the classic procedure of Karp and Miller together with the model of ω-Petri nets.

Chapter 4 gives the basic definitions related to the formalism of parametric Petri nets, introduces two instances of decidability problems in parametric Petri nets, the universal and the existential instances, based on the definitions of classic properties of Petri nets. We also present the problem of the synthesis. We then provide a proof of the undecidability results of existential and universal coverability. Therefore, we refine the hierarchy of PPN to provide subclasses where universal or existential coverability are decidable.

Chapter 5 introduces these subclasses of our parameterised models and presents results of translation and monotonicity. We carry out a complete survey of decidability of parametric coverability in this restricted parametric setting with the corresponding complexities. We notably give constructions for proving the ExpSpace-completeness of the universal coverability for preT-PPNs. We then turn our attention to the problem of the synthesis.

In Chapter 6, we study the structure of the solution sets for preT-PPNs and postT-PPNs and show under which condition Valk and Jantzen's algorithm can be used to construct finite representation of those sets. To complete this study, we propose a direct synthesis algorithm for preT-PPNs. We also discuss the case of distinctT-PPNs.

In Chapter 7, we emancipate from the problem of the coverability in order to establish decision frontiers in this parametric settings by studying the problem of existential and universal reachability and proving their undecidability for the subclasses of preT-PPNs and postT-PPNs. Interestingly, existential reachability becomes decidable when we restrict the use of parameters to initial markings only. Note that the remaining problem of universal reachability in P-PPNs is still an open problem.

Chapter 8 and 9 respectively conclude this thesis dissertation and present directions for future work.

CHAPTER 2

Discrete Mathematical Background

"Everything has beauty, but not everyone sees it."
— Confucius

Contents

This chapter reviews basic notions and definitions about discrete mathematics from the field of mathematical logic, set theory and graph theory, from algebra, order theory and combinatorics. We also recall the notion of upward and downward closed sets. Those notions will be mainly reused in the definitions of the following chapters and in the different proofs. Readers familiar with those fields may skip this chapter or do a cursory reading and come back when specific references are made.

2.1 Notions of Set Theory and Order Theory

2.1.1 Set Theory

We recall here classic notions and notations of the set theory in the classic form of Zermelo-Fraenkel with the axiom of choice.

Intuitively, a set is a collection of distinct objects. The empty set is denoted by $\emptyset$, membership by $\in$, set inclusion by $\subseteq$ whereas strict inclusion is denoted by $\subset$. We denote set intersection by $\cap$, set union by $\cup$ and $\setminus$ is used to denote the set difference. We denote by $\mathbb{Z}$ the set of integers, and by $\mathbb{N}$ the set of natural numbers. We denote by ω an arbitrary large number such that for each $n \in \mathbb{N}$, $n + \omega = \omega$, $\omega - n = \omega$, $\omega \leq \omega$ and $n < \omega$. We denote by $\mathbb{N}_\omega$ is the union $\mathbb{N} \cup \{\omega\}$.

Let X be a finite set. We denote by 2^X the powerset of X, that is to say the set of all subsets of X and by $|X|$ the size of X. If $X \subseteq \mathbb{N}^k$, $\neg X$ denotes its complement in $\mathbb{N}^k$.

The Cartesian product of two sets A and B, denoted by $A \times B$, is the set of ordered pairs $\{(a, b) \mid a \in A, b \in B\}$. The Cartesian product can be generalised to the n-ary Cartesian product over n sets $X_1, \ldots, X_n$ such that $X_1 \times \cdots \times X_n = \{(x_1, \ldots, x_n) \mid \forall 1 \leq i \leq n, x_i \in X_i\}$.

We define correspondences between two sets through the notion of relation:

Definition 1 : Binary relation
A binary relation R between two sets X and Y is a subset of $X \times Y$. Given $(x, y) \in R$ we say that x is related to y by R, which is denoted by xRy. The domain of R, denoted by $\mathtt{dom}(R)$, is the set $\{x \in X \mid \exists y \in Y, (x, y) \in R\}$. The set Y is the co-domain of R whereas the image of R, denoted by $\mathtt{im}(R)$ is the set $\{y \in Y \mid \exists x \in X, (x, y) \in R\}$, that is to say the subset of the co-domain effectively reached.

We call the binary relation on X the binary relation between X and X itself. Given a binary relation $R \in X \times Y$, we define R^{-1} as the subset of $Y \times Y$ such that for all $x \in X$, for all $y \in Y$, $y\mathcal{R}^{-1}x$ iff $x\mathcal{R}y$. Given a binary relation $f \in X \times Y$, f is a partial function*iff* $(x, y) \in f$ and $(x, z) \in f$ implies that $y = z$. In this case, we can write equivalently $(x, y) \in f$ or $f(x) = y$ since there is no ambiguity and y is called the image of x by f. If $\mathtt{dom}(f) = X$, f is said to be a (total) function, we write $f : X \to Y$. The set of all (total) functions from X to Y is written Y^X. In the sequel we may use the term mapping to refer to a total function in order to emphasise that we establish a correspondence between elements in one set with elements in another set.

Given X and Y two sets, for any subset $X' \subseteq X$ and function $f \in Y^X$, we define the restriction $f_{|X'}$ of f to X' as the unique function from X' to Y such that $f_{|X'}(x) = f(x)$ for all $x \in X'$. We extend this notation to sets of functions: given $F \subseteq Y^X$, $F_{|X'}$ denotes its projection on X' that is to say $F_{|X'} = \{f_{|X'} \mid f \in F\}$. We now consider two sets A and B such that $X = A \cup B$ and $A \cap B = \emptyset$, and functions $g \in Y^A$ and $h \in Y^B$, we write $g \cup h \in Y^X$ the function defined by $(g \cup h)_{|A} = g$ and $(g \cup h)_{|B} = h$. We call $g \cup h$ the *union* of g and h.

Definition 2 : Sequence
A sequence is an enumerated collection of objects from a set A in which repetitions are allowed. Formally, it is a function $f_s : D \to A$ whose domain D is a convex subset of the set of integers, i.e. $\forall i, j \in D$ with $i \leq j$, $\forall k \in \mathbb{N}$, $i \leq k \leq j \Rightarrow k \in D$. For $i \in D$, the element associated to the index i, $f_s(i)$ is simply denoted by s_i, and the sequence itself is denoted by $(s_n)_{n \in D}$ or simply (s_n).

The length of a sequence $|(s_n)|$ correspond to the cardinal of D. If D is finite, (s_n) is said finite, if not, D is necessary countable, by convention we can thus define $|s_n| = \omega$. Finally, we denote $()$ the empty sequence which has a length equal to zero. Let A be a set, we denote by A^* the set of all finite sequences of elements of A. In language theory, A is called alphabet and those sequences can be seen as words. In this context, the empty sequence $()$ is often written ϵ and is called the empty word. Let $w \in A^*$ be a finite sequence. We write $|w|$ the length of w. Given $a \in A$, $|w|_a$ is the number of occurrences of a in w.

Given a finite sequence $s = s_1, s_2, \ldots, s_n$, we call every sequence t of the form $s_1, \ldots, s_m$ with $m \leq n$ a prefix of s. We write $t \sqsubseteq s$. Given L a language over the alphabet A, that is to say a subset of A^*, we denote by $Pref(L)$ the prefix closure of the language L, *i.e.* $Pref(L) = \bigcup_{s \in L} \{t \mid t \sqsubseteq s\}$.

2.1.2 Order Theory

We now introduce formally the intuitive notion of order using binary relations. We first need to introduce some properties for relations.

Definition 3 : Relation properties
Given a binary relation R on a set S we say that R is

- *reflexive:* $\forall x \in S,\ xRx$
- *irreflexive:* $\forall x \in S,\ \neg(xRx)$
- *symmetric:* $\forall x, y \in S,\ xRy \Leftrightarrow yRx$
- *asymmetric:* $\forall x, y \in S,\ xRy \Rightarrow \neg(yRx)$
- *antisymmetric:* $\forall x, y \in S,\ (xRy \wedge yRx) \Rightarrow x = y$
- *transitive:* $\forall x, y, z \in S,\ (xRy \wedge yRz) \Rightarrow xRz$
- *total:* $\forall x, y \in S,\ xRy \vee yRx$

Given a binary relation R on a set X, the transitive closure of R written R^+ is the minimal transitive relation on X that contains R. The reflexive transitive closure of R, written R^*, is the minimal transitive and reflexive relation on X that contains R. Based on those properties we now formalise the notion of order.

Definition 4 : Quasi order
A relation R that is reflexive and transitive is said to be a quasi order (qo for short).

A pair $(S, \lesssim)$ is a quasiordered set if $\lesssim$ is a quasiorder on S. Note that for the sequel, we use $\leq$ when the relation is antisymmetric and $\lesssim$ when it is not. For x and y elements of S and given a *qo* R on S, x and y are said *comparable* if either $(x, y) \in R$ or $(y, x) \in R$. A relation $<$ is a *strict order* on a set S if it is irreflexive and transitive (which implies asymmetry). A relation $\sim$ is an *equivalence relation* on a set S if it is reflexive, symmetric and transitive. Given any quasi order $\lesssim$ on a set S we can define: (i) a strict order $<$ given by $x < y$ iff $x \lesssim y \wedge \neg(y \lesssim x)$, (ii) an equivalence relation $\sim$ given by $x \sim y$ iff $x \lesssim y \wedge y \lesssim x$, (iii) its dual quasi order $\gtrsim$ given by $y \gtrsim x$ iff $x \lesssim y$.

Definition 5 : Well quasi order (See, *e.g.*, [Higman, 1952] and [Kruskal, 1972])
A well quasi-ordering (wqo for short) is a qo $\lesssim$ on a set S such that, for any infinite sequence $s = x_0, x_1, x_2, \ldots$ in S, there are indices $i < j$ with $x_i \lesssim x_j$.

Let us also recall the following property:

Lemma 1
Given a set S and a well quasi order $\lesssim$ on this set, let $p_0, p_1, \ldots, p_n, \ldots$ be an infinite sequence of elements of S. Then, there is an infinite sequence $p_{i_1}, p_{i_2}, \ldots, p_{i_n}, \ldots$ such that $p_{i_1} \lesssim p_{i_2} \lesssim \ldots \lesssim p_{i_n} \lesssim \ldots$. (with $i_1 < i_2 < \cdots < i_n < \ldots$).

This can be easily understood using a Ramsey argument as follows: consider the set $I = \{i \mid$ there exists no j *s.t.* $i < j$ and $p_i \leq p_j\}$. If I were infinite, then we could extract a subsequence of the terms indexed by the elements of I which would contradict the definition of wqo. Therefore I is finite. We can thus consider any p_n with n greater than the maximal element of I as the first element of such an infinite increasing subsequence. By construction of n and since $\mathbb{N} \setminus I$ is infinite, we can find an element $m > n$ such that $m \notin I$ and $p_n \leq p_m$. We can thus iterate the process and build a sequence satisfying the previous property.

Note that well quasi-orders can be defined in a various but equivalent manner, by ensuring that there exists no infinite strictly decreasing sequence (which is equivalent to a property called well foundedness) and that there is no infinite antichains (sequences of pairwise incomparable elements).

Some examples of wqos are listed below:

- $(S, =)$ where S is a finite set equipped with the equality relation on its elements
- $(\mathbb{N}, \leq)$, the set of natural numbers with standard ordering is a wqo (which is total).
- $(\mathbb{N}^k, \leq)$ where $\leq$ is the qo on $\mathbb{N}^k$ component-wise (this is known as the Dickson's lemma [Dickson, 1913] recalled below)
- $(\mathbb{N}^k_\omega, \leq)$ where $\leq$ is the qo on $\mathbb{N}^k_\omega$ component-wise

Lemma 2 : Dickson's lemma [Dickson, 1913]
$(\mathbb{N}^k, \leq)$, the set of vectors of k natural numbers with component-wise ordering, i.e. given two vectors m_1 and m_2, $m_1 \leq m_2$ if for all indexes i, $m_1(i) \leq m_2(i)$, is a wqo.

2.1.3 Deepening of the Order Theory

We reuse definitions and concepts from [Finkel and Goubault-Larrecq, 2009, Finkel and Goubault-Larrecq, 2012] which are summed up in [Finkel and Leroux, 2015].

Upward Closed Sets

An upward closed set of the well quasi ordered set $(\mathbb{N}^k, \leq)$ is a subset U of $\mathbb{N}^k$ such that if $x \in U$, $y \in \mathbb{N}^k$ and $x \leq y$ then $y \in U$. The upward closure of a vector u, written $\uparrow u$ is the set $\{m \in \mathbb{N}^k \mid u \leq m\}$. Given a set U, we write $\uparrow U$ for the upward closure of U, defined as $\uparrow U = \bigcup_{u \in U} \uparrow u$. This implies that $\uparrow U$ is the least upward closed set in which U is included. Any upward closed set U can be represented by a finite set F, called basis, such that $U = \uparrow F$. An element x of F is called minimal when for all element $y \in F$ such that $y \leq x$, then $x \leq y$. The set of all the minimal elements of $F \subseteq \mathbb{N}^k$ still form a basis of U independently of F. This basis is minimal for inclusion among all bases and is thus called the minimal upward basis of F.

Downward Closed Sets

A downward closed set of the well quasi ordered set $(\mathbb{N}^k, \leq)$ is a subset D of $\mathbb{N}^k$ such that if $x \in D$, $y \in \mathbb{N}^k$ and $y \leq x$ then $y \in D$. The downward closure of a vector d, written $\downarrow d$ is the set $\{m \in \mathbb{N}^k \mid m \leq d\}$. Given a set D, we write $\downarrow D$ the downward closure of D, defined as $\downarrow D = \bigcup_{d \in D} \downarrow d$. This implies that $\downarrow D$ is the least downward closed set in which D is included. Moreover, the downward closure of a finite set (of vectors in $\mathbb{N}^k$) is finite. To symbolically represent downward closed sets, we use the extension $\mathbb{N}^k_\omega$. The definitions remain otherwise the same. If D is a downward closed set, we can write $D = \mathbb{N}^k \cap \downarrow F$ where F is a finite set of $\mathbb{N}^k_\omega$. We call F a downward basis of D. An element x of F is called maximal when for all element $y \in F$ such that $x \leq y$, then $y \leq x$. The set of all maximal elements of $F \subseteq \mathbb{N}^k$ still form a basis of D independently of F. This basis is minimal for the inclusion among all bases and is thus called the minimal downward basis of D.

Some Notable Properties

We also recall important results on upward and downward closed sets (see, *e.g.,* [Bouajjani and Mayr, 1999]): the union and the intersection of two upward (resp. downward) closed sets is an upward (resp. downward) closed set. The complement of an upward closed set is a downward closed set and vice-versa. Given the basis of an upward closed set, it is possible to compute the basis of its complement using for instance the procedure suggested in Example 5 of [Goubault-Larrecq, 2009], and vice versa by adapting this procedure.

Let us recall those procedures. First, we recall the reasoning of [Goubault-Larrecq, 2009]. This report provides as an example that the complement of $\uparrow(1, 3, 2)$ is exactly the intersection $\mathbb{N}^3 \cap \downarrow\{(0, \omega, \omega), (\omega, 2, \omega), (\omega, \omega, 1)\}$. Indeed, not being greater than or equal to $(1, 3, 2)$ means

exactly having the first component less than or equal to 0, or the second component less than or equal to 2, or the third complement less than or equal to 1.

For the general case, given a vector $x = (i_1, i_2, \ldots, i_k)$ the complement of $\uparrow x$ in $\mathbb{N}^k$ is equal to $\mathbb{N}^k \cap \downarrow\{(\omega, \ldots, \omega, i_j - 1, \omega, \ldots, \omega) \mid 1 \leq j \leq k, i_j \geq 1\}$. Now if we consider a family of vectors $\{x_1, \ldots, x_m\}$, the complement of $\uparrow \{x_1, \ldots, x_m\}$ can then be computed as the intersection of the complements of $\uparrow x_1, \ldots, \uparrow x_m$. Moreover, we can notice that the following intersection $\downarrow \{y_1, \ldots, y_m\} \cap \downarrow \{z_1, \ldots, z_p\}$ can be computed as $\downarrow \{\min(y_i, z_j) \mid 1 \leq i \leq m, 1 \leq j \leq p\}$, where $\min(y_i, z_j)$ are computed component-wise, for instance $\min((1, \omega, 3, \omega, 2), (3, 5, 0, \omega, \omega)) = (1, 5, 0, \omega, 2)$.

Conversely, we can adapt this reasoning as follows: the complement of $\downarrow (\omega, 3, 2)$ is exactly $\mathbb{N}^3 \cap \uparrow\{(0, 4, 0), (0, 0, 3)\}$. Indeed, not being lower than or equal to $(\omega, 3, 2)$ means exactly having the first component greater than ω which is not possible, or the second component greater than or equal to 4, or the third complement greater than or equal to 3.

For the general case, given a vector $x = (i_1, i_2, \ldots, i_k)$ the complement of $\downarrow x$ in $\mathbb{N}^k$ is equal to $\mathbb{N}^k \cap \uparrow\{(0, \ldots, 0, i_j + 1, 0, \ldots, 0) \mid 1 \leq j \leq k, i_j \neq \omega\}$. Now if we consider a family of vectors $\{x_1, \ldots, x_m\}$, the complement of $\downarrow \{x_1, \ldots, x_m\}$ can then be computed as the intersection of the complements of $\downarrow x_1, \ldots, \downarrow x_m$. Moreover, we can notice that the following intersection $\uparrow \{y_1, \ldots, y_m\} \cap \uparrow \{z_1, \ldots, z_p\}$ can be computed as $\uparrow \{\max(y_i, z_j) \mid 1 \leq i \leq m, 1 \leq j \leq p\}$, where $\max(y_i, z_j)$ are computed component-wise, for instance $\max((1, \omega, 3, \omega, 2), (3, 5, 0, \omega, \omega)) = (3, \omega, 3, \omega, \omega)$.

Finally, Valk and Jantzen proposed in [Valk and Jantzen, 1985] a necessary and sufficient condition, recalled in Lemma 3, to ensure that a finite basis of an upward closed set is effectively computable.

Lemma 3 : [Valk and Jantzen, 1985]
Given an upward closed set $U \subseteq \mathbb{N}^k$, a finite basis of U is effectively computable iff for each $v \in \mathbb{N}^k_\omega$, the emptiness of $\downarrow v \cap U$ is decidable, which is also equivalent to ask whether for all element $v \in \mathbb{N}^k_\omega$, it is decidable to answer whether $\downarrow v \cap \mathbb{N}^k \subseteq \neg U$.

2.2 Notions of Graph Theory, Algebra and Enumerative Combinatorics

2.2.1 Graph Theory

We now consider a set and a binary relation over itself. This pair of elements forms a directed graph.

Definition 6 : Directed graph
A directed graph G is a pair (V, A) where V is a set of vertices and $A \subseteq V \times V$ is a set of pairs of vertices, called arcs.

In graph theory, it can be relevant to consider graphs where the nodes can be of different kinds. When vertices can be divided into two disjoint sets U and V, such that no arc relates two vertices from the same set, we can refine the above definition:

Definition 7 : Bipartite directed graph
A bipartite directed graph G is a triplet (U, V, A) where U and V are two disjoints sets of vertices and $A \subseteq (U \times V) \cup (V \times U)$ is a set of pairs of vertices, called arcs.

Given a graph, we define sequences of nodes connected by consecutive arcs as a path.

Definition 8 : Path
Given a directed graph $G = (V, A)$ and $n \in \mathbb{N}$, A path $w : \{1, \dots, n\} \to V$ is a finite sequence of nodes such that $\forall i \in \{1, \dots, n-1\}, (w(i), w(i+1)) \in A$.

We define a rooted tree as a triplet (T, r, S) where:

- $S \subseteq T \times T$, $r \in T$ is the root, and for all $t \in T$, rS^*t (with S^* the reflexive transitive closure of S)
- $\forall t \in T$, $t \neq r \Rightarrow \exists$ a unique $t' \in T$ s.t. $t'St$
- $\forall t \in T$, $\neg(tS^+t)$ (with S^+ the transitive closure of S)

Informally, it is a directed graph where any two vertices are connected by one path and every arc can be assigned a natural orientation away from one given vertex (the root). We call the depth of a node the length of a path from the root to this node. By convention, the depth of the root is thus equal to zero.

Lemma 4 : König infinity lemma
Let $\mathscr{T}$ be a rooted (directed) tree in which each vertex has a finite number of successors (finite branching) and there is no infinite path directed away from the root. Then $\mathscr{T}$ is finite.

Each of those structures can be augmented by labelling functions assigning to each vertex or each arc an object.

We consider two alphabets Σ_1 and Σ_2. We consider a labelled tree defined as a graph by a tuple $\mathscr{C} = (N, r, B, \Lambda, \Gamma)$ where (N, r, B) is a rooted tree, $\Lambda : N \to \Sigma_1$ labels nodes with Σ_1, and $\Gamma : B \to \Sigma_2$ labels arcs with Σ_2. We say that x is an ancestor of y iff $(x, y) \in B^+$ where B^+ is the transitive closure of the relation B. Then, given a node $n \in N$, $\mathsf{Ancestor}_{\mathscr{C}}(n)$, is the set of ancestors of n in $\mathscr{C}$ plus n itself. Given two nodes x and y such that $x \in \mathsf{Ancestor}_{\mathscr{C}}(y)$, we write $x \rightsquigarrow y$ to denote the unique sequence of nodes along the path from x to y, that is to say $x = n_0, n_1, \dots, n_k = y$ such that each $(n_i, n_{i+1}) \in B$. We denote by $\mathsf{path}_{\mathscr{C}}(x, y) \in B^*$ the sequence of edges leading from x to y, formally it is equal to $(n_0, n_1), (n_1, n_2), \dots, (n_{k-1}, n_k)$. The corresponding label is given by $\mathsf{pathlabel}_{\mathscr{C}}(x, y) \in \Sigma_2^*$. Given two trees $\mathscr{T}_1 = (N_1, r_1, B_1, \Lambda_1, \Gamma_1)$ and $\mathscr{T}_2 = (N_2, r_2, B_2, \Lambda_2, \Gamma_2)$, a mapping ϕ from N_1 to N_2 is a labelled tree isomorphism *iff*

- ϕ maps the root of $\mathscr{T}_1$ to the root of $\mathscr{T}_2$ *i.e.* $\phi(r_1) = r_2$.
- $\forall (u, v) \in N_1$, $(u, v) \in B_1 \Leftrightarrow (\phi(u), \phi(v)) \in B_2$ with $\Gamma_1(u, v) = \Gamma_2(\phi(u), \phi(v))$
- for every node n of N_1, $\Lambda_1(n) = \Lambda_2(\phi(n))$.

Given a tree $\mathscr{C} = (N, r, B, \Lambda, \Gamma)$, $\mathscr{C}' = (N' \subseteq N, r', B' \subseteq B, \Lambda', \Gamma')$ is a prefix of $\mathscr{C}$ *iff* for each $y \in N'$, either $y = r = r'$ or if there exists a node $x \in N$ such that $(x, y) \in B$ then $x \in N'$ and $(x, y) \in B'$. Note that $\mathscr{C}'$ is indeed a tree and that Λ' and Γ' are respectively the restrictions of Λ and Γ to N' and B'.

2.2.2 Elementary Algebra

Given a set of variables X, we define a *linear expression* on X by the following grammar: $\lambda ::= k \mid k * x \mid \lambda + \lambda$ where $k \in \mathbb{Z}$ and $x \in X$.

Let $V \subseteq \mathbb{N}_\omega$, a V-valuation for X is a function from X to V. We refer to $\mathbb{N}_\omega$-valuations as *extended* valuations and to $\mathbb{N}$-valuations simply as valuations. The set $V^\emptyset$ of valuations from $\emptyset$ to V is reduced to a singleton $\{\emptyset_V\}$ where $\emptyset_V$ is the empty function. If X is finite, considering some arbitrary order on X, an (extended) valuation can be seen as a vector of size $|X|$.

Given a value a of $\mathbb{N}_\omega$, we denote as $\vec{a}$ the uniform (extended) valuation that maps every element of X to a. Given an extended valuation v, we write $\omega(v)$ for the subset of X such that $x \in \omega(v)$ iff $v(x) = \omega$. We write $\mathbb{N}(v)$ for the subset of X such that $x \in \mathbb{N}(v)$ iff $v(x) \in \mathbb{N}$.

Given a linear expression λ on X and an extended valuation v on $X' \subseteq X$, $v(\lambda)$ is the linear expression obtained when substituting each element x in X' from λ, by the corresponding value $v(x)$. If $X' = X$ we obtain an element of $\mathbb{N}_\omega$.

2.2.3 Combinatorics

Given a function $f : X \to Y$, f is said to be injective *iff* given two elements x and x' of X, $f(x) = f(x')$ implies that $x = x'$. That is to say that f never maps distinct elements of its domain to the same element of its co-domain. Note that given x in X, $y = f(x)$, when f is injective, x is called the fiber of y by f.

A function f is said to be surjective *iff* for every y of Y, there exists $x \in X$ such that $y = f(x)$. That is to say that the image of f is equal to its co-domain.

If f is both injective and surjective, f is said bijective. Informally, this means that the elements from the domain and the codomain of f are in one to one correspondence by f. Given a finite set X, we call the bijections (necessarily total) from X to X permutations.

Given a finite set X, S_X denotes the symmetric group on X (*i.e.* the set of all permutations of elements of X). If X is finite such that $|X|$=n, S_X is finite and contains $n!$ elements where $n!$ denotes the factorial of n. The notion of permutation is related to the act of arranging all the members of a set into some sequence or order, or if the set is already ordered, reordering its elements.

We also recall a well known result usually attributed to Dirichlet as the Dirichlet's drawer principle but more commonly referred to as the pigeonhole principle.

Lemma 5 : Pigeonhole principle
There does not exist an injective function whose co-domain is smaller than its domain.

If n items are put into m containers, with $n > m$, then at least one container must contain more than one item. By extension, if n is infinite, then at least one container contains an infinity of items. Thus, if $n = \omega$, at least one container contains ω items.

Numerous proofs will rely on this principle.

2.3 Computability Theory

Computability theory is a branch of mathematical logic and theory of computation. This theory notably addresses the notion of computable sets. We turn our interest toward decidability of decision problems in the sense originally introduced through [Turing, 1937].

2.3.1 Turing Machines

A Turing machine is a theoretical machine that reads and write symbols one at a time at a position given by the head of the machine which moves on the cells of an infinite tape by following a program. Let us provide a formal definition. Given an alphabet A we define $\tilde{A} = A \cup \{\square\}$ where $\square$ will be used to represent an empty cell from the tape.

Definition 9 : Turing Machine
A Turing machine on an alphabet A is a pair $(Q, \to)$ such that:

- *Q is a set of states containing at least the initial state* init*, the accepting state* accept *and the rejecting state* reject
- *$\to \subseteq Q \times \tilde{A} \times \tilde{A} \times \{-1, 0, +1\} \times Q$ is the transition relation*

We define the configuration of a Turing machine as a triplet (q, f, i) where $q \in Q$, $f : \mathbb{Z} \to \tilde{A}$ represents the contents of the tape and $i \in \mathbb{Z}$ represents the current position of the head. The initial configuration defined on the input w is the triplet $(\mathsf{init}, f_0, 0)$ where for $0 \leq i \leq |w| - 1$, $f_0(i) = w(i)$ and $f(i) = \square$ otherwise.

Given a Turing machine, its configuration (q, f, i) yields in one step the configuration (q', f', i') iff $\exists (q, a, b, x, q') \in \to$ such that:

- $a = f(i)$ (*i.e.* the head reads a on the tape)
- $b = f'(i)$ (*i.e.* the head writes b on the tape)
- $\forall j \neq i,\ f'(j) = f(j)$
- $i' = i + x$

Given a configuration if there is only one transition yielding another configuration, the machine is said deterministic. Otherwise it is said non-deterministic. Given an input w, if the machine reaches the state accept the machine stops and return true and if the machine reaches the state reject the machine stops and return false.

2.3.2 Decision Problems and Decidability

In computer science, a problem is characterised by a set of input instances and a task to perform on these input instances. The problem is encoded into a specialised format for efficient transmission, storage or processing by the machine.

A decision problem is a question which can be answered by a Boolean value depending on the input values. Note that the set of possible input values may be infinite. For instance asking whether an integer $n \in \mathbb{N}$ is a prime number is a decision problem. A decision problem can be assimilated to the set of possible inputs, called solution set, for which the answer is *true*.

A decision problem is said decidable *iff* there exists an algorithm which terminates after a finite amount of time (which may depend on the input value) and correctly decides whether the input is in the solution set or not. It is semi-decidable if there exists an algorithm that correctly decides when an input is in the solution set but may give no answer (*i.e.* may not terminate) for input not in the solution set. A problem that is not decidable is called undecidable. Note that some examples of decidable decision problems are presented in the next chapter when introducing the background on Petri nets, more precisely in Section 3.2.

In order to formalise the notion algorithm we need to rely on a specific formal model of computation. In particular we use here the formal model of Turing machine presented above. Indeed, any Turing machine can be translated in an effective algorithm. This means that every problem that is decidable through a Turing machine is decidable through an algorithm. Reciprocally, we admit that the decidability of a problem implies its decidability through a Turing machine. This latter implication is often referred to as the *Church-Turing thesis*.

Therefore, the definition of decidability is equivalent to the existence of a Turing machine which terminates after a finite number of steps (which may depend on the input value) and correctly decides whether the input is in the solution set or not.

2.3.3 Computational Complexity

We now focus on classifying decidable problems according to their difficulty by formalising that for a given problem, finding its solution requires a certain amount of resources, whatever the algorithm used. In particular, in this section, we provide the definition of the class EXPSPACE since the majority of the results proved in this manuscript belong to it.

To quantify the amount of resources needed, we refer to the Turing machines: if we consider decidable problems, we can deduce that there exists a Turing machine solving any instance of this problem, this instance being the input of the machine. Based on this observation, we define the complexity in time as the number of steps performed by the machine to stop (which might depend on the encoding). We also define the complexity in space as the maximal number of cells used (*i.e.* containing a element of the alphabet different from $\square$) by the machine along its execution, from the beginning until the termination.

Given two functions $f, g : \mathbb{N} \to \mathbb{N}$, we compare the asymptotic behaviour of f and g by writing that $g(n) = O(f(n))$ when n increases toward infinity which means that there exists some constants $c \in \mathbb{R}^+$, the set of positive real numbers, and $N \in \mathbb{N}$ such that for all $n \geq N$, $g(n) \leq cf(n)$. In particular, if we call $g(n)$ the complexity of the Turing machine on an input

of size n. We denote the asymptotic complexity of a Turing machine by $O(f(n))$ according to the previous definition.

The class ExpSpace is the set of all decision problems solvable by a deterministic Turing machine in $O(2^{p(n)})$ space, where $p(n)$ is a polynomial function of n. The class NExpSpace is the set of all decision problems solvable by a non-deterministic Turing machine in $O(2^{p(n)})$ space, where $p(n)$ is a polynomial function of n. Those two classes are known to be equal:

Corollary 6 : Corollary of Savitch's theorem [Savitch, 1970]
NExpSpace is equal to ExpSpace.

A decision problem is ExpSpace-complete if it is in ExpSpace, and every problem in ExpSpace has a polynomial-time reduction to it. That is to say that there is a polynomial-time [1] algorithm that transforms instances of one to instances of the other with the same answer. Intuitively, it means that the complexity has an ExpSpace upper bound and that it is as hard as any ExpSpace problem which provides a lower bound. This lower bound is referred to as ExpSpace-hardness.

We provided an extensive list of mathematical notions and background notations used throughout this thesis dissertation. We are now in position to introduce elements of the state of the art regarding Petri Nets which will be done in the following chapter.

1. An algorithm is said to be polynomial-time (or to have a polynomial running time) if there exists $k \in \mathbb{N}$ and $C > 0$ such that given any input of size n its running time is at most Cn^k.

CHAPTER 3

Preliminaries on Petri Nets

"Those who do not want to imitate anything, produce nothing."

— Salavador Dali

Contents

This chapter intends to introduce Petri nets in such a way that only classic properties and results are presented. Those results will be extended to a parametric context in the following chapters. Petri nets (or place/transition net) are a basic model of parallel and distributed systems introduced by Carl Adam Petri [Petri, 1962]. Petri nets are designed to model discrete event systems by exhibiting behaviours such as concurrency, conflict or dependency between events in both a graphical and formal manner through a strong mathematical background.

3.1 Definitions and Examples

3.1.1 Formal Definition

A Petri net is a bipartite graph consisting of places, transitions and arcs. An example is provided in Figure 3.1. Transitions represent events that may occur, they are depicted by squares (or sometimes bars in the literature). Places represent conditions, they are depicted by circles.

Definition 10 : Petri nets
A Petri net is a 4-tuple $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post})$ *where* P *is a finite set of places,* T *is a finite set of transitions,* Pre *and* $\mathsf{Post} \in \mathbb{N}^{P \times T}$ *are the backward and forward incidence functions, where for any* $p \in P$ *and* $t \in T$, $\mathsf{Pre}(p,t) = n$ *with* $n > 0$, *when there is an arc from place* p *to transition* t *with weight* n *and* $\mathsf{Post}(p,t) = n$ *with* $n > 0$ *when there is an arc from transition* t *to place* p *with weight* n. *If there is no arc between* p *and* t, *the corresponding values are equal to zero.*

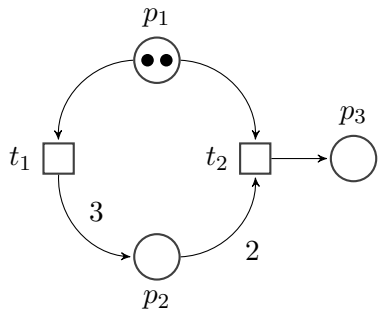
Given a Petri net $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post})$, and a transition t of T, we define $\mathsf{Pre}(t)$ as the univariate function on P at the point t which associates to each p of P the weight $\mathsf{Pre}(p,t)$. This function can be represented as that vector $(\mathsf{Pre}(p_1,t), \mathsf{Pre}(p_2,t), ..., \mathsf{Pre}(p_{|P|},t))$. We define $\mathsf{Post}(t)$ in a similar way. Seeing a transition t as an event occurring in a system, the input weights of t, $\mathsf{Pre}(t)$, can be seen as the pre-conditions of this event. The ouput weights $\mathsf{Post}(t)$ can be seen as its post-conditions. Note that Pre and Post can equivalently be seen as matrices from $\mathbb{N}^{|P| \times |T|}$. In the sequel, we may consider one formalism or the other depending on the context, in particular, each graphical representation of Pre or Post will rely on its matrix representation.

Definition 11 : Marking
A marking *of a Petri net* $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post})$ *is an* $\mathbb{N}$*-valuation for* P.

If $m \in \mathbb{N}^P$ is a marking, $m(p_i)$ represents a number of *tokens* in place p_i. Note that markings can be represented as vectors of $\mathbb{N}^{|P|}$.

Definition 12 : 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 3.1.



$$\mathsf{Pre} = \begin{array}{c} \begin{matrix} t_1 & t_2 \end{matrix} \\ \begin{bmatrix} 1 & 1 \\ 0 & 2 \\ 0 & 0 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} \qquad m_0 = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix}$$

$$\mathsf{Post} = \begin{array}{c} \begin{matrix} t_1 & t_2 \end{matrix} \\ \begin{bmatrix} 0 & 0 \\ 3 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix}$$

Figure 3.1 – A marked Petri net

3.1.2 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 $m \geq \mathsf{Pre}(t)$.

A *transition system* T is a directed graph with an initial state s_0, $(S, s_0, \rightarrow)$. It is used to describe the potential behaviour of discrete systems. It is convenient to associate a name to each element of the relation $\rightarrow$. Given an alphabet A, a *labelled transition system* T' is a triplet $(S', s'_0, \rightarrow')$ where $s'_0 \in S'$ is the initial state and $\rightarrow' \subseteq S \times A \times S$.

Definition 13 : Petri nets semantics
The semantics of Petri nets is a labelled transition system $\mathcal{S}_{\mathcal{T}} = (Q, q_0, \rightarrow)$ *where,* $Q = \mathbb{N}^{|P|}$, $q_0 = m_0$, $\rightarrow \subseteq Q \times T \times Q$ *such that, for all* $t_i \in T$, $m, m' \in \mathbb{N}^P$

$$m \xrightarrow{t_i} m' \Leftrightarrow \begin{cases} m \geq \mathsf{Pre}(t_i) \\ m' = m - \mathsf{Pre}(t_i) + \mathsf{Post}(t_i) \end{cases} \tag{3.1}$$

This relation can be extended to sequences of transitions as follows:

(i) $m \xrightarrow{\epsilon} m'$ if $m = m'$

(ii) $m \xrightarrow{wt} m'$ if $\exists m'', m \xrightarrow{w} m'' \wedge m'' \xrightarrow{t} m'$ where $w \in T^*$ and $t \in T$.

We write $\xrightarrow{*}$ the reflexive transitive closure of $\rightarrow$, i.e., $m \xrightarrow{*} m'$ when there exists $w \in T^*$ such that $m \xrightarrow{w} m'$. A path from m_0 in the semantics is called an *execution*.

3.1.3 Well Structured Transition Systems

Since markings can be seen as vectors of $\mathbb{N}^{|P|}$, we can consider a wqo invoking Dickson's Lemma (see Lemma 2). This conveys a special structure called *well structured transition system* which has been widely studied.

Definition 14 : Well structured transitions system [Abdulla et al., 1996, Finkel and Schnoebelen, 2001]
A well structured transition system (WSTS for short) is a tuple $S = (S, s_0 \rightarrow, \leq)$ *where* $(S, s_0, \rightarrow)$ *is a transition system,* $\leq \subseteq S \times S$ *is a quasi order and:*

- $\leq$ *is a wqo*
- $\leq$ *is upward compatible with* $\rightarrow$ *i.e.* $\forall s_1 \leq t_1$ *and transition* $s_1 \rightarrow s_2$, $\exists t_1 \xrightarrow{*} t_2$ *with* $s_2 \leq t_2$

In fact, compatibility can satisfy several additional criteria.

Definition 15 : Strong compatibility [Finkel and Schnoebelen, 2001]
A WSTS S has strong compatibility iff $\forall s_1 \leq t_1$ *and transitions* $s_1 \rightarrow s_2$, $\exists$ *a transition* $t_1 \rightarrow t_2$ *with* $s_2 \leq t_2$

Definition 16 : Strict compatibility [Finkel and Schnoebelen, 2001]
A WSTS S has strict compatibility iff $\forall s_1 < t_1$ *and transition* $s_1 \rightarrow s_2$, $\exists t_1 \xrightarrow{*} t_2$ *with* $s_2 < t_2$

Petri nets are WSTS's with strong strict compatibility. This property is also called strong monotonicity. We will see in the sequel how this property plays a key role in many analysis techniques of Petri nets.

3.1.4 Equivalent Modelling Formalism

One may notice that we sometimes invoke results and arguments initially used or proved on the models of *vector addition systems* and *vector addition systems with states* directly to our context. In fact those two models are equivalent to Petri nets. Vector addition systems (or VAS for short) have been introduced in [Karp and Miller, 1969].

Definition 17 : Vector addition system (VAS)
A VAS G of dimension n is a pair (d, W) where $d \in \mathbb{N}^n$ is called the start vector and W is a finite set of vectors in $\mathbb{Z}^n$.

The set of vectors that can be generated by G (called the *reachability set*, see *infra*) is then defined as the set of all vectors of the form $x = d + v_1 + \cdots + v_j$ where for all $i = 1 \dots j$, v_i belong to W and for all i such that $1 \leq i \leq j$, $d + v_1 + \cdots + v_i \geq 0$. VAS have then been generalised to vector addition systems with states (or VASS for short) in [Hopcroft and Pansiot, 1979].

Definition 18 : Vector addition system with states (or VASS)
A VASS G of dimension n is a finite directed graph (P, T) where transitions are labelled by vectors of $\mathbb{Z}^n$ together with an initial state $p_0 \in P$ and an initial vector $d \in \mathbb{N}^n$.

A configuration of a a VASS $G = (P, T, p_0, d)$ of dimension n consists in a tuple containing a control state from the set P and a integer vector of dimension n. The idea is that a transition can be fired from the current configuration *iff* it is in the corresponding control state and the sum of its effect with the vector of its current configuration is non negative. The firing of a transition will thus result in changing the current control state and updating the current vector accordingly. Both are computationally equivalent to Petri nets as detailed in [Reutenauer, 1989]. There exist indeed polynomial translations from VASS to VAS and PN and from PN to VAS (and VASS) [Reutenauer, 1989, Avellaneda and Morin, 2012].

3.2 Behavioural Properties of Petri Nets

We can now turn our attention to behavioural properties of Petri nets. In this context, one of the most important problem is to know whether a marking is reachable from the initial state. This problem is called the *reachability problem* and most verification problems are based on it. Formally:

Definition 19 : Reachability
Let $\mathcal{S} = (\mathcal{N}, m_0)$, where $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post})$, a marking m of $\mathbb{N}^P$ is reachable in $\mathcal{S}$ if $m_0 \xrightarrow{*} m$.

The *reachability set* $\mathcal{RS}(\mathcal{S})$ of $\mathcal{S}$ is the set of all reachable markings of $\mathcal{S}$. The reachability problem is decidable in PN but its complexity is still an open question. Notwithstanding, it is known to be ExpSpace-hard [Cardoza et al., 1976]. The original reachability proof was provided by Mayr in [Mayr, 1984]. It has been later simplified by Kosaraju in [Kosaraju, 1982] and Lambert in [Lambert, 1992]. The main idea of the construction behind this proof is also presented in [Reutenauer, 1989]. In other papers, this problem is solved using inductive invariants and Presburger definable sets such as in [Finkel and Leroux, 2015, Leroux, 2009].

Another behavioural property of interest is called *coverability*. It allows to verify safety properties.

Definition 20 : Coverability
Let $\mathcal{S} = (\mathcal{N}, m_0)$, where $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post})$, and m a marking of $\mathbb{N}^P$, m is coverable in $\mathcal{S}$ iff $\exists m' \in \mathcal{RS}(\mathcal{S}), m' \geq m$.

Coverability permits to answer properties which are tested using observers since it can be seen as the firability of a transition. The *coverability set* $\mathcal{CS}(\mathcal{S})$ of $\mathcal{S}$ is the set of markings coverable in $\mathcal{S}$. Coverability is decidable in marked Petri nets [Karp and Miller, 1969]. The construction [1] provided by Karp and Miller is of non-primitive recursive space complexity. The

1. This classic construction is described in Section 3.3.2.

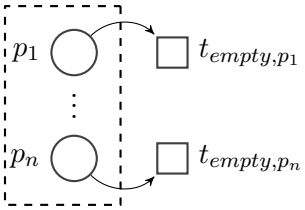


Figure 3.2 – Reducing coverability to reachability

original proof that the coverability problem is ExpSpace-complete is provided in [Rackoff, 1978]. The problem of coverability has been widely studied and is still studied nowadays. Indeed, more recently, [Finkel and Leroux, 2015] suggested an approach based on inductive invariants to solve coverability. The coverability set is an over approximation of the reachability set in the sense that $\mathcal{CS}(\mathcal{S}) =\downarrow \mathcal{RS}(\mathcal{S})$. Given a marked PN $\mathcal{S}$, and a marking m, we denote by $\mathsf{cov}(\mathcal{S}, m) \in \{True,\ False\}$ the coverability of m in $\mathcal{S}$. We recall that the coverability set of a marked Petri net is computable in the sense that its minimal downward basis is computable (see, *e.g.*, [Finkel and Leroux, 2015]). Here, the minimal downward basis is a finite set of extended markings. Given a marked Petri net $\mathcal{S}$, we write $\mathcal{BCS}(\mathcal{S})$ for the minimal downward basis of $\mathcal{CS}(\mathcal{S})$.

In fact, the construction of Karp and Miller cited above allows us to answer other interesting behavioural properties. Two of them are presented below.

Definition 21 : Boundedness
A marked Petri net $\mathcal{S}$ is k-bounded iff for all $m \in \mathcal{RS}(\mathcal{S})$, $m \leq \vec{k}$ where $\vec{k}$ is the vector equals to k on all its components. It is bounded iff there exists $k \in \mathbb{N}$ such that $\mathcal{S}$ is k-bounded.

The simultaneous unboundedness asks a set of places to be unbounded together as follows:

Definition 22 : Simultaneous unboundedness [Demri, 2013]
Given $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post})$ and $\mathcal{S} = (\mathcal{N}, m_0)$, considering a subset $X \subseteq P$, $\mathcal{S}$ is simultaneous X-unbounded if for any $B \geq 0$, there is a run w such that $m_0 \xrightarrow{w} m$ and for $i \in X$, we have $m(i) \geq B$. If X is reduced to a singleton $\{p\}$, $\mathcal{S}$ is said p-unbounded.

Note that the simultaneous unboundedness has been studied in [Demri, 2013] on VASS and that paper provides its complexity recalled in the following theorem.

Theorem 7 : [Demri, 2013]
The simultaneous unboundedness problem for Petri nets is ExpSpace-complete.

We now recall some relations between those properties that will be used in this manuscript. We focus here on two simple translations between the previous properties. Nevertheless, note that there exist other more complex reductions that would not be useful for the understanding of this thesis.

Coverability can be easily reduced to *reachability* by adding a subnet that we call observer as depicted in Figure 3.2. Indeed, covering a marking m is equivalent to reach the marking m in the net augmented where for each place p a transition $t_{empty,p}$ such that: $\mathsf{Pre}(p, t_{empty,p}) = 1$ and 0 for all other components, $\mathsf{Post}(t_{empty,p})$ is equal to 0 on all components.

Notice that *coverability* can be easily reduced to *simultaneous unboundedness* by the use of an observer as depicted in Figure 3.3. The transition t_{obs} has an input condition equal to the marking we want to cover m. Its output effect provides a token in a place p_{obs}, that, once is marked, can become unbounded through the firing of t_{cumul}. With this construction, m is coverable in the net iff it is simultaneous p_{obs}-unbounded.

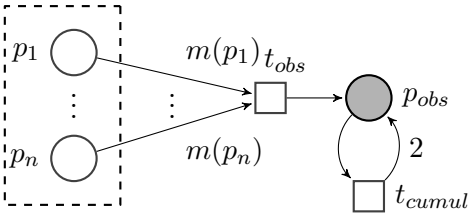


Figure 3.3 – Reduction of coverability to the place unboundedness

3.3 Analysis of Petri Nets

Thanks to their strong theoretical and formal background, mathematically grounded results can be derived from the systems modelled through this formalism. Standard techniques of verification have been established and improved in order to prevent error prototyping and time consuming ineffective trials. If some properties can be found directly through the structure of the net, we turn our interest toward a fundamental topic of verification, which is what to expect (or not to expect) from the system during operation. We will focus on the most generic method of analysis, which is a state space approach (or behavioural approach), since we latter plan to extend the results to a more generic model. Note that this approach relies on the enumeration of some reachable markings and can thus become very expensive due to state-space explosion.

The following sections addresses the fundamental results of coverability, boundedness and termination analysis through the search for self covering sequences during the executions.

3.3.1 Existence of Self-Covering Sequences

A *self-covering sequence* is a sequence of the form:

$$m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \dots \xrightarrow{t_i} m_i \xrightarrow{t_{i+1}} \dots \xrightarrow{t_j} m_j \text{ with } m_j \geq m_i$$

The termination problem asks whether all executions of the net are finite. The existence of a self-covering sequence is a necessary and sufficient condition of the non-termination of a net. The idea behind this statement is simple: once $m_j \geq m_i$ is reached, it is possible to fire the sequence $t_{i+1}, \dots, t_j$, leading to a marking $m'_j \geq m_j$. This can then be repeated infinitely, which is a witness run for the non-termination. Reciprocally, if there exists an infinite execution in the net, then applying the definition of wqo (see Definition 5), we can find two comparable markings in this sequence $m_i \leq m_j$ with $i < j$ which is exactly the definition of a self-covering sequence.

Let us consider the marked Petri net $(\mathcal{N}, m_0)$ depicted in Figure 3.4. We can for instance exhibit the following infinite run:

$$(1,0,0,0,0) \xrightarrow{t_1} (0,1,0,0,0) \xrightarrow{t_2} (0,0,1,0,0) \xrightarrow{t_3} (0,1,0,0,0) \xrightarrow{t_2} (0,0,1,0,0) \xrightarrow{t_3} (0,1,0,0,0) \dots$$

Moreover, if the self-covering sequence is strictly increasing, that is to say *iff* $m_j > m_i$, or more precisely, $\exists X \neq \emptyset \subseteq P,\ \forall p \in P, m_j(p) \geq m_i(p)$ *and* $\forall p \in X,\ m_j(p) > m_i(p)$, it becomes then a necessary and sufficient condition for the unboundedness of the net. Indeed, using strong monotonicity, it is possible to redo the sequence $t_{i+1}, \dots, t_j$ to obtain a marking strictly greater than m_j on the components of X. If the earlier example does not lead to an unbounded marking, the following one does:

$$(1,0,0,0,0) \xrightarrow{t_4} (0,0,0,1,0) \xrightarrow{t_5} (0,0,0,0,1) \xrightarrow{t_6} (0,0,0,2,0) \xrightarrow{t_5} (0,0,0,1,1) \xrightarrow{t_6} (0,0,0,3,0) \dots$$

After an arbitrary long repetition of firing of t_5, t_6 we would eventually reach any marking $(0,0,0,k,0)$ for $k \in \mathbb{N}$.

We will now see how Karp and Miller provided a behavioural analysis to systematically detect those self-covering sequences while performing an exploration of the reachable states of the net.

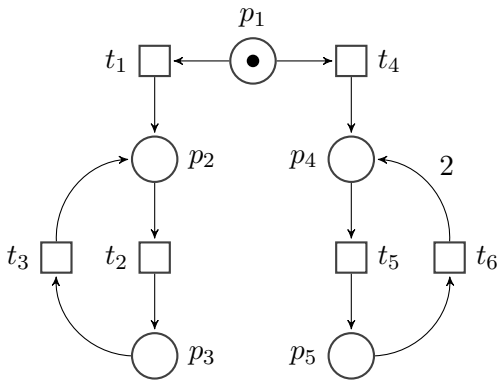


Figure 3.4 – Example of an unbounded marked Petri net

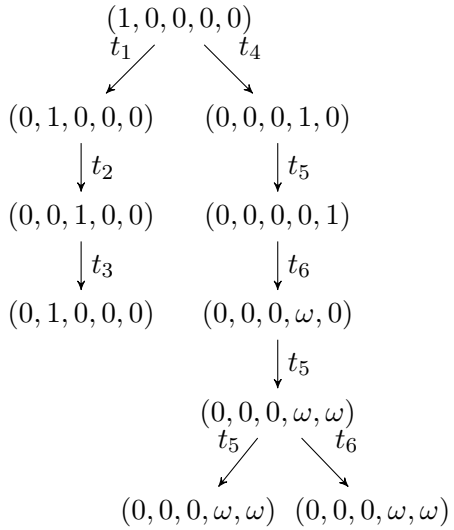


Figure 3.5 – The coverability tree of the marked Petri net from Figure 3.4

3.3.2 Karp and Miller Procedure

The Karp and Miller procedure constructs a finite representation of the potentially infinite coverability set of a Petri net through a tree where nodes are labelled by markings and arcs by transitions. This tree allows to answer the coverability and the (simultaneous) unboundedness problems for Petri nets. To make the procedure finish even if the Petri net studied presents infinitely many behaviours, an abstraction is used through an acceleration function. The idea behind acceleration relies on self-covering sequences. Indeed, as explained previously, due to monotonicity, a sequence $m_0 \to m_1 \to ... \to m_i \to ... \to m_j$, such that $m_i < m_j$, can be extended indefinitely by repeating the transitions involved between m_i and m_j. In order to express this repetition, Karp and Miller have suggested to replace the *strictly increasing coordinates* by ω meaning that we can store as many tokens as wanted in those places, by repeating the loop of transitions. The number ω is used to abstract the infinite set of markings coverable on the component(s) strictly growing with the loop. To ensure the termination, this procedure stops exploration once a node that has an ancestor with the same label is found. An example is provided through Figure 3.5.

Formally, the acceleration function Acc used to bypass loops is defined[2] as follows: $\mathsf{Acc}: 2^{\mathbb{N}_\omega^{|P|}} \times \mathbb{N}_\omega^{|P|} \to \mathbb{N}_\omega^{|P|}$ such that given a set of markings M (ancestors of m) and a marking

2. This definition is the one used in [Reynier and Servais, 2011], which is slightly different from the original acceleration of Karp and Miller since we accelerate directly w.r.t. the group of all predecessors rather than one by one. This permits to provide a more compact tree.

m, for each place p,

$$\mathsf{Acc}(M,m)(p) = \begin{cases} \omega & \text{if } \exists m' \in M \;\middle|\; \begin{array}{l} m' < m \\ \wedge \quad m'(p) < m(p) < \omega \end{array} \\ m(p) & \text{otherwise} \end{cases} \tag{3.2}$$

If a node n labelled by m has an ancestor n' labelled by m' such that $m' \leq m$, the function creates a marking m_ω such that for any place p, $m_\omega(p) = \omega$ if $m'(p) < m(p)$ and $m(p)$ otherwise.

Given a Petri net $\mathcal{S}$ we write $\mathcal{KM}(\mathcal{S})$ to denote its coverability tree obtained by the procedure *à la* Karp and Miller described in Algorithm 1 which corresponds to an improvement of the original method to build more compact structures (the acceleration beeing made directly regarding all the predecessors of the node, see [Reynier and Servais, 2011] for more details). By abuse of notation, we may assimilate the label of a node of the coverability tree with the node itself.

Algorithm 1 Coverability Algorithm

Require: A PN $\mathcal{S} = (P, T, \mathsf{Pre}, \mathsf{Post}, m_0)$
Return: A labelled tree $\mathscr{C} = (N, n_0, B, \Lambda)$
procedure
  Let n_0 be a new node such that $\Lambda(n_0) = m_0$
  $N \leftarrow n_0$; $Wait \leftarrow \{(n_0, t) | t \in T \wedge \Lambda(n_0) \geq \mathsf{Pre}(t)\}$; $B \leftarrow \emptyset$
  while $Wait \neq \emptyset$ **do**
    Pop(n',t) from *Wait*;
    $m \leftarrow Post(\Lambda(n'), t)$;
    if $\nexists y \in \mathsf{Ancestor}_{\mathscr{C}}(n')$ s.t. $\Lambda(y) = m$ **then**
      Let n be a new node s.t. $\Lambda(n) = \mathsf{Acc}(\Lambda(\mathsf{Ancestor}_{\mathscr{C}}(n')), m))$
      $N \leftarrow N \cup \{n\}$; $B \leftarrow B \cup \{(n', t, n)\}$;
      $Wait \leftarrow Wait \cup \{(n, u) | u \in T \wedge \Lambda(n) \geq Pre(u)\}$;
  return $\mathscr{C} = (N, n_0, B, \Lambda)$

Many properties can be checked using the coverability tree.

Theorem 8 : [Karp and Miller, 1969]
Given a marked Petri net $\mathcal{S}$ and a marking m, m is coverable in $\mathcal{S}$ iff there is a node in $\mathcal{KM}(\mathcal{S})$ labelled by a marking $m' \geq m$.

The following Lemma relates the notion of simultaneous unboundedness to the notion of coverability set.

Lemma 9 : [Karp and Miller, 1969, Demri, 2013]
Given a marked Petri Net $\mathcal{S}$, the following propositions are equivalent:

1. *$\mathcal{S}$ is simultaneous X unbounded*
2. *$\mathcal{KM}(\mathcal{S})$ contains a node labelled by a marking z such that $X \subseteq \omega(z)$*
3. *$\mathcal{BCS}(\mathcal{S})$ contains a marking z such that $X \subseteq \omega(z)$*

3.3.3 Introduction to the Rackoff Upperbound

We now focus on the establishment of the ExpSpace upper bound for coverability. We provide here the main intuition of a classic proof from [Rackoff, 1978] that will not be used directly in the sequel of this thesis but involves however interesting arguments. This has indeed become a common method of proof in the field of Petri nets. The main idea is to bound the length of witness runs among which the search should be performed.

The following explanation of the intuition of the proof originally presented in [Rackoff, 1978] is widely inspired by the one provided in [Bonnet et al., 2012] which was presented with a great clarity. We define R as the maximum absolute value of any weight (on an input or output arc) in the Petri net. Given a goal marking m_{cov}, we define R' as the maximum among the components of this marking and R. The main idea behind the Rackoff technique is to define a function $l : \mathbb{N} \to \mathbb{N}$ and prove that given a Petri net with n places, if m_{cov} can be covered, then it is coverable with a sequence of transitions of length at most $l(n)$ with $l(n)$ depending on R'. This is performed by induction on the number of places. In a Petri net with $i+1$ places, suppose $m_0 \xrightarrow{\sigma} m' \geq m_{cov}$ and m is the first marking obtained by a prefix of σ where one of the coordinates is more than $R \times l(i) + R' - 1$. If there is no such marking, then one can deduce that all intermediate markings coordinates are lower or equal to $R \times l(i) + R' - 1$. We can then bound the length of σ by $(R \times l(i) + R')^{i+1}$. Indeed, the $i+1$ places can take only $R \times l(i) + R'$ different values. Otherwise let $\sigma = \sigma_1\sigma_2$ such that $m_0 \xrightarrow{\sigma_1} m \xrightarrow{\sigma_2} m' \geq m_{cov}$. As explained, the length of σ_1 is bounded by $(R \times l(i) + R')^{i+1}$. If we then (temporarily) forget the place (or places) p such that $m(p) \geq R \times l(i) + R'$, we can use the induction hypothesis (on the net with strictly less than $i+1$ places) and conclude that starting from m, m_{cov} can be covered (in all places except p) with a sequence σ_2' of length at most $l(i)$. By definition of R', since $m \geq R \times l(i) + R'$ we can deduce that $\sigma_1\sigma_2'$ covers m_{cov} from m_0 and its length is at most $(R \times l(i) + R')^{i+1} + l(i) + 1$. So we can chose $l(i+1) = (R \times l(i) + R')^{i+1} + l(i) + 1$ and, solving the recurrence, it can be proved [3] that $l(i) \leq (6RR')^{(i+1)!}$.

From this reasoning we can construct a non deterministic procedure that will guess a run among those of size at most $l(n)$ that allows to cover m_{cov}. To this end, it is necessary to store a marking and a counter, indeed, we will compare m_{cov} to the intermediate stored marking and ensure that the run is of length lower than or equal to $l(n)$ through the counter. This can be done using at most $O(2^{cm\log(m)})$ bits for some constant c where m is an expression describing the size of the net involving the above constants that is to say, the initial marking m_{init}, the goal marking m_{cov} and the maximum absolute value of the weights involved in the net. Using the theorem from Savitch mentioned in Section 2.3.3, we obtain the ExpSpace upper bound. Note that this technique of proof is extended to the property of boundedness by Rackoff in his original paper [Rackoff, 1978] and to the notion of simultaneous unboundedness by Demri [Demri, 2013].

3.4 Simulations of Nets

It is classic to introduce mappings from a net to another where some special behaviours are easier to verify. We would like to be able to state that those mappings preserve some properties. More generally it can be interesting and time saving to notice that some nets, even if their structures are different, provide similar behaviours. This is formalised through the notion of simulation and bisimulation (see [Milner, 1980] and [Park, 1981] for their work on bisimulations). We consider marked Petri nets augmented by *silent actions* that abstract some behaviours of the Petri nets. Therefore, we introduce a labelling function Λ from the set of transitions T to Σ_ϵ, $\Lambda : T \to \Sigma_\epsilon$, such that $\Sigma_\epsilon = T \cup \{\epsilon\}$ and $\Lambda(t_i)$ equals either t_i or ϵ. We extend the previous notations by using $m \xrightarrow{t} m'$ or $m \xrightarrow{\Lambda(t)} m'$ depending on the context [4]. For instance, $m \xrightarrow{\epsilon^*} m'$ means that m leads to m' by using zero or more internal ϵ-transitions. In this context, the transitions of the transitions system induced by the semantics defined through Definition 13 are now labelled by the labels of transitions of the Petri net and not by the transitions themselves. Considering the previous alphabet Σ_ϵ and a labelled transitions system $(S, s_0, \to)$ with $\to \subseteq S \times \Sigma_\epsilon \times S$ and two states s and s' of S we write:

$$s \xrightarrow{\alpha}_\epsilon s' \Leftrightarrow \quad s \xrightarrow{\epsilon^*} \xrightarrow{\alpha} \xrightarrow{\epsilon^*} s' \quad \text{with } \alpha \in \Sigma \text{ (and thus } \alpha \neq \epsilon) \tag{3.3}$$

3. Note that in the context of [Bonnet et al., 2012], the constant R is defined for affine Petri nets. In the case of Petri nets, the formula still holds but the computation of R is easier.

4. In fact, using labelling is more general and allows to introduce non-deterministic behaviours.

Definition 23 : Weak-simulation
Given $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ *an alphabet and two labelled transitions systems,* $\mathcal{S}_1 = (Q, q_0, \rightarrow_1)$ *where* $\rightarrow_1 \subseteq Q \times \Sigma_\epsilon \times Q$ *and* $\mathcal{S}_2 = (S, s_0, \rightarrow_2)$ *where* $\rightarrow_2 \subseteq S \times \Sigma_\epsilon \times S$*, a binary relation* $\mathcal{R} \subseteq Q \times S$ *is a weak-simulation if*

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

If we can find a weak-simulation $\mathcal{R} \subseteq Q \times S$ such that $(q_0, s_0) \in \mathcal{R}$ 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 $\mathcal{R}' \subseteq Q \times S$ such that $\mathcal{S}_1$ weakly simulates $\mathcal{S}_2$, we say that $\mathcal{S}_1$ and $\mathcal{S}_2$ are *weakly co-similar*. In addition, when those two relations are the inverse of one another, we call it a bisimulation:

Definition 24 : Weak-bisimulation
Given $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ *an alphabet and two labelled transitions systems,* $\mathcal{S}_1 = (Q, q_0, \rightarrow_1)$ *where* $\rightarrow_1 \subseteq Q \times \Sigma_\epsilon \times Q$ *and* $\mathcal{S}_2 = (S, s_0, \rightarrow_2)$ *where* $\rightarrow_2 \subseteq S \times \Sigma_\epsilon \times S$*, a binary relation* $\mathcal{R} \subseteq Q \times S$ *is a weak-bisimulation*[a] *if*
$\mathcal{R}$ *is a weak-simulation and* $\mathcal{R}^{-1}$ *is a weak-simulation too where for every couple* x, y*,* $x\mathcal{R}y \Leftrightarrow y\mathcal{R}^{-1}x$*.*

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

Two labelled transitions systems $\mathcal{S}_1$ and $\mathcal{S}_2$ are *weakly bisimilar* if there is a weak bisimulation relating their initial states.

Given two labelled marked Petri nets $\mathcal{N}_1$ and $\mathcal{N}_2$, and their semantics $\mathcal{S}_1$ and $\mathcal{S}_2$, we say that:

- $\mathcal{N}_2$ weakly simulates $\mathcal{N}_1$ if $\mathcal{S}_2$ weakly simulates $\mathcal{S}_1$.
- $\mathcal{N}_1$ and $\mathcal{N}_2$ are weakly co-similar if $\mathcal{S}_1$ and $\mathcal{S}_2$ are *weakly co-similar*.
- $\mathcal{N}_1$ and $\mathcal{N}_2$ are weakly bisimilar if $\mathcal{S}_1$ and $\mathcal{S}_2$ are *weakly bisimilar*.

3.5 Notable Petri Nets Extensions

It is possible to increase the modelling power and the expressiveness of Petri nets through several extensions. Some of them, such as Coloured Petri nets [Jensen, 1987] are equivalent[5] to Petri nets and focus on improving the modelling capability whereas some others are more expressive. Indeed, some formalisms offer to use arcs with specific behaviours such as reset arcs [Araki and Kasami, 1976] or inhibitor arcs [Agerwala and Flynn, 1973, Hack, 1976], to add time to the model [Ramchandani, 1973, Merlin, 1974, Berthomieu and Diaz, 1991] on transitions, on places or on arcs (see [Boyer and Roux, 2008] for a comparison of those formalisms) or to consider real numbers markings [David and Alla, 1987].

We will more specifically turn our interest towards ω-Petri nets introduced in [Geeraerts et al., 2015] which present some similarity with the model that will be introduced in Chapter 4. In ω-Petri nets, input and output arcs can be weighted with ω:

Definition 25 : ω-Petri net [Geeraerts et al., 2015]
An ω*-Petri net* (ωPN) *is a tuple* $(P, T, \mathsf{Pre}, \mathsf{Post})$ *where* P *and* T *are respectively a finite set of places and transitions.* Pre *(resp.* Post*) is a function of* $P \times T$ *to* $\mathbb{N}_\omega$ *that gives the input (resp. output) effect of a transition* t *on a place* p*.*

5. To be more precise, Coloured Petri nets with finite colour's domain have the same expressiveness as Petri nets.

The main difference with classic Petri nets is that ω-transitions introduce additional non-determinism in the semantics:

Definition 26 : ωPN semantics [Geeraerts et al., 2015]
Given a marking m, and a transition t such that $m \geq Pre(t)$, firing t from m gives a new marking m' s.t. $\forall p \in P$, $m'(p) = m(p) - i + o$ where $i = \mathsf{Pre}(p,t)$ if $\mathsf{Pre}(p,t) \neq \omega$ and $i \in \{0, ..., m(p)\}$ if $\mathsf{Pre}(p,t) = \omega$, $o = \mathsf{Post}(p,t)$ if $\mathsf{Post}(p,t) \in \mathbb{N}$ and $o \geq 0$ if $\mathsf{Post}(p,t) = \omega$. We denote this by $m \xrightarrow{t} m'$.

With this semantics, $\mathsf{Post}(p,t) = \omega$ means that an arbitrary number of tokens can be generated in p. We illustrate those definitions with the example of Figure 3.6 which is taken from [Geeraerts et al., 2015]. It illustrates the skeleton of a distributed program where a main function launches (through t_1) an arbitrary parallel number of threads, each one executing a task (depicted by t_2).

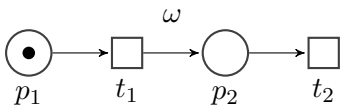


Figure 3.6 – An example of ω Petri nets

Nevertheless, this model can be directly related to classic Petri nets. Indeed, ω-output arcs can be simulated by tokens canons [6] whereas ω-input arcs can be simulated by transitions allowing to empty input places. Given an ω-Petri nets with a set of places P, it is possible to construct a classic Petri net with a set of places $P' = P \cup Q$ where Q contains a global lock lock_g and one lock lock_t per ω-transition t. Then, given a marking m from the initial ω-Petri net, it is possible to define a function f such that the $f(m)$ is a marking of the associated Petri net and such that, for all places of P, $f(m)(p) = m(p)$, $f(\mathsf{lock}_g) = 1$ and every $f(\mathsf{lock}_t) = 0$.

Lemma 10 : ωPN to PN [Geeraerts et al., 2015]
Let $\mathcal{N}$ be an ωPN, there exists a PN $\mathcal{N}'$ with the same set of places plus some temporary places such that for any marking m_0, $\mathcal{RS}(\mathcal{N}, m_0) = \mathcal{RS}(\mathcal{N}', f(m_0))$.

We have presented this simple result which will be useful in the sequel. However, for other properties such as termination, the reduction is not stragihtforward. It is important to notice that this translation from ω-Petri nets to Petri nets can be carried out in polynomial time.

The main concern of this first part was to review some essential mathematical notions and notations that this thesis will invoke. We also provided some background on Petri nets and some classic properties and verification techniques. On the basis of what has thus been presented, the next part introduces the formalism of parametric Petri nets and the related contributions of this thesis.

6. See construction from Figure 7.4 on page 109.

Part II
Contributions

CHAPTER 4

Parametric Petri Nets

"An idea is always a generalisation, and generalisation is a property of thinking. To generalise means to think."

— Georg Wilhelm Friedrich Hegel

Contents

This chapter intends to extend Petri nets with discrete parameters on the weights of arcs or directly on the initial marking to describe families of concurrent systems and thus provide more realistic classes of models. We formally introduce this formalism of parametric Petri nets (PPNs for short) and provide a semantics in order to explain how classic problems studied in Petri nets are impacted. We then show how this extension increases the modelling power of Petri nets and how it allows to simulate a *zero test*, making the parametric instances of the coverability problem undecidable.

4.1 Definition of Parametric Petri Nets

We would like to use less rigid modelling for systems where some data is not known *a priori*. Therefore, in this subsection, we extend the previous definitions by adding a set of parameters $\mathbb{P}$. Working with Petri nets and discrete parameters leads to consider two main situations: the first one involves parameters on initial markings, by replacing the initial 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 initial markings can easily be understood as modelling an unfixed amount of resources that one may want to optimise.

Figure 4.1 presents the modelling of a semaphore with a parametric initial value a. 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 b of resources, as depicted in Figure 4.2, and at the end, we need to pack end products in boxes of capacity c, as in Figure 4.3. This is part of a whole packaging process that one may want to optimise.

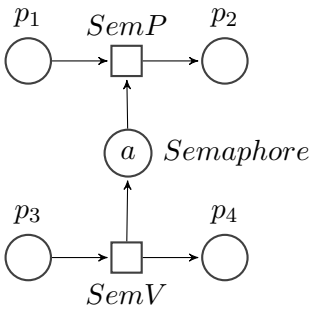


Figure 4.1 – A Petri net with a parametric initial marking modelling a semaphore with a parametric initial value

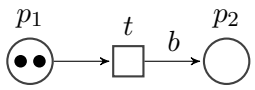


Figure 4.2 – Unpacking raw material

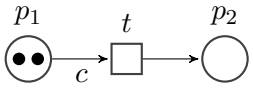
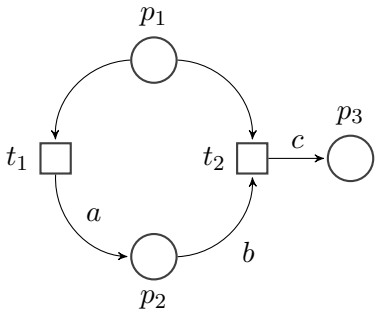


Figure 4.3 – Packing end products

The level of abstraction induced by parameters permits to leave those values unspecified in order to perform an early analysis.

Definition 27 : Parametric Petri net
A parametric Petri net, $\mathcal{N}$ is a 5-tuple $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$ such that P is a finite set of places of $\mathcal{N}$, T is a finite set of transitions of $\mathcal{N}$, $\mathbb{P}$ is a finite set of parameters of $\mathcal{N}$, Pre and $\mathsf{Post} \in (\mathbb{N} \cup \mathbb{P})^{P \times T}$

We provide a more complex example of parametric Petri net through Figure 4.4 where the matrix representations of Pre and Post are described.



$$\mathsf{Pre} = \begin{pmatrix} 1 & 1 \\ 0 & b \\ 0 & 0 \end{pmatrix} \quad \mathsf{Post} = \begin{pmatrix} 0 & 0 \\ a & 0 \\ 0 & c \end{pmatrix}$$

Figure 4.4 – A parametric Petri net

Modelling with parameters means using parameters over weights and markings rather than setting numeric values everywhere. Following the definitions of Petri nets, we now need to provide an initial marking to parametric Petri nets in order to make them executable.

Definition 28 : Marked parametric Petri net
A marked parametric Petri Net (PPN) is a pair $\mathcal{S} = (\mathcal{N}, \mu_0)$ *where* $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$ *is a parametric Petri net and* μ_0 *is the initial parametric marking*[a] *of* $\mathcal{N}$ *belonging to* $(\mathbb{N} \cup \mathbb{P})^P$.

a. Note that the components of the initial marking of the marked Parametric Petri net are not linear expression on $\mathbb{P}$ but either a parameter or a natural number. This extension will be discussed as future work in Chapter 9.

Considering a fix ordering on places, parametric initial markings can be represented as vectors of $(\mathbb{N} \cup \mathbb{P})^{|P|}$. Similarly, Pre and Post can be seen as matrices of $(\mathbb{N} \cup \mathbb{P})^{|P| \times |T|}$.

We define the *parametric transitions* of $\mathcal{S}$, $\Theta \subseteq T$ as the set of transitions with at least one parameter on an input or output arc: $\Theta = \{t \in T \mid \exists p \in P \ s.t. \ \mathsf{Pre}(p,t) \in \mathbb{P} \vee \mathsf{Post}(p,t) \in \mathbb{P}\}$. We refer to $T \setminus \Theta$ as the set of plain transitions in echo to the notations of [Geeraerts et al., 2015].

PPNs can be used to design systems where some parts have not been dimensioned 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.

Given a marked PPN $\mathcal{S} = (\mathcal{N}, \mu_0)$, where $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$, for any $\mathbb{N}$-valuation v on a subset X of $\mathbb{P}$, we define the v-instance of $\mathcal{S}$ as the marked PPN $v(\mathcal{S}) = (v(\mathcal{N}), v(\mu_0))$ where $v(\mathcal{N}) = (P, T, v(\mathsf{Pre}), v(\mathsf{Post}), \mathbb{P} \backslash X)$. By $v(\mathsf{Pre})$ and $v(\mathsf{Post})$ we denote the functions/matrices obtained by replacing in their entries each parameter a in $\mathtt{dom}(v)$ by $v(a)$. If $X = \mathbb{P}$, $v(\mathcal{N})$ and $v(\mathcal{S})$ are respectively a Petri net and a marked Petri net. Formally:

Definition 29 : Instantiation of parametric Petri nets
Let $\mathcal{S} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P}, \mu_0)$ *be a PPN. Let* $v \in \mathbb{N}^{\mathbb{P}}$, *we define* $v(\mathcal{S})$ *as the PN obtained from* $\mathcal{S}$ *by replacing each parameter* $a \in \mathbb{P}$ *by* $v(a)$, *its valuation by* v, *i.e.* $v(\mathcal{S}) = (P, T, \mathsf{Pre}', \mathsf{Post}', m_0)$ *where* $\forall i \in P, \forall j \in T$,

$$\mathsf{Pre}'(i,j) = \begin{cases} \mathsf{Pre}(i,j) & \text{if } \mathsf{Pre}(i,j) \in \mathbb{N} \cup \mathbb{P} \setminus \mathtt{dom}(v) \\ v(\mathsf{Pre}(i,j)) & \text{if } \mathsf{Pre}(i,j) \in \mathbb{P} \cap \mathtt{dom}(v) \end{cases}$$

$$\mathsf{Post}'(i,j) = \begin{cases} \mathsf{Post}(i,j) & \text{if } \mathsf{Post}(i,j) \in \mathbb{N} \cup \mathbb{P} \setminus \mathtt{dom}(v) \\ v(\mathsf{Post}(i,j)) & \text{if } \mathsf{Post}(i,j) \in \mathbb{P} \cap \mathtt{dom}(v) \end{cases}$$

$$m_0(i) = \begin{cases} \mu_0(i) & \text{if } \mu_0(i) \in \mathbb{N} \cup \mathbb{P} \setminus \mathtt{dom}(v) \\ v(\mu_0(i)) & \text{otherwise} \end{cases}$$

Given a total valuation v on $\mathbb{P}$, we define the relation $\rightarrow_v$ as the firing of a transition in the classic Petri Net $v(\mathcal{S})$. This relation also holds for sequence of transitions. If $m \xrightarrow{t}_v m'$, we may also write $m' = \mathsf{Post}_v(m,t)$.

4.2 Parametric Decision 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 a valuation 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 problems $\mathcal{P}$ (coverability, reachability, boundedness...), $\mathcal{S}$ a PPN and ϕ an instance of $\mathcal{P}$, parameterised problems are written as follows:

Definition 30 : $\mathcal{P}$-Existence problem
($\mathscr{E}$-$\mathcal{P}$): Is there a valuation $v \in \mathbb{N}^{\mathbb{P}}$ s.t. $v(\mathcal{S})$ satisfies the property ϕ ?

Definition 31 : $\mathcal{P}$-Universality problem
($\mathscr{U}$-$\mathcal{P}$): Does $v(\mathcal{S})$ satisfy the property ϕ for each $v \in \mathbb{N}^{\mathbb{P}}$?

In the context of parametric Petri nets, coverability leads to two main problems: the *existence problem*, written ($\mathscr{E}$-cov) and the *universal problem*, written ($\mathscr{U}$-cov). For instance, ($\mathscr{U}$-cov) is:

Given a marking $m \in \mathbb{N}^P$, m is $\mathscr{U}$-coverable [1] in $\mathcal{S} \Leftrightarrow \forall v \in \mathbb{N}^{\mathbb{P}}, \exists m' \in \mathcal{RS}(v(\mathcal{S}))$ s.t. $m' \geq m$

For example, considering the parametric Petri net of Figure 4.4 we order the places by their indices and the parameters lexicographically to use a vector representation. Let us define an initial marking as $\mu_0 = (d, 0, 0)$ and a goal marking as $m_{goal} = (0, 0, 1)$. The marking m_{goal} is existentially coverable, since valuation $v = (1, 1, 1, 2)$ is a solution. Nevertheless m_{goal} is not universally coverable, for instance valuations $(1, 1, 1, 1)$ or $(2, 3, 1, 2)$ lead to a deadlock where p_3 is not marked. In parallel, if we consider the parametric Petri net obtained by valuating the parameters b, c and d respectively to 0, 1 and 1, we obtained a parametric Petri net where the set of parameters is reduced to the singleton $\{a\}$ and in which m_{goal} is universally coverable. This latest consideration will be reused in Section 6.2 while working on parameter synthesis.

Remark 4.2.1. From any PN $\mathcal{S}$, we can build a PPN $\mathcal{S}'$ by adding an unused parameter. Then checking existential or universal coverability on $\mathcal{S}'$ is equivalent to checking coverability on $\mathcal{S}$. For instance, coverability is known to be ExpSpace-complete, and especially ExpSpace-hard. Since coverability can be reduced to a parametric coverability problem, the universal and existential problems are ExpSpace-hard too. The same reasoning applies for other properties such as (simultaneous) unboundedness.

4.3 Synthesis of Parameters

Beyond verification of properties, the use 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. This is the synthesis problem: given a property, compute the exact set of all parameter values such that, instantiated with these values, the system satisfies this property. Computing such a set of values can help to design the system. Moreover, it permits to quantify how reliable is this system. Such quantity is often called the *robustness* of the system. Indeed, in full knowledge of "good values" for the parameters, we may be able to quantify the distance between the instance considered and the closest "bad value".

Given a class of problems $\mathcal{P}$ (coverability, reachability, boundedness, ...), $\mathcal{S}$ a PPN and ϕ an instance of $\mathcal{P}$, the synthesis problem is written as follows:

Definition 32 : $\mathcal{P}$-Synthesis problem
($\mathscr{S}$-$\mathcal{P}$): Compute all the valuations v, such that $v(\mathcal{S})$ satisfies ϕ.

We call this set of valuations the *$\mathcal{P}$ synthesis set of a marked PPN $\mathcal{S}$ given the instance ϕ of the problem $\mathcal{P}$*. For instance, we will latter address the coverability synthesis set of a marked PPN $\mathcal{S}$ for a marking m that will be denoted by $\mathcal{CV}(\mathcal{S}, m)$. We also call it the *solution set* to the synthesis problem. Note that if the solution set to the synthesis problem can be computed in an explicit enough form, then answering the existential and universal problems are equivalent to deciding whether this set is respectively empty or the set $\mathbb{N}^{\mathbb{P}}$ itself (or a well defined subset of this set if the domain of the parameters was restricted).

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

4.4 Illustrative Examples

We now introduce some examples which highlight and motivate the use of parametric models and parametric verification and give the reader a better understanding of the theory. Two examples are provided which correspond to simple real-life examples.

4.4.1 Financial Loan

The first example concerns a financial loan and is inspired by the description of an offer for a personal loan on the website of the mainstream french bank BNP Paribas: *"a solution for financing different projects (renovations, car, leisure, furniture, computer equipment, travel, etc.) - the amount loaned, length of reimbursement, rate applied and monthly payments are defined when the contract is signed."*

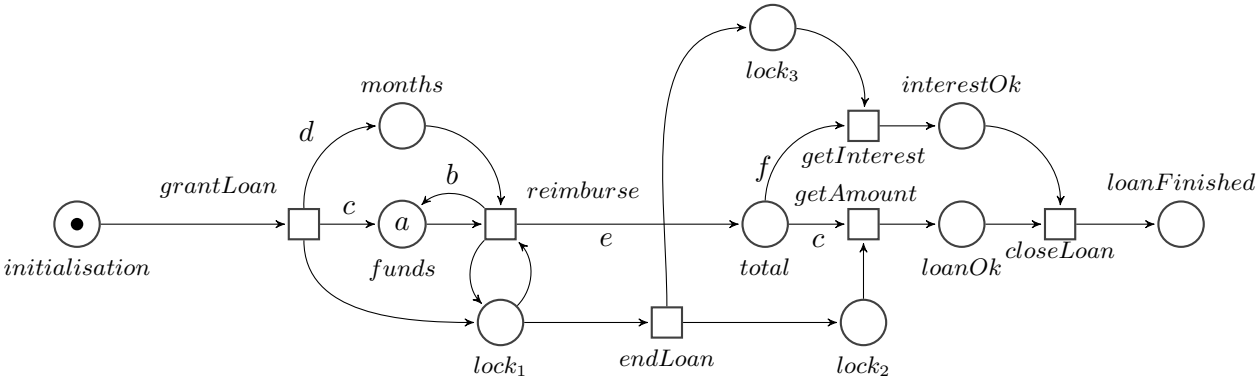


Figure 4.5 – Modelling a financial loan with parametric Petri nets

In the general case, a client has a certain amount of money, say a and is ensured to get a income b every month. To finance his project, the client needs to define with the bank the amount loaned c, the length of reimbursement d (in month) and the amount of each reimbursement e. At the end of the process, the bank expects to get back the initial amount loaned c plus the interest f. This interest is in fact directly related to the rate applied to the loan. Those variables are of course ruled by a simple constraint system that is summed up below:

$$\begin{cases} d \times e & \geq & c + f \\ a + c + d \times (b - e) & \geq & 0 \end{cases}$$

The PPN depicted by Figure 4.5 represents the process of this *personal loan*. The amount of money in possession of the client is depicted in place $funds$. The signature of the contract is symbolised through the firing of transition $grantLoan$ which imposes to the bank to loan an amount c for a term of d months. Each month, we can fire $reimburse$: the client receives its own income e that is added to its capital in $funds$ and in parallel an amount b is reimbursed to the bank. When we consider that the reimbursement is finished, we fire $endLoan$ that remove the token from $lock_1$ and allows us to enable the transitions of the second part of this example. We can then check if the bank can get its money back by testing if $loanOk$ can be marked and if the bank can get the interest by checking $interestOk$ can be marked or both by checking if $loanFinished$ can be marked.

If a deeper analysis were to be carried, the designer may fix some of the parameters. For instance, one can fix a, b and c according to the profile of the client. The only variables remaining are thus d, e and f. We could of course imagine providing a graphic modelling of more complex cases of computational finance.

4.4.2 Production Line

The second example concerns a production line composed of several cells in an assembly line fashion. Our company hesitates between different manufacturing lines to build similar finite products. The different manufacturing lines are depicted by the nets in Figure 4.6.

We first consider only the arcs depicted by solid lines. We thus imagine that all the manufacturing lines are available and want to analyse what is the best way to produce 50 finite products. Our production line is now depicted by the overlapping of all the nets from Figure 4.6. The first transition symbolises a delivery of raw material. We want to know the set of all valuations allowing us to generate 50 tokens in p_{goal}. If we associate some cost to each raw material, we will then be able to compute the minimal cost to produce our finite product and, by studying the trace, see what manufacturing lines are effectively used.

In fact this problem can be encoded and solved using a constraint system. Let us suppose that a costs 100, b costs 200, c costs 175, and d costs 125 per unit which defines a vector of costs C. We order the places and transitions by their indices and provide a matrix representation of this net in addition with some variables in Figure 4.7. At this point, consider only 0 in the Post matrix when it is written $0|1$. It is then possible to write the linear programming problem associated to this production line:

$$\begin{aligned} \textbf{minimise } & C^T X \\ \textbf{subject to } & m_0 + (\mathsf{Post} - \mathsf{Pre})K \geq m_{goal} \\ & K \in \mathbb{N}^{11} \\ \textbf{and } & X \in \mathbb{N}^4 \end{aligned}$$

Nevertheless, parametric Petri nets provide here a more graphical and readable representation of this study. One can easily isolate some intermediate events, replace a machine and see the effect on the production supply, which would be more abstract using only constraints.

Now, let us suppose that the production line is being modernised by adding a process to recycle waste and thus create some kind of retroaction in the production process. This can be easily depicted by introducing some events leading to produce tokens in the places from p_2 to p_5. This is depicted by the dashed arcs on Figure 4.6. The matrix representation can be adapted as well by now considering 1 each time a choice of the format $0|1$ is provided for the Post matrix in Figure 4.7. The Petri nets approach remains thus readable. Nevertheless, there is no direct trivial encoding in terms of constraint systems. Indeed, with the previous constraint system, the condition $m_0 + (\mathsf{Post} - \mathsf{Pre})K \geq m_{goal}$ is in fact related to the *state equation* of Petri nets. Given σ a sequence of transitions firable from m_0 and a vector K where the i^{th} component is equal to the number of occurrences of the transition t_i in σ, the marking obtained is exactly equal to $m_0 + (\mathsf{Post} - \mathsf{Pre})K$. However, given any sequence of transitions σ', the state equation does not check whether there is actually a sequence of intermediate markings such that σ' is firable. This is a well-known limitation of the state equation which in fact only provides a necessary condition (but not sufficient) for the problem of the reachability. More concretely, when we introduce the dashed arcs in the model, we create some loops which would enable new solutions that are not correct in terms of execution. Therefore, this constraint system would not answer our problem.

4.5 Undecidability of the General Case

Studying decidability for reset-nets and corresponding subclasses [Dufourd et al., 1998] shows that Petri nets are close to the frontiers of decidability: the reachability problem becomes undecidable when we increase the power of Petri nets. For instance, adding *reset arcs* makes reachability undecidable.

In this section, we focus on showing that adding parameters to Petri nets also leads to undecidability. More specifically, ($\mathscr{U}$-cov) and ($\mathscr{E}$-cov) are undecidable on PPNs.

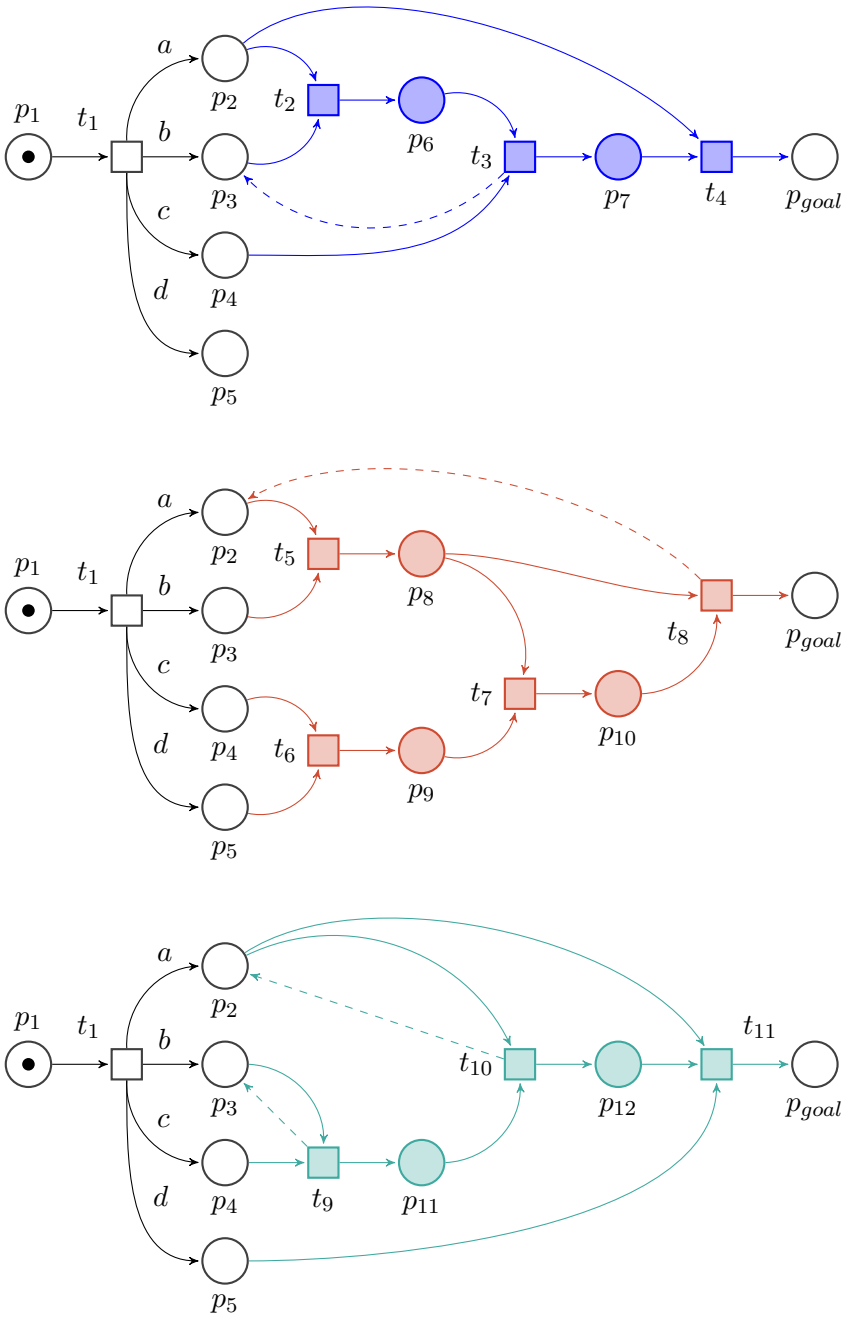


Figure 4.6 – Modelling a production line with parametric Petri nets

$$\mathsf{Pre} = \begin{array}{c} \begin{array}{ccccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} \end{array} \\ \left[\begin{array}{ccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{c} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \\ p_{11} \\ p_{12} \\ p_{goal} \end{array}$$

$$\mathsf{Post} = \begin{array}{c} \begin{array}{ccccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} \end{array} \\ \left[\begin{array}{ccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 & 0 & 0 & 0 & 0|1 & 0 & 0|1 & 0 \\ b & 0 & 0|1 & 0 & 0 & 0 & 0 & 0 & 0|1 & 0 & 0 \\ c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \end{array} \begin{array}{c} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \\ p_{11} \\ p_{12} \\ p_{goal} \end{array}$$

$$m_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad m_{goal} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 50 \end{pmatrix} \quad X = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \quad C = \begin{pmatrix} 100 \\ 200 \\ 175 \\ 125 \end{pmatrix}$$

Figure 4.7 – The constraint system corresponding to the production line

Since we will proceed by reduction from the halting problem and the counter boundedness problem for counter machines to answer our problem, we first recall some definitions. A 2-counter machine has two integer counters and a finite sequential program made of three types of instructions: increment a counter, decrement a counter, and branch if some counter is 0. 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 in p_i: increase c_k by one and go to p_j, where $p_i \in P$, $p_j \in P \cup \{halt\}$ and $k \in \{1, 2\}$
- decrement in p_i: decrease c_k by one and go to p_j, where $p_i \in P$, $p_j \in P \cup \{halt\}$ and $k \in \{1, 2\}$
- zero-test in p_i: if $c_k = 0$ go to state p_j else go to state p_l, where $p_i \in P$, $p_j, p_l \in P \cup \{halt\}$ 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 in [Minsky, 1967].

Theorem 11 : Undecidability of $\mathscr{E}$-cov on PPN
The $\mathscr{E}$-coverability problem for PPN is undecidable[a].

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

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

- Each counter c_i is modelled 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 modelled by the transitions and arcs depicted in Figure 4.8.
- 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 $\neg C_1$ and $\neg C_2$. The value of this parameter will therefore represent the upper bound of the counter over the instructions sequence. By construction, at all time, $m(C_i) + m(\neg C_i) = a$ with $i \in \{1, 2\}$. First we show that $\mathcal{S}_\mathcal{M}$ simulates $\mathcal{M}$ by verifying the behaviour of each instruction:

- *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* state is marked iff the incrementation instruction is performed whereas the counter is already equal to a. 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.
- *Zero test:* As C_i models the counter, and as we know the sum of tokens in C_i and $\neg C_i$ is a, there is no token in C_i iff there are a 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.

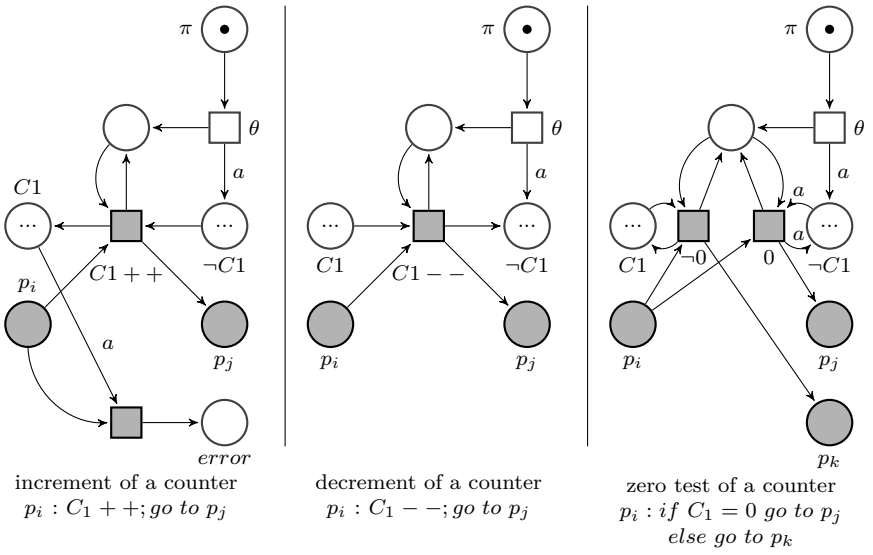


Figure 4.8 – Modelling a counter with PPN

We will show that given a 2-counter machine $\mathcal{M}$, *(a)* $\mathcal{M}$ halts (it reaches the *halt* state) iff *(b)* there exists a valuation v such that $v(\mathcal{S}_\mathcal{M})$ covers the corresponding p_{halt} place.

- $(a) \Rightarrow (b)$ First, let us assume that $\mathcal{M}$ halts. Since $\mathcal{M}$ halts, the execution of the machine is finite. On this execution the two counters are bounded respectively by c_{lim1} and c_{lim2}. Let c_{lim} be the maximum of those two values. Let v be the valuation such that $v(a) = c_{lim}$. By the previous explanation, $\mathcal{S}_\mathcal{M}$ simulates $\mathcal{M}$. Moreover, the valuation v ensures that $\mathcal{S}_\mathcal{M}$ does not reach a deadlock state where *error* is marked. Therefore, when $\mathcal{M}$ reaches *halt*, $\mathcal{S}_\mathcal{M}$ 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 v such that $v(\mathcal{S}_\mathcal{M})$ 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 a is less than the maximum value of the counters and error will be reachable during some increment instruction resulting in a deadlock, or the value of a 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 v, we will reach an instruction "*increment* c_i", where i is 1 or 2, and $c_i = v(a)$. Therefore, a token will be added in *error* leading to a deadlock. So $\mathcal{S}_\mathcal{M}$ will not cover any marking with one token in p_{halt}.

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

Theorem 12 : Undecidability of $\mathscr{U}$-cov on PPN
The $\mathscr{U}$-coverability problem for PPN is undecidable.

Proof. We proceed by reduction from *the counter boundedness problem in 2-counter machines.* We use the same construction as in the previous proof. We denote by m_{error} the marking where

$m_{error}(p) = 0$ for each $p \in P$ except $m_{error}(error) = 1$. We will show that given a 2-counter machine $\mathcal{M}$, *(a)* the counters are unbounded along the instruction sequence of $\mathcal{M}$ (counters boundedness problem) iff *(b)* for each valuation v, $v(\mathcal{S}_\mathcal{M})$ covers 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 eventually be added in $error$.
- $(b) \Rightarrow (a)$ Reciprocally, by contrapositive, we want to show that if the counters are bounded, there exists a valuation v such that $v(\mathcal{S}_\mathcal{M})$ does not cover m_{error}. This comes directly from the previous proof. Since the counters are bounded along the instructions sequence, we consider a valuation v such that $v(a) = c_{lim}$ where c_{lim} is an upper bound of the values of the counters. By construction, there is no possibility to add a token in $error$, otherwise, it means that $\mathcal{S}_\mathcal{M}$ took an incrementation transition meaning that c_{lim} is not an upper bound.

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

We can directly infer that universal reachability and existential reachability are undecidable. Moreover, the synthesis coverability (and thus the synthesis reachability) cannot be solved.

We provided an extension of the Petri nets using parameters and derive two parametric decision problems from the classic ones. However, this paradigm allows us to model zero test making those problems undecidable. In the next chapter, we introduce syntactical subclasses and study the decidability of parametric coverability with the aim of making parameters usable in practice.

CHAPTER 5

Decidable Subclasses of PPNs for Parametric Coverability

"Start by doing what is necessary; then do what is possible; and suddenly you are doing the impossible."
— Francis of Assisi

Contents

This chapter introduces syntactical subclasses of parametric Petri nets depending on whether the parameters are used for initial markings, input or output arcs of transitions. We prove that this restricted setting provides some strong properties on the semantics of those nets. We then proceed with an exhaustive study of the parametric coverability for those subclasses including decidability results and complexities.

5.1 Introduction of Subclasses

On the one hand, our parametric model increases the modelling power of Petri nets but on the other hand, using parameters leads to complex models where properties of interest 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 modelling. We will therefore introduce some subclasses of PPNs 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 synchronisations among arbitrary numbers of identical process, and finally some combinations of those.

A P-parametric marked Petri net (P-PPN for short) is a classic Petri net where the places are initially marked by a natural number or a parameter.

Definition 33 : P-parametric marked Petri net
A P-parametric marked Petri net (P-PPN), $\mathcal{S} = (\mathcal{N}, \mu_0)$ *where* $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$ *is a Parametric Petri net such that* Pre *and* $\mathsf{Post} \in \mathbb{N}^{P \times T}$ *and* μ_0 *is a parametric initial marking of* $(\mathbb{N} \cup \mathbb{P})^P$.

A T-parametric marked Petri net (T-PPN for short) is a parametric marked Petri net where the initial marking is not parameterised.

Definition 34 : T-parametric marked Petri net
A T-parametric Petri net (T-PPN), is a PPN $\mathcal{S} = (\mathcal{N}, m_0)$ *where* $\mathcal{N}$ *is a parametric Petri net and* m_0 *is a marking of* $\mathbb{N}^P$

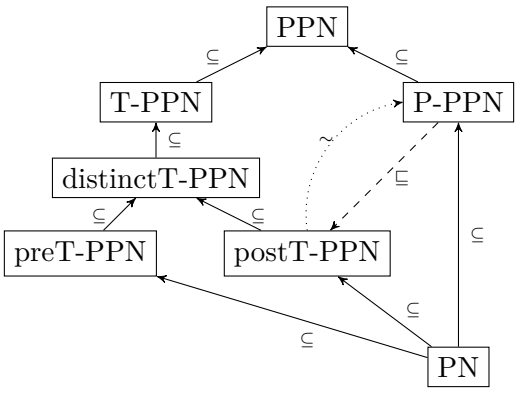


Figure 5.1 – $\xrightarrow{\subseteq}$: is a syntactical subclass of
$\dashrightarrow^{\sqsubseteq}$: is a weak-bisimulation subclass of
$\cdots\!\!>^{\sim}$: is a weak-cosimulation subclass of

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 disjoint subsets of parameters *i.e.* $\mathsf{par}(\mathsf{Pre}) \cap \mathsf{par}(\mathsf{Post}) = \emptyset$ where par is the application that maps to the set of parameters involved in a matrix (or a vector). We call this subclass *distinctT-parametric Petri net*[1] (or distinctT-PPN for short). We can even refine the subclass of distinctT-PPN by considering the two distinct classes of *Pre-T-parametric Petri net* (preT-PPN), where $\mathsf{Post} \in \mathbb{N}^{|P|}$ (or $\mathsf{Post} \in P \times T \to \mathbb{N}$ if Post is seen as a function) and *Post-T-parametric Petri net* (postT-PPN), where $\mathsf{Pre} \in \mathbb{N}^{|P|}$ (or $\mathsf{Pre} \in P \times T \to \mathbb{N}$ if Pre is seen as a function). Those subclasses are depicted in Figure 5.1 where the solid lines represent the syntactical inclusion of classes. Note that the meaning of the dashed arrow and the dotted arrow is introduced and justified in the following section (see Section 5.2).

1. Studying the former undecidability proof, it is relevant to think that using different parameters for the input and the output would reduce the modelling power.

5.2 Links between P-PPNs and PostT-PPNs

Intuitively, using parameters on output arcs or directly in the initial marking seems related. Parametric arcs should indeed lead to parameterised markings. In this section, we elaborate on the relationship between the classes of P-PPNs and postT-PPNs.

5.2.1 Translating P-PPNs to postT-PPNs

Given a P-PPN $\mathcal{S}$, we construct a postT-PPN $\mathcal{S}'$ as follows: in order to simulate the behaviour of parameterised places, we translate those places in a parameterised initialisation process that has 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 $\mathsf{Post}(\theta)$ initialises a P-PPN, as showed in Figure 5.2. 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 $\mathcal{S}'$ and $\mathcal{S}$ are weakly-bisimilar by showing that each behaviour of $\mathcal{S}$ can be done in $\mathcal{S}'$ if we begin by firing θ and reciprocally.

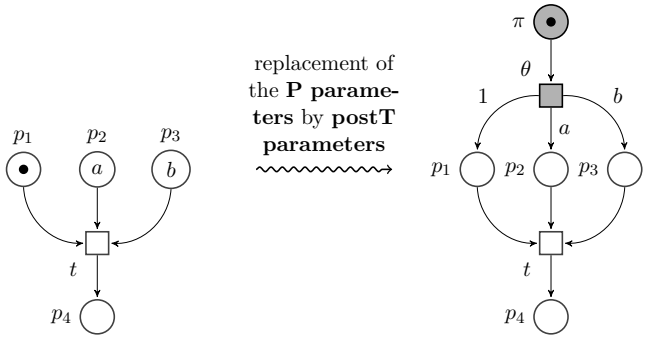


Figure 5.2 – From P-PPN to postT-PPN

As recalled earlier in Section 3.4, we extend our PPNs with labelling functions allowing to consider *silent actions*. Formally $\mathcal{S} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P}, \Lambda, \mu_0)$, has a set of transition T and T' denotes the set of transition of the constructed postT-PPN, $\mathcal{S}' = (P', T', \mathsf{Pre}', \mathsf{Post}', \mathbb{P}, \Lambda', \mu_0')$ such that $T' = T \cup \Theta$ with $T \cap \Theta = \emptyset$. In fact, here $\Theta = \{\theta\}$. For each $t \in T$, $\Lambda(t) = t$ and $\Lambda(\theta) = \epsilon$. Note that for all v, each path in $v(\mathcal{S})$ can be done in $v(\mathcal{S}')$ by adding θ at the beginning. And reciprocally, each path in $\mathcal{S}'$ begins by θ so is written θw where w is a path in $v(\mathcal{S})$.

Lemma 13
$\forall v \in \mathbb{N}^{\mathbb{P}}$, $v(\mathcal{S})$ and $v(\mathcal{S}')$ are weakly bisimilar.

Proof. Let $\nu \in \mathbb{N}^{\mathbb{P}}$ a valuation of the parameters. We want to show that $\nu(\mathcal{S})$ and $\nu(\mathcal{S}')$ are weakly bisimilar. Let $\nu(\mu_0)$ be the parametric initial marking of $\nu(\mathcal{S})$ and $\nu(m_0') = m_0'$ the initial marking of $\nu(\mathcal{S}')$. The only transition firable from m_0' is θ and $m_0' \xrightarrow{\theta} \nu(\mu_0)$ as shown in Figure 5.2. From $\nu(\mu_0)$, $\mathcal{S}$ and $\mathcal{S}'$ are isomorphic. So $\nu(\mathcal{S}_1)$ and $\nu(\mathcal{S}_2)$ are weakly-bisimilar. □

These result underlines that using parameters on output arcs is more powerful than using parameters on markings. We can conclude that T-PPNs are at least as expressive as P-PPNs and thus as expressive as PPNs.

Remark 5.2.1. This translation adds 1 place and 1 transition and at most $|P| + 1$ arcs. Thus this transformation is linear [2] in the size of the P-PPN.

2. The corresponding algorithm would be linear time.

5.2.2 Translating postT-PPNs to P-PPNs

We now show that from a postT-PPN, $\mathcal{S}_1 = (P_1, T_1, \mathsf{Pre}_1, \mathsf{Post}_1, \mathbb{P}_1, \Lambda_1, m_1^0)$ we can construct a P-PPN, $\mathcal{S}_2 = (P_2, T_2, \mathsf{Pre}_2, \mathsf{Post}_2, \mathbb{P}_2, \Lambda_2, \mu_2^0)$ that weakly-simulates the behaviours of the postT-PPN. Reciprocally, the postT-PPN also weakly-simulates the behaviours of the P-PPN built.

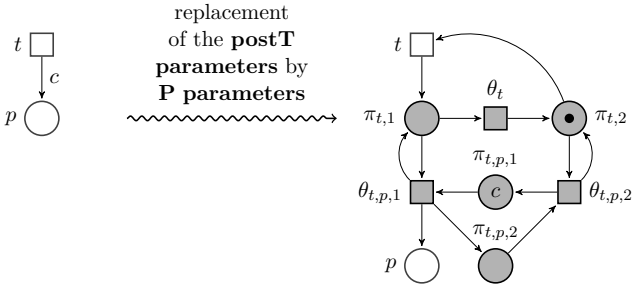


Figure 5.3 – From postT-PPN to P-PPN

For each transition t and place p such that the arc (t, p) is weighted by a parameter c, we construct the net depicted in Figure 5.3 which replaces this arc[3]. 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 $T_1 \cup \{\epsilon\}$ 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 14
$\forall v \in \mathbb{N}^{\mathbb{P}}$, $v(\mathcal{S}_1)$ and $v(\mathcal{S}_2)$ are weakly cosimilar.

Proof. We will prove the 2 weak-simulations.

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

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

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

Remark 5.2.2. This is not a weak-bisimulation. Indeed, if $\mathcal{S}_2$ adds 3 tokens in p (leading to a marking m_2) whereas $\mathcal{S}_1$ adds $v(c) = 4$ 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 $\mathcal{S}_1$ only. Here the two simulations relations are not reciprocal: m_1 would simulates m_2 but m_2 would not.

Remark 5.2.3. This translation adds 4 places and 3 transitions and 12 arcs for each parametric arc of the postT-PPN. Thus this transformation is linear[4] in the number of parametric arcs (which is bounded by the number of arcs) of the postT-PPN.

We have clearly established the link between postT-PPNs and PPNs inducing that T-PPNs are as expressive as PPNs. Therefore, by abuse, we may refer to T-PPNs simply as PPNs. In the sequel, we study the behaviour of the two elementary classes of preT-PPNs and postT-PPNs. We will derive some results on P-PPNs and distinctT-PPNs from this survey in Section 5.9.

3. Notice that if several labelled arcs come from the same transition, some places and transitions of the Figure 5.3 should be duplicated according to indices.
4. The corresponding algorithm would be linear time.

5.3 Monotonicity in PreT-PPNs and PostT-PPNs

When we restrict the use of parameters to input arcs, we ensure that any marking coverable in a v-instance remains coverable for any v'-instance such that $v' \leq v$. Intuitively, decreasing the valuation leads to a more permissive firing condition. Symmetrically, when we restrict the use of parameters to output arcs, we ensure that any marking coverable in a v-instance, remains coverable for any v'-instance such that $v \leq v'$. Intuitively, firing the same parametric transition while increasing the valuation leads to greater (and thus more permissive) markings. Those results are formalised in Lemma 15.

Lemma 15
Let $\mathcal{S}_{pre}$ and $\mathcal{S}_{post}$ be a marked preT-PPN and postT-PPN of initial markings m_0 and s_0 respectively.

- *For every transition sequence w of $\mathcal{S}_{pre}$ and for every valuation v, if $m_0 \xrightarrow{w} m$ in $v(\mathcal{S}_{pre})$, then for every valuation $v' \leq v$, there exists $m' \geq m$ such that $m_0 \xrightarrow{w} m'$ in $v'(\mathcal{S}_{pre})$.*
- *For every transition sequence w of $\mathcal{S}_{post}$ and for every valuation v, if $s_0 \xrightarrow{w} s$ in $v(\mathcal{S}_{post})$, then for every valuation $v' \geq v$, there exists $s' \geq s$ such that $s_0 \xrightarrow{w} s'$ in $v'(\mathcal{S}_{post})$.*

Proof. We only provide the proof for the preT-PPN. The case of postT-PPN can be adapted easily.

Given $w_v = t_1, t_2, ..., t_i, ..., t_n$, we proceed by induction on the length of w_v

- *Base case: n=0*
  If the sequence is empty, then $m_0 \geq m_0$ and by definition m_0 is reachable in v and v' instances.
- *Inductive step:* Let us assume the lemma holds for a sequence of length n
  Given $w = t_1, t_2, ..., t_i, ..., t_n, t_{n+1}$, we apply the induction hypothesis on $w_n = t_1, t_2, ..., t_n$ such that $m_0 \xrightarrow{w_n} m_n \xrightarrow{t_{n+1}} m_{n+1}$ in $v(\mathcal{S})$. By the induction hypothesis, for $v' \leq v$ we have $m_0 \xrightarrow{w_n} m'_n \geq m_n$ in $v'(\mathcal{S})$. Moreover t_{n+1} is firable in $v(\mathcal{S})$ so $m_n \geq v(\mathsf{Pre}(t_{n+1})) \geq v'(\mathsf{Pre}(t_{n+1}))$. So $m'_n \geq v'(\mathsf{Pre}(t_{n+1}))$ and $m_0 \xrightarrow{w} m'_{n+1}$ in $v'(\mathcal{S})$.
  Moreover since $v(\mathsf{Pre}(t_{n+1})) \geq v'(\mathsf{Pre}(t_{n+1}))$ and $\mathsf{Post}(t_{n+1}) \geq \vec{0}$, then $m'_{n+1} \geq m_{n+1}$, so the proposition is true at rank $n+1$.

By induction, it is true for every sequence of length n which proves Lemma 15. □

From this property we can directly deduce that

Corollary 16
The $\mathscr{U}$-coverability problem on postT-PPN is ExpSpace-complete.
The $\mathscr{E}$-coverability problem on preT-PPN is ExpSpace-complete.

Proof. Using Lemma 15, this can be directly proved by testing for the corresponding coverability problem on the instance under the lowest valuation of the domain, *i.e.* $\vec{0}$ here. □

5.4 Decidability of Existential Coverability for PostT-PPNs

In this section, we use ω-Petri nets as a reduction tool for the proofs. Let us consider a postT-PPN $\mathcal{N}$ and let us associate to this model the ωPN $\mathcal{N}'$ such that we replace each parametric arc of $\mathcal{N}$ by an ω arc. This transformation is a polynomial time transformation [5] from postT-PPN

5. More specifically we obtain an ω-output-PN or ωOPN for short, which corresponds to the natural subclass of ωPNs where $\mathsf{Pre} \in P \times T \mapsto \mathbb{N}$.

to ωPN (see [Geeraerts et al., 2015]) and we will show that it preserves existential coverability. It is then sufficient to invoke a transformation from ωPN to PN underlined in [Geeraerts et al., 2015] and recalled in Lemma 10.

Lemma 17 : From postT-PPNs to ω-Petri nets
Let $\mathcal{N}$ be a postT-PPN (which involves parameters of a set $\mathbb{P}$), let $\mathcal{N}'$ be its corresponding ω-PN (with the same sets of places and transitions) and let m_0 be their common initial marking. Given a marking $m \in \mathcal{RS}(\mathcal{N}', m_0)$, there exists a valuation v such that there exists a marking $m' \geq m$ with $m' \in \mathcal{RS}(v(\mathcal{N}), m_0)$. Moreover[a]*, $\cup_{v \in \mathbb{N}^{\mathbb{P}}} \mathcal{RS}(v(\mathcal{N}), m_0) \subseteq \mathcal{RS}(\mathcal{N}', m_0)$.*

a. Notice that this is only an inclusion. Indeed, contrarily to postT-PPNs, in ωPNs, the effect of an arc can change along the same execution.

Proof. We write $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post})$ a postT-PPN and $\mathcal{N}' = (P, T, \mathsf{Pre}, \mathsf{Post}')$ the corresponding ω-PN where the parameters on weights have been replaced by ω. Note that only the matrix Post changes. The first point comes directly from the semantics of ω-Petri nets. Given $m \in \mathcal{RS}(\mathcal{N}', m_0)$, there exists a run $m_0 \xrightarrow{w} m$. If w does not contain any ω-transition, then $m \in \mathcal{RS}(\mathcal{N}_p, m_0)$ thus $m \in \mathcal{RS}(\mathcal{N}, m_0)$. Otherwise, we consider all ω-transitions involved in w and compute their effective output effect on this particular run. That is to say, if an ω-transition t occurs in w between a marking m_i and m_{i+1}, its effective output effect is equal to $m_{i+1} - m_i + \mathsf{Pre}(t)$. We therefore construct a finite set of vectors. Now, let us consider the maximum value of the components of those vectors and call it k. We can then consider the valuation $\vec{k}$. By induction on the length of w we prove that w is firable in $\vec{k}(\mathcal{N})$ and its firing generates a marking greater or equal to m.

- *Base case:* the case where w is an empty sequence is straightforward.
- *Inductive step:* given $n \in \mathbb{N}$, we suppose that if w has a length equal to n and w is firable in $\mathcal{N}'$ generating a marking m_1, then w is firable in $\vec{k}(\mathcal{N})$ plus the marking generated m_2 is greater than or equal to m_1. We now consider that w has a length equal to $n+1$. Then $w = w't$ where $|w'| = n$. We can apply the induction assumption. We obtain that w' is firable in $\vec{k}(\mathcal{N})$. Moreover, we write that $m_0 \xrightarrow{w'} m_3 \xrightarrow{t} m'_3$ in $\mathcal{N}'$, we can thus assert that $m_0 \xrightarrow{w'} m_4$ in $\vec{k}(\mathcal{N})$ with $m_4 \geq m_3$. Since t is firable from m_3, by monotonicity, it is firable from m_4. Moreover, by definition of k, $k \geq \max(m'_3 - m_3 + \mathsf{Pre}(t))$ therefore, in $\vec{k}(\mathcal{N})$, $\vec{k}(\mathsf{Post}(t)) \geq m'_3 - m_3 + \mathsf{Pre}(t)$. By construction, $m_0 \xrightarrow{w'} m_4 \xrightarrow{t} m'_4$ and $m'_4 = m_4 + \vec{k}(\mathsf{Post}(t)) - \mathsf{Pre}(t)$ which implies that $m'_4 \geq m_3 + \vec{k}(\mathsf{Post}(t)) - \mathsf{Pre}(t) \geq m'_3$. This proves that the proposition holds at rank $n+1$.

By induction, we can thus conclude that, $m_0 \xrightarrow{w} m' \geq m$ in $v(\mathcal{N})$.

For the second point, given a valuation and a marking reachable within $v(\mathcal{N})$ through a run w, it is easy to construct a run w' composed of the same transitions in the same order in $\mathcal{N}'$ with the same effect. Indeed, each time an ω transition is fired in w', the effect of the ω output arc should be equal to the value associated through v to the parameter of the corresponding parametric output arc.

□

We can thus directly deduce the following theorem by reducing existential coverability in postT-PPNs to coverability in ωPN which belongs to ExpSpace by [Geeraerts et al., 2015].

Theorem 18 : Complexity of Existential Coverability
The existential coverability problem on postT-PPNs is ExpSpace-complete.

Proof. Given $\mathcal{N}$ a postT-PPN we construct $\mathcal{N}'$ its translation (in polynomial time) in an ωOPN, by replacing each parametric arc by an ω arc. By Lemma 17 we can deduce that m is coverable in $\mathcal{N}'$ iff m is existentially coverable in $\mathcal{N}$:

- If m is coverable in $\mathcal{N}'$ then there exists a valuation v such that m is coverable in $v(\mathcal{N})$ by the first point of Lemma 17.
- If m is existentially coverable in $\mathcal{N}$, there exists a valuation v such that there exists $m' \in \mathcal{RS}(v(\mathcal{N}), m_0)$ with $m' \geq m$. Thus, $m' \in RS(\mathcal{N}, m_0)$ by the second point of Lemma 17.

So we only need to answer coverability of m in $\mathcal{N}'$ which is in ExpSpace. It follows that existential coverability for postT-PPNs is in ExpSpace.

Moreover, Remark 4.2.1 provides the ExpSpace-hardness of this problem, so it is ExpSpace-complete. □

5.5 Decidability of Universal Coverability for PreT-PPNs

We prove here that universal coverability is decidable for preT-PPNs. To this end, as a preliminary result, we show that it is sufficient for a marking to be coverable in infinitely many instances (under uniform [6] valuations) of the parametric Petri nets to be universally coverable. Indeed, for any valuation v, we can find a uniform valuation [7] $\vec{k}$ such that $v \leq \vec{k}$ and apply Lemma 15.

Lemma 19
Given a marking m and a marked preT-PPN $\mathcal{S}$, the following two propositions are equivalent :

1. *m is universally coverable in $\mathcal{S}$*
2. *$\{k \in \mathbb{N} \mid \texttt{cov}(\vec{k}(\mathcal{S}), m)\}$ is infinite*

Proof. We invite the reader to refer to the proof of Lemma 22 on page 67.

Lemma 22 is indeed a generalisation of Lemma 19 to the property of simultaneous unboundedness. □

Let us now provide the main reasoning. If a marking is universally coverable, two main cases are possible: we can either reach this marking without using any parametric transition, and then the corresponding run works for any valuation, or we need at least one parametric transition. In the latter case, since there is an infinite number of parameter valuations and a finite number of parametric transitions, there is at least one such transition that must be used, as the first parametric transition in the run, for an infinite number of valuations, and therefore, clearly the input places of its parametric arcs are not bounded.

Let us formalise this intuition. We recall that the coverability set $\mathcal{CS}(\mathcal{S})$ of a marked Petri net $\mathcal{S}$ is computable in the sense that its minimal downward basis is computable (see, *e.g.,* [Finkel and Leroux, 2015]). Here, the minimal downward basis is a finite set of extended markings. Given a marked Petri net $\mathcal{S} = (\mathcal{N}, m_0)$, we recall that $\mathcal{BCS}(\mathcal{S})$ for the minimal downward basis of $\mathcal{CS}(\mathcal{S})$. Given a marking m and a subset Q of places, we recall that $m_{|Q}$ denotes the marking m restricted to Q. Similarly, given a Petri net $\mathcal{N}$ and a subset of Q of places, let $\mathcal{N}_{|Q}$ denote the Petri net obtained from $\mathcal{N}$ by keeping only the places in Q. All transitions of $\mathcal{N}$ remain in $\mathcal{N}_{|Q}$ but the arcs from/to the places not in Q are simply removed.

Formally:

6. We recall that uniform valuations are valuations such that all components are mapped to the same value $\mathbb{N}$. Given an integer $k \in \mathbb{N}$, the corresponding uniform valuation is written $\vec{k}$.

7. We recall that the uniform valuation $\vec{k}$ is the valuation that associates to every parameter of its domain, $\texttt{dom}(()v)$ here, the value k.

Lemma 20
Let $\mathcal{S} = (\mathcal{N}, m_0)$ be a marked preT-PPN and let $\mathcal{N}_p$ denote the Petri net obtained from $\mathcal{N}$ by removing all parametric transitions. A marking m is universally coverable in $\mathcal{S}$ iff

1. *m is coverable in $(\mathcal{N}_p, m_0)$ or*
2. *there exists $z \in \mathcal{BCS}(\mathcal{N}_p, m_0)$ such that $\omega(z) \neq \emptyset$ and $m_{|\mathbb{N}(z)}$ is universally coverable in $(\mathcal{N}_{|\mathbb{N}(z)}, z_{|\mathbb{N}(z)})$*

Proof. $\Rightarrow$ We first assume that m is universally coverable in $(\mathcal{N}, m_0)$. Two cases arise: Either m is coverable in $(\mathcal{N}_p, m_0)$ and then there exists z in $\mathcal{BCS}(\mathcal{N}_p, m_0)$ such that $m \leq z$, which proves directly the implication. Otherwise m is not coverable in $(\mathcal{N}_p, m_0)$. Let us now consider this second case. Since m is universally coverable in $(\mathcal{N}, m_0)$, for every integer $k > 0$, there exists a run in $\vec{k}(\mathcal{N})$ from m_0 to a marking $u_k \geq m$. This run contains at least one parametric transition, otherwise m would be coverable in $(\mathcal{N}_p, m_0)$. We can thus decompose it as:

$$m_0 \xrightarrow{*} m_k \xrightarrow{t_k} m'_k \xrightarrow{*} u_k \tag{5.1}$$

where t_k is a parametric transition and the prefix $m_0 \xrightarrow{*} m_k$ contains no parametric transition. Therefore, m_k belongs to $\mathcal{CS}(\mathcal{N}_p, m_0)$. Moreover, we know that $\mathcal{BCS}(\mathcal{N}_p, m_0)$ is finite. Thus, using the pigeon hole's principle, there exists z in $\mathcal{BCS}(\mathcal{N}_p, m_0)$ such that $z \geq m_k$ for infinitely many k. We define the set $V = \{\vec{k} \mid z \geq m_k\}$. This set is infinite. Moreover, for each of these k, $z(p) \geq m_k(p) \geq k$ for some place p such that $\mathsf{Pre}(p, t_k) \in \mathbb{P}$ (such a place necessarily exists since t_k is a parametric transition). Since there are only finitely many places, the pigeon hole's principle now implies that $z(p) = \omega$ for some place p, and $\omega(z)$ is therefore not empty. To complete the implication, we now prove that $m_{|\mathbb{N}(z)}$ is universally coverable in $(\mathcal{N}_{|\mathbb{N}(z)}, z_{|\mathbb{N}(z)})$. Lemma 19 states that it is sufficient to prove that $m_{|\mathbb{N}(z)}$ is coverable in $\vec{k}(\mathcal{N}_{|\mathbb{N}(z)}, z_{|\mathbb{N}(z)})$ for every $\vec{k} \in V$. We get from Equation 5.1 that, in the Petri net $\vec{k}(\mathcal{N}_{|\mathbb{N}(z)})$, the following run holds:

$$m_{k|\mathbb{N}(z)} \xrightarrow{t_k} m'_{k|\mathbb{N}(z)} \xrightarrow{*} u_{k|\mathbb{N}(z)}$$

Moreover, we can notice that $z_{|\mathbb{N}(z)} \geq m_{k|\mathbb{N}(z)}$ since $z \geq m_k$. Therefore, by monotonicity of the firing rule, we can fire the same run starting from $z_{|\mathbb{N}(z)}$. We thus get a run $z_{|\mathbb{N}(z)} \xrightarrow{*} u'_k$ in $\vec{k}(\mathcal{N}_{|\mathbb{N}(z)})$ and by monotonicity $u'_k \geq u_{k|\mathbb{N}(z)}$. Finally, by construction $u_{k|\mathbb{N}(z)} \geq m_{|\mathbb{N}(z)}$, thus $u'_k \geq m_{|\mathbb{N}(z)}$ and $m_{|\mathbb{N}(z)}$ is coverable in $\vec{k}(\mathcal{N}_{|\mathbb{N}(z)}, z_{|\mathbb{N}(z)})$ which concludes this implication.

$\Leftarrow$ Conversely, let us assume that there exists z in $\mathcal{BCS}(\mathcal{N}_p, m_0)$ such that (1) or (2) holds. First, if (1) holds, then m is trivially universally coverable in $(\mathcal{N}, m_0)$. Let us now assume that (1) does not hold (and thus (2) holds). Let k be a positive integer. By the second condition, $m_{|\mathbb{N}(z)}$ is coverable in $\vec{k}(\mathcal{N}_{|\mathbb{N}(z)}, z_{|\mathbb{N}(z)})$. Thus, there exists a run $z_{|\mathbb{N}(z)} \xrightarrow{*} m'$ in $\vec{k}(\mathcal{N}_{|\mathbb{N}(z)})$ with $m' \geq m_{|\mathbb{N}(z)}$. Let us consider a marking $y \in \mathbb{N}^P$ such that $y_{|\mathbb{N}(z)} = z_{|\mathbb{N}(z)}$. Since $z(p) = \omega$ for every place p in $P \backslash \mathbb{N}(z)$, we get that $z \geq y$, regardless of the value of $y_{|P\backslash\mathbb{N}(z)}$. Therefore $y \in \mathcal{CS}(\mathcal{N}_p, m_0)$ hence $y \in \mathcal{CS}(\vec{k}(\mathcal{N}), m_0)$. Moreover, we can still fire the same run in $\vec{k}(\mathcal{N})$ and get that $y \xrightarrow{*} m''$ where $m''_{|\mathbb{N}(z)} = m'$. Moreover, $m''_{|\mathbb{N}(z)} \geq m_{|\mathbb{N}(z)}$ and we can impose that $y_{|\omega(z)}$ is big enough to lead to $m''_{|\omega(z)} \geq m_{|\omega(z)}$ (otherwise, we should just choose a y greater on $\omega(z)$). Since $y \in \mathcal{CS}(\vec{k}(\mathcal{N}), m_0)$, we can deduce that m is coverable in $(\vec{k}(\mathcal{N}), m_0)$. This is true for every $k \geq 0$, we can therefore invoke Lemma 19 and conclude that m is universally coverable in $(\mathcal{N}, m_0)$, which concludes the proof. □

Remark now that the dimension (number of places) of the marked preT-PPN $(\mathcal{N}_{|\mathbb{N}(z)}, z_{|\mathbb{N}(z)})$ is strictly smaller than the dimension of $(\mathcal{N}, m_0)$. We can derive a decision procedure $\mathsf{ucov}_{\mathsf{pre}}$ for universal coverability for preT-PPNs from Lemma 20. Considering an initial marking m_0, a

preT-PPN $\mathcal{N}$ and a marking m to cover:

$$\mathsf{ucov}_{\mathsf{pre}}(\mathcal{N}, m_0, m) = \Big(\mathsf{cov}\big((\mathcal{N}_p, m_0), m\big)\Big) \vee \Big(\bigvee_{\substack{z \in \mathcal{BCS}(\mathcal{N}_p, m_0) \\ \omega(z) \neq \emptyset}} \mathsf{ucov}_{\mathsf{pre}}\big((\mathcal{N}_{p_{|\mathbb{N}(z)}}, m_{0|\mathbb{N}(z)}), m_{|\mathbb{N}(z)}\big)\Big)$$

Notice that in the case where a Petri net has no place, coverability will always be true. Thus can we immediately deduce the following theorem:

Theorem 21
Universal coverability is decidable in PreT-PPNs.

5.6 ExpSpace Upper Bound for Universal Coverability in PreT-PPNs

In previous section, we answered the decidability of universal coverability for preT-PPNs. We now focus on its complexity through that of the more general universal simultaneous unboundedness. We will prove that both are ExpSpace-complete.

5.6.1 Overview and Preliminaries

To prove that universal coverability in preT-PPNs belong to ExpSpace, we will prove that universal simultaneous unboundedness in preT-PPNs is ExpSpace. Together with remark of Section 4.2.1, knowing that coverability can be reduced to the simultaneous unboundedness problem as recalled in Section 3.2, we can then conclude that both problems are ExpSpace-complete.

Formally:

Definition 35 : Universal Simultaneous Unboundedness
Given a parametric Petri net, and a subset of places X of this net, the parametric net is universal simultaneous X unbounded *iff for every possible valuation v of its parameters, the v-instance of this net is simultaneous X unbounded.*

We first show that it is sufficient for a net to be simultaneous unbounded on a set of places in infinitely many instances [8] (under uniform valuations) of the parametric Petri net to be universally simultaneous unbounded on this set of places. Indeed, for any valuation v, we can find a uniform valuation $\vec{k}$ such that $v \leq \vec{k}$ and apply Lemma 15.

Lemma 22
Given a marked preT-PPN $\mathcal{S}$, a marking m, and X a subset of places of $\mathcal{S}$, the following two propositions are equivalent :

1. *$(\mathcal{S}, m_0)$ is universally simultaneous X unbounded*
2. *$\{k \in \mathbb{N} \mid (\vec{k}(\mathcal{S}), m_0)$ is simultaneous X unbounded$\}$ is infinite*

Proof. (1) $\Rightarrow$ (2): Let $\mathscr{V} = \{\vec{k} \mid k \in \mathbb{N}\}$. Set $\mathscr{V}$ is infinite. Suppose that for all valuations $v \in \mathbb{N}^{\mathbb{P}}$, $v(\mathcal{S})$ is simultaneous X unbounded. Since $\mathscr{V} \subseteq \mathbb{N}^{\mathbb{P}}$, we have in particular that for any $v \in \mathscr{V}$, $v(\mathcal{S})$ is simultaneous X unbounded *i.e.* (2).

(2) $\Rightarrow$ (1): Let $\mathscr{V} = \{\vec{k} \mid k \in \mathbb{N}$ and $(\vec{k}(\mathcal{S}), m_0)$ is simultaneous X unbounded$\}$. Suppose $\mathscr{V}$ is infinite. Let us consider $v \in \mathbb{N}^{\mathbb{P}}$. We define $k_1 = \max_{\lambda \in \mathbb{P}}(v(\lambda))$ *i.e.* the maximum of v componentwise. Clearly the number of elements in $\mathscr{V}$ that are less or equal to $\vec{k_1}$ is finite (they form a bounded subset of $\mathbb{N}^P$), so there exists $\vec{k_2} \in \mathscr{V}$ such that $\vec{k_2} \geq \vec{k_1}$. So $\vec{k_2}(\mathcal{S})$ is simultaneous X unbounded. This means that, for every value $B > 0$ we can consider a transitions sequence w s.t. in $\vec{k_2}(\mathcal{S})$, $m_0 \xrightarrow{w} m_{k_2}$ such that for all $p \in X$, $m_{k_2}(p) \geq B$. Therefore by Lemma 15, in $v(\mathcal{S})$, $m_0 \xrightarrow{w} m_v \geq m_{k_2} \geq B$. So $v(\mathcal{S})$ is simultaneous X unbounded and we have (1). □

8. This result is very close to the one stated in Lemma 19.

We can now address the problem of universal simultaneous unboundedness. To solve this problem, we reduce it to the existence of a classic Petri net built upon our parametric model satisfying an adequately chosen simultaneous unboundedness property. The classic Petri net is in fact obtained by evaluating a preT-PPN, called *incremental net*, under the uniform valuation $\vec{0}$. The incremental net has a polynomial size in the original preT-PPN and it directly depends on the original preT-PPN plus a sequence of distinct parametric transitions. This section is thus driven by the idea that universal simultaneous X unboundedness on a preT-PPN $\mathcal{S}$ is equivalent to the existence of a sequence σ of distinct parametric transitions of $\mathcal{S}$, such that the incremental model built on $\mathcal{S}$ and σ evaluated under $\vec{0}$ satisfies a simultaneous unboundedness property depending on X and σ.

5.6.2 Notion of Incremental Model

Before providing the theoretical definition, let us consider the main intuition of our construction. If a net is universally simultaneous unbounded on a set of places X, as for coverability, two main cases are possible: we can either find a path such that the places of X are unbounded without using any parametric transition, and then the corresponding run works for any valuation, or we need at least one parametric transition.

In the latter case, since there is an infinite number of parameter valuations and a finite number of parametric transitions, using the pigeonhole principle, there is at least one such transition that must be used as the first parametric transition in the run for an infinite number of valuations. The input places of its parametric arcs are therefore not bounded. Thus, the valuation of the input parametric arcs of this transition is not limiting anymore since we can generate an arbitrary large amount of tokens in the corresponding places. Therefore, we will later evaluate [9] those parameters to 0 in order to perform the verification on a classic Petri net.

Nevertheless, we need to ensure that the input places of the parametric arcs are not bounded (without using that transition). This is exactly the goal of this incremental model. Indeed, once fired, we could then consider a new net where the first parametric transition can be involved as well as non parametric transitions and investigate for the newly unbounded places. Either we can unbound the places of the set X or we can reuse the previous reasoning and choose a new parametric transition that has to be involved in infinitely many instances. What is important to note here is that at each firing of a new parametric transition, that never occurred in the run, we need to ensure that its input places of parametric input arcs were unbounded using only previous transitions of the run and to remember what are the new places that can be unbounded through the use of this new transition. We will now formalise how it is possible to remember the boundedness of the input places of parametric arcs by presenting exhaustively the model of incremental nets.

Given a preT-PPN $\mathcal{N} = (P, T', \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$ and a partition of its transitions $T' = T \cup \Theta$ between its plain and parametric transitions, we denote by $\mathcal{N}_p$ the Petri net obtained from $\mathcal{N}$ by removing all transitions of Θ from $\mathcal{N}$. An example is given in Figure 5.4. Let us now consider a finite sequence $\sigma \in Pref(S_\Theta)$ where S_Θ is the symmetric group over Θ seen as a language. Let $|T| = m$, $|P| = n$ and $|\sigma| = k$. We define the incremental model $\mathcal{I}$ of $\mathcal{N}$ along σ. We write $\mathcal{I} = \texttt{incr}(\mathcal{N}, \sigma)$ to denote this preT-PPN. This model is illustrated by the example [10] at the right hand side of Figure 5.4. Its construction consists of the following main steps:

(i) Consider $\mathcal{N}_p$ and k copies of $\mathcal{N}_p$, where to each of those $k+1$ subnets is associated a global lock place, ensuring that exactly one copy is active at any given instant. The copies correspond to the black subnet of this example, whereas the global locks p_0^i's are depicted in blue and dotted arcs.

(ii) Add a copy of the first i transitions of σ to each i^{th} copy of $\mathcal{N}_p$, for $1 \leq i \leq k$.

9. Note that any other finite valuation would be suitable since the input places of the parametric arcs are unbounded.

10. The exact meaning of the notations used to refer to the different components of this example will be provided after this informal intuition of the construction.

(iii) Between the $i-1^{th}$ and i^{th} subnets, add a copy of the $i+1^{th}$ transition of σ, depicted in plain green arcs in Figure 5.4, for $1 \leq i \leq k$. Notice that this copy presents a special behaviour: its input effect impacts the $i-1^{th}$ subnet and its lock p_0^{i-1} whereas its output effect impacts the i^{th} subnet and its lock p_0^i. This ensures that we change of active subnet only after the firing of a the first occurrence of a precise parametric transition.

(iv) Finally, we ensure that given every copy of a transition, including the intermediate copies that allow to change the active copy, it modifies simultaneously the places in the associated copy as explained above, but also all copies of greater index (*i.e.* those that have not been activated yet). Those arcs ensure that every later subnet always has the "same" marking as the active copy. They are depicted by dashed red arcs in Figure 5.4. Note that we synchronise the different copies and do not merge them because we use them to remember the order in which the different input places to parametric transitions become unbounded.

Let us suppose we evaluated this incremental model in order to perform an execution. At the beginning of any execution, given a precise subnet, let us say the i^{th} subnet, it follows the behaviour of the subnets with lower index because of synchronisations introduced in item (iv). Then, once this copy becomes active, after the firing of a given parametric transition introduced in item (iii), it will dictate the behaviour of the global net (and thus of the subnets with greater index through synchronisations). Once the next copy becomes active, our original i^{th} subnet cannot change its state anymore. It is now literally an historic state of the global run of the incremental net.

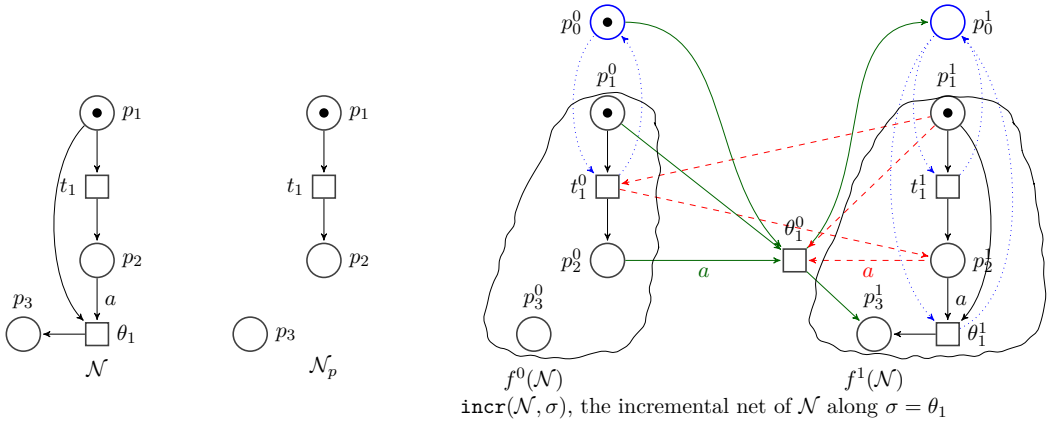


Figure 5.4 – Construction of an incremental net

More formally, the incremental net $\texttt{incr}(\mathcal{N}, \sigma)$ is the preT-PPN $(\mathcal{P}, \mathcal{T}, \textsc{Pre}, \textsc{Post}, \mathbb{P})$ such that $\mathcal{P} = \{p_j^i \mid 0 \leq i \leq k \wedge 0 \leq j \leq n\}$ where p_0^i represents the lock related to the i^{th} copy of $\mathcal{N}$ whereas p_j^i with $j > 0$ represents the copy of the place p_j of P in the i^{th} subnet and $\mathcal{T} = \{t_j^i \mid 0 \leq i \leq k \wedge 1 \leq j \leq m\} \cup \{\theta_j^i \mid 1 \leq j \leq i \leq k\} \cup (\cup_{1 \leq i \leq k} \{\theta_i^0\})$ where t_j^i represents the copy of the transition t_j of T in the i^{th} subnet, θ_j^i represents the copy of the transition θ_j of Θ in the i^{th} subnet, θ_i^0 represents a copy of the transition θ_i from σ which is used to change the active copy (from the $i-1^{th}$ to the i^{th}).

We define with this construction a net mapping to relate places and transitions from both models. Given the two nets $\mathcal{N}$ and $\mathcal{I}$ defined as above, considering previous notations, for each $0 \leq i \leq |\sigma|$ we define the application $f^i_{\mathcal{N}\to\mathcal{I}}$ that links the original net $\mathcal{N}$ to its i^{th} copy in $\mathcal{I}$ (except the corresponding lock). We define $f^i_{\mathcal{N}\to\mathcal{I}} : T \cup \{\theta_j \in \sigma \mid j \leq i\} \cup P \to \mathcal{T} \cup \mathcal{P}$ such that for $t_j \in T$ (resp. $\theta_j \in \sigma$ and $p_j \in P$), $f^i_{\mathcal{N}\to\mathcal{I}}(t_j) = t_j^i$ (resp. $f^i_{\mathcal{N}\to\mathcal{I}}(\theta_j) = \theta_j^i$ and $f^i_{\mathcal{N}\to\mathcal{I}}(p_j) = p_j^i$). We can then define f^{-1} the function that maps components of the copies of $\mathcal{N}$ in $\mathcal{I}$ to their original fiber by the previous application. Formally, f^{-1} is defined by: $f^{-1} : \cup_{0 \leq i \leq k} \texttt{im}(f^i_{\mathcal{N}\to\mathcal{I}}) \to T \cup P \cup \Theta$ and associates to $t_j^i \in \mathcal{T}$ (resp. $\theta_j^i \in \mathcal{T}$ and $p_j^i \in \mathcal{P}$) $f^{-1}(t_j^i) = t_j$ (resp. $f^{-1}(\theta_j^i) = \theta_j$ and $f^{-1}(p_j^i) = p_j$). Finally, we define the application

$h_{\mathcal{I}\to\mathcal{N}} : \cup_{1\leq i\leq k}\{\theta_i^0\} \subseteq \mathcal{T} \to \Theta$ that maps the intermediate parametric transitions between each copies of $\mathcal{N}$ in $\mathcal{I}$, θ_i^0 to their original fiber from $\mathcal{N}$ and occurring in σ, that is to say the i^{th} transition of σ.

Those applications allow us to define formally the functions Pre and Post. Given i' and j', let $x_{j'}^{i'}$ denote either t_j^i or θ_j^i from $\{t_j^i \mid 0 \leq i \leq k \wedge 1 \leq j \leq m\} \cup \{\theta_j^i \mid 1 \leq j \leq i \leq k\}$ in the following expressions:

- $\textsc{Pre}(p_j^i, x_{j'}^{i'})(resp.\ \textsc{Post}(p_j^i, x_{j'}^{i'})) = \begin{cases} 0 \text{ if } (i < i') \text{ or } (i > i' \text{ and } j = 0) \\ 1 \text{ if } i = i' \text{ and } j = 0 \\ \mathsf{Pre}(f^{-1}(p_j^i), f^{-1}(x_{j'}^{i'})) \text{ otherwise} \\ (resp.\ \mathsf{Post}(f^{-1}(p_j^i), f^{-1}(x_{j'}^{i'})) \text{ otherwise }) \end{cases}$
- $\textsc{Pre}(p_j^i, \theta_{i'}^0) = \begin{cases} 0 & \text{if } (i+1 < i') \text{ or } (i+1 > i' \text{ and } j = 0) \\ 1 & \text{if } i+1 = i' \text{ and } j = 0 \\ \mathsf{Pre}(f^{-1}(p_j^i), h^{-1}(\theta_{i'}^0)) & \text{otherwise} \end{cases}$
- $\textsc{Post}(p_j^i, \theta_{i'}^0) = \begin{cases} 0 & \text{if } (i < i') \text{ or } (i > i' \text{ and } j = 0) \\ 1 & \text{if } i = i' \text{ and } j = 0 \\ \mathsf{Post}(f^{-1}(p_j^i), h^{-1}(\theta_{i'}^0)) & \text{otherwise} \end{cases}$

Given a net $\mathcal{N}$ and the function $f^i_{\mathcal{N}\to\mathcal{I}}$ we extend the definition of $f^i_{\mathcal{N}\to\mathcal{I}}$ to sets by $f^i_{\mathcal{N}\to\mathcal{I}}(X) = \{f^i(x) \mid x \in X\}$ and nets by defining $f^i_{\mathcal{N}\to\mathcal{I}}(\mathcal{N})$ as $(f^i(P), f^i(T), \textsc{Pre}_{f^i(P)\times f^i(T)}, \textsc{Post}_{f^i(P)\times f^i(T)}, \mathbb{P})$. When the context is clear, we omit the subscript $\mathcal{N} \to \mathcal{I}$. As examples, $f^0(\mathcal{N})$ and $f^1(\mathcal{N})$ are provided in Figure 5.4. Finally, we associate to a marking m of $\mathcal{N}$ the marking $f(m)$ defined by for all p of $\cup_{0\leq i\leq k}(f^i(P))$, $f(m)(p) = m(f^{-1}(p))$. Notice that this ignores the locks introduced in the net. Given the initial marking m_0, we thus define the initial marking of the incremental net μ_0 as $f(m_0)$ for the copies of the places, and 0 in all locks except the first one which receives 1. Formally, $\mu_0(p_j^i) = m_0(p_j)$ if $j \neq 0$, 1 if $i = j = 0$ and 0 otherwise.

The idea behind this construction is double. First, we can enforce the order of the first occurrence of a parametric transition which is dictated by the sequence σ. Second, we can access the exact amount of tokens stored in a place before the firing of the first occurrence of a parametric transition and thus keep an historic of the state of a run, just before the firing of this parametric transition, through the copies of the original net. Based on those two observations, we will be able to observe if the input places of the parametric arcs of the first occurrence of a parametric transition in a run are bounded or not.

5.6.3 Complexity of Universal Simultaneous Unboundedness

We will now prove that universal simultaneous unboundedness can be reduced to the existence of a sequence σ of distinct parametric transitions such that the incremental net built upon this sequence is simultaneous unbounded on an adequately defined set of places. Indeed, we must ensure that each input place of a parametric arc of the first occurrence of a parametric transition is unbounded. Based on the previous construction, one can notice that given a parametric transition θ_i occurring in σ, its input places are only impacted by the transitions occurring in the first i copies of the net. Thus, once θ_i is fired in the incremental net, the new feasible transitions will not impact the amount of tokens stored in the i first subnets. We will thus be able to verify if the input places were bounded or not before its firing, by observing the places of the copy that occurs just before the first firing of this transition. For each parametric transition of σ, we should thus verify that the input places in the corresponding copies are unbounded, and finally verify that the places that should be unbounded as part of the original property are indeed unbounded in the last copy of the net and that for each instance of the incremental net under a uniform valuation. Nevertheless, since the corresponding input places of parametric transitions are unbounded, it is sufficient to verify this property for only one instance of the incremental net, and in particular we will later choose the $\vec{0}$-instance. Indeed, if such a property is verified

for any $\vec{k}$-instance, then, it could be verified for any $\vec{k'}$-instance (with $k' > k$) by exhibiting the witness run and performing more loops.

Given $t \in \Theta$, $\Pi(t) \subseteq P$ is used to represent the places p of P such that $\mathsf{Pre}(p,t)$ or $\mathsf{Post}(p,t)$ belongs to $\mathbb{P}$.

Lemma 23
Let $\mathcal{N} = (P, T', \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$ be a preT-PPN, such that $T' = T \cup \Theta$ where Θ represents the parametric transitions of $\mathcal{N}$ and T its plain transitions. For every set of places of $X \subseteq P$, the following propositions are equivalent:

1. *$(\mathcal{N}, m_0)$ is universally simultaneous X unbounded*
2. *$\exists \sigma = t_1, ..., t_l \in Pref(S_\Theta)$, considering the incremental model $\mathcal{I}$ of $\mathcal{N}$ along σ, $\mathcal{I} = \boldsymbol{incr}(\mathcal{N}, \sigma)$, $\exists k \in \mathbb{N}$, $(\vec{k}(\mathcal{I}), \mu_0)$ is simultaneous Y unbounded where $Y = f^l_{\mathcal{N}\to\mathcal{I}}(X) \cup (\bigcup_{t_i \in \sigma} f^{i-1}_{\mathcal{N}\to\mathcal{I}}(\Pi(t_i)))$.*
3. *$\exists \sigma = t_1, ..., t_l \in Pref(S_\Theta)$, considering the incremental model $\mathcal{I}$ of $\mathcal{N}$ along σ, $\mathcal{I} = \boldsymbol{incr}(\mathcal{N}, \sigma)$, $\forall k \in \mathbb{N}$, $(\vec{k}(\mathcal{I}), \mu_0)$ is simultaneous Y unbounded where $Y = f^l_{\mathcal{N}\to\mathcal{I}}(X) \cup (\bigcup_{t_i \in \sigma} f^{i-1}_{\mathcal{N}\to\mathcal{I}}(\Pi(t_i)))$.*

Following the notations, $Pref(S_\Theta)$ corresponds to the finite set of sequences of distinct parametric transitions. Remark that $f^l_{\mathcal{N}\to\mathcal{I}}(X)$ represents the copy of the places of X in the last subnet of the $\mathcal{I}$. Set $\bigcup_{t_i \in \sigma} f^{i-1}_{\mathcal{N}\to\mathcal{I}}(\Pi(t_i))$ is a bit more complex: for each transition $t_i \in \sigma$, $\Pi(t_i)$ represents the input places of the parametric arcs. We therefore address here the unboundedness of the copies of those places in the corresponding subnet of the incremental net.

Let us first introduce a few definitions. In the course of the proof we will relate parts of the construction using isomorphisms.

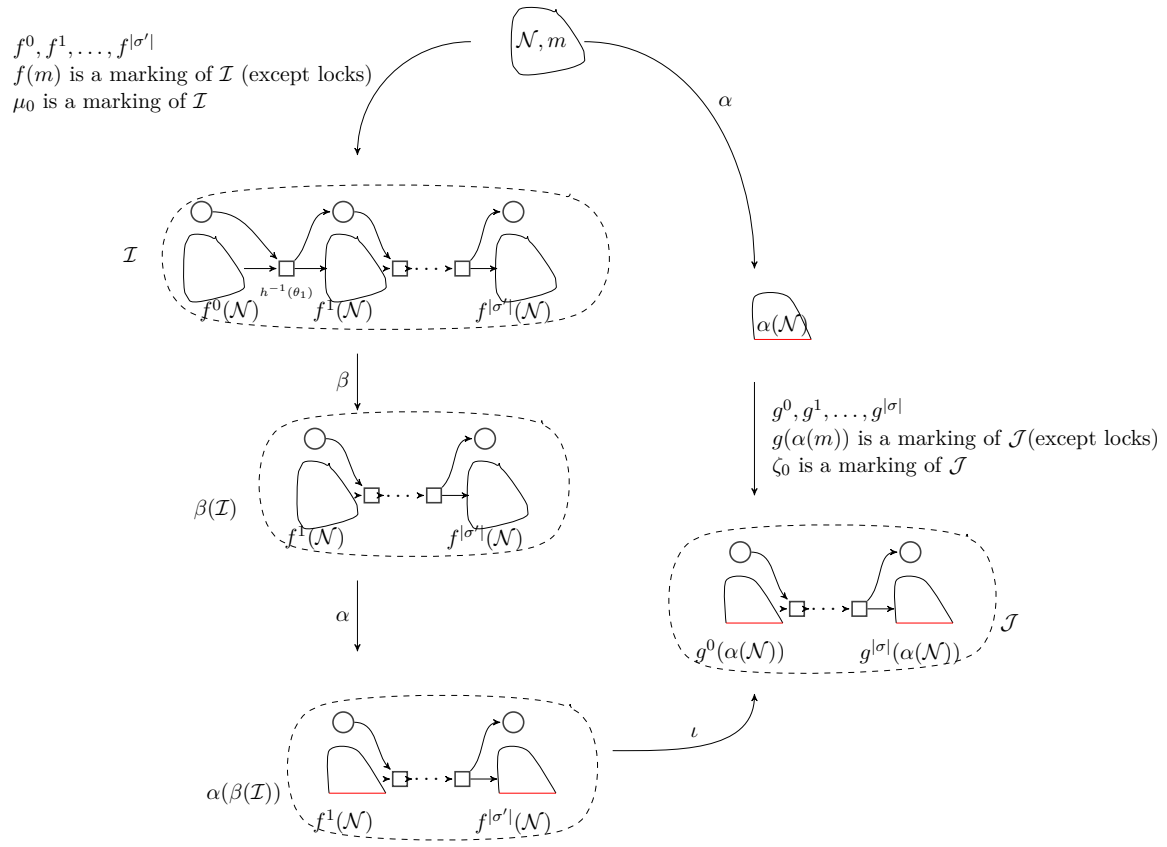
Definition 36 : Extension of [Couvreur et al., 2011]
Let $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$ and $\mathcal{N}' = (P', T', \mathsf{Pre}', \mathsf{Post}', \mathbb{P})$ be two PPNs and m_0, m'_0 their respective initial markings. Let $h : P \cup T \mapsto P' \cup T'$ be a mapping such that $h(P) \subseteq P'$ and $h(T) \subseteq T'$. We say that h is an isomorphism (of structure) from $\mathcal{N}$ to $\mathcal{N}'$ if h is a bijection and for each transition $t \in T$, we have $\mathsf{Pre}'(h(t)) = h(\mathsf{Pre}(t))$ and $\mathsf{Post}'(h(t)) = h(\mathsf{Post}(t))$.

We extend h to markings as follows: for all $p \in \boldsymbol{im}(h)$, $h(m)(p) = m(h^{-1}(p))$. If $m'_0 = h(m_0)$ we call it an isomorphism of net.

Considering the notations of Section 5.6.2, $(f^0_{\mathcal{N}\to\mathcal{I}}(\mathcal{N}), f^0_{\mathcal{N}\to\mathcal{I}}(m_0))$ and $(\mathcal{N}_p, m_0)$ are isomorphic, which is immediate by construction.

Proof of Lemma 23. (1) $\Rightarrow$ (3): We proceed by strong induction on the number $|\Theta|$ of parametric transitions in $\mathcal{N}$.

- *Base case:* if $|\Theta| = 0$, $\mathcal{N}$ is a classic Petri Net, which means $(\mathcal{N}, m_0)$ is simultaneous X unbounded. We recall that $\mathcal{I}$ denotes the incremental model of $\mathcal{N}$ along a sequence σ of parametric transitions and μ_0 its associated marking. Since $|\Theta| = 0$, σ must be ϵ. Thus, $\mathcal{I}$ only contains $f^0(\mathcal{N})$, which is isomorphic to $\mathcal{N}_p$, plus a lock. So for all k, $\vec{k}(f^0_{\mathcal{N}\to\mathcal{I}}(X))$ is simultaneous $f^0_{\mathcal{N}\to\mathcal{I}}(X)$ unbounded. Thus (3) holds.
- *Inductive step:* we consider $n \in \mathbb{N}$ and we assume that given a preT-PPN with $i \leq n$ parametric transitions, (3) holds. Let us assume $|\Theta| = n + 1$. Since $(\mathcal{N}, m_0)$ is universally simultaneous X unbounded, two cases arise:
  - either $(\mathcal{N}_p, m_0)$ is simultaneous X unbounded. Thus $\sigma = \epsilon$ satisfies the property (3) as explained in the base case.
  - otherwise, $(\mathcal{N}_p, m_0)$ is not simultaneous X unbounded. Nevertheless, by definition of universal simultaneous unboundedness, for every integer $k \geq 0$, $(\vec{k}(\mathcal{N}), m_0)$ is simultaneous X unbounded. Then, using Lemma 9, there exists a marking z_k reachable



For the sake of clarity, remark that we omit most of the arcs of the incremental net detailed in Figure 5.4.

Figure 5.5 – Relations between the different parts of the construction

in $\mathcal{KM}(\vec{k}(\mathcal{N}), m_0)$ such that $X \subseteq \omega(z_k)$. This means that, in the coverability graph of $(\vec{k}(\mathcal{N}), m_0)$, there is a run ρ_k leading to a node labelled by z_k. This run contains at least one parametric transition, otherwise $(\mathcal{N}_p, m_0)$ would be simultaneous X unbounded. We can thus decompose ρ_k as

$$m_0 \xrightarrow{*} m_k \xrightarrow{\theta_k} m'_k \xrightarrow{*} z_k$$

where θ_k is a parametric transition and the prefix $m_0 \xrightarrow{*} m_k$ contains no parametric transition. Therefore, m_k belongs to $\mathcal{CS}(\mathcal{N}_p, m_0)$. Moreover, we know that its basis $\mathcal{BCS}(\mathcal{N}_p, m_0)$ is finite. Thus, using the pigeon hole's principle, there exists z in $\mathcal{BCS}(\mathcal{N}_p, m_0)$ such that $z \geq m_k$ for infinitely many k. We define $V = \{\vec{k} \mid z \geq m_k\}$. This set is infinite. To each $\vec{k}$ of V, we can associate a parametric transition θ_k which is the first parametric transition involved in the run ρ_k. Nevertheless, there is a finite number of parametric transitions. We can thus invoke again the pigeon hole's principle and define an infinite subset of V, called V_1, such that for all $k \in V_1$, the θ_k's are equal, and without loss of genericity, we assume they are all equal to θ_1. For every k of V_1, for every $p \in P$, we know that $z(p) \geq m_k(p)$. In particular, we can now assert that, for every $p \in \Pi(\theta_1)$, $z(p) \geq m_k(p) \geq \vec{k}(\mathsf{Pre}(p, \theta_1)) = k$ by definition of $\Pi(\theta_1)$, where $\Pi(\theta_1) \subseteq P$ was defined has the subset of places p such that $\mathsf{Pre}(p, \theta_1) \in \mathbb{P}$. Therefore, $\Pi(\theta_1) \subseteq \omega(z)$. Notice that since $\theta_1 \in \Theta$, $\Pi(\theta_1) \neq \emptyset$.

Let α be the projection that associates to $\mathcal{N}$ the Petri net where the places of $\omega(z)$ have been removed. Through this operation [11], every arc that start or ends in a place

11. The relations between the different nets involved in this proof: $\mathcal{N}$, $\alpha(\mathcal{N})$, $\mathcal{I}$, $\mathcal{J}$, etc. are summarised in

of $\omega(z)$ is also removed. This projection can be extended to markings in the sense that given m, $\alpha(m) = m_{|\mathbb{N}(z)}$. We now consider the new preT-PPN $\alpha(\mathcal{N})$. It is in fact equal to $(\mathbb{N}(z), T', \mathsf{Pre}_{|\mathbb{N}(z)\times T'}, \mathsf{Post}_{|\mathbb{N}(z)\times T'}, \mathbb{P})$.

In this setting, we can partition T' into the set of plain and parametric transitions: $T' = T_1 \cup \Theta_1$. Remark that θ_1 is now an element of T_1 since its parametric arcs have been removed through the restriction to $\mathbb{N}(z)$. Thus, $\alpha(\mathcal{N})$ has strictly less than $n+1$ parametric transitions. We also define the operation inverse α_ω^{-1} operating on a marking of $\alpha(\mathcal{N})$ which reintroduces the places removed and valuate them to ω. The result is thus an extended marking of $\mathcal{N}$.

Let $\alpha(y)$ be the marking obtained by firing θ_1 from $\alpha(z)$. This operation is well defined since θ_1 is a plain transition in $\alpha(\mathcal{N})$. We can define y such that $y = \alpha_\omega^{-1}(\alpha(y))$.

Moreover, for every $k \in V_1$, $y \geq m'_k$ by construction and we can fire the suffix of ρ_k, and obtain by monotonicity, in the coverability graph of $(\vec{k}(\mathcal{N}), y)$, that there is a run such that $y \xrightarrow{*} y_k \geq z_k$. Thus $X \subseteq \omega(y_k)$, and $X \cap \mathbb{N}(z) \subseteq \omega(\alpha(y_k))$. Therefore by Lemma 9 for every uniform valuation $\vec{k}$ of V_1, $(\vec{k}(\alpha(\mathcal{N})), y)$ is simultaneous $X \cap \mathbb{N}(z)$ unbounded. We can invoke Lemma 22 to deduce that $(\alpha(\mathcal{N}), \alpha(y))$ is universally simultaneous $X \cap \mathbb{N}(z)$ unbounded.

As explained previously, this preT-PPN $\alpha(\mathcal{N})$ has at most n parametric transitions. By applying the induction hypothesis, there exists a sequence σ of parametric transitions from $\Theta_1 \subseteq (\Theta \setminus \{\theta_1\})$ satisfying item (3). We define the associated incremental net as $\mathcal{J}$ and its initial marking ζ_0. Following the definition of Section 5.6.2 we define $g^0, g^1, \ldots, g^{|\sigma|}$ from $\alpha(\mathcal{N})$ to $\mathcal{J}$ and given a marking m of $\alpha(\mathcal{N})$, we denote by $g(m)$ the marking of $\mathcal{J}$ which corresponds to duplicate m on each copies of $\alpha(\mathcal{N})$. Note that since we start from $\alpha(\mathcal{N})$ with an initial marking equal to $\alpha(y)$, the initial marking ζ_0 of $\mathcal{J}$ coincides with $g(\alpha(y))$ on the copies of the places of $\alpha(\mathcal{N})$ (*i.e.* if we omit the locks).

Now, let us consider the sequence $\sigma' = \theta_1\sigma$. We can define the incremental model $\mathcal{I}$ of $\mathcal{N}$ along σ' and its associated marking μ_0 which coincide with $f(m_0)$ on the copies of the places of $\mathcal{N}$. Following the definition of Section 5.6.2 we define $f^0, f^1, \ldots, f^{|\sigma'|}$ from $\mathcal{N}$ to $\mathcal{I}$ and given a marking m of $\mathcal{N}$, we denote by $f(m)$ the marking of $\mathcal{I}$ which corresponds to duplicate m on each copies of $\mathcal{N}$. We also define h that maps the intermediate parametric transitions of $\mathcal{I}$ that allows to change the active subnet to the original ones of σ'. Remark that h^{-1} is well defined.

Given $0 \leq i \leq |\sigma'|$, $f^i(\mathcal{N})$ is isomorphic to a subnet of $\mathcal{N}$ (indeed, only some parametric transitions are being removed). We can thus extend the definition of α to each of those nets. We now extend this projection to $\mathcal{I}$ as the union of the projections on each subnets of $\mathcal{I}$. By abuse of notation, we keep calling this operation α. We also extend its inverse α_ω^{-1} operating on a marking of $\alpha(\mathcal{I})$ which reintroduces the places removed and valuate them to ω. The result is thus a marking of $\mathcal{I}$.

We define β as the projection that associates to $\mathcal{I}$ a copy of this net where the following components have been removed:

- the places and transitions of $f^0(\mathcal{N})$
- the lock corresponding to $f^0(\mathcal{N})$
- the parametric transition that changes the active subnet from $f^0(\mathcal{N})$ to $f^1(\mathcal{N})$.

As previously we define its inverse β_0^{-1} operating on a marking of $\beta(\mathcal{I})$ which reintroduces the places removed by β and valuate them to 0, including the lock.

Let us now consider $\alpha(\beta(\vec{k}(\mathcal{I})))$. This net is isomorphic [12] to $\mathcal{J}$. Indeed, given $0 \leq i \leq |\sigma|$, $g^i(\alpha(\mathcal{N}))$ is isomorphic to $\alpha(f^{i+1}(\mathcal{N}))$. In particular for $i = 0$, notice that

Figure 5.5.

12. Informally, considering the notation of the Section 5.6.2, the application between $\mathcal{J}$ and $\alpha(\beta(\mathcal{I}))$ is defined by increasing all superscript of places and transitions by 1.

θ_1 belongs to $\alpha(\mathcal{N})$ and therein is not a parametric transition since all its parametric arcs have been removed through α. Therefore, by construction this transition has a copy in $g^0(\alpha(\mathcal{N}))$. Moreover, for $i > 1$ it also has a copy in every $f^i(\mathcal{N})$ by construction of $\mathcal{I}$. Now remark that $f^0(\mathcal{N})$, the only subnet that is not isomorphic to one of the $g^i(\alpha(\mathcal{N}))$, has been removed in $\beta(\mathcal{I})$. Then through α the subset $\omega(z)$ of places of $\mathcal{N}$ that were copied in $\mathcal{I}$ (and therefore $\beta(\mathcal{I})$) are removed in $\alpha(\beta(\mathcal{I}))$ to match the set of places of $\mathcal{J}$, which is built upon $\alpha(\mathcal{N})$.
We can thus denote by ι the isomorphism between $\alpha(\beta(\mathcal{I}))$ and $\mathcal{J}$. Given a marking m of $\mathcal{N}$, we thus have that:

$$\iota \circ \alpha \circ \beta \circ f(m) = g \circ \alpha(m)$$

Let us now study the markings of those nets. Given a valuation $\vec{k}$, we consider $(\vec{k}(\mathcal{J}), \zeta_0)$. It is simultaneous $g^{|\sigma|}(X) \cup \left(\bigcup_{t_i \in \sigma} g^{i-1}(\Pi(t_i))\right)$ unbounded by the induction hypothesis. Thus, there exists a sequence of transitions w in its coverability graph leading to a marking η such that $g^{|\sigma|}(X) \cup \left(\bigcup_{t_i \in \sigma} g^{i-1}(\Pi(t_i))\right) \subseteq \omega(\eta)$. Let us consider $(\alpha(\beta(\mathcal{I})), \iota^{-1}(\zeta_0))$, then $\iota^{-1}(w)$ leads to a marking $\iota^{-1}(\eta)$ with $\iota^{-1}\Big(g^{|\sigma|}(X) \cup \left(\bigcup_{t_i \in \sigma} g^{i-1}(\Pi(t_i))\right)\Big) \subseteq \omega(\iota^{-1}(\eta))$ by definition of ι^{-1}.

Now let us consider $(\vec{k}(\mathcal{I}), \beta_0^{-1}(\alpha_\omega^{-1}(\iota^{-1}(\zeta_0))))$. Sequence $\iota^{-1}(w)$ is still firable. Indeed, α_ω^{-1} consists in adding ω's which are not blocking for the firing of a transition and will stay unchanged through the firing of the sequence. Moreover β_0^{-1} adds some zeros in the places of $f^0(\mathcal{N})$ but no transitions of $f^0(\mathcal{N})$ are used in this sequence, thus no places of $f^0(\mathcal{N})$ are inputs to a transition of $\iota^{-1}(w)$. From those facts, we can conclude that the marking obtained in the coverability graph of $\vec{k}(\mathcal{I})$ after firing $\iota^{-1}(w)$ is exactly $\beta_0^{-1}(\alpha_\omega^{-1}(\iota^{-1}(\eta)))$.
Let us now focus on ζ_0. It can be divided into two parts, first the locks, and second the copies of the places of $\alpha(\mathcal{N})$. On this second part, it is exactly equal to $g(\alpha(y))$. Thus the marking $\beta_0^{-1}(\alpha_\omega^{-1}(\iota^{-1}(\zeta_0)))$ can be as well divided in two parts: the locks and the copies of the places of $\mathcal{N}$. On the first part, it is equal to ζ_0 completed with 0 in the initial lock. On the second part, it is exactly equal to $\beta_0^{-1}(\alpha_\omega^{-1}(\iota^{-1}(g(\alpha(y))))))$. Now, let $u = \beta_0^{-1}(\alpha_\omega^{-1}(\iota^{-1}(g(\alpha(y))))))$ and recall that $g \circ \alpha = \iota \circ \alpha \circ \beta \circ f$. We have:

$u = \beta_0^{-1}(\alpha_\omega^{-1}(\iota^{-1}(\iota(\alpha(\beta(f(y))))))))$	
$= \beta_0^{-1}(\alpha_\omega^{-1}(\alpha(\beta(f(y))))))))$	Because $\iota \circ \iota^{-1}$ is the identity function
$= \beta_0^{-1}((\beta(f(y))))))))$	Because $\alpha_\omega^{-1}(\alpha((x)) = x$ iff x equal ω on every place removed by α. Here $y_{\|\omega(z)} = \vec{\omega}$ thus $f(y)$ is equal to ω on every copy of $\omega(z)$.
$\leq f(y)$	Because $\beta_0^{-1}(\beta((x)) \leq x$ since β_0^{-1} replaces by 0 the values of the places affected by β.

Now, recall that f^0 is an isomorphism between $\mathcal{N}_p$ and $f^0(\mathcal{N})$. In $\mathcal{KM}(\mathcal{N}_p, m_0)$, there is a run to a marking z' greater than z since $z \in \mathcal{BCS}(\mathcal{N}_p, m_0)$. Therefore, there exists a run in $f^0(\mathcal{N})$ starting from $f^0(m_0)$ leading to a marking $f^0(z')$. This run can also be seen as a run w_1 of $\vec{k}(\mathcal{I})$, going from the initial marking μ_0 to a marking ψ_0 equal to $f(z')$ completed with zeros on the locks corresponding to the subnets $f^i(\mathcal{N})$ for $1 \leq i \leq |\sigma'|$ and 1 on the lock corresponding to $f^0(\mathcal{N})$. From this marking, we can fire $h^{-1}(\theta_1)$ since $f^0(\Pi(\theta_1)) \subseteq \omega(\psi_0)$ and thus have $\mu_0 \xrightarrow{w_1} \psi_0 \xrightarrow{h^{-1}(\theta_1)} \psi_1$ where ψ_1 can

be decomposed in two parts, first the locks, and second the copies of the places of $\mathcal{N}$. This second part is greater or equal to $f(y)$ by construction. Thus, $\psi_1 \geq u$. We can thus consider monotonicity, and conclude that $\iota^{-1}(w)$ is firable from ψ_1 such that in the coverability graph of $\vec{k}(\mathcal{I})$: $\psi_0 \xrightarrow{w_1} \psi_0 \xrightarrow{h^{-1}(\theta_1)} \psi_1 \xrightarrow{\iota^{-1}(w)} \psi_2$ where $\psi_2 \geq \iota^{-1}(\eta)$. Therefore, $f^{|\sigma'|}(X) \subseteq \omega(\psi_2)$ and $\big(\bigcup_{t_i \in \sigma} f^{i-1}(\Pi(t_i))\big) \subseteq \omega(\psi_2)$. Moreover, $f^0(\Pi(\theta_1)) \subseteq \omega(\psi_2)$ since $f^0(\Pi(\theta_1)) \subseteq \omega(\psi_0)$ and ψ_2 is a successor of ψ_0. Thus, since $\sigma' = \theta_1\sigma$ we obtain exactly that $f^{|\sigma'|}(X) \cup \big(\bigcup_{t_i \in \sigma'} f^{i-1}_{\mathcal{N}\to\mathcal{I}}(\Pi(t_i))\big) \subseteq \omega(\gamma)$ which concludes the induction.

(3) $\Rightarrow$ (1): Intuitively, the last copy $f^{|\sigma|}(\mathcal{N})$ of the incremental net $\mathcal{I}$ of $\mathcal{N}$ along σ is synchronised with every previous copy. So every sequence performed on $\mathcal{I}$ can be folded into the last copy of this net. Moreover, this last copy is isomorphic to a subnet of $\mathcal{N}$ itself. Therefore, the run will be firable in $\mathcal{N}$ itself. In the sequel, we omit the subscript $\mathcal{N} \to \mathcal{I}$ of the functions $f^i_{\mathcal{N}\to\mathcal{I}}$ and $h_{\mathcal{I}\to\mathcal{N}}$. We denote by P the places of $\mathcal{N}$.

More formally, we reason on the coverability graph and reuse the notations introduced in Section 5.6.2. From (3) we can deduce that, given $\vec{k}$, we can exhibit a run

$$\mu_0 = y_0 \xrightarrow{\pi_0} z_0 \xrightarrow{\tau_1} y_1 \xrightarrow{\pi_1} z_1 \xrightarrow{\tau_2} y_2 \xrightarrow{\pi_2} z_2 \dots z_{l-1} \xrightarrow{\tau_l} y_l \xrightarrow{\pi_l} z_l \text{ where } \begin{cases} \forall 1 \leq i \leq l,\ \tau_i \in h^{-1}(\Theta) \\ \forall 0 \leq i \leq l,\ \pi_i \in (f^i(T \cup \Theta))^* \\ f^l(X) \cup \big(\bigcup_{t_i \in \sigma} f^{i-1}(\Pi(t_i))\big) \subseteq \omega(z_l) \end{cases}$$

Moreover, by construction of the incremental net, the tokens in places $f^j(P)$ are created by transitions of the set $\cup_{i \leq j}(f^i(T))$. Since for each parametric transition θ_j, $f^{j-1}(\Pi(\theta_j))$ is equal to ω, the ω's of $f^{j-1}(\Pi(\theta_j))$ have to appear between π_0 and π_{j-1}. We can thus deduce that

$$\forall 1 \leq j \leq l,\ f^{j-1}(\Pi(\theta_j)) \subseteq \omega(z_{j-1})$$

Each time a new parametric transition is involved in the run, the input places of its parametric arcs already contain ω in this run in the coverability graph. Since $f^{|\sigma|}(\mathcal{N})$ is synchronised on all the previous copies to mimic their behaviour along the run, it is possible to simply consider the run folded into $f^{|\sigma|}(\mathcal{N})$:

$$f^l(m_0) = y_{0|f^l(P)} \xrightarrow{f^l(f^{-1}(\pi_0))} z_{0|f^l(P)} \xrightarrow{f^l(h(\tau_1))} y_{1|f^l(P)} \dots z_{l-1|f^l(P)} \xrightarrow{f^l(h(\tau_l))} y_{l|f^l(P)} \xrightarrow{f^l(f^{-1}(\pi_l))} z_{l|f^l(P)}$$

Where $\omega((z_l)_{|f^l(P)})$ contains $f^l(X)$ and, as explained above, before each first occurrence of a parametric transition, $f^l(h(\tau_i))$, $f^l(\Pi(h(\tau_i))) \subseteq \omega((z_{i-1})_{|f^l(\mathcal{N})})$.

Now let us recall that f^l is an isomorphism between a subnet of $\mathcal{N}$ and $f^l(\mathcal{N})$. Therefore, there exists an isomorphic run in the coverability graph of $(\vec{k}(\mathcal{N}), m_0)$ which provides that same set of places equal to ω. In particular, since X is included in this set, we can deduce that $\vec{k}(\mathcal{N})$ is simultaneous X unbounded. Moreover, by invoking Lemma 22, we can conclude that $\mathcal{N}$ is universally simultaneous X unbounded.

(3) $\Rightarrow$ (2): straightforward

(2) $\Rightarrow$ (3): We reuse here the decomposition used in (3) $\Rightarrow$ (1) but push it a bit further. Indeed, since the input places of parametric transitions are unbounded, it is sufficient to verify this property for only one instance of the incremental net. If such a property is verified for a $\vec{k}$-instance, then, it could be verified for any $\vec{k''}$-instance (with $k'' > k$) by exhibiting the witness run and performing more loops.

Formally, from (3) we know that, there exists $\vec{k}$, such that we can exhibit a run

$$\mu_0 = y_0 \xrightarrow{\pi_0} z_0 \xrightarrow{\tau_1} y_1 \xrightarrow{\pi_1} z_1 \xrightarrow{\tau_2} y_2 \xrightarrow{\pi_2} z_2 \dots z_{l-1} \xrightarrow{\tau_l} y_l \xrightarrow{\pi_l} z_l \text{ where } \begin{cases} \forall 1 \leq i \leq l,\ \tau_i \in h^{-1}(\Theta) \\ \forall 0 \leq i \leq l,\ \pi_i \in (f^i(T \cup \Theta))^* \\ f^l(X) \cup \big(\bigcup_{t_i \in \sigma} f^{i-1}(\Pi(t_i))\big) \subseteq \omega(z_l) \end{cases}$$

Moreover, as explained in (3) ⇒ (1), by construction of the incremental net, the tokens in places $f^j(P)$ are created by transitions of the set $\cup_{i\leq j}(f^i(T \cup \Theta))$, which was summed up as:

$$\forall 1 \leq j \leq l,\ f^{j-1}(\Pi(\theta_j)) \subseteq \omega(z_{j-1})$$

Each time a new parametric transition is involved in the run, the input places of its parametric arcs already contain ω in this run in the coverability graph (for the active copy and for all the following one, thus for all the input places of this transition if we consider the synchronisations). Therefore, if we consider the same run under a different valuation $k'' > k$, we can assert that there will be no difference for the coverability graph since the only effect of the valuation would be to consume more tokens from places that already contains ω. Using Lemma 9 we thus directly obtain that $\forall k'' > k$, $(\vec{k''}(\mathcal{I}), \mu_0)$ is simultaneous $f^l_{\mathcal{N}\to\mathcal{I}}(X) \cup \left(\bigcup_{t_i\in\sigma} f^{i-1}_{\mathcal{N}\to\mathcal{I}}(\Pi(t_i))\right)$ unbounded.

Finally, supposing that $\exists k \in \mathbb{N}$, $(\vec{k}(\mathcal{I}), \mu_0)$ is simultaneous $f^l_{\mathcal{N}\to\mathcal{I}}(X) \cup \left(\bigcup_{t_i\in\sigma} f^{i-1}_{\mathcal{N}\to\mathcal{I}}(\Pi(t_i))\right)$ unbounded, we can easily adapt the result of Lemma 15 to prove that $\forall k' \leq k$, $(\vec{k'}(\mathcal{I}), \mu_0)$ is simultaneous $f^l_{\mathcal{N}\to\mathcal{I}}(X) \cup \left(\bigcup_{t_i\in\sigma} f^{i-1}_{\mathcal{N}\to\mathcal{I}}(\Pi(t_i))\right)$ unbounded.

We have thus proved that (2) ⇒ (3). □

From Lemma 23 we can observe that answering the universal simultaneous X unboundedness on a preT-PPN can be reduced to guessing an element σ of the finite set $Pref(S_\Theta)$ such that $(\vec{0}(\mathcal{I}), \mu_0)$ is simultaneous Y unbounded. Since checking simultaneous X unboundedness can be done in ExpSpace as recalled in Theorem 7, we obtain a NExpSpace procedure. Then, a well-known theorem by Savitch [Savitch, 1970] stating that there is therefore an ExpSpace deterministic procedure answering this problem and Remark 4.2.1 allow us to deduce the following theorem:

Theorem 24
The universal simultaneous unboundedness problem for preT-PPNs is ExpSpace-complete.

Proof. The size of $\mathcal{N}$ is defined as usual, plus the number of parameters. The input consists of $\mathcal{N}$ plus a subset of its places X. The ExpSpace-hardness lower bound is a consequence of Remark 4.2.1.

We now establish the upper bound for the universal simultaneous unboundedness problem. Given a preT-PPN $\mathcal{N}$, we construct a non-deterministic Turing machine which guesses an element σ of $Pref(S_\Theta)$. Note that there are $2^{|\mathbb{P}|+1}$ elements possible in $Pref(S_\Theta)$.

We then ask whether given $\mathcal{I} = \texttt{incr}(\mathcal{N}, \sigma)$ and μ_0 its associated initial marking, $(\vec{0}(\mathcal{I}), \mu_0)$ is simultaneous Y unbounded, where $\mathcal{I}$ is the incremental net of $\mathcal{N}$ along σ with the initial marking associated μ_0, and Y is a set [13] of size at most polynomial in the size of the input.

Constructing the incremental net can be done in polynomial time because it consists only in filling matrices of size polynomial in the size of the input. It is therefore also only polynomially larger than $\mathcal{N}$.

Checking simultaneous X unboundedness can be done in ExpSpace as recalled in Theorem 7.

We thus obtain a NExpSpace procedure. By a well-known theorem by Savitch [Savitch, 1970], there is therefore an ExpSpace deterministic algorithm for the universal simultaneous unboundedness problem. This problem is thus ExpSpace-complete. □

Corollary 25
Universal coverability in preT-PPNs is ExpSpace-complete.

13. The exact formula of Y is given in Lemma 23.

5.7 Adapting Karp and Miller for PostT-PPNs

We saw in Section 3.3.2 of Chapter 3 that the Karp and Miller tree is one of the very first results in the scope of decidability in Petri nets. This construction permits to solve the problem of coverability and simultaneous unboundedness. Note that in our context, we cannot extend the proof for existential coverability using ω-PNs to existential simultaneous unboundedness. Indeed, there is a difference between places that are unbounded regardless to the valuation used and places that can store an arbitrarily large number of tokens through the use of arbitrarily large valuations of the parameters. Furthermore, in some sense, Karp and Miller's tree also provides some information about reachability (for the components that are bounded). Moreover, the target marking needs not be fully specified, some places can be left unconstrained while others can be imposed to be unbounded or to exceed a precise number of tokens. It seems thus natural to extend this construction to our subclasses.

5.7.1 An Extension of Karp and Miller Algorithm

Here, we want to adapt the Karp and Miller algorithm to provide some decidability results for existential coverability and existential simultaneous unboundedness. Formally:

Definition 37 : Existential Simultaneous Unboundedness
Given a parametric Petri net, and a subset of places X of this net, the parametric net is existential simultaneous X unbounded *iff there exists a valuation v of its parameters such that the v-instance of this net is simultaneous X unbounded.*

As presented above, our extension needs to deal with the fact that places where parameters can be generated can reach an arbitrary high value (by using an arbitrary large valuation), but are not necessary unbounded in a given instance. The following algorithm is different from the one introduced in [Geeraerts et al., 2015] adapted to ω-PNs which is based on an ω-semantics. In this latter algorithm, the ω's in the markings do not mean unboundedness (and thus non termination) and a stronger condition on the paths in the tree obtained is then needed to answer the termination. In fact if a marking of the coverability tree built contains an ω, this ω is either due to a classic acceleration operation, meaning that the corresponding place can store as many tokens as wanted through the repetition of a self increasing sequence or on the contrary, it is the result of the firing of a unique ω transition, which implies that a finite number of tokens have been generated in this place. The main limitation is thus that after several iterations of the algorithm, one cannot decide directly by looking at a marking which ω results from which operation. For instance, considering the simple postT-PPN depicted in Figure 5.6, we would obtain in the coverability tree a marking $(0, \omega)$ even if this net is clearly not existential p_2 unbounded. In fact, this net is universally bounded. Here, we try to represent directly in the

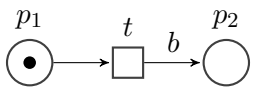


Figure 5.6 – A universally bounded postT-PPN

tree the difference that exists between a parametric number of tokens (that can be arbitrarily large) and an unbounded number of tokens. To model this duality in the *arbitrary large value* we suggest to use a special extension of $\mathbb{N}$. We use ω to represent unbounded markings whereas $*$ will be used to represent an arbitrary large value that is due to the effect of an arbitrary large valuation. Formally, we denote by $\mathbb{N}_*$ the union $\mathbb{N} \cup \{*\}$ and by $\mathbb{N}_{\omega,*}$ the union $\mathbb{N} \cup \{\omega\} \cup \{*\}$ such that :

- $\forall n \in \mathbb{N}, n < * < \omega$
- $\forall n \in \mathbb{N},\ * - n = *$ and $* + n = *$

- $\forall n \in \mathbb{N}_*,\ \omega - n = \omega$ and $\omega + n = \omega$

Note that there is no need to define $* - *$ or $\omega - \omega$ in our context. We can easily derive that similarly to $(\mathbb{N}_\omega, \leq)$, $(\mathbb{N}_{*,\omega}, \leq)$ is wqo. Therefore, using the component-wise order, $(\mathbb{N}^k_{*,\omega}, \leq)$ is wqo too. Given a vector (or a function) m taking values in $\mathbb{N}_{\omega,*}$, we denote by $\mathbb{N}_*(m)$ the set of components of m that belong to $\mathbb{N}_*$

First, let us notice that given a postT-PPN with a set of parameter $\mathbb{P}$, using monotonicity, one can trivially show that it is equivalent to answer the existential coverability if we consider the postT-PPN with the same structure but where all parameters have been set equal to the same parameter say a, thus reducing $\mathbb{P}$ to $\{a\}$. The intuition is to value all parameters to $*$, and if all the input places of a transition are marked by $*$ to propagate this symbol to the output places. Formally, given a marking m of $\mathbb{N}^P_{*,\omega}$ we define $\mathsf{Post}_*(m,t) \in \mathbb{N}^P_{*,\omega}$ as follow:

$$\mathsf{Post}_*(m,t)(p) = \begin{cases} m(p) + * & \text{if } \mathsf{Post}(p,t) \in \mathbb{P} \\ m(p) - \mathsf{Pre}(p,t) + \mathsf{Post}(p,t) & \text{otherwise} \end{cases}$$

Given a path in a tree $\mathscr{C}$ from a node n_1 to a node n_2, written $n_1 \rightsquigarrow n_2$, $\mathsf{effect}_{\mathscr{C}}(n_1 \rightsquigarrow n_2)$ is defined by the sum of the effect of the transitions along this path *i.e.* $\sum_{t \in \mathsf{pathlabel}(n_1,n_2)} \mathsf{Post}_*(t) - \mathsf{Pre}(t)$. We now adapt the acceleration function in a tree $\mathscr{C}$ to allow the propagation of $*$'s:

$$\mathsf{Acc}_{\mathscr{C}}(N,n)(p) = \begin{cases} \omega & \text{if } \exists n' \in N \left| \begin{array}{ll} & \Lambda(n') < \Lambda(n) \\ \wedge & \mathsf{effect}_{\mathscr{C}}(n' \rightsquigarrow n)_{|\mathbb{N}_*(n')} > 0 \\ \wedge & \Lambda(n')(p) < \Lambda(n)(p) < \omega \end{array} \right. \\ * & \text{otherwise and if } \exists n' \in N \left| \begin{array}{ll} & \Lambda(n') < \Lambda(n) \\ \wedge & \mathsf{effect}_{\mathscr{C}}(n' \rightsquigarrow n)_{|\mathbb{N}(n')} > 0 \\ \wedge & \Lambda(n')(p) < \Lambda(n)(p) < * \end{array} \right. \\ m(p) & \text{otherwise} \end{cases} \tag{5.2}$$

Algorithm 2 Karp and Miller for PostT-PPNs

Require: A postT-PPN $\mathcal{S} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P}, m_0)$
Return: A labelled tree $\mathscr{C} = (N, n_0, B, \Lambda)$
procedure
  Let n_0 be a new node such that $\Lambda(n_0) = m_0$
  $N \leftarrow n_0$; $Wait \leftarrow \{(n_0, t) | t \in T \wedge \Lambda(n_0) \geq \mathsf{Pre}(t)\}$; $B \leftarrow \emptyset$
  while $Wait \neq \emptyset$ **do**
    Pop(n',t) from *Wait*;
    $m \leftarrow \mathsf{Post}_*(\Lambda(n'), t)$;
    if $\nexists y \in \mathsf{Ancestor}_{\mathscr{C}}(n')$ s.t. $\Lambda(y) = m$ **then**
      Let n be a new node s.t. $\Lambda(n) = m$
      update $\Lambda(n) = \mathsf{Acc}_{\mathscr{C}}(\Lambda(\mathsf{Ancestor}_{\mathscr{C}}(n')), n))$
      $N \leftarrow N \cup \{n\}$; $B \leftarrow B \cup \{(n', t, n)\}$;
      $Wait \leftarrow Wait \cup \{(n, u) | u \in T \wedge \Lambda(n) \geq \mathsf{Pre}(u)\}$;
  return $\mathscr{C} = (N, n_0, B, \Lambda)$

The following proofs are very close to the original ones from Karp and Miller except for Theorem 28 where the construction of the valuation is a bit technical.

5.7.2 Termination

Theorem 26 : Termination
The procedure terminates and returns a finite tree.

Proof. We proceed by *reductio ad absurdum*. Note that we abuse the notations and we assimilate the label of a node to the node itself. The procedure does not terminate *iff* the *while loop* does not terminate. Moreover, since the procedure builds a tree, we can claim that no former nodes are added to the set of nodes added to the set $Wait$ during the construction, thus, the *while loop* does not terminate *iff* the set $Wait$ never becomes empty which directly implies that the tree built is infinite. Now, notice that this tree is finite branching: one node has at most $|T|$ successors. Using König's Lemma (see Lemma 4), we can infer that there exists an infinite path in this tree.

Now, we can invoke Lemma 1 on the infinite sequence of labels of the nodes on this path. We can thus extract an infinite subsequence (respecting the order of the infinite path) of markings of $\mathbb{N}_{*,\omega}$:

$$m_0 \leq m_1 \leq \cdots \leq m_i \leq \cdots \leq m_j \leq \dots$$

By construction, those markings are distinct otherwise, the exploration would have stopped.

$$m_0 < m_1 < \cdots < m_i < \cdots < m_j < \dots$$

Let us extract an infinite subsequence and prove that one ω or one $*$ is generated. On this sequence, there exists a pair m_i and m_j with $i < j$. Then, $\mathsf{effect}(m_i \rightsquigarrow m_j) \geq 0$ on the components in $\mathbb{N}$. Moreover, $\mathsf{effect}(m_i \rightsquigarrow m_j)$ can be negative on the components equal to ω or $*$. In any case, necessarily, one component of $\mathsf{effect}(m_i, m_j)$ must be strictly positive, otherwise, the markings m_i and m_j would be equal which is not possible. Thus if we consider the acceleration function, we are in the first or second situation, depending on where the negative components occurs. This implies that the acceleration create at least a $*$ or an ω.

Starting from m_j, the rest of the sequence is still infinite and we can repeat this reasoning. Since there is a finite number of components, and we will each time be able to add a symbol $*$ or ω to at least one component, this process will eventually reach a marking containing only $*$'s and ω's which contradicts the existence of the initial infinite sequence.

We can thus conclude that this tree is finite.

□

5.7.3 Correctness

Theorem 27 : Completeness
Given a marked postT-PPN $\mathcal{S}$ and a marking m, if there exists a valuation v such that m can be covered in $v(\mathcal{S})$, then, there exists a node n in the tree built by the procedure such that its label $\Lambda(n)$ is greater than or equal to m.

Proof. Let us suppose there exists a valuation v such that $m_0 \xrightarrow{w} m' \geq m$ in $v(\mathcal{S})$ with $w = t_1 t_2 \dots t_k$ and consider the corresponding (finite) sequence of markings.

First, find each occurrence of a parametric transition in w and replace their effect by $*$ in the sequence of markings. We will now apply to the newly obtained sequence of markings the following operation as many time as possible:

1. find the first element u' of the sequence such that for some earlier element u'', $u'' \leq u'$ with non negative effect on the components not equal to $*$ or ω.
2. if $u'' = u'$, delete all the elements following u'
3. otherwise, if the effect is positive on the components equal to $*$, replace the positive components by ω in u' and for each marking beyond u' in the sequence

4. otherwise, (there exists a component equal to $*$ where the effect is negative), replace the positive components by $*$ in u' and for each marking beyond u' in the sequence

This process builds up a sequence of markings that is exactly the sequence of labels of nodes from a path of the tree build through Algorithm 2. The final label in this sequence is a vector greater or equal to m' and thus to m.

□

Theorem 28 : Soundness

Given a postT-PPN $\mathcal{S}$, and the tree built by the procedure, given a node n of the tree, the following propositions hold:

- $\forall B \in \mathbb{N},\ \exists v \in \mathbb{N}^{\mathbb{P}},\ \exists m \in \mathcal{RS}(v(\mathcal{S}))$ *such that,* $\forall p$*:*
  $(\Lambda(n)(p) = \omega$ *or* $\Lambda(n)(p) = *) \Rightarrow m(p) \geq B$
  and $(\Lambda(n)(p) \notin \{*, \omega\}) \Rightarrow m(p) = \Lambda(n)(p)$
- $\exists v \in \mathbb{N}^{\mathbb{P}},\ \forall B \in \mathbb{N},\ \exists m \in \mathcal{RS}(v(\mathcal{S}))$ *such that,* $\forall p$*:*
  $(\Lambda(n)(p) = \omega) \Rightarrow m(p) \geq B$
  and $(\Lambda(n)(p) \notin \{*, \omega\}) \Rightarrow m(p) = \Lambda(n)(p)$

The difference between the first and second items of this theorem is the inversion of the universal and existential quantifiers targeting respectively a bound and a valuation. This claims that when the valuation is fixed, then the components that can reach $*$ through the algorithm are not unbounded. But on the other hand, if the valuation is not fixed, then it is possible to choose a valuation big enough to overpass any chosen bound B. A direct consequence of Theorem 27, Theorem 28 and Remark 4.2.1 is provided in the following corollary:

Corollary 29

The existential simultaneous unboundedness problem on postT-PPNs is ExpSpace*-hard.*

Proof of Theorem 28. Given any node n, and $B > 0$, we will build a valuation v and a marking m satisfying the first property. We will then prove the second one using that valuation.

There is a path in the tree defined by $n_0 \rightsquigarrow n$. We can assume that the first h components of $\Lambda(n)$ are equal to ω, the components from $h+1$ to k are equal to $*$ and the remaining $k+1$ to n belong to $\mathbb{N}$. Plus we assume that the ω's (and the $*$'s if possible) are generated by growing indices. Note it is possible that a $*$ becomes an ω along the procedure, therefore, the order of components imposed by ω's may not match with the order of appearance of the $*$'s. Nevertheless, at most k $*$'s are generated and h ω's along this path which is what mostly interests us.

$$\Lambda(n) = \begin{bmatrix} \omega \\ \vdots \\ \omega \\ * \\ \vdots \\ * \\ j_{k+1} \\ \vdots \\ j_n \end{bmatrix} \qquad m_{goal} = \begin{bmatrix} B \\ \vdots \\ B \\ B \\ \vdots \\ B \\ j_{k+1} \\ \vdots \\ j_n \end{bmatrix} \qquad m_{goal,l} = \begin{bmatrix} B+l \\ \vdots \\ B+l \\ B \\ \vdots \\ B \\ j_{k+1} \\ \vdots \\ j_n \end{bmatrix} \qquad \begin{matrix} p_1 \\ \vdots \\ p_h \\ p_{h+1} \\ \vdots \\ p_k \\ p_{k+1} \\ \vdots \\ p_n \end{matrix} \tag{5.3}$$

We denote by m_0 the initial marking and define m_{goal} as in Equation (5.3). Our goal is to find a valuation v such that there exists a marking $m \geq m_{goal}$ reachable in the v-instance. Then,

given any $l \geq 0$, we define $m_{goal,l}$ as in Equation (5.3). We will then explain why there exists a marking $m_l \geq m_{goal,l}$ reachable in the v-instance.

We define L as $|\mathsf{pathlabel}(n_0, n)|$ that is to say the length of this path (the number of arcs). Let M be equal to $\max_{p \in P}(m_{goal}(p))$ and R be the maximum absolute value of the integer weights involved in the net.

We now need to exhibit a valuation and a run in the Petri net leading to a marking satisfying the property. The main idea is to analyse the path in the tree in order to build a valuation and a sequence accordingly. Following Algorithm 2, if a new ω appears, it is necessarily the consequence of the acceleration function, we will thus need to estimate how many times the corresponding loop must be performed. Whereas, if a new $*$ appears, it can come from the firing of a parametric transition with the operator Post_*, which corresponds to the firing of a single parametric transition in the path or it can come from the acceleration function, which intuitively corresponds to the transfer of an arbitrary large (but finite) amount of tokens due to the value of the parameters. We will therefore need to estimate how many times those transfer sequences need to be fired to transfer those tokens.

1. First, we estimate the valuation by back propagation of the constraints induced by the transfer of the $*$ symbols.
2. Second we estimate the number of times each loop creating an ω (called ω loop in the sequel) must be performed. If we consider the i^{th} ω to appear, we denote this number by x_i in the sequel. Note that the resolution here is more complex than in the classic Karp and Miller proof, indeed, the number of times each loop is performed must consider some loop executions to compensate the tokens consumed during the $*$ transfer from the places already equal to ω. To this aim, we will use a simple but brutal over approximation.
3. Finally we provide the intuition of the construction of the final run.

Step 1:
In this step, we reorder the components by order of appearance of the $*$'s. We consider the case where only one $*$ is introduced at each time. If several $*$'s were to appear in one step, the reasoning would be similar. We decompose the path as follows:

$$n_0 \rightsquigarrow n_1 \rightsquigarrow \cdots \rightsquigarrow n_k \rightsquigarrow n$$

such that in n_1 the first $*$ is introduced and in n_k the last $*$ is introduced (we consider the "worst" case where k $*$'s are generated). Note that by definition, each of those paths has a length that is lower or equal to L. Let us now study the acceleration function defined by Equation (5.2). It is important to note that the repetition of ω loops will not affect the number of tokens generated by parametric arcs of the parameters since their effect must be non-negative on places marked by $*$. Thus, we keep the ω's as they are in those paths and make a first back propagation of constraints on components equal to $*$ to find a value for the valuation in order to instantiate those $*$'s.

Considering n_k, to reach a marking satisfying the property, it would be sufficient to instantiate each $*$ in this marking by $RL + M$ tokens. Let us recall how $*$ appears: either it is generated by the firing of one transition, otherwise, it comes from the acceleration.

- In the first case, we can choose directly the value necessary for the parameter, that is to say, the maximum between what is needed for this new component (*i.e.* $RL + M$ as explained above) and what would be needed for the other components.
- In the latter case, this implies that there is a sequence increasing with a negative effect on at least one component already equal to $*$ (otherwise, we would value it to ω through the acceleration function by definition). If some negative components are on the ω's this is not limiting since by construction it is sufficient to consider more executions of the corresponding loops, and it will not influence the valuation of the parameters. What matter here are thus the components with a negative effect on the $*$'s already created.

The valuation that we are going to build will consider the most expensive scenario in terms of consumption of tokens. A $*$ transfer sequence has a length bounded by L transitions and each transition can consume at most R tokens from any place, thus, if the latest $*$ introduced in n_k is a propagation of the $*$'s from n_{k-1}, it is sufficient to ensure that the $*$'s from n_{k-1} can effectively reach the value $(M + RL) + R^L(M + RL)$. Indeed $R^L(M + RL)$ tokens are used to generate $M + RL$ tokens through a run of length L consuming R tokens per transition. Plus we need to ensure that $M + RL$ tokens are still available after this transfer. We can iterate this process until we reach the first $*$ created. Let us denote by $(v_n)_{n\in\mathbb{N}}$ the arithmetico-geometric [14] sequence such that: $v_{n+1} = R^L v_n + (M + RL)$ with $v_0 = M + RL$. To get v_0 tokens in the $*$ from n_k, we need v_1 tokens in the $*$ from n_{k-1}. To get v_1 tokens in the $*$ from n_{k-1}, we need v_2 tokens in the $*$ from n_{k-2}, and so on. By finishing this reasoning, we need at most $V = \sum_{i=0}^{i=k-1} v_i$ tokens in the $*$ from n_1. Moreover, since this $*$ is the first to be introduced, it necessarily comes from the use of the operator Post_*. Using the first item stated previously, we can therefore consider the valuation $\vec{V}$. This valuation would cover the most greedy case, that is to say, all the $*$'s are created by transfer sequences which have negative components on all the previous $*$'s, such that those sequences are of length L and each transition involved consume R tokens from each places considered.

Step 2: We reorder the components by order of appearance of the ω's. Let us build from the original sequence a run and a marking satisfying the property by evaluating how much time it is necessary to perform the different loops in order to instantiate the ω's.

For the i^{th} ω, there is a subsequence such that its effect is positive on the component i and non negative on the other components, from $i + 1$ to n. The effect can be negative on the components already equal to ω (*i.e.* from components from p_1 to p_{i-1}). We consider $-\Delta$ the smallest negative components among the effect of all those subsequences. In fact Δ represent the maximum amount of token that a loop which is repeated to create an ω can consume per component at each occurrence of the loop. We can define the x_i's just like in the classic proof by Karp and Miller: each x_i represent the number of times the loop creating the i^{th} ω needs to be performed. The set $\{x_1, \dots x_h\}$ satisfies thus :

$$\begin{aligned} x_h &\geq M + RL \\ x_{h-1} &\geq M + RL + \Delta x_h \\ &\dots \\ x_1 &\geq M + RL + \Delta(x_h + \dots + x_2) \end{aligned}$$

Those x_i's are not sufficient however. Indeed, when we transfer a $*$ we may consume tokens from some places equal to ω's. Nevertheless, we know how many times each transfer sequence needs to be fired, using the previous arithmetico-geometric sequence. We know that a sequence creating a $*$ by acceleration will consume less than V tokens (this is clearly a brutal over approximation). So let us impose that each ω covers the needs from the previous inequation plus V. For the latest ω we need at least $(M + RL)$ for the remaining part after the $*$'s plus $(k - h)V$ tokens, for each of the stars that may appear after this ω and thus consume some of its tokens.

For the previous ω, (the one in component $h - 1$) the worst case would happen if in the h^{th} component an $*$ appears before to become an ω. Therefore, we need to compensate at most $k - h + 1$ $*$'s by acceleration. We thus need at least $(M + RL) + (k - h + 1)V$, plus we need to compensate the tokens consumed by the next omega, $\Delta \times n_k$ and so on. We will therefore add

14. Clearly if $R^L = 1$ it means that $R = 1$ or $L = 0$. If $L = 0$, $n = n_0$ thus there is no $*$'s nor ω's and the answer is direct. If $R = 1$, we obtain a classic arithmetic sequence of which generic term v_i can be written $(M+RL)+iR$. Else if $L > 0$ and $R > 1$, given an arithmetico-geometric sequence $(u_n)_{n\in\mathbb{N}}$ defined by $\forall n \in \mathbb{N},\ u_{n+1} = au_n + b$ with $a \neq 1$, we recall that if we write $r = \frac{b}{1-a}$, then $u_n = a^n(u_0 - r) + r$.

those constraints to the inequation system [15] of the second point to obtain a new system whose solutions are some x_i's.

$$\begin{aligned} x_h &\geq M + RL + (k-h)V \\ x_{h-1} &\geq M + RL + (k-h+1)V + \Delta x_h \\ &\dots \\ x_1 &\geq M + RL + (k-1)V + \Delta(x_h + \dots + x_2) \end{aligned}$$

Such a system can be solved due to its triangular form.

Step 3: Finally, let us put it all together. We first consider the Petri net $\vec{V}(\mathcal{S})$. Now we run through the path $n_0 \rightsquigarrow n$ and apply the following procedure: As long as no $*$ nor ω are introduced, keep the path as it is. When a $*$ is found, if it comes from the firing using Post_* keep the path as it is. Otherwise, if a $*$ is found, say the i^{th} to appear, go back to the beginning of the acceleration sequence, and repeat this sequence $\sum_{i=0}^{i=h-i} v_i$ times and continue to run through the path. Otherwise, if an ω is found, say the i^{th} to appear, go back to the beginning of the acceleration sequence, and repeat this sequence x_i times and continue to run through the path. By construction this run will lead to a marking greater than or equal to m_{goal} which prove the first property.

To prove the second property, using the initial discussion in this proof, it is sufficient to notice that the x_i are not bounded from above, therefore, it is possible to repeat them sufficiently to cover the l tokens that have been added. □

5.8 Adapting Karp and Miller for PreT-PPNs

We now consider the case of preT-PPNs. We provide an algorithm in Subsection 5.8.1. The proof of termination and correctness are provided in Subsections 5.8.2 and 5.8.3 respectively. Note that the algorithm presented is an adaptation of Karp and Miller algorithm [16].

5.8.1 An Extension of Karp and Miller Algorithm

The idea is to extend the construction to preT-PPN by introducing a special behaviour for parametric transitions. We provide Algorithm 3 which we call *extended Karp and Miller algorithm* or KM^+ for short and which allows us to fire parametric transitions once we verified that those transitions are firable for each instance of the parametric Petri net. This verification relies on an easy criterion: intuitively, a parametric transition will be universally firable if the input places of the parametric arcs are not bounded (formalised in Equation 5.4).

We apply the classic algorithm for non parametric transitions. The acceleration function Acc used in the algorithm (see Line 10) is thus a classic acceleration of which definition is recalled in Equation 3.2 of Section 3.3.2. For a parametric transition u, the firing condition (Equation 5.4 involved at line 12) is slightly different: we verify that the label of the node n covers the firing precondition (when the input arc are valued with natural numbers) and that the components of the label of n in a place corresponding to a parametric input arc are equal to ω. Formally :

$$P_{param}(n,u) = \forall p \in P,\ \begin{cases} \mathsf{Pre}(p,u) \in \mathbb{P} & \Rightarrow \Lambda(n)(p) = \omega \\ \mathsf{Pre}(p,u) \in \mathbb{N} & \Rightarrow \Lambda(n)(p) \geq \mathsf{Pre}(p,u) \end{cases} \tag{5.4}$$

Once such a transition is considered in the set $Wait$, we fire it by considering the zero valuation $\mathbf{0}$ as shown at line 8. This could be any other valuation since $\omega - x = \omega$ for any $x \in \mathbb{N}$.

15. Note that all those inequations are very pessimistic. One could suggest a more refined system where V would be replaced by some partial sum of the terms v_i depending on the number of $*$'s that have already appeared. For the sake of compactness and simplicity, we consider only this very rough system.

16. More precisely the version provided in the paper of Reynier and Servais [Reynier and Servais, 2011].

Algorithm 3 Universal Coverability Algorithm

Require: A preT-PPN $\mathcal{S} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$ and an initial marking m_0
Return: A labelled tree $\mathscr{C} = (N, n_0, B, \Lambda)$
procedure
  Let n_0 be a new node such that $\Lambda(n_0) = m_0$
  $N \leftarrow n_0$; $Wait \leftarrow \{(n_0, t) | \mathsf{Pre}(t) \in \mathbb{N}^P \wedge \Lambda(n_0) \geq \mathsf{Pre}(t)\}$; $B \leftarrow \emptyset$
  while $Wait \neq \emptyset$ **do**
    Pop(n',t) from *Wait*;
    $m \leftarrow \mathsf{Post}_{\mathbf{0}}(\Lambda(n'), t)$;
    if $\nexists y \in \mathsf{Ancestor}_{\mathscr{C}}(n')$ s.t. $\Lambda(y) = m$ **then**
      Let n be a new node s.t. $\Lambda(n) = \mathsf{Acc}(\Lambda(\mathsf{Ancestor}_{\mathscr{C}}(n'), m))$
      $N \leftarrow N \cup \{n\}$; $B \leftarrow B \cup \{(n', t, n)\}$;
      $Wait \leftarrow Wait \cup \{(n, u) | P_{param}(n, u)\}$;
  return $\mathscr{C} = (N, n_0, B, \Lambda)$

5.8.2 Termination

Theorem 30
For any preT-PPN $\mathcal{S}$, the tree $KM^+(\mathcal{S})$ is finite.

In the classic Karp and Miller algorithm applied to any v-instance of a preT-PPN, a parametric transition θ may be fired even if its input places for parametric arcs (we call this set of places $\Pi(\theta)$) are not marked with ω. Indeed, $v(\mathsf{Pre}(p, \theta))$ tokens in all places $p \in \Pi(\theta)$ are sufficient. In KM^+, a parametric transition needs a node labelled with a marking presenting ω in each input places of a parametric arc to be fired. Since this condition is more restrictive, it implies that if a parametric transition is fired in $KM^+(\mathcal{S})$, it would also be fired in $KM(v(\mathcal{S}))$. Nevertheless, if a parametric transition is fired in $KM(v(\mathcal{S}))$, it would be fired in $KM^+(\mathcal{S})$ only if it meets the previous condition. Therefore, $KM^+(\mathcal{S})$ paths are prefixes of paths of the tree build by the Karp and Miller procedure *i.e.* $KM^+(\mathcal{S})$ is obtained from $KM(v(\mathcal{S}))$ by cutting out the branches where parametric transition are fired before reaching an ω marking. The sequel provides a formal proof.

Lemma 31
Given a preT-PPN $\mathcal{S}$, for any v, there exists a prefix $\mathscr{P}$ of $KM(v(\mathcal{S}))$ such that $\mathscr{P}$ and $KM^+(\mathcal{S})$ are isomorphic.

Proof. Given a valuation v, we show that each path from the label of the root to the label of a node of $KM^+(\mathcal{S})$ is also a path from the label of the root to the label of a node of $KM(v(\mathcal{S}))$. Considering only those paths, we construct a subtree of $KM(v(\mathcal{S}))$ which would be the image $KM^+(\mathcal{S})$ by a tree isormorphism ϕ.

We will prove that for all n of $\mathbb{N}$ the following property holds:

$$\forall v \in \mathbb{N}^{\mathbb{P}}, \exists \mathscr{P} \text{ a prefix of } KM(v(\mathcal{S})), \exists \phi \text{ a tree isomorphism between } \mathscr{P} \text{ and } \mathscr{P}',$$
$$\text{where } \mathscr{P}' \text{ is the prefix of } KM^+(\mathcal{S}) \text{ composed of the nodes of depth at most } n$$

We proceed by induction on n and will consider a node η of $KM^+(\mathcal{S})$ of depth n in order to construct ϕ along the induction.

- *Base case:* we consider $n = 0$, the only node of depth 0 is thus the root which has the same label in both trees (the initial marking). ϕ maps the root of $KM^+(\mathcal{S})$ to the root $KM(v(\mathcal{S}))$. So the property is true.
- *Inductive step:* given $n \in \mathbb{N}$, we suppose the property holds. Thus, given a valuation v, there exists an isomorphism ϕ such that each path to the label of a node of $KM^+(\mathcal{S})$ of length n is also a path of $KM(v(\mathcal{S}))$. We consider a node η^* of depth $n+1$ in $KM^+(\mathcal{S})$.

We write t the last transition used in this path such that $\Lambda(\eta) \xrightarrow{t} \Lambda(\eta^*)$ where η is thus a node of depth n in $KM^+(\mathcal{S})$. Using the induction assumption on η, there exists a valuation v and an isomorphism ϕ such that the same path can be fired toward a node η' in $KM(v(\mathcal{S}))$ such that $\Lambda(\eta) = \Lambda(\eta')$ (with $\phi(\eta) = \eta'$). Now, let us consider t. Either t is a parametric transition or not.

- If t is not parametric, KM^+ mimics the classic algorithm. Thus applying the acceleration function, will create a node η'' with label $\Lambda(\eta^*)$ in $KM(v(\mathcal{S}))$. Therefore, considering the same valuation v, it is sufficient to extend ϕ as $\phi \cup (\eta^* \mapsto \eta'')$.
- If t is a parametric transition, we define $\Pi(t) = \{p \in P \mid \mathsf{Pre}(p,t) \in \mathbb{P}\}$. Line 12 of Algorithm 3 implies that $\Lambda(\eta)(p)$ equals ω for each $p \in \Pi(t)$. Moreover, for each $i \in \mathbb{N}, \omega - i = \omega$. Therefore $\Lambda(\eta) \xrightarrow{t}_0$ and $\Lambda(\eta') \xrightarrow{t}_v$ leads to the same marking m^* such that $m^*(p)$ equals ω if $p \in \Pi$, and $\Lambda(\eta)(p) - \mathsf{Pre}(p,t) + \mathsf{Post}(p,t)$ otherwise (or ω if there is a need to accelerate those components). The next steps of the two algorithms thus have the same behaviour: using the induction assumption, m^* is compared to the same ancestors in both trees. Therefore, the acceleration function provides the same result and a node η'' with label $\Lambda(\eta^*)$ will be created in $KM(v(\mathcal{S}))$ (this node will the the image of η^* by ϕ).

In both cases, the path from $\Lambda(\eta_0)$ to $\Lambda(\eta^*)$ studied in $KM^+(\mathcal{S})$ exists in $KM(v(\mathcal{S}))$. Since $KM^+(\mathcal{S})$ is finite branching, this can be repeated for each node of depth $n+1$ allowing us to extend the isomophism ϕ to each of those nodes. We can thus deduce that the property holds at rank $n+1$. The nodes obtained and the corresponding ancestors form a prefix of $KM(v(\mathcal{S}))$.

By induction we have proved Lemma 31 since the function ϕ constructed induces trivially a tree isomorphism between $KM^+(\mathcal{S})$ and the prefix of $KM(v(\mathcal{S}))$ considered. □

Considering Lemma 31 and the finiteness of the classic coverability tree of Karp and Miller proved in [Karp and Miller, 1969], the proof of Theorem 30 is immediate.

5.8.3 Correctness

First we need to introduce some results in order to provide an intuition for the proof. To determine the *universal coverability set*, we would like to compute the limit (if it exists) to the infinite intersection of coverability sets of every instance of the parametric Petri net. In this context, we first try to reduce the set of valuations on which this limit is computed. We already stated in Lemma 19 that it is sufficient for a marking to be coverable in infinitely many uniform instances.

We also already explained in a previous section the technique to answer universal coverability of a marking in preT-PPNs. Here, since we adapt Karp and Miller algorithm, in addition to the previous reasoning, we need to justify that every path studied in KM^+ can be instantiated in every instance of the parametric Petri net. We investigate a set of runs with good properties *i.e.* runs using the same support of parametric transitions with the same "slowest" parametric transition to appear (*i.e.* the same last first occurrence). Figure 5.8 on page 88 presents a set of runs w_i fired in different v_i-instances of a preT-PPN with $n+1$ parametric transitions (named t_{λ_1} to $t_{\lambda_{n+1}}$) such that each run w_i is equal to $x_{v_i} t_{\lambda_{n+1}} y_{v_i}$ where each t_{λ_j} is involved in x_{v_i} except $t_{\lambda_{n+1}}$. The construction of this set of runs is detailed in the sequel. The identical "slowest" parametric transition allows us to prove Lemma 32 by induction: if there is no parametric transition involved, then the result is obvious, as stated previously. Then, assuming that Lemma 32 is true for at most n parametric transitions and considering $n+1$ parametric transitions, we build a set of runs using those $n+1$ parametric transitions such that the induction hypothesis can be applied to the prefix x_{v_i} of each run. Even if the result may seem quite natural, the proof is a bit technical. It is entirely detailed at the end of this section.

To manipulate the parametric transitions occurring in a transitions sequence in the upcoming proofs, we introduce the notion of parametric support as follows:

Definition 38 : Parametric support
Given a PPN, we write Θ its set of parametric transitions. The parametric support of a transitions sequence w, written $\mathsf{Sup}_{\mathsf{par}}(w)$, is the set of parametric transitions involved in w.

$$\mathsf{Sup}_{\mathsf{par}}(w) = \{t \in \Theta \ s.t. \ |w|_t \geq 1\}$$

Lemma 32
Let $n \in \mathbb{N}$, $\mathcal{S} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P})$ a preT-PPN and a marking $m \in \mathbb{N}^P$. If $\mathcal{S}$ has at most n parametric transitions ($|\Theta| \leq n$) and for an infinite number of uniform valuations v, m is coverable in $v(\mathcal{S})$ then there is a vertex x in $KM^+(\mathcal{S})$ s.t. $m \leq \Lambda(x)$.
Moreover, if for an infinite number of uniform valuations v , we can find a set of firing sequences w_v leading to a marking covering m in the v-instantiated PPN, such that those sequences have the same parametric support, then, there is a path in the tree constructed by KM^+ from the root to the node, covering m, which involves the same parametric support.

This technical lemma together with Lemma 31 allows us to conclude to the correctness of the algorithm:

Corollary 33
The following propositions are equivalent :

1. *m is $\mathscr{U}$-coverable in $\mathcal{S}$*
2. *There is a node in $KM^+(\mathcal{S})$ of which label is greater than or equal to m*

Proof of Corollary 33. We prove the two implications:

- (1) $\Rightarrow$ (2):
  We consider a marking m universally coverable in $\mathcal{S}$. By Lemma 19 we can define an infinite set:

  $$\{v \in \mathbb{P}^{\mathbb{N}} | \exists k \in \mathbb{N} \ s.t. \ \forall \lambda \in \mathbb{P}, \ v(\lambda) = k \ \wedge \ m \ coverable \ in \ v(\mathcal{S})\}$$

  i.e. m is coverable for an infinite number of uniform valuations. By Lemma 32, there is a node in $KM^+(\mathcal{S})$ such that its label is greater than or equal to m.
- (2) $\Rightarrow$ (1): Reciprocally,
  Let us suppose that there exists a node n in $KM^+(\mathcal{S})$ such that its label is greater than or equal to m. By Lemma 31, for any valuation v, m is coverable in $KM(v(\mathcal{S}))$. The correctness of Karp and Miller Algorithm provides that m is coverable in any instance of $\mathcal{S}$. So m is universally coverable in $\mathcal{S}$.

Thus we have proved that the two properties are equivalent. □

Proof of Lemma 32. We prove by induction that for all $n \in \mathbb{N}$ this implication is true.

Base case: n=0

Let $\mathcal{S}$ be a classic Petri net *i.e.* $\mathcal{S}$ has no parametric transition. Supposing that m is coverable in an infinite number of instances of $\mathcal{S}$ means m is coverable in $\mathcal{S}$ itself. Let us build the KM^+ tree of $\mathcal{S}$. Since there is no parametric transition, our algorithm builds the classic coverability

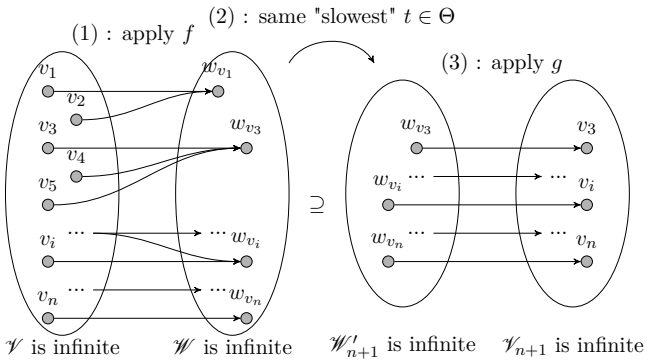


Figure 5.7 – The different sets of the proof

tree of Karp and Miller. Therefore, by the correctness of Karp and Miller algorithm, a node of the tree will be labelled by a marking covering m. Moreover, since no parametric transition is involved, the second point of the property clearly holds. So the implication holds for $n = 0$.

Inductive step: given $n \in \mathbb{N}$, we assume the implication holds at this rank.

Let us consider $\mathcal{S}$ a preT-PPN such that $|\Theta| \leq n + 1$. Let m be $\mathscr{U}$*-cov* in $\mathcal{S}$. We define the set of uniform valuations $\mathscr{V}$ such that for any $v \in \mathscr{V}$, m is coverable in $v(\mathcal{S})$. By Lemma 19 $\mathscr{V}$ is infinite.
If $|\Theta| \leq n$, then we can apply the induction assumption and conclude that the implication holds. Otherwise, $|\Theta| = n+1$. For each valuation v we consider an application $f : \mathbb{N}^{\mathbb{P}} \mapsto T^*$. Application f associates to each valuation a unique transitions sequence w_v such that $m_0 \xrightarrow{w_v}_v m_v$ where $m_v \geq m$ and $|\mathsf{Sup}_{\mathsf{par}}(w_v)|$, the number of distinct parametric transitions used in w_v, is minimal. In the sequel, we assimilate w_v to $f(v)$. Figure 5.7, presents a summary of the different sets used in this proof. We are currently studying step (1). We consider $\mathscr{W}$ the union of those sequences:

$$\mathscr{W} = \bigcup_{v \in \mathscr{V}} w_v$$

We have to consider two cases: either $\mathscr{W}$ is finite or infinite.

First case: $\mathscr{W}$ is finite. We will exhibit a sequence which works for each valuation. Since each sequence w_v of $\mathscr{W}$ is finite, it can be interpreted as a finite list of markings of $v(\mathcal{S})$, written $\mathcal{RS}_v(w_v)$ *i.e.* the markings reached along the firing of w_v in $v(\mathcal{S})$. For each w_v we can therefore build a maximal marking, m_{max,w_v} of which components are equal to the maximal components of $\mathcal{RS}_v(w_v)$: given p, $m_{max,w_v}(p) = \max_{m \in \mathcal{RS}(w_v)}(m(p))$. We can then define a maximal value a among the components of those markings : $a = 1 + \max_{p,w_v}(m_{max,w_v}(p))$. It represents a quantity of tokens that any marking of $\mathcal{RS}_v(w_v)$ for any w_v of $\mathscr{W}$ cannot store in any place. We define a family of $|P|$ markings $m_i = a.\delta_{i,j}$ for $1 \leq i, j \leq |P|$ where $\delta_{i,j}$ is the delta of Kronecker [17]. None of those markings is coverable by a marking of any $\mathcal{RS}_v(w_v)$. Nevertheless, $\mathscr{V}$ is infinite, so we can consider a valuation $v^* \geq a$ in $\mathscr{V}$. We know by construction that w_{v^*} allows to cover m in $v^*(\mathcal{S})$ but this run cannot use any parametric transitions, since the m_i's are not coverable by construction of v^*, so $\mathsf{Sup}_{\mathsf{par}}(w_{v^*}) = \emptyset$. Therefore, w_{v^*} is firable in any instance of $\mathcal{S}$. And in particular, if we consider $\mathcal{S}' = (P, T \backslash \Theta, \mathsf{Pre}, \mathsf{Post})$, m remains coverable. We found a PPN (in fact a classic PN) with less than n parametric transitions where m is universally coverable. We can therefore apply the induction assumption and conclude that the implication holds in this case.

17. By definition $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.

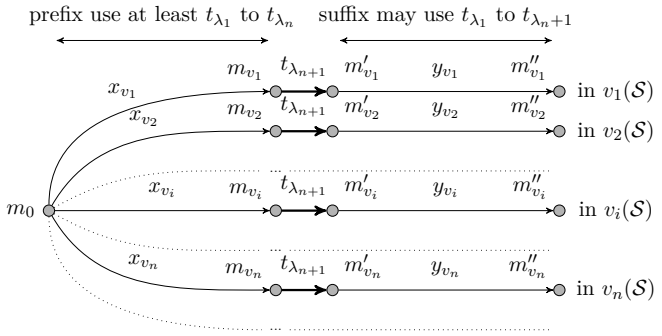


Figure 5.8 – w_{v_l} covering m for the valuations of $\mathscr{V}_{n+1,n+1}$

Second case: $\mathscr{W}$ is infinite. We will exhibit a pattern which works for an infinity of valuations. First, we isolate a set of runs which use the same parametric transitions. By the previous case, we can claim that there is no w_v with 0 parametric transition otherwise $\mathscr{W}$ would be reduced to this run (since w_v are chosen so that $|\mathsf{Sup}_{\mathsf{par}}(w_v)|$ is minimal). We can assign to each w_v the cardinal of its parametric support within 1 and n+1. Using the infinite extension of the pigeonhole principle, there is a pigeonhole with an infinite number of representative *i.e.*

$$\exists\ j,\ 1 \leq j \leq n+1 \textit{ such that } \mathscr{W}_j = \bigcup_{v \in \mathscr{V}\, s.t. |\mathsf{Sup}_{\mathsf{par}}(w_v)|=j} w_v \text{ is infinite.}$$

We now consider the amount j of different parametric transitions involved. We study two subcases: either $j \leq n$ or $j = n+1$.

If $j \leq n$, we can remove the unused parametric transition from Θ. We construct a parametric PPN which meets the induction assumption and the implication is true.

Let us consider $j = n+1$ *i.e.* there is an infinity of runs, corresponding to an infinity of uniform valuations using necessarily every parametric transitions to cover m.
We can now assign to each run of $\mathscr{W}_{n+1}$ the label of the parametric transition of which the first occurrence is the last one among the other parametric transitions involved along the runs. By the pigeonhole we find a set $\mathscr{W}'_{n+1}$ which is infinite. The sequences of $\mathscr{W}'_{n+1}$ have the same parametric support and the last first occurrence of a parametric transition targets the same parametric transition. Step (2) of Figure 5.7 illustrates the sets. We rename this parametric transition $t_{\lambda_{n+1}}$. We rename the other parametric transitions of Θ as t_{λ_h} where $1 \leq h \leq n$. For any run of $\mathscr{W}'_{n+1}$, there is a uniform valuation associated. We define the set $\mathscr{V}_{n+1} = g(\mathscr{W}'_{n+1})$ where

$$\begin{array}{rcccl} g & : & \mathscr{W} & \to & \mathscr{V} \\ & & w & \mapsto & v \quad \textit{such that } f(v) = w \\ & & & & \quad \textit{and for all } v' \textit{ such that } f(v') = w,\ v \leq v' \end{array}$$

In fact, we take the lowest valuation for each run as in Step (3) of Figure 5.7. Since $\mathscr{W}'_{n+1}$ is infinite, $\mathscr{V}_{n+1}$ is infinite. We rename the elements of $\mathscr{V}_{n+1}$ as v_l. We can impose for $l < k$, $v_l < v_k$ since we have a total order on distinct elements and the v_l are distinct. Each run w_{v_l} uses the $n+1$ parametric transitions. Moreover, $t_{\lambda_{n+1}}$ is the last parametric transition that is used for the first time.

We consider the set of places $\Pi(t_{\lambda_{n+1}}) = \{p \in P \mid \mathsf{Pre}(p, t_{\lambda_{n+1}}) \in \mathbb{P}\}$. So for any $p \in \Pi(t_{\lambda_{n+1}})$, $v_1(\mathsf{Pre}(p, t_{\lambda_{n+1}})) < v_2(\mathsf{Pre}(p, t_{\lambda_{n+1}})) < ...$ For each w_{v_l} we define the prefix x_{v_l} and the suffix y_{v_l} such that $w_{v_i} = x_{v_l} t_{\lambda_{n+1}} y_{v_l}$ and $t_{\lambda_{n+1}}$ is not involved in x_{v_l}. We define m_{v_l} such that $m_0 \overset{x_{v_l}}{\to}_{v_l} m_{v_l}$ as depicted in Figure 5.8. Moreover $m_{v_l} \overset{t_{\lambda_{n+1}}}{\to}_{v_l} m'_{v_l}$ and $m'_{v_l} \overset{y_{v_l}}{\to}_{v_l} m''_{v_l}$.

$$\text{Let } \mathscr{M} = \bigcup_{v_l \in \mathscr{V}_{n+1}} m_{v_l}$$

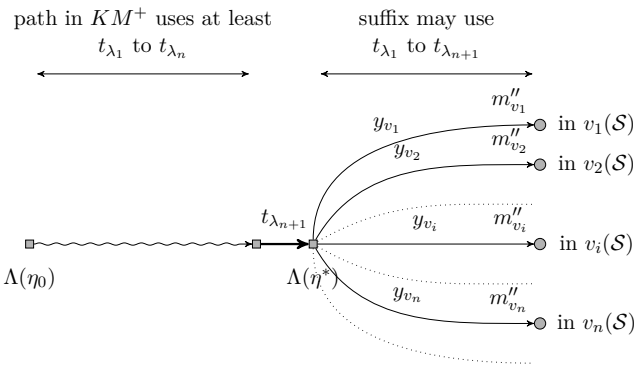


Figure 5.9 – Construction of the node η^*

If $\mathscr{M}$ is finite, each component of each marking is bounded by a value b where $b = 1 + \max_{m \in \mathscr{M},\ p \in P} m(p)$. Since $\mathscr{V}_{n+1}$ is infinite, we can find $v_b \in \mathscr{V}_{n+1}$ such that $v_b \geq b$. For this valuation v_b, $t_{\lambda_{n+1}}$ is not firable in $v_b(\mathcal{S})$ as each component of the markings of $\mathscr{M}$ is bounded by b. So there is a contradiction. Thus $\mathscr{M}$ is infinite.

Since we have a wqo on markings, by Lemma 1, we can extract an infinite growing sub sequence $\mu_1 \leq \mu_2 \leq ... \leq \mu_k \leq ...$ from $\mathscr{M}$ where each μ_k is in fact a m_{v_l}. From this sequence, we extract a strictly increasing subsequence: $\mu_{i_1} < \mu_{i_2} < ... < \mu_{i_k} < ...$ Such an infinite subsequence exists, otherwise, it would imply that infinitely many μ_k are equal which would mean that, at a certain point, $(\mu_k)_{k \in \mathbb{N}}$ is uniform, equal to μ^*. Since the μ_k correspond to different (and thus strictly increasing) valuations, and by definition, μ_k must satisfy the firing of $t_{\lambda_{n+1}}$, μ^* should satisfy the firing of $t_{\lambda_{n+1}}$ for an infinity of increasing valuations, which is impossible.

We now extract from this sequence a subsequence such that the strictly increasing μ correspond to ordered uniform valuations. Let us consider $\mu'_{i_1} = \mu_{i_1}$ (we denote as v_{l_1} the valuation used for the run leading to μ'_{i_1}), there is an infinity of μ_{i_k} greater. Among this infinite set, we look for μ'_{i_2} of which the corresponding valuation is greater than v_{l_1}. Such a valuation exists otherwise we would construct an infinite subset of $\mathscr{V}_{n+1}$ which would be bounded by v_{l_1}. Then we start back with the μ_{i_k} chosen. We end up with a sequence $\mu'_{i_1} < \mu'_{i_2} < ... < \mu'_{i_k} < ...$ corresponding to $v_{l_1} < v_{l_2} < ... < v_{l_k} < ....$ Those valuations define a new infinite set $\mathscr{V}^*$. Moreover, for each $p \in \Pi(t_{\lambda_{n+1}})$, $\mu'_{i_1}(p) < \mu_{i_2}(p) < ... < \mu'_{i_k}(p) < ....$ Indeed, let us imagine that $\mu'_{i_k}(p)$ for $p \in \Pi(t_{\lambda_{n+1}})$ is bounded. At a given rank r, it would reach a uniform value. Nevertheless, it corresponds to a uniform valuation v_{l^*}. So the corresponding sequence of valuations is bounded on $v_{l^*}(p)$ which is not possible since the v_{l_i} are strictly increasing.

The marking μ'_{i_1} is the first element, so it is coverable in an infinity of v-instance where $v \in \mathscr{V}^*$. Moreover by construction the prefix of the corresponding w_v uses the same n parametric transitions. By the induction assumption, there is a node η_1 in $KM^+(\mathcal{S})$ such that $\Lambda(\eta_1) \geq \mu'_{i_1}$ and a path to this node which uses those n parametric transitions.

This also holds for each μ'_{i_k}. By Theorem 30, $KM^+(\mathcal{S})$ is finite, so one node covers infinitely many μ'_{i_k}. Moreover, since the μ'_{i_k} are growing on the component corresponding to the parametric input of $t_{\lambda_{n+1}}$, the corresponding node present an ω on those components. Now, in KM^+ algorithm, we can fire $t_{\lambda_{n+1}}$. It does not affect the ω as we see at line 8 of the Algorithm 3. The node obtained is written η^*. This construction is shown in Figure 5.9.

By construction $\Lambda(\eta^*)$ covers all the m'_v with $v \in \mathscr{V}^*$ and the path in the KM^+ tree uses every parametric transition, then $\Lambda(\eta^*)$ has ω's on each place which is an input of a parametric arc. More formally, for each p, if there exists $t \in \Theta$ s.t. $\mathsf{Pre}(p,t) \in \mathbb{P}$, then $\eta^*(p) = \omega$. Therefore, from η^* any parametric transition would be firable in KM^+ (as long as the non-parametric part of the firing condition is satisfied).

Let us consider a suffix of a w_v selected *i.e.* $y_{v_{l_1}}$ for instance. By construction, $\Lambda(\eta^*) \geq m'_{v_{l_1}}$.

So $\Lambda(\eta^*)$ allows to fire $y_{v_{l_1}}$ in the algorithm. We can invoke the correctness of Karp and Miller algorithm to conclude. Since we start from a marking covering $m'_{v_{l_1}}$ and fire a sequence covering m, it leads to construct a node of which label covers m. Moreover, the path in the tree constructed by KM^+ uses every parametric transition by construction. So the implication holds at rank $n+1$.

In every sub-case considered, we conclude that the implication holds, therefore considering the implication true at rank n, we proved that the implication holds at rank $n+1$.

Conclusion:
By induction we have proved that for any value of $n \in \mathbb{N}$ the implication is true, so Lemma 32 is true. □

5.9 Consequences on P-PPNs and DistinctT-PPNs

From the results underlined for postT-PPNs, using the translation provided in Section 5.2 we can directly deduce the following corollary:

Corollary 34
$\mathscr{E}$-coverability in P-PPN is ExpSpace-complete.
$\mathscr{U}$-coverability in P-PPN is ExpSpace-complete.

The previous study can also be used for distinctT-PPNs. Parameters can be partitioned between $\mathbb{P}_{Pre}$ and $\mathbb{P}_{Post}$, respectively sets of parameters involved on inputs and outputs which are disjoint.

Corollary 35
$\mathscr{E}$-coverability in distinctT-PPN is ExpSpace-complete.
$\mathscr{U}$-coverability in distinctT-PPN is ExpSpace-complete.

Proof. We prove the two items:

- Given a distinctT-PPN $\mathcal{S}$ and a set of parameter $\mathbb{P}$, we consider the partial valuation $\vec{0}_{|\mathbb{P}_{Pre}}$, which maps every parameter of $\mathbb{P}_{Pre}$ to 0. We therefore get a postT-PPN, $\mathcal{S}_{\mathsf{post}} = \vec{0}_{|\mathbb{P}_{Pre}}(\mathcal{S})$, on which the problem is decidable. Moreover, the postT-PPN $\mathcal{S}_{\mathsf{post}}$ is the one with the greatest number of behaviours for coverability as claimed in Lemma 15. Considering that point, if we cannot find any instance of $\mathcal{S}_{\mathsf{post}}$ satisfying the property, we cannot find any instance of the original distinctT-PPN $\mathcal{S}$ satisfying it either since any marking coverable in the postT-PPN where the "input parameters" values are greater than $\vec{0}$ would be coverable in $\mathcal{S}_{\mathsf{post}}$ where the "input parameters" were valuated to 0 by monotonicity. Reciprocally, if we find an instance that works, then the corresponding valuation completed with 0 on $\mathbb{P}_{Pre}$ is a witness valuation for coverability. Thus, $\mathscr{E}$-coverability for the distinctT-PPN is decidable.

  Moreover, it can be done by forcing every parameters on input arcs to zero (linear in the size of the PPN) and solving $\mathscr{E}$-coverability on the corresponding postT-PPN. Thus, this problem is in ExpSpace. As previously, from Remark 4.2.1 we deduce the ExpSpace-hardness. This problem is thus ExpSpace-complete.
- The proof is very similar to the previous one. We consider the partial valuation $\vec{0}_{|\mathbb{P}_{Post}}$, which maps every parameter of $\mathbb{P}_{Post}$ to 0. The preT-PPN built is the one with the smallest number of behaviours for coverability by adapting the result of Lemma 15. Moreover, this problem is known decidable on preT-PPN. Considering that point, given a marking m :
  - if the answer on this preT-PPN is false, then m is not universally coverable on this distinctT-PPN.

- else if the answer is yes, adapting Lemma 15 claims that every instances will a behaviour covering m.

Thus $\mathscr{U}$-coverability for the distinctT-PPN is decidable. Moreover, it can be done by forcing every parameters on output arcs to zero (linear in the size of the PPN) and solving $\mathscr{U}$-coverability on the corresponding preT-PPN. Thus, this problem is in ExpSpace. As previously, from Remark 4.2.1 we deduce the ExpSpace-hardness. This problem is thus ExpSpace-complete.

□

We have restricted the use of discrete parameters depending on whether we parameterise input arcs or output arcs. The study of the decidability of the parametric coverability leads to the results summed up in Table 5.1 where the complexities of the corresponding problems are recalled. We proved that all these decision problems are ExpSpace-complete, and even provide the ExpSpace-completeness of the universal simultaneous unboundedness for the class of preT-PPNs. Moreover, putting the two types of parameters together while forbidding any parameter to be used as both an input and output weight preserves the decidability of the emptiness and the universality of the solution set. In the next chapter, we turn our interest towards the problem of coverability synthesis.

	$\mathscr{U}$-Cov	$\mathscr{E}$-Cov
PPN	Undecidable *(Theorem 12)*	Undecidable *(Theorem 11)*
T-PPN	Undecidable *(Th. 12 & Lem. 13)*	Undecidable *(Th. 11 & Lem. 13)*
preT-PPN	ExpSpace-c *(Corollary 25)*	ExpSpace-c *(Corollary 16)*
postT-PPN	ExpSpace-c *(Corollary 16)*	ExpSpace-c *(Theorem 18)*
P-PPN	ExpSpace-c *(Corollary 34)*	ExpSpace-c *(Corollary 34)*
distinctT-PPN	ExpSpace-c *(Corollary 35)*	ExpSpace-c *(Corollary 35)*

Table 5.1 – Global summary for the decidability of parametric coverability (with complexities)

CHAPTER 6

Solving the Synthesis

"Success is a science; if you have the conditions, you get the result."

— Oscar Wilde

Contents

In this chapter, we address the *coverability synthesis problem*. When we restrict the use of parameters to input arcs only, or output arcs only, we give some structure to the solution set of the *coverability synthesis problem*: we prove that it is then downward-closed wrt. the usual order on integer vectors for preT-PPNs, and upward-closed for postT-PPNs. We show how a procedure by Valk and Jantzen from [Valk and Jantzen, 1985] can be used for computing a finite minimal basis of the solution set for postT-PPNs or its complement for preT-PPNs. This requires deciding universal coverability in preT-PPNs and existential coverability in postT-PPNs. We also provide an algorithm computing directly a minimal basis of the solution set for preT-PPNs. Finally, we prove for distinctT-PPNs that the solution set cannot be represented using any formalism for which the emptiness of the intersection with equality constraints is decidable.

6.1 Special Structure of the Coverability Synthesis Set for PreT-PPNs and PostT-PPNs

Note that the monotonicity properties presented in Lemma 15 directly provide a notable structure for the solution set of the synthesis for those two subclasses presented in Corollary 36. When we restrict the use of parameters to input arcs, we ensure that any marking coverable in a v-instance remains coverable for any v'-instance such that $v' \leq v$. Intuitively, decreasing the valuation leads to a more permissive firing condition. Therefore, if a marking is coverable for a given valuation, it remains coverable for any lower valuation. Symmetrically, when we restrict

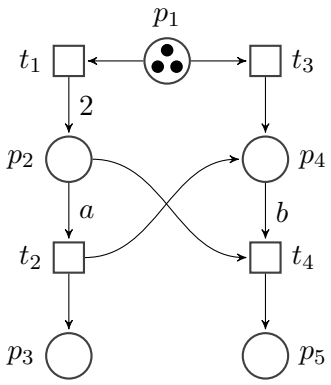


Figure 6.1 – A preT-PPN $\mathcal{S}_1$ where $\mathbb{P} = \{a, b\}$ with two parametric arcs

the use of parameters to output arcs, we ensure that any marking coverable in a v-instance, remains coverable for any v'-instance such that $v \leq v'$. Intuitively, firing the same parametric transition while increasing the valuation leads to greater (and thus more permissive) markings. Therefore, if a marking is coverable for a given valuation, it remains coverable for any greater valuation. We recall that $\mathcal{CV}(\mathcal{S}, m,)$ denotes the set of valuations that are solutions to the coverability synthesis problem of the marking m in the PPN $\mathcal{S}$.

This can be formalised through the following corollary of Lemma 15:

Corollary 36
Given $\mathcal{S}_{pre}$, $\mathcal{S}_{post}$ a marked preT-PPN and a marked postT-PPN respectively and goal markings m and s for each of those nets,

- *$\mathcal{CV}(\mathcal{S}_{pre}, m)$ is downward closed.*
- *$\mathcal{CV}(\mathcal{S}_{post}, s)$ is upward closed.*

Given a preT-PPN $\mathcal{S}$ and a marking m, one way to compute $\mathcal{CV}(\mathcal{S}, m)$ is thus to find its finite minimal basis. A naive enumeration is not possible however since this set may be infinite. In particular, the strategy that consists in testing for universality and, in the negative case, enumerating until a witness of non coverability is found would in general provide only a subset of $\mathcal{CV}(\mathcal{S}, m)$. In fact, the main difficulty here resides in the fact that the elements of the minimal basis have to be found among the complete lattice induced by $\leq$ on $\mathbb{N}_\omega^{\mathbb{P}}$. We now suggest to use a well known procedure by Valk and Jantzen [Valk and Jantzen, 1985] which aims to compute a finite basis of a given upward closed set.

6.2 Reduction of Valk and Jantzen Condition for PreT-PPNs and PostT-PPNs

In order to represent a finite basis of a downward closed set of valuations, we need to extend valuations to $\mathbb{N}_\omega$. Given a preT-PPN and an extended valuation v, we extend the predicate $\texttt{cov}(v(\mathcal{S}), m)$ to extended valuations as follows: $\texttt{cov}(v(\mathcal{S}), m) \overset{def}{\Leftrightarrow} \forall v' \in \mathbb{N}^{\omega(v)}$, $\texttt{cov}(v'(v_{|\mathbb{N}(v)}(\mathcal{S})), m)$.

Figure 6.1 presents a preT-PPN with two parameters a and b. If we consider the valuation v defined by $v(a) = 1$ and $v(b) = \omega$, $\texttt{cov}(v(\mathcal{S}), m)$ is therefore equivalent to the universal coverability of m in $v_{|\{a\}}(\mathcal{S})$ where $v_{|\{a\}}$ is a valuation defined by $v(a) = 1$, that is to say "can we cover m in $\vec{1}_{|\{a\}}(\mathcal{S})$ for any value of b ?". Note that this extension of $\texttt{cov}(v(\mathcal{S}), m)$ is consistent with the classic behaviour: if $\mathbb{N}(v) = \mathbb{P}$, then $\texttt{cov}(v(\mathcal{S}), m)$ asks the coverability of m in the marked Petri Net $v(\mathcal{S})$.

We recall that in postT-PPNs, universal coverability is true iff $\texttt{cov}(\vec{0}(\mathcal{S}), m)$. In a similar

manner to preT-PPN, we extend the notation of $\texttt{cov}$, by: given a postT-PPN and an extended valuation v, we extend the predicate $\texttt{cov}(v(\mathcal{S}), m)$ to extended valuations as follows: $\neg\texttt{cov}(v(\mathcal{S}), m) \overset{def}{\Leftrightarrow} \forall v' \in \mathbb{N}^{\omega(v)}, \neg\texttt{cov}(v'(v_{|\mathbb{N}(v)}(\mathcal{S})), m)$.

This definition is similar to the definition extended for preT-PPNs where coverability has been replaced by non-coverability. Since universal coverability is decidable for preT-PPNs (see Theorem 21) and existential coverability is decidable for postT-PPNs (see Theorem 18), it is possible to extend the function $\texttt{cov}$ defined in Section 3.2 to the context of extended valuations of $\mathbb{N}_\omega^{\mathbb{P}}$:

Corollary 37
Given a preT-PPN or a postT-PPN $\mathcal{S}$, a marking m and an extended valuation $v \in \mathbb{N}_\omega^{\mathbb{P}}$, $\texttt{cov}(v(\mathcal{S}), m)$ is decidable.

With those extended notations, we now prove that it is possible to compute a finite basis of $\mathcal{CV}(\mathcal{S}, m)$ where $\mathcal{S}$ is a preT-PPN or a postT-PPN and m a goal marking. To this end, we suggest to use an algorithm by Valk and Jantzen [Valk and Jantzen, 1985] to compute a finite representation of those sets. Nevertheless, to ensure that this algorithm is suitable to our context and that those bases are effectively computable, we need to clarify two points:

- First, this algorithm is used to compute a basis of an upward closed set $U \subseteq \mathbb{N}^k$.
- Second, the necessary and sufficient condition recalled in Lemma 3 must be satisfied: we must ensure that for every element $v \in \mathbb{N}_\omega^k$, it is possible to decide whether the intersection between $\downarrow v$ and U is empty.

To address the first point, by Corollary 36 notice that in the case of postT-PPNs, the set $\mathcal{CV}(\mathcal{S}, m)$ is upward closed, and the procedure could be applied directly on it. In the case of preT-PPNs, since $\mathcal{CV}(\mathcal{S}, m)$ is downward closed, we need to consider its complement $\neg\mathcal{CV}(\mathcal{S}, m)$, which is thus upward closed.We also recalled in Section 2.1.3 that, given a finite basis of an upward closed set, it is possible to compute a finite basis of its complement. It is therefore equivalent to being able to compute a finite basis of the set of valuations for which it is not possible to cover m or to being able to compute a finite basis of the set of valuations for which it is possible to cover m. Thus, a finite basis of $\neg\mathcal{CV}(\mathcal{S}, m)$ is effectively computable *iff* a finite basis of $\mathcal{CV}(\mathcal{S}, m)$ is computable. Considering this reasoning, we address the second point through the following lemma:

Lemma 38
The Valk and Jantzen condition can be reduced to the following criteria:

- *we can compute a finite representation of the coverability synthesis set in preT-PPNs iff universal coverability is decidable in preT-PPNs*
- *we can compute a finite representation of the coverability synthesis set in postT-PPNs iff existential coverability is decidable in postT-PPNs*

We prove the two items by proving the two implications for both items.

Proof. We first consider the case of preT-PPNs. Let us denote the proposition *we can compute a finite representation of the coverability synthesis set in preT-PPNs* by (1) and the proposition *universal coverability is decidable in preT-PPNs* by (2).

(1) $\Rightarrow$ (2): Let us consider a marked preT-PPN $\mathcal{S}$ and a marking m. We assume a finite basis of $\mathcal{CV}(\mathcal{S}, m)$ is effectively computable. Thus, we can get a representation of its complement, $\neg\mathcal{CV}(\mathcal{S}, m)$ using a procedure similar to the one of Example 5 in [Goubault-Larrecq, 2009] as we already explained in Section 2.1.3. Since $\neg\mathcal{CV}(\mathcal{S}, m)$ is upward closed, as the complement of the downward closed set $\mathcal{CV}(\mathcal{S}, m)$ from Lemma 36, we can invoke the result of Valk and Jantzen from [Valk and Jantzen, 1985] which implies that for all element $v \in \mathbb{N}_\omega^k$, it is decidable to answer

whether $\downarrow v \cap \neg\mathcal{CV}(\mathcal{S}, m) \neq \emptyset$. This is equivalent to decide whether $\downarrow v \cap \mathbb{N}^k \subseteq \mathcal{CV}(\mathcal{S}, m)$. Let us consider in particular $v = \vec{\omega}$. Then $\downarrow v \cap \mathbb{N}^k = \mathbb{N}^k$.

This exactly means that universal coverability of m in $\mathcal{S}$ is decidable.

(2) $\Rightarrow$ (1): We suppose universal coverability in preT-PPNs is decidable. Let us consider a preT-PPN $\mathcal{S}$ and a marking m. Then, for any valuation $v \in \mathbb{N}_\omega^k$, coverability of m in $v(\mathcal{S})$ is decidable. Indeed, if $v \in \mathbb{N}^k$ it is trivially decidable, otherwise, it means that universal coverability of m in $v_{|\mathbb{N}(v)}(\mathcal{S})$ is decidable, which is true by assumption. Thus, for any $v \in \mathbb{N}_\omega^k$, it is decidable to answer whether $\texttt{cov}(v(\mathcal{S}), m)$ is true, which directly gives whether $\downarrow v \cap \mathbb{N}^k \subseteq \mathcal{CV}(\mathcal{S}, m)$ by Lemma 15. This is equivalent to answering for all element $v \in \mathbb{N}_\omega^k$ whether $\downarrow v \cap \neg\mathcal{CV}(\mathcal{S}, m) \neq \emptyset$ which is exactly the condition given by Lemma 3 to use Valk and Jantzen algorithm to compute a finite basis of $\neg\mathcal{CV}(\mathcal{S}, m)$. Now, as recalled previously, we can compute a finite basis of its complement $\neg(\neg\mathcal{CV}(\mathcal{S}, m))$, that is to say $\mathcal{CV}(\mathcal{S}, m)$.

We now consider the case of postT-PPNs. Let us denote the proposition *we can compute a finite representation of the coverability synthesis set in postT-PPNs* by (3) and the proposition *existential coverability is decidable in postT-PPNs* by (4).

(3) $\Rightarrow$ (4): Let us consider a marked postT-PPN $\mathcal{S}$ and a marking m. Set $\mathcal{CV}(\mathcal{S}, m)$ being upward closed, we can invoke Valk and Jantzen's result and deduce that for all $v \in \mathbb{N}_\omega^k$, it is decidable to answer whether $\downarrow v \cap \mathbb{N}^k \subseteq \neg\mathcal{CV}(\mathcal{S}, m)$. Let us now notice that non coverability is monotonic in postT-PPNs in the sense that if m is not coverable in $v(\mathcal{S})$, then it is not coverable in any $v'(\mathcal{S})$ for all valuation $v' \leq v$. This is indeed exactly the contrapositive of Lemma 15. Thus $\downarrow v \cap \mathbb{N}^k \subseteq \neg\mathcal{CV}(\mathcal{S}, m)$ is equivalent to $\neg\texttt{cov}(v(\mathcal{S}), m)$. In particular, $\neg\texttt{cov}(\vec{\omega}(\mathcal{S}), m)$ is decidable, that is to say universal non coverability is decidable in postT-PPNs. We can now conclude by remarking that universal non coverability is equivalent to non existential coverability.

(4) $\Rightarrow$ (3): We suppose existential coverability in postT-PPNs is decidable. Since universal non coverability is equivalent to non existential coverability, we can now directly adapt the proof of the first item to get this result. □

Corollary 39 : Computability of the Solution Set of the $\mathcal{S}$-cov Problem

- *Given a marked preT-PPN $\mathcal{S}$ and a marking m, we can compute a finite representation of $\mathcal{CV}(\mathcal{S}, m)$.*
- *Given a marked postT-PPN $\mathcal{S}$ and a marking m, we can compute a finite representation of $\mathcal{CV}(\mathcal{S}, m)$.*

6.3 An Algorithm for a Direct Computation for PreT-PPNs

For the sake of completeness of this study, we now provide a procedure to compute directly the (minimal) basis of the solution set for preT-PPN rather than computing its complement. We suggest to synthesise such a basis by enumerating and testing exhaustively some valuations satisfying some criterion of monotonicity in preT-PPNs. For instance, let us consider the case where $|\mathbb{P}| = 2$, we intuitively need to begin by asking whether it is possible to cover the marking targeted for any valuation of the parameters. In the affirmative case, then the procedure can terminate. Otherwise, we need to restart by asking if the marking is universally coverable when we fix one of the parameter to 0. Let us focus on $\mathcal{S}_1$ from Figure 6.1. Universal coverability of $m_{goal1} = (0, 0, 1, 0, 1)$ is not possible. Nevertheless, m_{goal1} is universally coverable when a is set to 0. We must now pay attention to the fact that, if we choose a value for b, we can then permit a to be greater. Indeed, with b set to 1, it is possible to chose $a = 5$. We could first be tempted to set b to 0 hoping to find the maximal possible value for a. In this example, setting b to 0 allows to find a maximal value of 5 for a. To sum up, $\mathcal{CV}(\mathcal{S}_1, m_{goal1}) = \downarrow\{(0, \omega), (1, 5), (3, 2), (5, 1)\}$. Nevertheless, a difficulty rises from the net $\mathcal{S}_2$ of Figure 6.2. Indeed, in this case, if we fix b to 0,

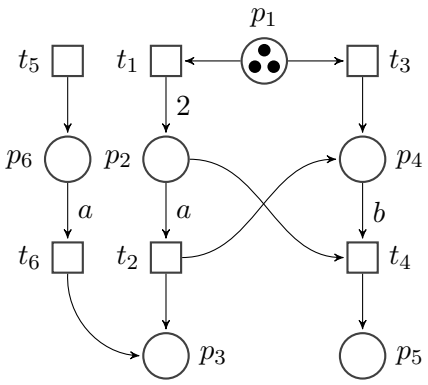


Figure 6.2 – A preT-PPN $\mathcal{S}_2$ where $\mathbb{P} = \{a, b\}$ with three parametric arcs

then a has no maximal value since any valuation of a will allow to cover $m_{goal2} = (0, 0, 1, 0, 1, 0)$. In fact, $\mathcal{CV}(\mathcal{S}_2, m_{goal2}) = \downarrow\{(0, \omega), (1, 5), (\omega, 2)\}$.

6.3.1 Preliminaries

We introduce a few notations that will shorten unions of partial valuations. Given a preT-PPN $\mathcal{S}$, an extended valuation $v \in \mathbb{N}_\omega^\Lambda$ where $\Lambda \subseteq \mathbb{P}$ and a subset Ω of $\mathbb{P}$, we define v^Ω as the extended valuation where for all $\lambda \in \Omega$, $v^\Omega(\lambda) = \omega$ and for all $\lambda' \in \Lambda \backslash \Omega$, $v^\Omega(\lambda') = v(\lambda')$. Given $\lambda \in \mathbb{P}$ and $k \in \mathbb{N}$, we also define $v_{\lambda=k}$ as the extended valuation where $v_{\lambda=k}(\lambda) = k$ and for all $\mu \neq \lambda$, $v_{\lambda=k}(\mu) = v(\mu)$. We will also combine both notations, provided that $\lambda \notin \Omega$. All those notations are also valid for valuations v such that $\omega(v) = \emptyset$.

We first extend Lemma 19 based on the extended definition of coverability from Corollary 37.

Lemma 40
Given a marked preT-PPN $\mathcal{S}$, a marking m of $\mathbb{N}^P$ and an extended valuation $v \in \mathbb{N}_\omega^\mathbb{P}$, the following three propositions are equivalent:

1. $\mathtt{cov}(v(\mathcal{S}), m)$
2. $\forall \lambda \in \omega(v), \{k \in \mathbb{N} \mid \mathtt{cov}(v_{\lambda=k}(\mathcal{S}), m)\}$ *is infinite.*
3. $\exists \lambda \in \omega(v), \{k \in \mathbb{N} \mid \mathtt{cov}(v_{\lambda=k}(\mathcal{S}), m)\}$ *is infinite.*

Proof. First, let us notice that (1) $\Rightarrow$ (2) $\Rightarrow$ (3) are immediate, by definition of $\mathtt{cov}(v(\mathcal{S}), m)$. We now prove that (3) $\Rightarrow$ (1). Notice that we can rewrite $v_{\lambda=k}$ as $\vec{k}_{|\{\lambda\}} \cup \omega_{|\omega(v)\backslash\{\lambda\}} \cup v_{|\mathbb{N}(v)}$. Let us consider $\mathcal{S}' = v_{|\mathbb{N}(v)}(\mathcal{S})$. It is a marked preT-PPN. Moreover, we know that there is an infinity of $k \in \mathbb{N}$ such that $\mathtt{cov}(\vec{k}_{|\{\lambda\}} \cup \omega_{|\omega(v)\backslash\{\lambda\}}(\mathcal{S}'), m)$ is true. By the definition of the extension of coverability to extended valuation, $\forall v' \in \mathbb{N}^{\omega(v)\backslash\{\lambda\}}, \mathtt{cov}(v'(\vec{k}_{|\{\lambda\}}(\mathcal{S}')), m)$ is true. Especially, this is true for $v' = \vec{k}$. Thus, we have found an infinite set of uniform valuations $\vec{k}$ such that $\mathtt{cov}(\vec{k}(\mathcal{S}'), m)$. Therefore, by Lemma 19, m is universally coverable in $\mathcal{S}'$ which concludes the proof. □

In the sequel, we abuse the term basis to designate a downward basis. We present in Figure 6.2 a preT-PPN $\mathcal{S}_2$, built to catch one of the difficulties of the coverability synthesis. In this preT-PPN, m_0 is equal to $(3, 0, 0, 0, 0, 0)$. We consider a marking to cover m_{goal2} equal to $(0, 0, 1, 0, 1, 0)$.

The exploration is based on the fact that there is no need to search for a bound of a. One can remark that when we fix b to 0, and then progressively increase b, the maximal value found for a decreases until it reaches a minimal value, 0 here, which is notably reached if we fix b to ω.

Let us provide an example by focusing on parameter a. $\mathcal{CV}(\mathcal{S}_2, m_{goal2})$ can be seen as the infinite union for each possible value for a of each valuation $v_{a=k}$ where v belongs to $\mathcal{CV}(\vec{k}_{|\{a\}}(\mathcal{S}_2), m)$

(v is thus a valuation of the parameter b only) formally: $\bigcup_{k\in\mathbb{N}}\bigcup_{v\in\mathcal{CV}(\vec{k}_{|\{a\}}(\mathcal{S}_2),m)}\{v_{a=k}\}$. This set might be infinite, but since each solution set $\mathcal{CV}(\vec{k}_{|\{a\}}(\mathcal{S}_2), m_{goal2})$ is downward closed, we can consider a finite basis to represent it. Let us enumerate those bases by instantiating the parameter a, starting from 0: $\mathcal{CV}(\vec{0}_{\{a\}}(\mathcal{S}_2), m_{goal2})$ is equal to $\downarrow\{(\omega)\}$, $\mathcal{CV}(\vec{1}_{\{a\}}(\mathcal{S}_2), m_{goal2})$ is equal to $\downarrow\{(5)\}$ and $\mathcal{CV}(\vec{k}_{\{a\}}(\mathcal{S}_2), m_{goal2})$ is equal to $\downarrow\{(2)\}$ for any $k \geq 2$. Thus, if we valuate a to ω we get that $\mathcal{CV}(\vec{\omega}_{\{a\}}(\mathcal{S}_2), m_{goal2})$ is the set $\downarrow\{(2)\}$. Now, for each value k tested, we can consider the minimal basis of the corresponding downward closed set and make the union of the valuation of this basis and $\vec{k}_{|\{a\}}$. This manipulation is formalised in Definition 39 through the notion of partial basis. For 0 we get $\downarrow\{(0,\omega)\}$, for 1 we get $\downarrow\{(1,5)\}$ and for any $k \geq 2$ we get $\downarrow\{(k,2)\}$. We see that this sequence converges toward $\downarrow\{(\omega,2)\}$ and that there is no need to seek for new valuations over $k = 2$. Considering a well chosen sequence, we will prove in Lemma 41 that such a value necessarily exists.

Definition 39 : Partial Basis

Given $\mathcal{S}$ a preT-PPN involving a set of parameters $\mathbb{P}$, v an extended valuation (such that $\mathtt{dom}(v) \subseteq \mathbb{P}$), m a marking of $\mathbb{N}^P$ fixed, we define the partial basis $\mathit{CVFB}(\mathcal{S}, m, v)$ as the set of all the unions of v with all the partial valuations of the minimal downward basis of $\mathcal{CV}(v(\mathcal{S}), m)$. By definition of the minimal downward basis, this is thus the subset of $\mathbb{N}^{\mathbb{P}}_{\omega}$ such that:

1. $\mathit{CVFB}(\mathcal{S}, m, v)_{|\mathtt{dom}(v)} = \{v\}$
2. $\mathit{CVFB}(\mathcal{S}, m, v) \subseteq \mathcal{CV}(\mathcal{S}, m)$
3. $\mathcal{CV}(v(\mathcal{S}), m) \subseteq\downarrow \mathit{CVFB}(\mathcal{S}, m, v)_{|\mathbb{P}\setminus\mathtt{dom}(v)}$
4. $\forall x, y \in \mathit{CVFB}(\mathcal{S}, m, v)\ x \neq y \Rightarrow x \not\leq y$ *and* $y \not\leq x$

Remark 6.3.1. The projection of a downward closed set is still a downward closed set. Therefore, this definition makes sense and describes a unique subset of $\mathbb{N}^{\mathbb{P}}_{\omega}$.

With this definition, in particular, $\downarrow \mathtt{CVFB}(\mathcal{S}, m, \emptyset_{\mathbb{N}}) \cap \mathbb{N}^{\mathbb{P}}$ is trivially equal to $\mathcal{CV}(\mathcal{S}, m)$. In order to prove that the previous decomposition can be computed through a finite union of partial bases, Lemma 41 is used to exhibit the least finite union that permits to compute the whole basis by exhibiting a fix point of a specific sequence of downward closure of partial bases.

Lemma 41

Given $\mathcal{S}$ a preT-PPN involving a set of parameters $\mathbb{P}$, v an extended valuation m a marking of $\mathbb{N}^P$ fixed, and a parameter $\lambda \in \mathbb{P} \setminus \mathtt{dom}(v)$, let

$$(X_n)_{n\in\mathbb{N}} \begin{cases} X_0 &= \mathit{CVFB}(\mathcal{S}, m, v_{\lambda=\omega}) \\ X_{n+1} &= X_n \cup \mathit{CVFB}(\mathcal{S}, m, v_{\lambda=n}) \end{cases}$$

Then the sequence $(\downarrow X_n)_{n\in\mathbb{N}}$ is ultimately constant.

Since we change the value of the parameter λ, we search for the greatest value k_{max} such that for every integer k greater than k_{max}, if m is coverable in $v_{\lambda=k}$, then m is also coverable in $v_{\lambda=\omega}$.

Proof. We note $\Lambda = \mathtt{dom}(v)$ and we define Λ' as the union $\Lambda \cup \{\lambda\}$. Notice that $\mathtt{dom}(v_{\lambda=\omega})$ and $\mathtt{dom}(v_{\lambda=n})$ are equal to Λ'. For greater readability, since $\mathcal{S}$, m and Λ' are fixed, we denote $\mathtt{CVFB}(\mathcal{S}, m, v_{\lambda=k})$ as $\mathtt{B}(k)$. With this notation $X_0 = \mathtt{B}(\omega)$ and $X_{n+1} = X_n \cup \mathtt{B}(n)$. Thus, for any $n \in \mathbb{N}$, $X_n \subseteq X_{n+1}$. First, we show that the lemma follows from the following statement:

$$\exists n^* \in \mathbb{N},\ \forall n \geq n^*,\ \forall x_n \in \mathtt{B}(n),\ \exists y \in \mathtt{B}(\omega) \text{ such that } y \geq x_n \tag{6.1}$$

Indeed, assuming (6.1), for every $n \geq n^*$, $\downarrow \mathtt{B}(n) \subseteq\downarrow \mathtt{B}(\omega)$. Moreover, since $X_{n+1} = X_n \cup \mathtt{B}(n)$, $\downarrow X_{n+1} \subseteq\downarrow \mathtt{B}(\omega) \cup \downarrow X_n$. Thus $X_0 = \mathtt{B}(\omega) \subseteq X_n$ and $\downarrow \mathtt{B}(\omega) \subseteq\downarrow X_n$. Thus $\downarrow X_{n+1} \subseteq\downarrow X_n$. Moreover, $\downarrow X_n \subseteq\downarrow X_{n+1}$. Therefore $\downarrow X_n =\downarrow X_{n+1}$. Thus $(\downarrow X_n)_{n\geq n^*}$ is constant.

Now we prove statement 6.1 ab absurdo.

$$\text{Assume } \forall n^* \in \mathbb{N},\ \exists n \geq n^*,\ \exists x_n \in \mathtt{B}(n),\ \text{such that } \forall y \in \mathtt{B}(\omega),\ y \not\geq x_n$$

Let us consider the infinite sequence of x_n's. By Lemma 1, we can extract a growing subsequence (x_{n_i}) such that $x_{n_1} \leq x_{n_2} \leq \cdots \leq x_{n_i} \leq \ldots$ with $n_1 < n_2 < \cdots < n_i < \ldots$.

Moreover, by the definition of `CVFB` (see Definition 39, item (1) and (2)), we have $\mathtt{cov}(x_{n_j}(\mathcal{S}), m)$ is true and $x_{n_j}(\lambda) = n_j$. By Lemma 15 on monotonicity, given any $n_i \leq n_j$, $\mathtt{cov}(x_{n_j\,\lambda=n_i}(\mathcal{S}), m)$ is true. Nevertheless, $x_{n_i} \leq x_{n_j}$ by construction and thus $x_{n_i} \leq x_{n_j\,\lambda=n_i}$ Thus, invoking the definition of `CVFB`, item (4), $x_{n_i} = x_{n_j|\lambda=n_i}$. Further, since $x_{n_i}(\lambda) = n_i$ and $x_{n_i|\Lambda} = v$, $(x_{n_i|\mathbb{P}\setminus\Lambda'})_{i\in\mathbb{N}}$ is in fact constant equal to some value e. So every x_{n_i} is equal to $e \cup v_{\lambda=n_i}$. We can thus deduce that $\forall i \in \mathbb{N}$, $\mathtt{cov}(e \cup v_{\lambda=n_i}(\mathcal{S}), m)$ is true. Thus, by applying Lemma 40, $\mathtt{cov}(e \cup v_{\lambda=\omega}(\mathcal{S}), m)$ is true. Therefore by Definition of `CVFB`, item (3), $\exists y \in \mathtt{B}(\omega)$ such that $y_{|P\setminus\Lambda'} \geq e$. Moreover, since $y \in \mathtt{B}(\omega)$, $y_{|\Lambda'} = v_{\lambda=\omega}$. Therefore, $y \geq e \cup v_{\lambda=\omega} \geq e \cup v_{\lambda=n_i} = x_{n_i}$. This contradicts the hypothesis. Thus Equation 6.1 is proved. □

6.3.2 Procedure

Given a set of vectors $X \subseteq \mathbb{N}^n_\omega$, we define $\max(X)$ as the minimal basis of $\downarrow X$. Based on those notations, we present, in Algorithm 4, a recursive procedure `synthesisDB` that computes the minimal basis of $\mathcal{CV}(\mathcal{S}, m)$. Its inputs are a marked preT-PPN $\mathcal{S}$, a marking of $m \in \mathbb{N}^P$, an extended valuation v such that $\mathtt{dom}(v) \subseteq \mathbb{P}$. Its output is a set of extended valuations.

Algorithm 4 Coverability Synthesis Algorithm for PreT-PPNs, `synthesisDB`

Require: A preT-PPN $\mathcal{S} = ((P, T, \mathsf{Pre}, \mathsf{Post}, \mathbb{P}), m_0)$, a goal marking m, a set of parameters Λ, and an extended valuation v
Return: A subset of $\mathbb{N}^{\mathbb{P}}_\omega$
procedure
  if $\mathtt{dom}(v) = \mathbb{P}$ **then**
    if $\mathtt{cov}(v(\mathcal{S}), m)$ **then**
      return $\{v\}$
    else
      return $\emptyset$
  else choose $\lambda \in \mathbb{P}\setminus\mathtt{dom}(v)$
    let $\begin{cases} X_0 &= \mathtt{synthesisDB}(\mathcal{S}, m, v \cup \vec{\omega}_{|\{\lambda\}}) \\ X_{n+1} &= X_n \cup (\mathtt{synthesisDB}(\mathcal{S}, m, v \cup \vec{n}_{|\{\lambda\}})) \end{cases}$
    $n^* \leftarrow \min\{n \in \mathbb{N} \text{ such that } \max(X_n) = \max(X_{n+1})\}$
    return $\max(X_{n^*})$

In the following subsections, we will prove that Theorem 42 holds:

Theorem 42
Given $\mathcal{S}$ a preT-PPN involving a set of parameters $\mathbb{P}$, m a marking of $\mathbb{N}^P$ fixed, `synthesisDB`$(\mathcal{S}, m, \emptyset_{\mathbb{N}})$ terminates and, upon termination, `synthesisDB`$(\mathcal{S}, m, \emptyset_{\mathbb{N}})$ returns the minimal downward basis of $\mathcal{CV}(\mathcal{S}, m)$.

6.3.3 Completeness and Soundness

We first prove that this algorithm is complete and sound through the following theorem:

Theorem 43
Given $\mathcal{S}$ a preT-PPN involving a set of parameters $\mathbb{P}$, v an extended valuation, m a marking of $\mathbb{N}^P$ fixed,

$$\texttt{synthesisDB}(\mathcal{S}, m, v) = \texttt{CVFB}(\mathcal{S}, m, v)$$

Proof. We note $\Lambda = \texttt{dom}(v)$. We prove that $\texttt{synthesisDB}(\mathcal{S}, m, v)$ satisfies the four items of Definition 39. Then by unicity of $\texttt{CVFB}(\mathcal{S}, m, v)$ (see Remark 6.3.1), we can deduce the equality. We proceed by induction on $|\mathbb{P}\backslash\Lambda|$:

- base case: we consider $|\mathbb{P}\backslash\Lambda| = 0$, then $\Lambda = \mathbb{P}$. Two possibilities arise from the algorithm: it returns either $\{v\}$ or $\emptyset$. In both case (1) (2) (3) and (4) are trivially true.
- inductive step: given $n \in \mathbb{N}$, $n < |\mathbb{P}|$ we assume that for any Λ, such that $|\mathbb{P}\backslash\Lambda| = n$ the proposition holds. Let us consider Λ such that $|\mathbb{P}\backslash\Lambda| = n + 1$. Given $\lambda \in \mathbb{P}\backslash\Lambda$, we define Λ' as $\Lambda \cup \{\lambda\}$. Thus, for every $v' \in \mathbb{N}_\omega^{\Lambda'}$ $\texttt{synthesisDB}(\mathcal{S}, m, v') = \texttt{CVFB}(\mathcal{S}, m, v')$ by the induction hypothesis. More precisely, if we consider the sequence defined in the algorithm, each X_i is thus the union $\texttt{CVFB}(\mathcal{S}, m, v_{\lambda=\omega}) \cup \left(\bigcup_{0\leq k\leq i-1} \texttt{CVFB}(\mathcal{S}, m, v_{\lambda=k})\right)$. Moreover, we can invoke Lemma 41 to justify that the fix point n^* exists and, by its proof, that for each $k > n^*$, $\downarrow\texttt{CVFB}(\mathcal{S}, m, v_{\lambda=k}) \subseteq\downarrow\texttt{CVFB}(\mathcal{S}, m, v_{\lambda=\omega}) \subseteq\downarrow X_{n^*}$. The algorithm then returns the maximum elements of X_{n^*}. First, let us consider X_{n^*}. It is a finite union of sets satisfying (1) (2) (3) and (4). Therefore, (1) and (2) remain true on X_{n^*}. Then taking the *max* of X_{n^*} will not add new elements so (1) and (2) still hold for the result returned by the procedure. We now focus on proposition (3) of the definition. By induction hypothesis, (3) provides that $\forall b \in \mathcal{CV}(\mathcal{S}, m)$, such that $b_{|\Lambda} = v$, $\exists e \in \texttt{CVFB}(\mathcal{S}, m, b_{|\Lambda'})$ such that $e \geq b$. Let us consider a pair (b, e) defined by the previous statement. Two cases arise: if $b(\lambda) \leq n^*$, we have that $\downarrow\texttt{CVFB}(\mathcal{S}, m, b_{|\Lambda'})$ is directly included in $\downarrow X_{n^*}$, and as we consider only the maximal elements, we will be able to choose some ϵ such that $\epsilon \geq e \geq b$. Otherwise, $b(\lambda) > n^*$ then, by the proof of Lemma 41, $\downarrow\texttt{CVFB}(\mathcal{S}, m, v_{\lambda=b(\lambda)}) \subseteq\downarrow X_{n^*}$. And as we consider the maximal elements of X_{n^*}, we are still able to choose ϵ satisfying $\epsilon \geq e \geq b$. Thus, $\forall b \in \mathcal{CV}(\mathcal{S}, m)$, such that $b_{|\Lambda} = v$, $\exists \epsilon \in \max(X_{n^*}) = \texttt{synthesisDB}(\mathcal{S}, m, b_{|\Lambda})$ such that $\epsilon \geq b$. This means exactly that (3) holds for this set. Let us conclude by noting that max just affects the minimality of the basis: by (2) and (3) we know that $X_{n^*|\mathbb{P}\backslash\Lambda}$ is a basis of $\mathcal{CV}(v(\mathcal{S}), m)$. Thus, $max(X_{n^*})_{\mathbb{P}\backslash\Lambda}$ is the minimal basis of $\mathcal{CV}(v(\mathcal{S}), m)$ and $max(X_{n^*})$ is just its extension with v on Λ. Thus, (4) holds, by construction.

By induction we proved (1) (2) (3) and (4). We can conclude by unicity of the minimal basis and construction of CVFB that $\texttt{synthesisDB}(\mathcal{S}, m, v)$ is equal to $\texttt{CVFB}(\mathcal{S}, m, v)$. □

6.3.4 Termination

Theorem 44
Given $\mathcal{S}$ a preT-PPN involving a set of parameters $\mathbb{P}$, v an extended valuation, m a marking of $\mathbb{N}^P$ fixed, the procedure $\texttt{synthesisDB}(\mathcal{S}, m, v)$ terminates.

Proof. We note $\Lambda = \texttt{dom}(v)$. We proceed by induction on $|\mathbb{P}\backslash\Lambda|$. Notice first that this algorithm, when it terminates, returns a finite set.

- base case: we consider $\lambda = \mathbb{P}$, then $|\mathbb{P}\backslash\Lambda| = 0$. Two possibilities arise from the algorithm: it returns either $\{v\}$ or $\emptyset$. In both case it terminates.
- inductive step: given $n \in \mathbb{N}$, we assume that for any Λ, such that $|\mathbb{P}\backslash\Lambda| = n$ the proposition holds. Let us consider Λ such that $|\mathbb{P}\backslash\Lambda| = n+1$. Given $\lambda \in \mathbb{P}\backslash\Lambda$, we define Λ' as $\Lambda \cup \{\lambda\}$. By induction hypothesis, for every v', we know that $\texttt{synthesisDB}(\mathcal{S}, m, v')$ terminates, and as stated in Theorem 43, $\texttt{synthesisDB}(\mathcal{S}, m, v') = \texttt{CVFB}(\mathcal{S}, m, v')$. Thus, by application of Lemma 41 there exists n^* such that $(\downarrow X_n)_{n\geq n^*}$ is constant. Thus, $(max(X_n))_{n\geq n^*}$

is constant too. Moreover, X_{n^*} is computable as a finite union of finite sets. It is then possible to take the maximal elements of this set, as it is a finite set.

By induction, we proved the termination of Algorithm 4. □

Note that we could symmetrically carry out a development dual to that for preT-PPNs, and derive an algorithm which computes the minimal downward basis of the set of valuation allowing non-coverability for postT-PPNs.

We proposed here a survey of the coverability synthesis for preT-PPNs and postT-PPNs to compute directly some basis for the sets of valuation allowing to cover (or not to cover) a marking. We now study what can be inferred from these results to the class of distinctT-PPNs.

6.4 Limit for DistinctT-PPNs

Let us finally consider the case of PPNs in which the set of parameters used as input weights, and the set of parameters used as output weights, are disjoint. For this class, called distinctT-PPNs, the emptiness and the universality of the solution set to the synthesis problem for coverability are decidable (see Corollary 35). Interestingly, we can adapt an idea originally used for L/U-automata in [Jovanović et al., 2015] to prove that the structure of this set is however much more complex than for preT-PPNs or postT-PPNs. In particular, one cannot represent this set with a finite set, a finite union of downward and/or upward closed sets or a finite union of convex polyhedra.

Lemma 45
If it can be computed, the solution of the synthesis of coverability in distinctT-PPN cannot, in general, be represented using any formalism for which emptiness of the intersection with equality constraints is decidable.

Proof. We adapt an idea originally used for L/U-automata in [Jovanović et al., 2015]. We recall that following Theorem 11, existential coverability on PPNs is undecidable. Let us suppose we can compute the set of valuations under which a given marking of $\mathbb{N}^P$ is coverable in a distinctT-PPN. Then, let us consider a general PPN $\mathcal{S}$. For each parameter λ used on both input and output arcs, we replace its occurrence on input arcs by λ_i and on output arcs by λ_o. We have therefore constructed a distinctT-PPN. We then solve the synthesis problem and compute the set of valuations *Good*. Let K be the set of equality constraints $\lambda_i = \lambda_o$ for each parameter λ that was replaced. Clearly deciding whether there exists a valuation v such that m is coverable in $v(\mathcal{S})$ is equivalent to deciding whether $Good \cap K$ is empty or not. Existential coverability being undecidable on T-PPNs, we can conclude that such an operation cannot be computed. □

We achieved here to prove a powerful result for two strict syntactical subclasses of parametric Petri nets: interestingly, the set of all valid valuations of parameters, allowing to cover a given marking, is effectively computable for parametric Petri nets where parameters are restricted to only input arcs or only output arcs. Indeed, we have shown how the computability of the synthesis set for coverability in preT-PPNs and postT-PPNs can be reduced to a decision problem, respectively, universal coverability and existential coverability, which is then used in Valk and Jantzen's procedure. We also develop this study to provide a direct procedure to compute the basis of the solution set to the coverability synthesis for preT-PPNs rather than computing first a basis of its complement and then its basis. Putting the two types of parameters together while forbidding any parameter to be used as both an input and output weight preserves the decidability of the emptiness and universality of the solution set. However, we have proved that, even with this restriction, the solution set can in general not be represented using any formalism

for which emptiness of the intersection with equality constraints is decidable, which seems a big restriction in practice.

Since we have now provided a survey of the parametric coverability problems, it would be interesting to turn our attention toward establishing decision frontiers for the subclasses of PPNs. This is the concern of the next chapter.

CHAPTER 7

Establishing Frontiers of Decidability Problems

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

— Alan Turing

Contents

In this Chapter, we emancipate from the problem of coverability and investigate decidability frontiers for the subclasses of PPNs. The currently known results are summed up in Table 7.1. We extended the results of decidability to simultaneous unboundedness thanks to the extensions of the procedure of Karp and Miller on the one hand, and to the monotonicity on the other hand.

Lemma 46

We have the following results:

- *Universal simultaneous unboundedness on postT-PPNs, P-PPNs and distinctT-PPNs is ExpSpace-complete.*
- *Existential simultaneous unboundedness on preT-PPNs is ExpSpace-complete.*
- *Existential simultaneous unboundedness on P-PPNs and distinctT-PPNs is ExpSpace-hard.*

In order to close some of the remaining open problems, we study reachability for those subclasses and prove an interesting gap of modelling power between postT-PPNs and P-PPNs.

7.1 Undecidability of Parametric Reachability for PostT-PPNs

We first consider the class of postT-PPNs. We will prove that universal reachability and existential reachability are both undecidable for this class by adapting the proofs by reduction from 2-counter machine of Section 4.5.

Given a Minsky 2-counter machine $\mathcal{M}$, we want to construct a postT-PPN that simulates it, $\mathcal{S}_{\mathcal{M}}$. Note that since we can still use parametric output arcs the main structure of the translation established for generic PPNs in Figure 4.8 remain unchanged. Indeed, parametric input arcs were only used for the zero test. Nevertheless, we consider here reachability properties. We will

	$\mathscr{U}$-problem			$\mathscr{E}$-problem		
	Reach.	S.Unbound	Cov.	Reach.	S.Unbound	Cov.
PPN	U	U	U	U	U	U
T-PPN	U	U	U	U	U	U
preT-PPN	?	D	D	?	D	D
postT-PPN	?	D	D	?	D	D
P-PPN	?	D	D	?	D	D
distinctT-PPN	?	D	D	?	D	D

Table 7.1 – Intermediate summary for the decidability results (D stands for decidable, U stands for undecidable)

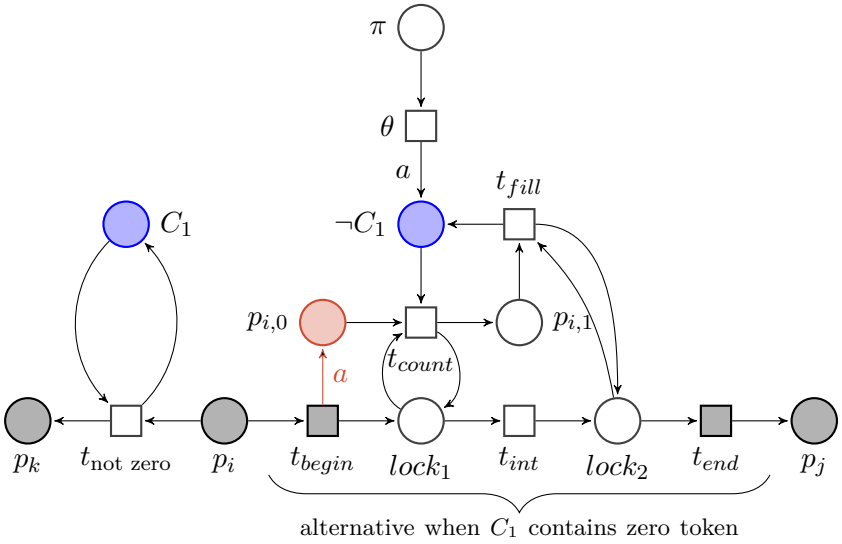


Figure 7.1 – Modelling a zero test with a postT-PPN and reachability

now detail a construction, where only parametric output arcs are involved, and which allows us to perform zero test *fairly* or *unfairly*. In the latter case, some tokens will be trapped in some observer places, meaning the only way to find a run reaching markings with zero token in those observer places is to find a run which performs only *fair* zero test. Let us first detail this construction depicted in Figure 7.1.

We recall that C_i models the counter, and at all times we know the sum of tokens available in C_i and $\neg C_i$ is a. Thus, there is no token in C_i iff there are a tokens in $\neg C_i$. However, we cannot directly test if there are exactly a tokens in $\neg C_i$ since we cannot consume them with a parametric input arc. The idea is therefore to create an observer that will count if there is at least a tokens in $\neg C_1$. This is done by first generating a tokens in a place $p_{i,0}$. Then, tokens from $p_{i,0}$ and $\neg C_1$ are jointly consumed and used to fill a place $p_{i,1}$. This is possible as long as $lock_1$ remained marked. Then, the intermediate event t_{int} occurs: $lock_2$ becomes marked, so that the tokens stored in $p_{i,1}$ can be used to refill $\neg C_1$ through the firing of t_{fill}. At the end of the process, $\neg C_1$ has at most a tokens. So there is a nominal behaviour that correctly simulates a zero test. We will now investigate for the other possible behaviours.

Here, what is important is to notice that it is possible to perform an *unfair* zero test by firing the sequence from t_{begin} to t_{end} even if there are less than a tokens in $\neg C_1$ say $k < a$ (thus $a-k > 0$ tokens in C_1). Nevertheless, this will be remembered since, with this construction, at least $a-k$ tokens will then be trapped in $p_{i,0}$. On the contrary, if the sequence $t_{begin}(t_{count})^a t_{int}(t_{fill})^a t_{end}$ is performed, we say that the zero test is *fair*.

Remark 7.1.1. Let us formalise this intuition. We can observe and prove the following properties:

- if there are a tokens in $\neg C_1$ and 1 token in p_i, then the zero test can be performed fairly, leading to a marking with a tokens in $\neg C_1$ and 1 token in p_j whereas other places (in particular $p_{i,0}$, $p_{i,1}$, $lock_1$, $lock_2$ and p_i) are empty. As stated above, this is trivial since it is sufficient to fire $t_{begin}(t_{count})^a t_{int}(t_{fill})^a t_{end}$ to obtain the result (but it is also possible to perform unfair zero tests).
- if there are strictly less than a tokens, say k, in $\neg C_1$ and 1 token in p_i, regardless of the number of tokens initially in $p_{i,0}$. Then, no possible fair zero test can be done. It is only possible to perform an *unfair* zero test to reach a marking with 1 token in p_j but the place $p_{i,0}$ will have a strictly positive amount of tokens. This is immediate by observing the construction: we can fire $t_{begin}(t_{count})^h t_{int}(t_{fill})^l t_{end}$ with $h \leq k < a$. Since t_{begin} generates a tokens in $p_{i,0}$, at most h tokens can be consumed from $p_{i,0}$ and some tokens thus remains trapped in $p_{i,0}$.
- Finally each time t_{begin} is fired, then a tokens are generated in $p_{i,0}$ and at most a tokens can be consumed from $p_{i,0}$. Therefore, if at some moment in the execution some tokens remain in $p_{i,0}$, then performing the same zero test again cannot help to empty $p_{i,0}$ since it will generate a new tokens in addition to those already present in $p_{i,0}$.

We have now all the material necessary to prove the following theorem:

Theorem 47
$\mathscr{E}$-reachability is undecidable in postT-PPNs.
As a consequence $\mathscr{E}$-reachability is undecidable in distinctT-PPNs.

Proof. We will show that given a 2-counter machine $\mathcal{M}$, *(a)* $\mathcal{M}$ halts (it reaches the *halt* state) iff *(b)* there exists a valuation v such that $v(\mathcal{S}_\mathcal{M})$ reaches a marking where p_{halt} is marked and place $p_{i,0}$ of every zero test are empty. To fully determine this problem, we will in fact consider the marking m_{halt} where only the place p_{halt} contains one token. In order to make sure that the other places can be emptied, we modify a bit the construction: given any place p_i different from p_{halt} and from the $p_{i,0}$'s, we add a transition t_i such that $\mathsf{Pre}(p_i, t_i) = 1$, $\mathsf{Pre}(p_{halt}, t_i) = 1$, $\mathsf{Post}(p_{halt}, t_i) = 1$, and Pre and Post are equal to 0 otherwise. Those transitions allow to empty the places of the net once p_{halt} is reached. With this feature, *(b)* is equivalent to the existence a valuation v such that $v(\mathcal{S}_\mathcal{M})$ can reach m_{halt}.

- $(a) \Rightarrow (b)$ First, let us assume that $\mathcal{M}$ halts. Since $\mathcal{M}$ 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 v be the valuation such that $v(a) = c_{lim}$. Invoking the previous explanation on this construction, $\mathcal{S}_\mathcal{M}$ clearly simulates $\mathcal{M}$. Moreover, the valuation v ensures that $\mathcal{S}_\mathcal{M}$ does not reach a deadlock state where p_{error} is marked. Therefore, when $\mathcal{M}$ reaches *halt*, $\mathcal{S}_\mathcal{M}$ will add 1 token in p_{halt}. Moreover, since the run was performed in the machine, for any zero test performed in the machine, the simulated run can perform a *fair* test in the constructed net. Thus, there exists a run in $\mathcal{S}_\mathcal{M}$ that does not leave tokens trapped in any $p_{i,0}$'s. So, a marking where there is one token in p_{halt} and no tokens in $p_{i,0}$'s is reachable, and it is then sufficient to fire the transitions t_i to empty the other places.
- $(b) \Rightarrow (a)$ We proceed by contrapositive. Let us assume that $\mathcal{M}$ does not halt. We want to show that there is no valuation a such that $v(\mathcal{S}_\mathcal{M})$ adds a token in p_{halt}. Let us consider the following two distinct alternatives:
  - If the counters are bounded along the execution, either the value of a is less than the maximum value of the counters and p_{error} will be reached during some increment resulting in a deadlock, or the value of a is big enough so that p_{error} is never marked, but, in this case, then, since the machine does not halt, it means that it does not reach *halt*. So there is no instruction that leads to *halt* in $\mathcal{M}$. The only possibility to reach *halt* would thus be to cheat at least one zero test. Nevertheless, this would create

some tokens in the corresponding $p_{i,0}$ that could never be consumed as explained in the previous remark. Thus, no marking with jointly one token in p_{halt} and no tokens in any $p_{i,0}$ can be reached.

- If at least one counter is not bounded, then for any given valuation v, we will reach an instruction "*increment* c_i", where i is 1 or 2, and $c_i = v(a)$. Therefore, a token will be added in p_{error} leading to a deadlock. So $\mathcal{S}_{\mathcal{M}}$ will not reach a terminal state.

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

□

Theorem 48
$\mathscr{U}$-Reachability is undecidable in postT-PPNs.
As a consequence $\mathscr{U}$-reachability is undecidable in distinctT-PPNs.

Proof. We proceed by reduction from *the boundedness problem for a 2-counter machine*. We denote by m_{error} the marking where $m_{error}(p) = 0$ for each $p \in P$ except $m_{error}(p_{error}) = 1$. We use a construction similar to the one of the previous proof : this time for each place p_i different from p_{error} and the $p_{i,0}$'s, a transition t_i can be used to empty p_i iff p_{error} is marked.

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 v, $v(\mathcal{S}_{\mathcal{M}})$ can reach m_{error}.

- $(a) \Rightarrow (b)$ First, let us assume that on a given instructions 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}. Moreover, since there is no need to perform an unfair zero test, then, for all valuation, the run that mimics exactly the behaviour of the two counter machine will add one token in p_{error}, leaving every $p_{i,0}$'s empty. The other places can then be emptied making (b) true.
- $(b) \Rightarrow (a)$ Reciprocally, by contrapositive, we want to show that if the counters are bounded, there exists a valuation v such that $v(\mathcal{S}_{\mathcal{M}})$ does not reach m_{error}. This comes directly from the previous proof. Since the counters are bounded along the instructions sequence, we consider a valuation v such that $v(a) = c_{lim}$ where c_{lim} is an upper bound of the values of the counters. By construction, there is no fair possibilities to add a token in p_{error}, otherwise, it means that $\mathcal{S}_{\mathcal{M}}$ took an incrementation transition meaning that c_{lim} is not an upper bound. Thus, the only possible way to mark p_{error} would be to perform an unfair zero test in order to consider a different execution of the machine. Nevertheless, this implies that there is no possibility to empty the corresponding $p_{i,0}$ making the marking m_{error} not reachable either.

The undecidability of the counter boundedness problem on the 2-counter machine gives the undecidability of the $\mathscr{U}$-reachability problem. □

We now consider the parametric reachability problems in the context of preT-PPNs.

7.2 Undecidability of Parametric Reachability for PreT-PPNs

Considering preT-PPNs, it is not possible to generate a tokens anymore. Nevertheless, we are authorised to consume a tokens through parametric input arcs. We first introduce a construction, depicted in Figure 7.2, that allows to generate arbitrary many tokens in a place and keep a memory and a control on the amount of tokens generated using only preT-PPNs. This construction will be used to simulate the effect of a parametric output arc generating a tokens. Indeed, on this construction, it is easy to see that for every marking m of this net,

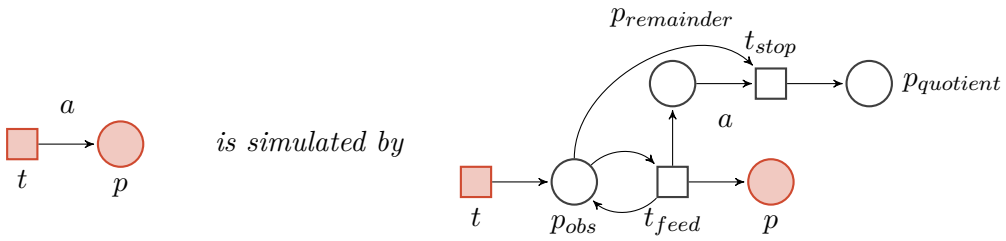


Figure 7.2 – Generating a tokens with a preT-PPN

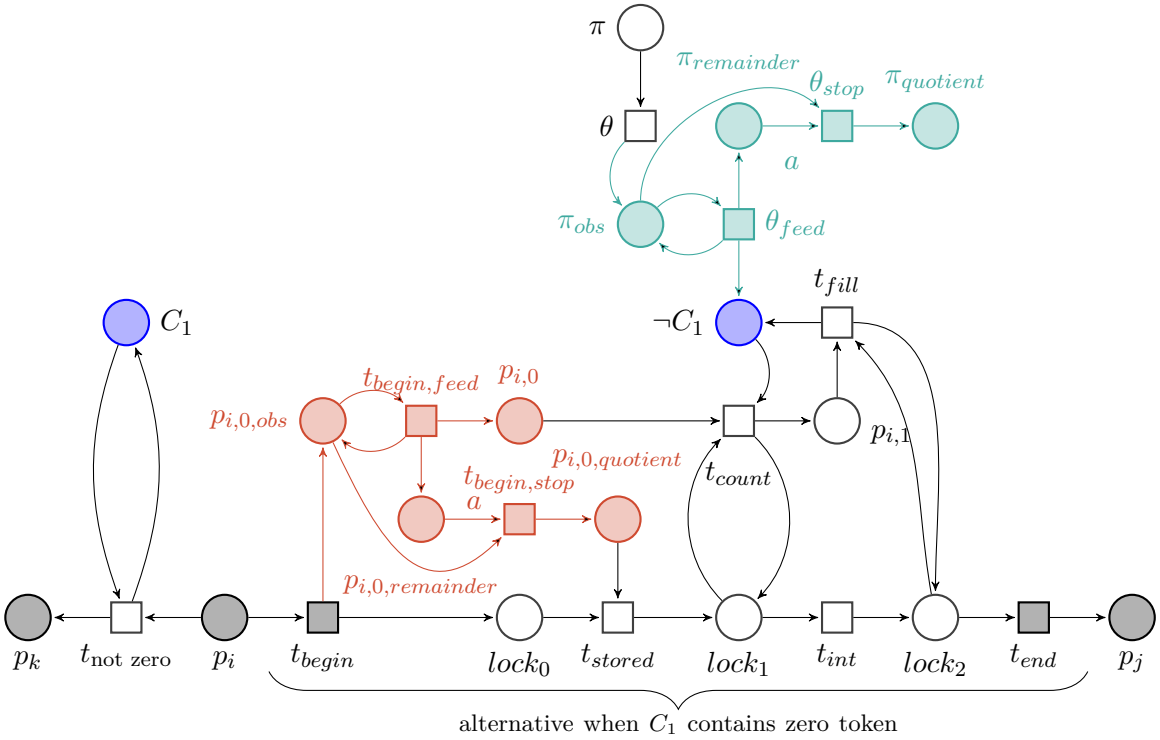


Figure 7.3 – Modelling a zero test with a preT-PPN

$m(p) = a \times m(p_{quotient}) + m(p_{remainder})$. Therefore, for every marking m, we can assert that $m(p) = a$ iff $m(p_{quotient}) = 1$ and $m(p_{remainder}) = 0$.

We will now provide an adaptation of the construction of previous section where each parametric output arc is replaced by the construction depicted in Figure 7.2. Then, we will elaborate on the properties satisfied by the net built and provide proofs for the undecidability of existential and universal reachability in preT-PPNs using this construction. Let us first focus on the construction of Figure 7.3 and explain how it simulates the zero test.

First let us study the initialisation process: to initialise the place $\neg C_1$, with this construction, it is now possible to generate an arbitrary number of tokens in $\neg C_1$, nevertheless, $\neg C_1 = a$ iff the two places corresponding to the remainder and the quotient contain respectively 0 and 1 token.

Concerning the zero test, the properties of previous remark can be adapted as follow:

Remark 7.2.1. The following properties hold:

- if there are a tokens in $\neg C_1$ and 1 token in p_i, then the zero test can be performed fairly, and it leads to a marking with a tokens in $\neg C_1$ and 1 token in p_j and $\pi_{quotient}$ whereas other places are empty. As stated above, this is trivial since it is sufficient to fire $t_{begin}(t_{begin,feed})^a t_{begin,stop} t_{stored} (t_{count})^a t_{int} (t_{fill})^a t_{end}$ to obtain the result (but it is also possible to perform unfair zero test).
- if there are strictly less than a tokens in $\neg C_1$ (say k) and 1 token in p_i. Then it is possible

to perform an *unfair* zero test to reach a marking with 1 token in p_j. Two behaviours can be considered:

- If a tokens or more are created in $p_{i,0}$, the place $p_{i,0}$ will have a strictly positive number of tokens after the firing of t_{end}. This is immediate by observing the construction: we can fire $t_{begin}(t_{begin,feed})^{a+l}t_{begin,stop}t_{stored}(t_{count})^h t_{int}(t_{fill})^j t_{end}$ with $j \leq h \leq k < a$ and $l \geq 0$. Since we generate $a + l$ tokens in $p_{i,0}$, at most $k < a$ tokens can be consumed from $p_{i,0}$ and some tokens thus remains trapped in $p_{i,0}$.
- If less than a tokens are generated in $p_{i,0}$, say $h < a$. Then $p_{i,0,remainder}$ will also contain $h < a$, thus $p_{i,0,quotient}$ cannot be marked. There is no other possibilities than generating more tokens in $p_{i,0,remainder}$ and thus in $p_{i,0}$ to continue the execution, leading to the previous item.

- Finally each time t_{begin} is fired, then at least a tokens are generated in $p_{i,0}$ and at most a tokens can be consumed from $p_{i,0}$. Therefore, if at some moment in the execution some tokens remain in $p_{i,0}$, then performing the same zero test again cannot help to empty $p_{i,0}$ since it will generate a new tokens in addition to those already present in $p_{i,0}$.

Let us now suggest a scenario to provide a better understanding. Imagine we perform this zero test twice, the first time is fair but more than a tokens are created, say $a + k$. Then k tokens remains in $p_{i,0}$ and the place $p_{i,0,remainder}$ after the firing of this test. Now, let us suppose we perform an unfair zero test, that is to say less than a tokens are in $\neg C_1$. We could generate less than $a - k$ tokens in $p_{i,0,remainder}$ such that now $p_{i,0}$ as a tokens. Nevertheless, we could never consume more than $h \leq k$ tokens, leaving tokens trapped in $p_{i,0}$, since it is not possible to consume from $p_{i,0}$ more tokens than there are in $\neg C_1$.

Theorem 49
$\mathscr{E}$-Reachability is undecidable in preT-PPNs.

Proof. We will show that given a 2-counter machine $\mathcal{M}$, *(a)* $\mathcal{M}$ halts (it reaches the *halt* state) iff *(b)* there exists a valuation v such that $v(\mathcal{S}_\mathcal{M})$ can reach the corresponding p_{halt} place, such that each place $p_{i,0}$ of the zero test is empty, the remainder place of the initial construction is empty and the quotient place of the initial construction has exactly one token. Moreover, we can impose that the other places of the net are empty by adding transitions that read if p_{halt} is marked and empty those other places.

- $(a) \Rightarrow (b)$ This is a direct adaptation from the corresponding proof for postT-PPNs and the discussion above.
- $(b) \Rightarrow (a)$ This can be adapted as well from the proof for postT-PPNs, in particular, if the counters are bounded along the execution, as previously, the only way to mark p_{halt} is to cheat the zero test leading to tokens trapped in $p_{i,0}$ by the discussion above. The other case is similar to the one in postT-PPNs.

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

□

Theorem 50
U-Reachability is undecidable in preT-PPNs.

Proof. We proceed by reduction from *the boundedness problem of a 2-counter 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 v, $v(\mathcal{S}_\mathcal{M})$ reaches the m_{error} (1 token in p_{error} and in the quotient corresponding to the initialisation of the counter, 0 tokens in the places $p_{i,0}$'s of the

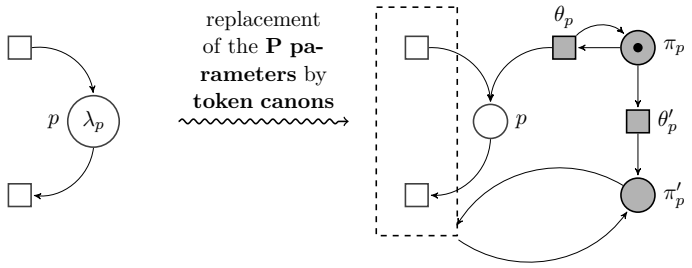


Figure 7.4 – From PPNs to PNs

zero test). To precise this marking we can ask every other place to be equal to zero and add transitions that reads if p_{error} is marked and allows to empty the other places.

- $(a) \Rightarrow (b)$ This is the same reasoning as for postT-PPNs.
- $(b) \Rightarrow (a)$ This is the same reasoning as for postT-PPNs since p_{error} could be marked only if a zero test is performed unfairly which would leave tokens trapped in a place $p_{i,0}$ marking the goal marking unreachable.

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

7.3 Decidability of Existential Reachability for P-PPNs

We proved in Section 5.2 of Chapter 5 that postT-PPN and P-PPN are effectively weakly-cosimilar. Therefore, given a postT-PPN we can build a P-PPN which is weakly-cosimilar. Moreover, since coverability can be reduced to firing a transition (by adding an observer transition), weak-simulation preserves coverability. Nevertheless, it does not preserve reachability. In this section, we prove that existential reachability is decidable on P-PPNs.

Theorem 51
$\mathscr{E}$-Reachability on P-PPN is decidable and ExpSpace*-hard.*

Proof. We consider a P-PPN $\mathcal{S}_1$. We will now build a PN $\mathcal{S}_2$ with *token-canons* that will supply the parameterised places of $\mathcal{S}_1$ as depicted in Figure 7.4. Each *token-canon* consists in two places π_p, π'_p and two transitions θ_p, θ'_p. Transition θ_p supplies place p of $\mathcal{S}_2$. Moreover, each transition of $\mathcal{S}_1$ 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 initialise $\mathcal{S}_2$ with 1 token in each π_p. So for each valuation v of $\mathcal{S}_1$, firing the sequence $\theta_p^{v(\lambda_p)}\ \theta'_p$ for each parameterised place p leads to a marking m_2 equals to the valuation of the initial marking of $\mathcal{S}_1$. Moreover, the θ-transitions added have been fired, so every π'_p is marked. The two nets have now the same behaviour: if we omit the places π_p and π'_p, any marking reachable in a v-instance of $\mathcal{S}_1$ can be reached in $\mathcal{S}_2$ and reciprocally. This shows that $\mathcal{S}_2$ simulates any valuation of $\mathcal{S}_1$. Therefore, 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 a possibly unbounded Petri net. Thus $\mathscr{E}$-reachability is decidable on P-PPN.

From Remark 4.2.1, we know that $\mathscr{E}$-reachability on P-PPN is at least ExpSpace-hard. Moreover, we apply here a transformation which is linear in the size of the Petri Net. Indeed we add 2 places, 2 transitions and $5 + 2 \times |T|$ arcs for each initial parameterised place. With this transformation, we get a classic marked Petri Net. Thus $\mathscr{E}$-reach on P-PPN is decidable and not harder than reachability on Petri nets.

□

	$\mathscr{U}$-problem			$\mathscr{E}$-problem		
	Reach.	S.Unbound	Cov.	Reach.	S.Unbound	Cov.
PPN	U	U	U	U	U	U
T-PPN	U	U	U	U	U	U
preT-PPN	U	D	D	U	D	D
postT-PPN	U	D	D	U	D	D
P-PPN	?	D	D	D	D	D
distinctT-PPN	U	D	D	U	D	D

Table 7.2 – Global summary for the decidability results

Intuitively, considering the property of reachability, it is possible to simulate a preT-PPN with a postT-PPN and vice-versa as underlined through the constructions of Figures 7.1 and 7.3. Informally, preT-PPNs and postT-PPNs are therefore equivalent from the point of view of reachability.

With P-PPNs, there is no possibility to generate or consume a parametric amount of tokens. The paramaterised amount of tokens is fixed in a place before the execution and there is no modelling features to replicate this parametric amount somewhere else in the net. This is an important modelling gap with postT-PPNs and preT-PPNs which may informally justify why existential reachability becomes decidable here.

The results are summed up in Table 7.2. The reader can notice that the decidability (or undecidability) of universal reachability in P-PPNs is not settled. We leave it as an open problem in this thesis. Considering the gap of modelling power between postT-PPNs and P-PPNs, it seems sensible to conjecture that universal reachability in P-PPNs is decidable and to this end, to study reachability proofs for Petri nets in order to adapt the reasoning to our parametric context. We will provide a small survey on the techniques of proof for reachability in the Chapter 9 about future work.

Part III
Conclusion and Future Work

CHAPTER 8

Conclusion

"Believe you can, you are halfway there."
— Theodore Roosevelt

Contents

The core objective of this thesis was to propose meaningful parametric infinite state systems with decidable decision problems in order to provide more realistic classes of models and to address more realistic verification issues. This goal was formalised through the following three research questions:

RQ_1 *How can we extend Petri nets with parameters in order to represent a whole family of nets through one given model?*

RQ_2 *How can we adapt decision problems to this parametric context? What problems are decidable? What are their complexities?*

RQ_3 *Can we answer the problem of the synthesis of parameters?*

To satisfy this goal at those three levels, this thesis presented contributions dealing with verification of parametric problems (universal, existential and synthesis) in Petri nets augmented by discrete parameters on their arcs or initial markings. We will now recall the main contributions presented throughout Part II following the organisation of those research questions.

8.1 Introduction of Parametric Petri Nets and their Subclasses

Our first concern expressed in RQ_1 was to provide some general models of Petri nets. The main motivation is to enhance the design stage of a concurrent system by a stage of early analysis. To this hand, we have introduced the use of discrete parameters in Petri nets. In this paradigm, called parametric Petri nets (or PPNs for short) parameters are involved on arcs (input or output arcs of transitions) or directly on the places through a parameterised initial marking. We introduced the intuitive syntactical subclasses of this parametric models by restricting the use of parameters to input arcs only (preT-PPNs), to output arcs only (postT-PPNs) or to places (P-PPNs). Note that a specific class where parameters are allowed on the arcs only such that the subset of parameters used on the input arcs and on the output arcs are disjoint, called distinctT-PPNs, is also considered in a concern of completeness of this survey. We studied the relations between those subclasses and proved that the class where parameters

are used only on arcs (T-PPNs) is as expressive as the general class of PPNs. Those parametric Petri nets provide more genericity in the design stage since they allow us to represent and analyse a family of concurrent systems through one model. This former point is the concern of RQ_2. Indeed, we had to adapt the most useful properties to this parametric context. To this end, we suggested two classic parametric families of well known decision problems involved in Petri nets analysis that are reachability, coverability and (simultaneous) boundedness problems. Those two families of decision problems can be observed based on the quantification of the values of the parameters. The existential problems ask whether there exists a valuation such that the instance of the parametric Petri net under this valuation satisfies the property, whereas the universal problems ask whether the property considered holds for every valuation. Concretely, this permits to understand if the parameterised parts of the net have an influence on the property considered leading to possible simplification in the design stage.

8.2 Study of Decidability Problems

With those new decision problems introduced, the second half of RQ_2 was of course to find which problems are decidable and in the affirmative case, to provide their complexities. We focused on the coverability problem, which can be used in Petri nets analysis to solve numerous safety problems. All the results that have been proved throughout this thesis are summed up in Table 8.1. We will here provide a quick overview of those results. We first proved that the coverability (and thus the reachability and the simultaneous unboundedness) parametric problems were undecidable for the generic classes of PPNs and T-PPNs. Therefore, we turn our interest to the syntactical subclasses in order to find decidable decision problems. In particular, we consider the case of preT-PPNs and postT-PPNs and proved some simple results of monotonicity allowing us to provide some straightforward proofs that directly derive from classic Petri nets analysis. This is the case for existential coverability and existential simultaneous unboundedness for preT-PPNs and universal coverability and universal simultaneous unboundedness for postT-PPNs. On the other hand, some results are more complex. Existential coverability for postT-PPNs is derived from a translation between this subclass and the class of ωPNs, whereas universal coverability for preT-PPN carry over some Rackoff upperbound to prove its ExpSpace-completeness. Note that we also provided extensions of the classic Karp and Miller algorithm to this parametric context for those two subclasses. It notably permits to provide a proof for the existential simultaneous unboundedness for postT-PPNs. Nevertheless this construction is also interesting since it is stronger than a simple black box which would answer if a marking is existentially (or universally) coverable. We are indeed able to compute the markings that are universally coverable or existentially coverable for any preT-PPN or postT-PPN (and P-PPN and distinctT-PPN by extension). With the concern to establish decision frontiers, we also provided proofs of undecidability based on reduction from Minsky 2-counters machines for different problems. Those results interestingly emphasise the modelling power gap between postT-PPNs and P-PPNs. Indeed, our results show that despite the strong link between those two subclasses, there exists a significant difference in terms of use of parameters. PostT-PPNs offer the possibility to generate a parameterised amount of tokens whereas P-PPNs are only able to consume tokens from a parametric initial amount of tokens. This might informally explain why, in the latter case, existential reachability becomes decidable which is a strong result. Our research work has some limitation and the results that are still missing will be outlined in the next chapter.

	$\mathscr{U}$-problem			$\mathscr{E}$-problem		
	Reach.	S.Unbound	Cov.	Reach.	S.Unbound	Cov.
PPN	Undecidable *(Th. 12)*	Undecidable *(Th. 12)*	Undecidable *(Th. 12)*	Undecidable *(Th. 11)*	Undecidable *(Th. 11)*	Undecidable *(Th. 11)*
T-PPN	Undecidable *(Th. 12)*	Undecidable *(Th. 12)*	Undecidable *(Th. 12 & Lem. 13)*	Undecidable *(Th. 11)*	Undecidable *(Th. 11)*	Undecidable *(Th. 11 & Lem. 13)*
preT-PPN	Undecidable *(Th. 50)*	ExpSpace-c *(Th. 24)*	ExpSpace-c *(Cor. 25)*	Undecidable *(Th. 49)*	ExpSpace-c*(Lem. 46)*	ExpSpace-c *(Corollary 16)*
postT-PPN	Undecidable *(Th. 48)*	ExpSpace-c*(Lem. 46)*	ExpSpace-c *(Corollary 16)*	Undecidable *(Th. 47)*	ExpSpace-h*(Cor. 29)*	ExpSpace-c *(Theorem 18)*
P-PPN	?	ExpSpace-c*(Lem. 46)*	ExpSpace-c *(Corollary 34)*	Decidable *(Th. 51)*	ExpSpace-h*(Lem. 46)*	ExpSpace-c *(Cor. 34)*
distinctT-PPN	Undecidable *(Th. 48)*	ExpSpace-c*(Lem. 46)*	ExpSpace-c *(Corollary 35)*	Undecidable *(Th. 47)*	ExpSpace-h*(Lem. 46)*	ExpSpace-c *(Corollary 35)*

Table 8.1 – Global Summary for the decidability results (with complexities)

	$\mathscr{S}$-problem		
	Reach.	S.Unbound	Cov.
PPN	✗	✗	✗
T-PPN	✗	✗	✗
preT-PPN	✗	✓	✓*(Lem. 39)*
postT-PPN	✗	✓	✓*(Lem. 39)*
P-PPN	?	✓	✓*(Lem. 13, 39)*
distinctT-PPN	✗	p	p *(Lem. 45)*

Table 8.2 – Global summary for the synthesis results (✓ stands for computable, p stands for no simple representation, ✗ stands for non computable in a formalism where the emptiness or the universality of the set can be decided and is a direct consequence of Th. 11 and 12)

8.3 Study of Synthesis Problems

As explained, the aim of parameters is also to help the designer of a system by providing a feedback on his modelling task. To this end, the most pertinent design issue that has to be considered is the computation of the parameter values ensuring the satisfaction of some expected properties, which is called the synthesis. Note that if the solution set of a synthesis problem can be computed in an explicit enough form, then both existential and universal versions of this problem can be directly answered. In this sense, the problem of the synthesis is more complex. Moreover, the knowledge of this set notably permits an estimation of the *robustness* of a given instance of a model. Therefore, RQ_3 is at the heart of this thesis. Using the monotonicity results from preT-PPNs and postT-PPNs, we proved that the solution set of the coverability synthesis satisfied a structure of downward closed sets and upward closed sets respectively. Those mathematical structures have been widely studied and, relying on the literature, given the bases of an upward closed set, it is possible to compute the bases of its complement (which is a downward closed set). Moreover, Valk and Jantzen provided an algorithm to compute a base of an upward closed set if some precise condition is satisfied. We showed that in our context, those two results allow us to reduce the computability of the solution of the coverability synthesis problem to the decidability of the universal coverability for preT-PPNs and the existential coverability for postT-PPNs. As explained in Chapter 6, this is a powerful result for those two syntactical subclasses of parametric Petri nets: one can indeed compute the set of all valid valuations of parameters, allowing to cover a given marking for parametric Petri nets where parameters are restricted to only input arcs or only output arcs. We recalled in Table 8.2 the synthesis problems such that their solution set is computable. We extended the results to the problem of simultaneous unboundedness since the arguments of monotonicity still hold for this property. It is important to underline the difference between "✗" and "p". The former, based on the undecidability of the decision problems claim that there is no computable representation in a formalism where emptiness or universality of a set can be decided (otherwise existential and universal problems would be decidable) whereas the latter claims that there is no computable representation in a formalism where the emptiness of the intersection with equality constraints is decidable.

We believe that being able to handle these parameterised models is an important advance in two ways. First, it significantly increases the level of abstraction in models, allowing to handle a much larger and therefore more *realistic classes of models*. Second, the existence of parameters can also address more relevant and *realistic verification issues*. Instead of just providing a binary answer on the satisfaction or not of an expected property, we can aim to *synthesise* 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. It is often challenging to find meaningful parametric infinite state systems with decidable decision problems. In this thesis, we introduced parametric Petri nets and investigated on those different axes. Nevertheless, this research opens up room for further improvements. In the next chapter, we propose some new directions that can be explored on this topic.

CHAPTER 9

Future Work

"What is past is prologue."

— William Shakespeare

Contents

In this chapter, we elaborate on different directions for future work. We have indeed identified several possible research directions to explore. We first discuss the direct continuity of this work by considering the results that remain to be proved such as universal coverability for P-PPNs or the complexity of universal simultaneous unboundedness for postT-PPNs. Then, we propose a reflexion based on a study made throughout this thesis on the implementation of a symbolic algorithm using constraints. Finally, we elaborate on the different theoretical extensions that could be considered first in a discrete context, and then in a continuous context.

9.1 Direct Continuation

As explained in the development of our thesis, some results are still missing. To complete our study it would be interesting to look in more details the proofs for the decidability of reachability in order to adapt one of them to answer to the problem of existential reachability in P-PPN, which remains open. We have a strong intuition that this problem is decidable using a result of [Leroux, 2013, Lemma XI.1.] stating that *Presburger subreachability sets are flatable*. This implies that given a marking m, we can test if some semilinear set is included in $Pre^*(m) = \{x \in \mathbb{N}^P \mid x \xrightarrow{*} m\}$. If we consider a P-PPN with a set of n parameters called $\mathbb{P} = \{a_1, \ldots, a_n\}$ and of initial marking $\mu = m_0 + \mu_0$ where $m_0 \in \mathbb{N}^P$ and $\mu_0 \in (\{0\} \cup \mathbb{P})^P$, this would imply the decidability of universal reachability of a marking m in this P-PPN by considering the linear set $\{m_0 + \sum_{i=1}^{i=n} k_i v_i(\mu_0) \mid \forall 1 \leq i \leq n,\ k_i \in \mathbb{N}\}$ where each v_i is the valuation that associates 1 to the parameter a_i and 0 otherwise.

Our study could also be completed by establishing the complexity upper bound of existential simultaneous unboundedness for postT-PPN which is currently solved using a procedure *à la* Karp and Miller. We have a strong intuition that the work proposed for ω-PNs in [Geeraerts et al., 2015] to extend Rackoff arguments coupled to the work of Stéphane Demri in [Demri, 2013] could be used to establish an ExpSpace upper bound.

Finally, it would be interesting to study the complexity of the synthesis procedure in order to find if some arguments *à la* Rackoff could help to provide an ExpSpace procedure, or if,

on the contrary, if it is possible to provide an Ackermannian upper bound (see, *e.g.,* [Figueira et al., 2011]). More specifically, we could provide a complexity analysis of the two procedures we suggested to answer the synthesis problem, that is the Valk and Jantzen procedure, and our recursive procedure involving partial bases, in order compare those two algorithms.

9.2 Toward a practical symbolic algorithm

The research work carried out for this thesis proposed a theoretical study. We suggest to investigate how a symbolic procedure could be developed to represent effectively a set of markings and to extend some operators as the computation of successors or the test of ordering of markings. To this end, it would be convenient to focus on some orders on powersets such as those studied by A. Marcone in [Marcone, 2001] written $\leq^{\forall}_{\exists}$, also studied in [Jančar, 1999] and used by P. Abdulla in [Abdulla and Nylén, 2000, Abdulla and Nylén, 2001]. Note that [Finkel and Goubault-Larrecq, 2012, Section 5], permits to build well quasi-orders as products of well quasi-orders.

A main limitation to this kind of symbolic representation is the apparition of quadratic constraints that cannot be handled using convex polyhedra or Presburger arithmetic. For instance, considering the net depicted in Figure 9.1, if a valuation if chosen, the net is bounded, but if we consider the parametric net, the reachable markings belong to the set $\{(1,0,0),(0,a,0)\} \cup \{(0,a-k,kb) \mid 0 \leq k \leq a\}$.

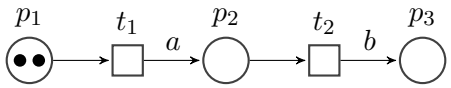


Figure 9.1 – Quadratic constraints on parameters in a postT-PPN

9.3 Discrete Extensions

Another axis to extend our work could be to provide discrete extensions to our model. First, one can easily imagine to allow some linear combinations of parameters on weights or initial markings instead of simple parameters and to study how decidability frontiers evolve. If we first consider that the linear combinations only use positive coefficients, we might find simple constructions to simulate those new nets using our simpler models. It is indeed possible to simulate the creation of such quantities using simple parametric arcs. Nevertheless, if negative coefficients are allowed, the theory is less intuitive and some choices should be made: Do we authorise negative valuations? Can we imagine a construction where the negative components of linear combinations for input arcs are turned into positive linear combinations on output arcs and vice-versa? And in the later case, can we relate those models to our simpler models?

It would also be a relevant extension to allow the user to verify parameterised properties. For instance, it would be meaningful to ask whether a given parametric marking can be covered. If such problem might be solved on P-PPNs and postT-PPNs where some markings can be initially parameterised or where a parametric number of tokens can be generated, it seems more complicated for the other subclasses that are preT-PPNs or distinctT-PPNs. Another interesting approach could be to introduce reset arcs or transfer arcs to our models in order to provide new scenarios. Note that reset arcs and transfer arcs make reachability undecidable but coverability is still decidable (see [Dufourd et al., 1998]).

9.4 Timed Extensions

Many quantitative extensions of Petri nets could be combined with our discrete parameters, such as probabilistic Petri nets [Emzivat et al., 2016] or continuous Petri nets [David and Alla,

1987]. In this section, we briefly discuss the case of time Petri nets [Merlin, 1974].

It is possible to extend Petri nets with time by attaching timing constraints to transitions. In this context, coverability is already undecidable even without parameters since it is possible to simulate inhibitor arcs (see [Jones et al., 1977]). Therefore we need to start with subclasses of time Petri nets. One possibility is to restrict the structure of the underlying untimed net: indeed [Akshay et al., 2016] considers structural subclass of time Petri nets where the untimed net is free choice (see *e.g.,* [Desel and Esparza, 2005]) and provides some decidability results, notably for the firability of a transition and for the termination of the net. It might thus be interesting to introduce parameters in this context.

Another possibility is to consider a weak time semantics (*i.e.* no urgency). In Timed Arc Petri nets, where the time corresponds to the age of tokens, with a weak semantics, coverability is decidable [Abdulla and Nylén, 2001]. It follows from [Boyer and Roux, 2008] that considering a weak time semantics gives similar results for other classes of Petri nets with time, in particular for time Petri nets.

CHAPTER 10

French Summary

"Ma patrie, c'est la langue française."

— Albert Camus

Contents

10.1 Introduction

Récemment, nous avons pu observer que notre société s'est tournée vers une ère du tout numérique. Dans ce cadre, les failles de sécurité ouvrent des brèches à des cyberattaques impactant directement notre vie. L'exemple récent de *WannaCry* en est la triste illustration. S'il est inquiétant de constater qu'une bonne partie de la population n'est pas consciente de l'impact désastreux que peut causer une erreur dans un système informatique, le véritable danger vient aussi du fait que nous manipulons des systèmes toujours plus complexes, toujours plus liés les uns aux autres. Leur compréhension précise et complète tend à nous échapper car elle n'est plus intelligible pour un cerveau humain. Les réseaux sans fils, les appareils connectés, les voitures autonomes, les interventions médicales assistées par ordinateur sont des systèmes dont la criticité n'est plus à démontrer : au delà de l'aspect financier, les systèmes informatiques peuvent avoir un impact direct sur nos vies. Il est de ce fait nécessaire de pouvoir s'assurer de la sûreté de tels systèmes, et ce malgré leur complexité croissante. Aussi, dans ce contexte, les méthodes formelles et la vérification de modèles semblent toutes désignées pour devenir une compétence clé de toute personne concevant de tels systèmes. Ces branches de l'informatique théorique traitent en effet de la modélisation de systèmes et de propriétés afin d'analyser exhaustivement et automatiquement si le modèle proposé satisfait ou non la propriété étudiée. Les outils de vérification de modèles ont déjà su faire leur preuve sur des cas d'étude concrets. Néanmoins, notre enthousiasme face à cette science doit être pondéré par la complexité des procédures mises

en œuvre. Souvent très gourmandes, elles peuvent aussi s'avérer inefficaces si l'utilisateur ne sait pas proposer de modèle complet du système étudié. Face à ces limitations, la vérification de modèles s'est vu étendue par l'ajout de paramètres. Dans la vérification paramétrée, le but est désormais de pouvoir réaliser une analyse du système même si sa spécification n'est pas entièrement connue. La modélisation se veut moins rigide et plus abstraite afin de proposer des phases d'analyses anticipées, desquelles des résultats sont tirés pour aider à la conception du système, et non pas uniquement pour le vérifier. Paradoxalement, nous souhaitons, en laissant des trous dans le système (les paramètres), en tirer une connaissance plus approfondie suite à son analyse.

10.1.1 Contributions

Afin de permettre une modélisation plus souple des systèmes, nous proposons de généraliser les réseaux de Petri par l'ajout de paramètres discrets représentant le poids des arcs ou le nombre de jetons présents dans les places. Dans ce modèle, tout problème de décision peut être décliné sous deux versions, une universelle, demandant si la propriété considérée est vraie quelles que soient les valeurs que prennent les paramètres et une existentielle, qui s'interroge sur l'existence de valeurs des paramètres telles que la propriété soit satisfaite. Concernant la couverture, nous prouvons que ces deux problèmes sont indécidables dans le cas général. De ce fait, nous introduisons des sous-classes syntaxiques basées sur la restriction des paramètres aux places, aux arcs en sortie ou en entrée des transitions. Dans ces différents cas, nous montrons que la couverture existentielle et universelle sont décidables et ExpSpace-complètes. Notre étude est alors élargie au problème de la synthèse de paramètres qui s'intéresse à calculer l'ensemble des valeurs de paramètres telles que la propriété considérée soit vraie. Sur les sous-classes introduites, concernant la couverture, nous montrons que les ensembles solutions à la synthèse ont des structures respectivement fermée supérieurement (cas des arcs de sortie) et fermée inférieurement (cas des arcs d'entrée). Nous prouvons alors que calculer ces ensembles est possible en appliquant un algorithme de la littérature, proposé par Valk et Jantzen, dont la condition d'application se réduit aux problèmes de décision étudiés précédemment. Nous terminons cette étude par la recherche de frontières de décision en étudiant les versions paramétrées de l'accessibilité pour nos sous-classes.

10.1.2 Organisation

Cette thèse débute par des rappels mathématiques utiles pour la théorie des réseaux de Petri ainsi que pour la compréhension des preuves présentées tout au long du manuscrit. Des rappels portant sur le modèle des réseaux de Petri et les techniques courantes d'analyse sont évoqués brièvement en Section 10.2. Nous introduisons par la suite le modèle des réseaux de Petri paramétrés ainsi que les problèmes de décision universel et existentiel. Nous prouvons alors que dans le cas général et pour le problème de la couverture, ces deux problèmes sont indécidables. Ceci est résumé en Section 10.3. Aussi, dans la Section 10.4, nous introduisons des sous-classes de ces réseaux paramétrés où l'utilisation de paramètres est restreinte aux arcs d'entrée, aux arcs de sortie ou aux places. Dans ces différents cas, les problèmes de décision précédemment étudiés deviennent décidables. Nous proposons alors des preuves de leur complexité. Comme résumé en Section 10.5, nous élargissons cette étude au problème de la synthèse et montrons que dans le cas de la couverture, pour les sous-classes où les paramètres sont autorisés sur des arcs d'entrée uniquement ou de sortie uniquement, l'ensemble solution au problème de la synthèse est calculable. Pour clore cette étude, nous considérons les versions paramétrées du problème de l'accessibilité dans ces mêmes sous-classes afin d'établir des frontières de décision. Ceci est l'objet de la Section 10.6.

10.2 Travaux Connexes et Notions Préliminaires

Plusieurs travaux traitent de modèles représentant un nombre arbitraire de processus identiques. D'une manière assez proche de nos travaux [Abdulla et al., 2013] présente notamment un framework dédié à la vérification de problèmes de sûreté. Concernant le modèle des réseaux de Petri et l'usage de paramètres, la littérature se veut quelque peu fragmentée. Les paramètres sont utilisés tantôt en vue de décrire différents niveaux de raffinements, tantôt afin de simplifier la représentation du modèle (ils sont alors à domaine fini) ou pour modéliser des évolutions du système au cours de son exécution (les valuations peuvent changer au cours des séquences d'exécution). Dans cette thèse, nous considérons des paramètres discrets à domaine infini (l'ensemble $\mathbb{N}$ des entiers naturels), représentant un unique niveau de raffinement et dont la valeur une fois établie pour l'exécution ne peut changer. Nous souhaitons en effet proposer un modèle général sur lequel des analyses quantitatives pourront être réalisées. Les modèles les plus proches de notre formalisme sont ceux présentés dans [Chiola et al., 1991] et, du point de vue de leur analyse, les ω-Petri nets introduits dans [Geeraerts et al., 2015].

Par la suite, nous abordons les notions mathématiques relevant de la théorie des ensembles et de la théorie des graphes. Nous revenons également sur l'algèbre, la théorie des ordres que nous approfondissons afin de considérer les notions d'espace fermé supérieurement et inférieurement, et la combinatoire élémentaire. Nous rappelons également des notions de la théorie de la décidabilité et de la complexité.

Comme expliqué, ce manuscrit se base sur le formalisme des réseaux de Petri servant à modéliser des systèmes concurrents. Un exemple de réseau de Petri est rappelé en Figure 10.1.

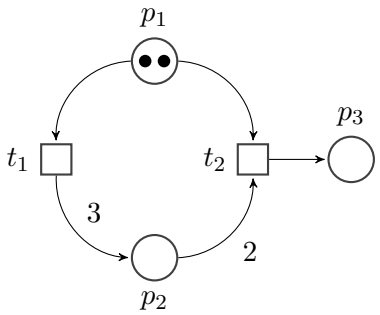


Figure 10.1 – Un réseau de Petri

Il comprend des places, représentées par les cercles, des transitions représentées par des carrés et des arcs les reliant. Un marquage correspond à l'état du réseau et est représenté par le vecteur du nombre de jetons par place (quitte à considérer un ordre sur les places). Une transition peut être franchie s'il y a suffisamment de jetons pour satisfaire la précondition (arcs entrant) de la transition, le tir de la transition consomme ces jetons et génère des jetons dans les places en sortie (autant de jetons que la valeur précisée sur l'arc et un jeton si aucune valeur n'est précisée). Nous abordons les propriétés usuelles que sont l'accessibilité, la couverture, la non bornitude, et la non bornitude simultanée. Nous présentons également dans ce manuscrit des techniques d'analyse et de preuve usuelles desquelles nous nous inspirerons dans les parties de contributions. Parmi ces techniques, on notera la procédure de Karp et Miller permettant de construire un arbre de couverture, ou la technique de preuve de Rackoff consistant à borner la taille de séquences d'exécution.

10.3 Introduction des Réseaux de Petri Paramétrés et Problèmes de Décision Associés

Nous proposons d'étendre le modèle des réseaux de Petri par l'ajout de paramètres discrets représentant le poids des arcs en entrée ou sortie des transitions ou les valeurs de jetons du

marquage initial. Un exemple est donné par la Figure 10.2.

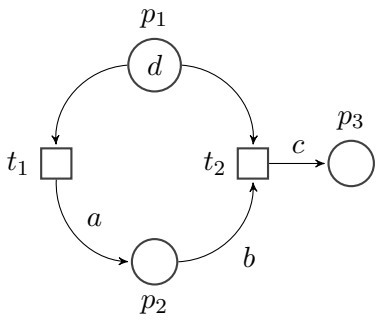


Figure 10.2 – Un Réseau de Petri Paramétré

Ce modèle est introduit comme *réseau de Petri paramétré* ou PPN. Nous proposons deux exemples illustratifs l'un concernant un modèle de prêt bancaire, l'autre relevant d'un problème d'optimisation linéaire sur une chaîne de production où le but est de minimiser le coût en matières premières. Étant donné une valuation des paramètres d'un réseau de Petri paramétré, le réseau obtenu en remplaçant toute occurrence d'un paramètre par la valeur qui lui est associée par cette valuation est appelé instance du réseau de Petri paramétré sous cette valuation. Si la valuation est une fonction totale de l'ensemble des paramètres, nous obtenons un réseau de Petri classique sur lequel nous pouvons appliquer la sémantique classique. Dès lors, étant donné un problème de décision, nous pouvons nous demander s'il existe une valuation telle que l'instance du réseau de Petri paramétré considéré satisfasse le problème, ou dualement, si ce problème est satisfait pour toute valuation des paramètres. Ces problèmes sont qualifiés respectivement de problème existentiel et universel. Nous prouvons alors que dans le cas général des PPNs, les problèmes de la couverture existentielle et de la couverture universelle sont indécidables. Ces preuves sont réalisées respectivement par réduction du problème de terminaison et de non bornitude des compteurs d'une machine à deux compteurs de Minsky, qui sont bien connus pour être eux mêmes non décidables. Nous proposons donc dans cette preuve de simuler une machine à deux compteurs via un PPN.

10.4 Sous-Classes Paramétrées et Décidabilité

Les problèmes de couverture paramétrés ayant été prouvés indécidables dans le cas général, nous nous employons à proposer des sous-classes syntaxiques de notre modèle où ces problèmes seront décidables afin de pouvoir les utiliser en pratique. Aussi, nous proposons de contraindre l'utilisation des paramètres:

- aux places uniquement (P-PPNs)
- aux poids des arcs uniquement (T-PPNs)
- aux poids des arcs en entrée des transitions uniquement (preT-PPNs)
- aux poids des arcs en sortie des transitions uniquement (postT-PPNs)
- aux poids des arcs uniquement et tel qu'un même paramètre ne puisse pas être utilisé à la fois sur un arc d'entrée et sur un arc de sortie (distinctT-PPNs)

À noter que cette dernière sous-classe est pertinente car c'est justement le fait d'utiliser le même paramètre en entrée et sortie d'une transition qui nous permettait d'effectuer le test à zéro dans la preuve d'indécidabilité précédente.

Si les inclusions syntaxiques entre ces sous-classes sont triviales, nous proposons une traduction évidente (bisimulation faible) des P-PPNs vers les postT-PPNs montrant par là même que les T-PPNs ont un pouvoir de modélisation aussi important que les PPNs. Nous proposons également une construction plus technique attestant d'une co-simulation faible des postT-PPNs, par les P-PPNs.

Par la suite, nous nous focalisons donc sur les deux sous-classes "élémentaires" que sont les preT-PPNs et les postT-PPNs. Nous prouvons tout d'abord des propriétés dites de monotonie. Dans le cas des preT-PPNs, pour toute valuation des paramètres v, étant donné un marquage m couvrable, alors m est toujours couvrable dans l'instance sous une valuation $v' \leq v$. Symétriquement, dans le cas des postT-PPNs, pour toute valuation des paramètres v, étant donné un marquage m couvrable, alors m est toujours couvrable dans l'instance sous une valuation $v' \geq v$.

Ces propriétés nous permettent d'établir des premiers résultats de décidabilité. En effet, la couverture existentielle des preT-PPNs et la couverture universelle des postT-PPNs peuvent être décidées par simple test de la propriété sur l'instance du réseau paramétré où tous les paramètres ont été évalués à zéro. Nous obtenons conjointement la complexité (ExpSpace) de ces problèmes de décision.

Nous proposons également deux preuves plus techniques pour la couverture universelle des preT-PPNs, et la couverture existentielle des postT-PPNs, respectivement par récurrence sur le nombre de places du réseau et par traduction vers les ω-PNs. À noter que si dans ce dernier cas, nous obtenons la complexité ExpSpacede la couverture existentielle pour les postT-PPNs, nous passons par une autre preuve plus complexe et plus générale dans le cas des preT-PPNs, mettant en avant le caractère ExpSpacede la non bornitude simultanée universelle et donc de la couverture universelle.

Nous prolongeons ces résultats par des extensions de la construction de Karp et Miller, pour ces deux sous-classes. En conclusion de cette première étude, nous élargissons ces résultats à la classe des distinctT-PPNs où la couverture existentielle et universelle sont également décidables.

10.5 Problème de la Synthèse

Dans le cadre des systèmes paramétrés, au delà des simples problèmes de décision, notre intérêt se porte sur le problème de la synthèse. Il s'agit dorénavant de calculer l'ensemble des valuations des paramètres tel que tout réseau évalué sous l'une de ces valuations satisfasse la propriété étudiée. En terme de vérification, la synthèse n'est plus un problème de décision, puisque sa solution est un ensemble de valuations et pas seulement un booléen. De ce fait, la synthèse permet d'obtenir des informations importantes sur le système.

Nous savons que pour le cas général des PPNs, le problème de l'existence est non-décidable. Aussi il n'est pas nécessaire de s'intéresser au problème de la synthèse dans ce cas. Nous nous focalisons donc sur les sous-classes que sont les preT-PPNs et les postT-PPNs. Nous avons démontré que ces deux sous-classes sont sujettes à des propriétés de monotonie. Grâce à ces propriétés, nous pouvons démontrer que, pour le problème de la couverture, les espaces solutions au problème de la synthèse sont fermés inférieurement et supérieurement respectivement pour les preT-PPNs et pour les postT-PPNs. Avec ces observations, nous mettons à profit une procédure proposée par Valk et Jantzen dont l'objectif est de calculer une base d'un ensemble fermé supérieurement. En effet, cette procédure possède une condition d'application nécessaire et suffisante qu'il nous est facile d'adapter à notre contexte. Pour les postT-PPNs, nous montrons que cette condition est équivalente à la décidabilité de la couverture existentielle. Pour les preT-PPNs, nous considérons la synthèse de la non couverture dont l'ensemble solution est fermé supérieurement en tant que complémentaire d'un ensemble fermé inférieurement. Dans ce cas, la condition est équivalente à la décidabilité de la couverture universelle. Nous pouvons alors calculer une base de son complémentaire (soit de l'ensemble solution à la synthèse de la couverture) par une procédure adaptée. Nous proposons également dans cette thèse un algorithme de synthèse afin de calculer directement la base de l'ensemble solution pour les preT-PPNs sans passer par le complémentaire. En toute logique, cet algorithme peut être adapté pour calculer la base de l'ensemble solution à la non couverture pour les postT-PPNs.

10.6 Frontières de Décidabilité

Nous nous intéressons à élargir cette étude en proposant des frontières de décidabilité pour les problèmes paramétrés. De ce fait, nous démontrons que les versions existentielle et universelle de l'accessibilité pour les preT-PPNs et les postT-PPNs sont indécidables en adaptant la construction simulant une machine à deux compteurs de Minsky proposée pour le cas général. Néanmoins, il est intéressant de noter que l'accessibilité existentielle est décidable pour la classe des P-PPNs. La preuve repose sur une construction classique de canon à jetons. Nous pouvons donc mettre en avant ici que les postT-PPNs, même s'ils semblaient proches des P-PPNs échappent à la décidabilité pour cette propriété. Une explication informelle pourrait être l'impossibilité de répliquer un montant paramétré de jetons: avec des postT-PPNs, il est possible de répliquer un montant paramétré de jetons, par le tir d'une transition dont l'un des arcs de sortie est paramétré, alors que les P-PPNs se limitent à la consommation d'une quantité initialement paramétrée de jetons.

10.7 Conclusion

Il peut être difficile de trouver des modèles paramétrés non bornés où des problèmes de décision sont décidables. Dans cette thèse, nous avons abordé ce problème sous l'angle des réseaux de Petri, modèle largement représenté dans le monde de la vérification des systèmes concurrents.

Nous pensons que l'analyse des systèmes paramétrés constitue une avancée considérable dans le monde de la vérification. En effet, ces systèmes permettent de proposer des modèles présentant un plus grand niveau d'abstraction, et ainsi représentatifs de classes de modèles plus larges et donc plus réalistes. Par ailleurs, comme nous l'avons vu, la vérification paramétrée ouvre la voie à des problématiques concrètes comme la synthèse. En connaissant l'ensemble des valeurs de paramètres telles que le système satisfasse une propriété donnée, il est possible d'établir une distance entre le système effectif et les valeurs limites, et ainsi d'estimer une robustesse de notre système. Ce manuscrit ouvre les portes à d'autres études, plus pratiques, mais également à l'extension de nos travaux par l'ajout d'aspects discrets (combinaisons linéaires de paramètres, propriétés paramétrées, ajout d'autres types d'arcs) ou de temps, de probabilité ou même la considération de réseaux de Petri continus.

Bibliography

[Abdulla et al., 2013] Abdulla, P., Haziza, F., and Holík, L. (2013). All for the price of few. In Giacobazzi, R., Berdine, J., and Mastroeni, I., editors, *Verification, Model Checking, and Abstract Interpretation*, volume 7737 of *Lecture Notes in Computer Science*, pages 476–495. Springer Berlin Heidelberg. 18, 19, 123

[Abdulla et al., 1996] Abdulla, P. A., Cerans, K., Jonsson, B., and Tsay, Y.-K. (1996). General decidability theorems for infinite-state systems. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 313–321. 35, 137

[Abdulla and Nylén, 2000] Abdulla, P. A. and Nylén, A. (2000). Better is better than well: on efficient verification of infinite-state systems. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No.99CB36332)*, pages 132–140. IEEE. 118

[Abdulla and Nylén, 2001] Abdulla, P. A. and Nylén, A. (2001). Timed petri nets and bqos. In Colom, J.-M. and Koutny, M., editors, *Applications and Theory of Petri Nets 2001: 22nd International Conference, ICATPN 2001 Newcastle upon Tyne, UK, June 25–29, 2001 Proceedings*, pages 53–70, Berlin, Heidelberg. Springer Berlin Heidelberg. 118, 119

[Agerwala and Flynn, 1973] Agerwala, T. and Flynn, M. (1973). Comments on capabilities, limitations and "correctness" of petri nets. In *Proceedings of the 1st Annual Symposium on Computer Architecture*, ISCA '73, pages 81–86, New York, NY, USA. ACM. 42

[Akshay et al., 2016] Akshay, S., Hélouët, L., and Phawade, R. (2016). Combining free choice and time in petri nets. In *2016 23rd International Symposium on Temporal Representation and Reasoning (TIME)*, pages 120–129. 119

[Alur et al., 1993] Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993). Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, pages 592–601. 19

[Aminof et al., 2014] Aminof, B., Kotek, T., Rubin, S., Spegni, F., and Veith, H. (2014). Parameterized model checking of rendezvous systems. In Baldan, P. and Gorla, D., editors, *CONCUR 2014 – Concurrency Theory: 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, pages 109–124, Berlin, Heidelberg. Springer Berlin Heidelberg. 18

[Araki and Kasami, 1976] Araki, T. and Kasami, T. (1976). Some decision problems related to the reachability problem for Petri nets. *Theoretical Computer Science*, 3(1):85–104. 42

[Avellaneda and Morin, 2012] Avellaneda, F. and Morin, R. (2012). Vector Addition Systems with States vs. Petri Nets. report, Laboratoire d'informatique Fondamentale de Marseille - LIF. 36

[Badouel and Oliver, 1999] Badouel, E. and Oliver, J. (1999). Dynamic Changes in Concurrent Systems: Modelling and Verification. report RR-3708, INRIA. 19

[Baier and Katoen, 2008] Baier, C. and Katoen, J. (2008). *Principles of Model Checking*. MIT Press. 15

[Berthomieu and Diaz, 1991] Berthomieu, B. and Diaz, M. (1991). Modeling and verification of time dependent systems using time petri nets. *IEEE transactions on software engineering*, 17(3):259–273. 42

[Bertrand et al., 2015] Bertrand, N., Fournier, P., and Sangnier, A. (2015). Distributed local strategies in broadcast networks. report, Inria Rennes. 18

[Biel et al., 2011] Biel, J., Macias, E., and Perez de la Parte, M. (2011). Simulation-Based Optimization for the Design of Discrete Event Systems Modeled by Parametric Petri Nets. In *2011 Fifth UKSim European Symposium on Computer Modeling and Simulation (EMS)*, pages 150–155. 19

[Bonnet et al., 2012] Bonnet, R., Finkel, A., and Praveen, M. (2012). Extending the Rackoff technique to Affine nets. *LIPIcs - Leibniz International Proceedings in Informatics*, 18. 41

[Bouajjani et al., 2000] Bouajjani, A., Jonsson, B., Nilsson, M., and Touili, T. (2000). Regular Model Checking. In Emerson, E. A. and Sistla, A. P., editors, *Computer Aided Verification*, number 1855 in Lecture Notes in Computer Science, pages 403–418. Springer Berlin Heidelberg. 18

[Bouajjani and Mayr, 1999] Bouajjani, A. and Mayr, R. (1999). Model checking lossy vector addition systems. In *16th Annual Symposium on Theoretical Aspects of Computer (STACS'09)*, pages 323–333, Trier, Germany. Springer. 26

[Boyer and Roux, 2008] Boyer, M. and Roux, O. H. (2008). On the compared expressiveness of arc, place and transition time petri nets. *Fundamenta Informaticae*, 88(3):225–249. 42, 119

[Bozga et al., 2001] Bozga, M., Mounier, L., and Lesens, D. (2001). Model Checking Ariane-5 Flight Program. In Ultes-Nitsche, S. G. U., editor, *6th International Workshop on Formal Methods for Industrial Critical Systems FMICS 2001*, pages 211–227, Paris, France. INRIA. 16

[Bozzelli and Torre, 2009] Bozzelli, L. and Torre, S. L. (2009). Decision problems for lower/upper bound parametric timed automata. *Formal Methods in System Design*, 35(2):121–151. 19

[Browne et al., 1989] Browne, M. C., Clarke, E. M., and Grumberg, O. (1989). Reasoning about networks with many identical finite state processes. *Information and Computation*, 81(1):13–31. 18

[Cardoza et al., 1976] Cardoza, E., Lipton, R., and Meyer, A. R. (1976). Exponential Space Complete Problems for Petri Nets and Commutative Semigroups (Preliminary Report). In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, pages 50–54, New York, NY, USA. ACM. 36

[Chiola et al., 1991] Chiola, G., Donatelli, S., and Franceschinis, G. (1991). On Parametric PT nets and their Modelling Power. *12th International Conference Application and Theory of Petri Nets (ICATPN'91)*, page 206. 20, 123

[Chiola et al., 1997] Chiola, G., Dutheillet, C., Franceschinis, G., and Haddad, S. (1997). A Symbolic Reachability Graph for Coloured Petri Nets. *Theor. Comput. Sci.*, 176(1-2):39–65. 19

[Christensen and Mortensen, 1997] Christensen, S. and Mortensen, K. H. (1997). Parametrisation of Coloured Petri Nets. *DAIMI Report Series*, 26(521). 19

[Couvreur et al., 2011] Couvreur, J.-M., Poitrenaud, D., and Weil, P. (2011). *Branching Processes of General Petri Nets*, pages 129–148. Springer Berlin Heidelberg, Berlin, Heidelberg. 71, 137

[David et al., 2015a] David, N., Jard, C., Lime, D., and Roux, O. H. (2015a). Discrete parameters in petri nets. In *Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings*, pages 137–156. 21

[David et al., 2015b] David, N., Jard, C., Lime, D., and Roux, O. H. (2015b). Discrete parameters in petri nets (informal presentation). In *2nd International Workshop on Synthesis of Complex Parameters, SynCoP 2015, April 11, 2015, London, United Kingdom*, pages 103–103. 21

[David et al., 2017] David, N., Jard, C., Lime, D., and Roux, O. H. (2017). Coverability synthesis in discrete petri nets. **(in press)**. In *28th International Conference on Concurrency Theory, CONCUR 2017, Berlin, Germany, September 4-9, 2017, Proceedings*, page to appear. 21

[David and Alla, 1987] David, R. and Alla, H. (1987). Continuous Petri nets. In *8th European Workshop on Application and Theory of Petri nets*, volume 340, pages 275–294. 19, 42, 119

[Delzanno, 2016] Delzanno, G. (2016). A unified view of parameterized verification of abstract models of broadcast communication. *International Journal on Software Tools for Technology Transfer*, 18(5):475–493. 18

[Demri, 2013] Demri, S. (2013). On selective unboundedness of VASS. *Journal of Computer and System Sciences*, 79(5):689–713. 37, 40, 41, 117, 137

[Desel and Esparza, 2005] Desel, J. and Esparza, J. (2005). *Free choice Petri nets*, volume 40. Cambridge university press. 119

[Dickson, 1913] Dickson, L. E. (1913). Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422. 26

[Dufourd et al., 1998] Dufourd, C., Finkel, A., and Schnoebelen, P. (1998). Reset nets between decidability and undecidability. In Larsen, K. G., Skyum, S., and Winskel, G., editors, *Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115. Springer Berlin Heidelberg. 52, 118

[Emzivat et al., 2016] Emzivat, Y., Delahaye, B., Lime, D., and Roux, O. H. (2016). Probabilistic Time Petri Nets. In Kordon, F. and Moldt, D., editors, *Application and Theory of Petri Nets and Concurrency: 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings*, pages 261–280, Cham. Springer International Publishing. 118

[Esparza et al., 2016] Esparza, J., Ganty, P., and Majumdar, R. (2016). Parameterized verification of asynchronous shared-memory systems. *J. ACM*, 63(1):10:1–10:48. 18

[Figueira et al., 2011] Figueira, D., Figueira, S., Schmitz, S., and Schnoebelen, P. (2011). Ackermannian and primitive-recursive bounds with dickson's lemma. In *Logic in Computer Science (LICS), 2011 26th Annual IEEE Symposium on*, pages 269–278. IEEE. 118

[Finkel and Goubault-Larrecq, 2009] Finkel, A. and Goubault-Larrecq, J. (2009). Forward analysis for WSTS, part I: Completions. In *26th Annual Symposium on Theoretical Aspects of Computer Science (STACS'09)*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 433–444, Freiburg, Germany. Leibniz-Zentrum für Informatik. 26

[Finkel and Goubault-Larrecq, 2012] Finkel, A. and Goubault-Larrecq, J. (2012). Forward analysis for WSTS, part II: complete WSTS. *Logical Methods in Computer Science*, 8(3). 26

[Finkel and Goubault-Larrecq, 2012] Finkel, A. and Goubault-Larrecq, J. (2012). The Theory of WSTS: The Case of Complete WSTS. In *Application and Theory of Petri Nets: 33rd International Conference, PETRI NETS 2012, Hamburg, Germany, June 25-29, 2012. Proceedings*, pages 3–31. Springer Berlin Heidelberg. 118

[Finkel and Leroux, 2015] Finkel, A. and Leroux, J. (2015). Recent and simple algorithms for Petri nets. *Software & Systems Modeling*, 14(2):719–725. 26, 36, 37, 65

[Finkel and Schnoebelen, 2001] Finkel, A. and Schnoebelen, P. (2001). Well-structured transition systems everywhere! *Theoretical Computer Science*, 256:63–92. 35, 137

[Geeraerts et al., 2015] Geeraerts, G., Heußner, A., Praveen, M., and Raskin, J. (2015). ω-Petri nets: Algorithms and complexity. *Fundam. Inform.*, 137(1):29–60. 20, 42, 43, 49, 64, 77, 117, 123, 137

[Goubault-Larrecq, 2009] Goubault-Larrecq, J. (2009). On a Generalization of a Result by Valk and Jantzen. Technical report, LSV. 26, 95

[Gracanin et al., 1993] Gracanin, D., Srinivasan, P., and Valavanis, K. (1993). Fundamentals of parameterized Petri nets. In *, 1993 IEEE International Conference on Robotics and Automation, 1993. Proceedings*, pages 584–591 vol.1. 19

[Hack, 1976] Hack, M. (1976). Petri net language. Technical report, Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, MA, USA. 42

[Hermanns, 2016] Hermanns, H. (2016). European commission : Cordis : Projects & results service : Power to the people. verified. `http://http://cordis.europa.eu/project/rcn/203431_en.html/`. [Online; accessed 18-July-2017]. 15

[Higman, 1952] Higman, G. (1952). Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(1):326–336. 25, 137

[Hopcroft and Pansiot, 1979] Hopcroft, J. and Pansiot, J.-J. (1979). On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159. 36

[Jančar, 1999] Jančar, P. (1999). A note on well quasi-orderings for powersets. *Information Processing Letters*, 72(5-6):155 – 160. 118

[Jensen, 1987] Jensen, K. (1987). Coloured petri nets. In *Petri nets: central models and their properties*, pages 248–299. Springer Berlin Heidelberg. 42

[Jones et al., 1977] Jones, N. D., Landweber, L. H., and Lien, Y. E. (1977). Complexity of some problems in Petri nets. *Theoretical Computer Science*, 4(3):277–299. 119

[Jovanović et al., 2015] Jovanović, A., Lime, D., and Roux, O. H. (2015). Integer Parameter Synthesis for Real-Time Systems. *IEEE Transactions on Software Engineering*, 41(5):445–461. 19, 101

[Karp and Miller, 1969] Karp, R. M. and Miller, R. E. (1969). Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195. 35, 36, 40, 85

[Kosaraju, 1982] Kosaraju, S. R. (1982). Decidability of reachability in vector addition systems (preliminary version). In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 267–281. ACM. 36

[Kruskal, 1972] Kruskal, J. B. (1972). The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297–305. 25, 137

[Lambert, 1992] Lambert, J. (1992). A structure to decide reachability in petri nets. *Theoretical Computer Science*, 99(1):79 – 104. 36

[Leroux, 2009] Leroux, J. (2009). The general vector addition system reachability problem by Presburger inductive invariants. In *2009 24th Annual IEEE Symposium on Logic In Computer Science*, pages 4–13. 36

[Leroux, 2013] Leroux, J. (2013). Presburger vector addition systems. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 23–32. 117

[Lindqvist, 1993] Lindqvist, M. (1993). Parameterized reachability trees for Predicate/Transition nets. In Rozenberg, G., editor, *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes in Computer Science*, pages 301–324. Springer Berlin Heidelberg. 19

[Marcone, 2001] Marcone, A. (2001). Fine Analysis of the Quasi-Orderings on the Power Set. *Order*, 18(4):339–347. 118

[Marsan et al., 1994] Marsan, M. A., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. (1994). *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition. 20

[Mayr, 1984] Mayr, E. W. (1984). An algorithm for the general petri net reachability problem. *SIAM Journal on computing*, 13(3):441–460. 36

[Merlin, 1974] Merlin, P. M. (1974). *A Study of the Recoverability of Computing Systems.* PhD thesis, Department of Information and Computer Science, University of California, Irvine. AAI7511026. 42, 119

[Milner, 1980] Milner, R. (1980). *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer. 41

[Minsky, 1967] Minsky, M. L. (1967). *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 55

[Park, 1981] Park, D. (1981). Concurrency and automata on infinite sequences. In Deussen, P., editor, *Theoretical Computer Science: 5th GI-Conference Karlsruhe, March 23–25, 1981*, pages 167–183, Berlin, Heidelberg. Springer Berlin Heidelberg. 41

[Petri, 1962] Petri, C. A. (1962). Kommunikation mit Automaten. Dissertation, Schriften des IIM 2, Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, Bonn. 33

[Rackoff, 1978] Rackoff, C. (1978). Rackoff, C.: The covering and boundedness problems for vector addition systems. Theoret. Comp. Sci. 6, 223-231. *Theoretical Computer Science*, 6(2):223–231. 37, 40, 41

[Ramchandani, 1973] Ramchandani, C. (1973). *Analysis of asynchronous concurrent systems by timed Petri nets.* PhD thesis, Massachusetts Institute of Technology. 42

[Reutenauer, 1989] Reutenauer, C. (1989). *Aspects mathématiques des réseaux de Petri*. Masson. 36

[Reynier and Servais, 2011] Reynier, P.-A. and Servais, F. (2011). Minimal coverability set for petri nets: Karp and miller algorithm with pruning. In *International Conference on Application and Theory of Petri Nets and Concurrency*, pages 69–88. Springer. 39, 40, 83

[Savitch, 1970] Savitch, W. J. (1970). Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192. 31, 76

[Turing, 1937] Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265. 29

[Valk, 1978] Valk, R. (1978). Self-modifying nets, a natural extension of Petri nets. In Ausiello, G. and Böhm, C., editors, *Automata, Languages and Programming*, number 62 in Lecture Notes in Computer Science, pages 464–476. Springer Berlin Heidelberg. 19

[Valk and Jantzen, 1985] Valk, R. and Jantzen, M. (1985). The residue of vector sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21(6):643–674. 21, 27, 93, 94, 95

List of Tables

List of Figures

List of Definitions

UNIVERSITE BRETAGNE LOIRE

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Nicolas DAVID

Réseaux de Petri à Paramètres Discrets

Discrete Parameters in Petri Nets

Résumé

Afin de permettre une modélisation plus souple des systèmes, nous proposons d'étendre les réseaux de Petri par des paramètres discrets représentant le poids des arcs ou le nombre de jetons présents dans les places. Dans ce modèle, tout problème de décision peut être décliné sous deux versions, une universelle, demandant si la propriété considérée est vraie quelles que soient les valeurs que prennent les paramètres et une existentielle, qui s'interroge sur l'existence d'une valeur pour les paramètres telle que la propriété soit satisfaite. Concernant la couverture, nous montrons que ces deux problèmes sont indécidables dans le cas général. Nous introduisons donc des sous classes syntaxiques basées sur la restriction des paramètres aux places, aux arcs en sortie ou aux arcs en entrée des transitions. Dans ces différents cas, nous montrons que la couverture existentielle et universelle sont décidables et EXPSPACE-complètes. Nous étudions alors le problème de la synthèse de paramètres qui s'intéresse à calculer l'ensemble des valeurs de paramètres telles que la propriété considérée soit vraie. Sur les sous classes introduites, concernant la couverture, nous montrons que les ensembles solutions à la synthèse ont des structures fermée supérieurement (cas des arcs de sortie) et fermée inférieurement (cas des arcs d'entrée). Nous prouvons alors que ces ensembles se calculent par un algorithme de la littérature, proposé par Valk et Jantzen, dont les conditions d'application se réduisent aux problèmes de décision étudiés précédemment. Enfin nous étudions les frontières de décision en nous intéressant aux versions paramétrées de l'accessibilité pour ces sous classes.

Mots clés

Réseaux de Petri, Paramètres, Synthèse, Décidabilité, Complexité, Couverture, Accessibilité.

Abstract

With the aim of increasing the modelling 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, the universal and the existential one: "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 nets, depending on whether the parameters are used on places, input or output arcs of transitions. For some classes, we prove that universal and existential coverability become decidable, making these classes more usable in practice. To complete this study, we prove that those problems are EXPSPACE-complete. We also address a problem of parameter synthesis, that is computing the set of values for the parameters such that a given marking is coverable in the instantiated net. Restricting parameters to only input weights (preT-PPNs) provides a downward-closed structure to the solution set. We therefore invoke a result for the representation of upward closed set from Valk and Jantzen. The condition to use this procedure is equivalent to decide the universal coverability. We also propose an adaptation of this reasoning to the case of parameters used only as output weights (postT-PPNs). In this case, the condition to use this procedure can be reduced to the decidability of the existential coverability. Finally, we broaden this study by establishing decision frontiers through the study of existential and universal reachability.

Key Words

Petri nets, Parameters, Synthesis, Decidability, Complexity, Coverability, Reachability.