Coverability and Expressiveness Properties of Well-Structured Transition Systems

Gilles Geeraerts

Thèse
présentée pour l'obtention du grade de
Docteur en Sciences
de l'Université Libre de Bruxelles
(Faculté des Sciences, Département d'Informatique)

Quaerendo invenies...

[...] ὅτι ἐρωτώμενοι οἱ ἄνθρωποι, ἐάν τις καλῶς ἐρωτᾷ, αὐτοὶ λέγουσιν πάντα ᾗ ἔχει καίτοι εἰ μὴ ἐτύγχανεν αὐτοῖς ἐπιστήμη ἐνοῦσα καὶ ὀρθὸς λόγος, οὐκ ἂν οἷοί τ´ ἦσαν τοῦτο ποιῆσαι.

Plato, *Phaedo*, 18.

This thesis has been written under the supervision of Prof. Jean-François Raskin (Université Libre de Bruxelles, Belgique). The Members of the Jury are:

- Prof. Bernard Boigelot (Université de Liège, Belgique)
- Prof. Ahmed Bouajjani (Université de Paris 7, France)
- Prof. Raymond Devillers (Université Libre de Bruxelles, Belgique)
- Prof. Thierry Massart (Université Libre de Bruxelles, Belgique)
- Prof. Joël Ouaknine (Oxford University, United Kingdom)
- Dr. Laurent Van Begin (Université Libre de Bruxelles, Belgique)

Résumé

Ces cinquante dernières années, les ordinateurs ont occupé une place toujours plus importante dans notre vie quotidienne. On les retrouve aujourd'hui présents dans de nombreuses applications, sous forme de *systèmes enfouis*. Ces applications sont parfois critiques, dans la mesure où toute défaillance du système informatique peut avoir des conséquences catastrophiques, tant sur le plan humain que sur le plan économique. Nous pensons par exemple aux systèmes informatiques qui contrôlent les appareils médicaux ou certains systèmes vitaux (comme les freins) des véhicules automobiles.

Afin d'assurer la correction de ces systèmes informatiques, différentes techniques de *Vérification Assistée par Ordinateur* ont été proposées, durant les trois dernières décénies principalement. Ces techniques reposent sur un principe commun: donner une description *formelle* tant du système que de la propriété qu'il doit respecter, et appliquer une méthode automatique pour *prouver* que le système respecte la propriété.

Parmi les principaux modèles aptes à décrire formellement des systèmes informatiques, la classe des *systèmes de transition bien structurés* [ACJT96, FS01] occupe une place importante, et ce, pour deux raisons essentielles. Tout d'abord, cette classe généralise plusieurs autres classes bien étudiées et utiles de modèles à espace d'états infini, comme les réseaux de Petri [Pet62](et leurs extensions monotones [Cia94, FGRVB06]) ou les systèmes communiquant par canaux FIFO avec pertes [AJ93]. Ensuite, des problèmes intéressants peuvent être résolus algorithmiquement sur cette classe. Parmi ces problèmes, on trouve le *problème de couverture*, auquel certaines propriétés intéressantes de sûreté peuvent être réduites.

Dans la première partie de cette thèse, nous nous intéressons au *problème de couverture*. Jusqu'à présent, le seul algorithme général (c'est-à-dire applicable à n'importe quel système bien structuré) pour résoudre ce problème était un algorithme dit *en arrière* [ACJT96] car il calcule itérativement tous les états potentiellement non-sûrs et vérifie si l'état initial du système en fait partie. Nous proposons *Expand, Enlarge and Check*, le premier algorithme *en avant* pour résoudre le problème de couverture, qui calcule les états potentiellement accessibles du système et vérifie si certains d'entre eux sont non-sûrs. Cette approche est plus efficace en pratique, comme le montrent nos expériences. Nous présentons également des techniques permettant d'accroître l'efficacité de notre méthode dans le cas où nous analysons des réseaux de Petri (ou une de leurs extensions monotones), ou bien des systèmes communiquant par canaux

FIFO avec pertes. Enfin, nous nous intéressons au calcul de l'ensemble de couverture pour les réseaux de Petri, un objet mathématique permettant notamment de résoudre le problème de couverture. Nous étudions l'algorithme de Karp & Miller [KM69], une solution classique pour calculer cet ensemble. Nous montrons qu'une optimisation de cet algorithme présenté dans [Fin91] est fausse, et nous proposons une autre solution totalement neuve, et plus efficace que la solution de Karp & Miller.

Dans la seconde partie de la thèse, nous nous intéressons aux *pouvoirs d'expression* des systèmes bien structurés, tant en terme de mots infinis que de mots finis. Le pouvoir d'expression d'une classe de systèmes est, en quelque sorte, une mesure de la diversité des comportements que les modèles de cette classe peuvent représenter. En ce qui concerne les mots infinis, nous étudions les pouvoirs d'expression des réseaux de Petri et de deux de leurs extensions (les réseaux de Petri avec arcs non-bloquants et les réseaux de Petri avec arcs de transfert). Nous montrons qu'il existe une hiérarchie stricte entre ces différents pouvoirs d'expression. Nous obtenons également des résultats partiels concernant le pouvoir d'expression des réseaux de Petri avec arcs de réinitialisation. En ce qui concerne les mots finis, nous introduisons la classe des *langages bien structurés*, qui sont des langages acceptés par des systèmes de transition bien structurés étiquetés, où l'ensemble des états accepteurs est clos par le haut. Nous prouvons trois *lemmes de pompage* concernant ces langages. Ceux-ci nous permettent de réobtenir facilement des résultats classiques de la littérature, ainsi que plusieurs nouveaux résultats. En particulier, nous prouvons, comme dans le cas des mots infinis, qu'il existe une hiérarchie stricte entre les pouvoirs d'expression des extensions des réseaux de Petri considérées.

Abstract

During the last fifty years, computers have been increasingly present in our daily life. They can be found in many applications, as *embedded systems*. These applications are sometimes critical, in the sense that any failure of the computer system can yield catastrophic effects, both from the human and the economical point of view. Examples of such systems are the computers that control medical appliances or some vital subsystems of cars (such as brakes).

In order to ensure correctness of such computer systems, several techniques of *Computer Aided Verification* have been proposed, mainly during the last thirty years. These techniques have a common principle: provide *formal* specifications of the system and of the property it has to enforce, and use some automatic method to *prove* that the property is fulfilled by the system.

Among the models that have been studied for describing computer systems, the class of *Well-Structured Transition Systems* [ACJT96, FS01] is one of the most important. Two main arguments can be provided in favour of that class. First of all, that class generalises several other well-studied an useful classes of infinite-state models, such as Petri nets [Pet62] (and their monotonic extensions [FGRVB06, Cia94]), or the lossy channel systems [AJ93]. Second, some interesting problems can be algorithmically solved on this class, such as the *coverability problem*, to which many *safety properties* can be reduced.

In the first part of the thesis, we consider the *coverability problem*. Hitherto, the only general algorithm (in the sense that it can be applied to any well-structured system) to solve the coverability problem was a so-called *backward* one [ACJT96]. A backward algorithm computes iteratively all the potentially unsafe states and check whether the initial state belongs to that set. We propose *Expand, Enlarge and Check*, the first *forward* algorithm for the coverability problem. That algorithm iteratively computes the states that are reachable from the initial one, and checks whether one of these states is unsafe. That approach is more efficient in practice, as our experiments show. We also explain how that method can be further improved in practice, when considering certain classes of systems, such as extensions of Petri nets, or lossy channel systems. Finally, we look into the computation of the *coverability set*, in the case of Petri nets. The coverability set is a mathematical tool that allows to solve the coverability problem. We consider the Karp & Miller procedure [KM69], a well-known solution to compute that set. We show that an optimisation of that procedure, introduced in [Fin91], is incorrect, and introduce a completely novel algorithm to compute the coverability set, that is more efficient than the Karp & Miller procedure.

In the second part of the thesis, we study *expressiveness properties* of the well-structured transition systems, in terms of infinite and finite words. The expressive power of a class of models can be regarded as a measure of the diversity of behaviours these models can express. As far as infinite words are concerned, we study the expressive power of Petri nets and two extensions of theirs, namely, the Petri nets with non-blocking arcs and the Petri nets with transfer arcs. We show that there exists a strict hierarchy between these different expressive powers. We also obtain partial results regarding Petri nets with reset arcs. As far as finite words are concerned, we introduce the class of *well-structured languages*, which are languages accepted by labelled well-structured transition systems, when upward-closed sets of accepting configurations are considered. We prove three *pumping lemmata* about these languages. These lemmata allow us to re-obtain easily some classical results of the literature, and to prove new ones. In particular, we prove, as in the case of finite words, that there exists a strict hierarchy between the expressive powers of the aforementioned extensions of Petri nets.

Acknowledgements

ALTHOUGH I am credited as sole author of this thesis, many people have been involved its elaboration. May these few lines be a tribute to their – voluntary or not – participation !

My foremost gratitude goes to Prof. Jean-François RASKIN, who has willingly accepted six years ago to be the promoter of my Master's thesis, and has thus become the promoter of the present PhD thesis, the year after. Asking whether his most precious quality is his professionalism or his cheerful attitude is, by far, an undecidable problem ! His insightful understanding of computer science, and, in particular, computer-aided verification, has allowed me to progress a lot. The numerous evenings spent listening together to classical music on his top-of-line hi-fi equipment will remain for me wonderful souvenirs and testimony of our friendship. Thank you, Jeff !

Most of the original content of this thesis is based on several papers, written in collaboration with (besides Jean-François) Laurent VAN BEGIN. This collaboration too has been to me an excellent experience, both from the professional and the humane point of view. I also take the opportunity to thank the anonymous reviewers who have read and rated our papers. Several improvements of our results come from the advices we have received from them.

Many thanks to Prof. Bernard BOIGELOT, Prof. Ahmed BOUAJJANI, Prof. Raymond DEVILLERS, Prof. Thierry MASSART, Prof. Joël OUAKNINE and Dr. Laurent VAN BEGIN for having kindly accepted to be members of the jury and review the thesis. This important task is always time consuming and deserves all my gratitude.

A special thanks has to be addressed to Prof. Raymond DEVILLERS, who has proofread the preliminary versions of this thesis in great details. Prof. DEVILLERS has become famous in the Computer Science Dept. as the ultimate 'subtle-error-finder', and his many comments and suggestions have unquestionably improved the overall quality of this document. The grain of sands I found between the pages of the last preliminary version I handed to him prove that Prof. DEVILLERS took his duty to read the document very seriously, even during his holidays, on the beach...

Several ideas that are developed in this thesis arose from fruitful discussions with fellow scientists, during conferences, scientific stays, or by e-mail. In particular, I would like to express my gratitude to Prof. Ahmed Bouajjani and Prof. Mihaela Sighireanu who have helped us with their implementation of the *Simple regular expressions* library; to Prof. Parosh Aziz Abdulla and to Dr. Johann Deneux, who have cheerfully welcomed me at the Uppsala University (Sweden) for a research stay; and finally to Prof. Peter Starke and Prof. Alain Finkel, who have kindly collaborated with us when we where looking into their algorithms to compute the minimal coverability set of Petri nets. I would like to emphasise that Prof. Starke has devoted time to help us although retired.

In addition to these direct contributions to the scientific work contained in this thesis, many other people have contributed to his elaboration by making my everyday life enjoyable enough that I could devote time and energy to the present work. In particular, it is a well-known fact that being the girlfriend of someone who writes a PhD thesis is *not* an easy task. Sandrine, however, has always put up with me, with constancy and love. She has been the one to support and hearten me every single time it was necessary. Without exception. Without condition. With love and care. She has accepted my weird work schedules, my laptop during the holidays, my bad mood when I had the feeling not to have progressed for the last who-knows-how-many weeks. I cannot tell for sure whether that thesis would have even been finished without her, and my gratefulness goes well beyond what words can express.

My colleague Eythan, *amicus optimus*, καλὸς κἀγαθός, has been one of my closest friends during these several years spent at the Computer Science Department. As friends and neighbours (for some time), we have spent lots of wonderful evenings discussing together. Thank you, Eythan !

My other colleagues of the Department as well as my friends contributed to create a daily atmosphere of fun and work too. Many thanks to all of you: Ahmed, Ana, Benjamin, Cédric, Claude-Robert, Fabienne, Frédéric, Frédéric, Gabriel, Gianluca, Gwenaël, Jean, Joël, Karim, Kevin, Laurence, Laurent, Léon, Marianne, Marie, Maryka, Max, Nicolas, Nicolàs, Olivier, Pascaline, Patrick, Pierre, Martin, Rachel, Robert, Rod, Sébastien, Stefan, Steve, Thierry, Véronique, Vincent, Walter, Yann-Aël, Yves,...

Domus helvetica tua fuit nobis, O Marci, domus felicitatis. Illic, magnissima pars huius operis scripta fuit. Illae lineae tibi gratias referant.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

"*Computers are stupid machines*". Everyone who had his first contact with a computer has probably been told the same. A computer lacks *intelligence*, by the human standards, because it is only a *machine for manipulating data according to a list of instructions*[1].

Nevertheless, we rely more and more on computers in our daily life. Computers are present everywhere. They control the braking systems of our cars. They care of our laundry by controlling our washing machines. They manage the balance of our bank account. They form the main components of any modern telephony or power supply network. They are sent to outer space (by rockets that they control), in order to carry out scientific experiments... However, despite the diversity of the tasks and applications that they are entrusted with, all the computers behave by executing programs that have been written by human beings.

The ubiquity of computing devices sets the problem of the *reliability* of computers. From the point of view of the programmer, this problem amounts to ensuring the *correctness* of the program that the computer has to execute in order to accomplish its task. Because computers are often present in *critical applications*, the correctness of the program is sometimes an unavoidable requirement: failures of such critical systems can sometimes be life-threatening, environmentally devastating, and/or tremendously costly.

It is a well-known fact that computers suffer from bug, or programming errors. The history of computing has retained the story of a moth getting trapped, on September 9th, 1945, between the relays of the Mark II computer at Harvard University, as *the first bug of history*. Ironically, the poor moth has been taped to the log book of the computer, and preserved since then (see Figure 1.1). It is interesting to remark that the log book entry reads *First actual case of bug found*, which clearly shows that the term *bug* was, by then, already in use to speak about a computer malfunction. Moreover, this

[1] `http://en.wikipedia.org/wiki/Computer`

malfunction is a hardware failure, and not directly related to a programming mistake (the most current meaning for *bug*). Besides this anecdotal fact, the short history of computing already provides us with a respectable amount of failures. Let us illustrate this by three famous examples:

- The first (unmanned) test flight of the *Ariane 5 space rocket*, which took place on June 4th, 1996, was a complete failure because of a software error. According to the report by the Enquiry Board [Ari96], the conversion of data from 64 bits floating point to 16 bits floating point during the first seconds of the flight caused an arithmetic overflow. As a consequence, the computer controlling the rocket's engines let it leave its trajectory, and the resulting aerodynamic forces caused the launcher to disintegrate.
- In 1985–1987, at least five patients died, and several other were severly injured, due to the malfunction of a radiation therapy device, the *Therac-25* [NT93]. Because of a race condition in the software that controlled the device, a rapid operator could accidentally order the *Therac-25* to deliver a massive and lethal dose of radiation.
- On January 15th, 1990, a large part of the AT&T telephony network in New York City went down, affecting about 60,000 people[2]. The failure was due to a bug in the new software to control long-distance switches that had just been installed. The switches would reboot every time they received a specific control message from another switch. Unfortunately, that specific control message was broadcast by any switch after rebooting. Thus, the reboot of a single switch would trigger a cascade of continual reboot on the whole network...

These situations show clearly how catastrophic a computer failure can be, both from the human and from the economical point of view. A recent report [NIS02] of the American National Institute of Standards and Technology has estimated that 'software bugs [...] are so detrimental that they cost the U.S. economy an estimated $59.5 billion annually [...]'.

Unfortunately, ensuring the correctness of programs is not an easy task. Some of the main difficulties that make the development of reliable software a difficult task are as follows:

- There is often an important gap between, on the one hand, the correctness criteria, that are most of the time fuzzy and expressed in some natural language and, on the other hand, the high degree of precision that is required when providing instructions to the computer. It is thus important to be able to specify the correct behaviours of the computer systems in an univocal way, corresponding to the true aims of the user.

[2]Source: `www.wired.com/news/technology/0,69355-1.html?tw=wn_story_page_next1`

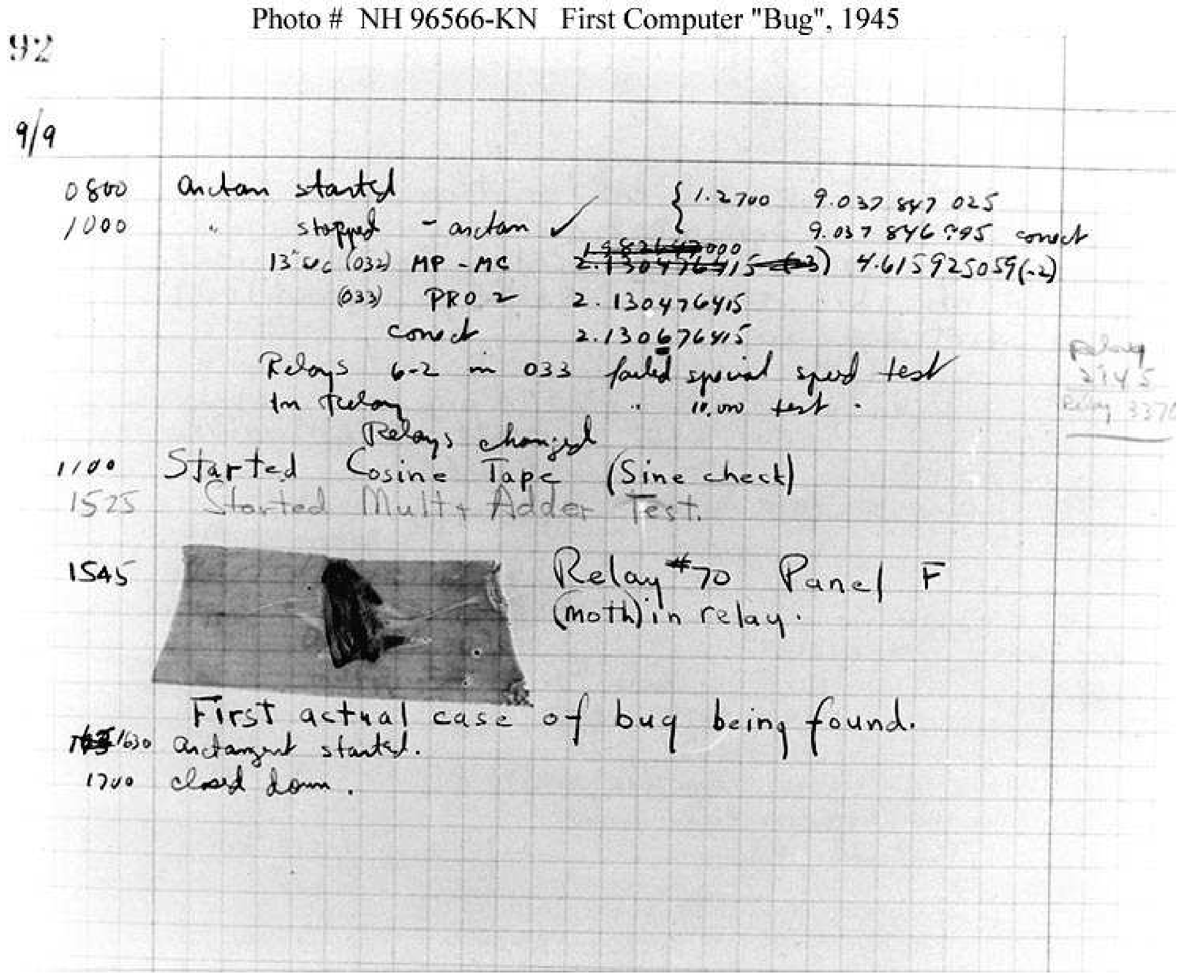


Photo # NH 96566-KN First Computer "Bug", 1945

92

9/9

0800 Anctan started { 1.2700 9.037 847 025
1000 " stopped - anctan ✓ 9.037 846 995 conect
13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415~~ (-3)) 4.615925059(-2)
(033) PRO 2 2.130476415
conect 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 11.000 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F (moth) in relay.
First actual case of bug being found.
~~1630~~ 1630 Antangent started.
1700 closed down.

Relay 2145
Relay 3370

Figure 1.1: An excerpt from the log book of the Mark II computer, on September 9th, 1945. This page shows a moth that has been trapped in a relay of the computer, causing it to malfunction. It has been removed and taped to the log book by the operators. This moth is widely regarded as *the first bug ever in computer science*. The log entry, however, reads *First actual case of bug found*, which shows that the term *bug* was already in use to design a malfunction of a computer. Actually, in a letter dated 1878, Thomas Edison already uses this word to mean a *little fault* or *difficulty*, according to the *computer bug* article on Wikipedia (`http://en.wikipedia.org/wiki/Computer_bug`).

- The amount of lines of source code for modern software can be huge. For instance, the single kernel of the Linux operating system has more that 1.5 million lines of source code. Such a quantity of information is difficult to manage by a single person. As a consequence, the development of large software is spread across teams of large dimensions (sometimes thousands of programmers). This is particularly true for open-source software, where the development is often massively cooperative.

- The advancement of programming languages and networking technology has promoted the development of *distributed* applications, in which the computing workload is distributed on several threads of executions, that interact and communicate. The exact semantics of the communication procedures, and the intricate interleaving of the threads make it a difficult task, for the programmer, to fully foresee all the possible behaviours of his program (at least, it is a much more difficult task than in the case of sequential, single-thread programs). Remark that the recent introduction of multi-core CPUs will dramatically increase the need for concurrent or multithread applications, in order to profit as much as possible from this new technology.

As a consequence, *software engineering* has greatly evolved, in order to cope with the growing demand for reliable and powerful computer systems. Complex methodologies (such as UML [Alh98],...) have been proposed in order to achieve a better analysis of the requirements, use cases, functionnalities, architectural issues... Tools to assist the designer and the programmer have been created, such as smart compilers that try to detect and foresee as many errors as possible at compile time; or integrated development environment that can, for instance, extract the canvas of C++ classes from an UML class diagram, thereby improving consistency between design and implementation. The use of strongly typed languages, such as ADA, for the development of sensitive applications has been promoted. Extensive testing of the various components of the system is nowadays part of every development cycle. And so forth...

However, although all those methods greatly improve our confidence in the final product, they are *not* sufficient. *Testing*, for instance, chiefly consists in defining typical sequences of inputs, i.e., test cases, and observing the behaviour of the system when provided with these inputs. Unfortunately, *test cases* usually do not cover all the possible situations that can arise during the lifetime of the system. For instance, race conditions that arise from the interleaving of several threads of executions are not easily detected by tests. Moreover, these test cases are often designed by humans who are – willingly or not – partial in their selection. As an example, it has been reckoned that the failure of the Ariane 5 software could have been detected before the launch, by proper testing [Ari96].

A major alternative to testing is offered by the family of *systematic verification techniques*. In this approach, the system under analysis and the requirement are both expressed formally, and the goal is to actually *prove* that the system meets its requirement thanks to the formal specifications[3]. Unfortunately, this general idea of *proving* the correctness of the system *as is* is not applicable in practice. Writing the proof of a program is a very tedious, error-prone task for a human being, even for a few lines of code. Moreover, it is well-known, since the seminal works of Turing [Tur36] that this approach cannot be made fully automatic, because basic problems such as the halting problem, for instance, are *undecidable*. As a matter of fact, Rice's theorem [Ric53] states that any non-trivial property of computable functions is undecidable.

Since a direct analysis of the system is not feasible, all the formal verification methods rely on the notion of *model of the system*, for which the property to verify is decidable. A suitable model of the system has to be fine enough in order to reflect faithfully the behaviours of the system, but also coarse enough in order to maintain decidability of the property to check[4]. Hence, it is particularly important to have at our disposal a broad range of classes of models that are suitable to express various kinds of computer systems. It is also absolutely crucial to measure as precisely as possible the respective expressive powers of these models, and to know which properties are decidable.

Nevertheless, when a suitable class of models has been fixed, the great advantage of a systematic verification technique on testing is that systematic verification is able of guaranteeing that the model of the system meets the specification, whereas a certain degree of uncertainty still remains with testing, because some cases may not have been covered. Systematic verification techniques, and formal methods in general, are being developed for the last thirty years, and are nowadays widely reckoned as an effective way of raising our confidence in a piece of hardware or software. Their importance is such that formal methods are extensively used in the hardware industry: companies such as Intel, SUN, Motorola and IBM develop model-checking tools [Ger01]. These methods also begin to be accepted by the software industry (at Microsoft for instance [BR02, AQR$^+$04]), and known by the public (for instance, two recent articles [Jac06, Sti06] of the science and technology journal *Scientific American* deal with verification techniques).

One of the most successful technique for systematic verification is known as *model-checking*[5] [CGP99]. It has been introduced around 1980, independently by Clarke and Emerson [CE81]; and Queille and Sifakis [QS82]. In this approach, the possible

[3]In general, all the techniques that are based upon the formal specification of the system and the requirements are called *formal methods*. Remark that these formal models can also be used to perform (rigorous) testing. Recently, several works have developped *formal approaches to testing* (see, for instance, [HNRW06]). As a consequence, the family of formal methods nowadays embodies testing techniques too.

[4]Remark that another advantage of analysing a model of the system instead of the system itself, is that the dimension of a model is usually much smaller than that of the whole system, which makes the process of verification more efficient in practice (see also the discussion on model-checking hereunder).

[5]Our presentation of the history of model-checking is inspired from the introduction of [CGP99], and closely follows it.

behaviours of the system are encoded by a finite-state automaton, whose transitions are labelled by actions of the system. This automaton can be regarded as defining finite word languages (when we are interested in reachability properties, for instance) or infinite word languages (when we are interested in response properties[6], for instance). The language defined by the automaton is thus a *model* of all the possible behaviours (traces, path of executions) of the system. On the other hand, the requirement is usually specified under the form of a formula from some linear-time or branching-time logic (such a LTL [VW86], CTL [CE81] or CTL* [CES83]). The formula defines a set of words which corresponds to (an over-approximation of) the behaviours of the systems that are acceptable.

Hence, with this framework, the correctness can be assessed by extracting a model (finite state automaton) from the system, formalising the requirement under the form of a formula, and checking that the automaton *satisfies* the formula. This last step boils down to checking that the language of the automaton is included in the set of words defined by the formula. If it is the case, we have proved that the (model of) the system meets its requirements. Otherwise, the model-checking procedure is able to expose a word accepted by the automaton but rejected by the formula. That word can then be interpreted as a trace of the system, and is thus a practical counter-example that can be used to debug the system.

In practice, one of the major impediments to the development of model-checking has been the so-called *state explosion problem*. Indeed, the size of the automaton that defines the system can sometimes be huge and intractable (especially when the system is specified under the form of *several separated* automata that correspond to the sub-components of the system, and have to be *synchronised* into one single automaton). Moreover, the formula itself is usually translated into an automaton, which can be too large to manage. The state explosion problem, however, has been quite efficiently dealt with thanks to the introduction of symbolic methods [McM93] based on OBDD representations [Bry86], or partial order reductions [God96]. It is now become possible to verify systems with more than 10^{120} states [CGP99], and *model-checking* can thus be regarded as a quite mature technology. Several success stories, such as the verification of a component from NASA's deep space one mission [HLP01], have confirmed this maturity.

Nevertheless, model-checking is not the silver bullet that can solve all the problems in verification. Its main limitation comes from the relatively poor expressiveness of finite state automata that are used to model the system. Indeed:

- It is sometimes interesting to reason about *real-time* properties of the system. Integrating temporal information about the behaviour of the system in a finite state automaton is not trivial.

[6]A *response property* states, roughly speaking, that each event of a certain type will eventually be followed by an event of a certain type.

- Some systems do not easily admit finite state models, and are more conveniently modelled by *infinite state* models. For instance, parametrised systems, which are made up of an unbounded number of copies of the same process running in parallel, and whose correctness has to be ensured for any number of processes. Instances of such systems are to be found among mutual exclusion protocols, cache coherency protocols…

- Distributed systems often use complex communication primitives. For instance, the JAVA programming language [Lea00] allows to write *multi-thread* programs, where the threads can contain atomic blocks and synchronise thanks to the `notify` and `notifyAll` methods. The semantics of these synchronisation primitives is not easy to express under the form of a finite state automaton.

The first of these three limitations has been attacked by Alur and Dill, who have introduced the theory of *timed automata* [AD94, Alu99]. A timed automaton is a finite state automaton extended with real-valued clocks. The theory of *hybrid systems*, proposed by Thomas Henzinger [Hen96], extends that of timed automata, by integrating in the model continuous variables whose derivative is arbitrary. Hybrid systems are thus a mix between discrete (finite automaton) and continuous (real variables) control. Since continuous models are outside the scope of this thesis, we refer the interested reader to the aforementioned works for further reading.

As far as the infinite number of states and the communication procedures are concerned, several (mostly recent) works have to be mentioned:

- In [Pet62], Karl Adam Petri introduces the model of Petri nets, which allows to model infinite state concurrent systems. In [KM69], Karp and Miller provide an algorithm to compute a *coverability set* of a Petri net. That set allows to decide, for instance, certain safety properties[7].

- In [Fin90], Alain Finkel studies a class of infinite state systems for which certain safety properties are decidable. This class is very close that of well-structured transition systems (see hereunder).

- In [Fin91], Finkel presents an improvement of the aforementioned Karp&Miller procedure.

- In [GS92], German and Sistla propose a method for verifying certain safety properties of concurrent systems containing an arbitrary number of identical processes that communicate through *rendez-vous* synchronisations. Their method relies on *counting abstraction*, which consists in building a model whose states retain how many processes are in each process state, thereby forgetting about their individual identities.

[7]A safety property states, roughly speaking that *something bad* will never happen. It is thus a special case of reachability.

- In [AJ93], Abdulla and Jonsson introduce a method to verify certain safety, reachability and eventuality properties of systems consisting of finite-state processes that communicate through unbounded lossy FIFO channels. These systems are called *lossy channel systems*. Since the channels are unbounded, these systems are infinite state.

- In [BW95, Boi99] Wolper and Boigelot describe how finite-state automata can be used to (finitely and compactly) encode (possibly infinite) sets of vectors of integer numbers satisfying a given Pressbürger arithmetic formula [Pre29]. This new datastructure, called Number Decision Diagram (or NDD for short) allows, together with the concept of *meta-transition* (a loop acceleration technique), to verify certain kinds of infinite-state systems. NDD (and certain variations such as QDD) have been implemented in the Lash toolkit [LAS], and succesfully used to verify infinite-state systems (communication protocols for instance).

- In [JK95], Jonsson and Kempe present a method to verify '*safety properties of a class of infinite state distributed algorithms*'. This class of systems is suitable to model, for instance, telephony networks.

- In [ACJT96] (see also [FS01]), an important step is made by Abdulla, Cerans, Jonsson and Tsay. In this work, they introduce the theory of *Well-Structured Transition Systems* that naturally generalises several infinite state models such as Petri nets and their monotonic extensions [Cia94]. They show that several interesting properties, such as the verification of certain safety properties, are decidable on this class. Since then, several new classes of models that are well-structured have been proposed.

- In [ABJ98], Abdulla, Bouajjani and Jonsson study the *lossy channel systems* in the framework of well-structured transition systems. They investigate forward analysis of these systems, and provide a new datastructure, the *simple regular expressions*, to finitely represent possibly infinite sets of reachable configurations. These algorithms have been implemented in the tool TReX [ABS01].

- In [EN98, EFM99], Emerson, Namjoshi, Esparza, Finkel and Mayr study *broadcast protocols*, a class of well-structured transition systems that are made up of an unbounded number of finite state processes that communicate through *rendez-vous* and *broadcast*.

- In [Del00], Delzanno uses broadcast protocols to model and verify cache coherency protocols.

- In [DR00, DRVB01] Delzanno, Raskin and Van Begin introduce efficient datastructures to store upward-closed sets of tuples of integers, which allows them to obtain an empirically efficient version of the algorithm introduced in [ACJT96], on Petri nets.

- In [AN01], Abdulla and Nylén introduce the *Timed Petri Nets*, another class of well-structured transition systems. That class extends the plain Petri nets by assigning real-valued ages to the tokens, that can be tested by the transitions. Such systems thus enjoy two dimensions of infinity: the unbounded number of tokens, and the unbounded values of the clocks. Timed Petri nets have been further studied in [ADMN04].

- In [DRVB02], Delzanno, Raskin and Van Begin use monotonic extensions of Petri nets, called Multi-Transfer Nets, in order to obtain models of multithread Java programs that can capture the semantics of `NotifyAll` (broadcast communication) and `Notify` (non-blocking *rendez-vous*). They instantiate the algorithm of [ACJT96] to that class, and rely on the efficient datastructure of [DR00, DRVB01] to decide certain safety properties.

- In [BFLP03], Bardin, Finkel, Leroux and Petrucci introduce fast, a tool to verify (without guarantee of termination) infinite state systems whose states are tuples of integers representing the values of (finitely many) counters, and whose transitions assign to counters arbitrary linear combination of the current counters' valuation (remark that some of these systems are well-structured, but not all of them). Their technique, rather similar to that of Boigelot and Wolper [Boi99] (see above) is based on the acceleration of loops.

- In [BH05], Bingham introduces new general methods to verify well-structured transition systems.

It should now be clear that *Well-Structured Transition Systems* form a class of infinite state systems which are worth of interest. Indeed, they naturally generalise several other classes that have been proposed as successful ways to model various types of computer systems. Moreover, the paper [ACJT96] has shown that several interesting properties are decidable on that class, which justifies the ever growing interest of the computer aided verification community in them.

The *purpose of this thesis* is to provide new results about *well-structured transition systems*, and, in particular, about the verification of their *coverability properties*, and their *expressiveness*. Our interest in these two specific points will be fully justified along Chapter 2 and Chapter 3. Nevertheless, let us already evoke these arguments here:

- The *coverability problem* consists in asking whether a certain set of (bad) configurations[8] is reachable from the initial state of the system. This problem is meaningful in practice because many *safety properties* can be reduced to it. It has been shown decidable in [ACJT96], by means of a *backward algorithm*, i.e.,

[8]It has to be *upward-closed*

an algorithm that iteratively computes the set of all the configurations that can reach the set of bad configurations in zero or more steps, and tests whether the initial configuration belongs to that set. A symmetrical (forward) algorithm could work by computing the set of all the configurations that are reachable from the initial one and test whether that set intersects with the bad ones. However, to the best of our knowledge, no general forward algorithm exists for well-structured transition systems. This is one of the open questions that we are about to address in the first part of this thesis (see the contents, hereunder).

- As in the case of finite automata in the framework of model checking, one can label the transitions of well-structured transition systems by actions of the program that has to be verified. It is then natural to study the *expressiveness* of these systems and their subclasses (such as Petri nets), because the expressiveness of a certain class of model is a good measure of its expressive power, i.e., an indication of which behaviours can be expressed by models of the class. We will see that several questions remain open regarding expressiveness of well-structured transition systems.

Plan of the thesis The thesis is organised as follows. It is divide into two parts, besides the two introductory chapters.

In Chapter 2, we recall all the preliminary notions that are necessary for the rest of the discussion. In particular, we recall various notions of orderings and explain how to handle closed sets. Then, we define well-structured transition systems and identify several of their subclasses, such as Petri nets, Extended Petri nets, Self-Modifying Petri nets, Lossy Channels Systems, Broadcast Protocols and Timed Petri nets. Finally, we define several decision problems regarding these systems and explain how these systems can be used to define languages. In Chapter 3, we recall the state of the art (the results presented there closely follow our short summary of the literature, in the present introduction). We recall, in particular, the Karp&Miller procedure, as well as decidability and expressiveness results.

In the *first part* of the thesis, we discuss coverability properties. We introduce, in Chapter 4, the *Expand, Enlarge and Check* schema of algorithms, which is the first *forward* algorithm to decide coverability on the whole class of well-structured transition systems (under some reasonable effectiveness requirements). In Chapter 5, we discuss several issues related to the practical implementation of this algorithm, in the case of Self-Modifying Petri nets, and Lossy Channel Systems. We provide experimental evidence of the efficiency of this new approach. In Chapter 6, we address the coverability problem on the specific class of Petri nets, by considering the improved algorithm of [Fin91], to compute the minimal coverability set of Petri nets. We show that this algorithm is incorrect, and may compute an under-approximation of the reachable states. We propose a new algorithm to compute a coverability set of Petri nets, based on a novel approach, and show that it is reasonably efficient in practice.

In the *second part* of the thesis, we discuss expressiveness properties. In Chapter 7, we study the expressiveness of extended Petri nets in terms of infinite word languages, and strictly separate the expressive powers of their different subclasses. In Chapter 8, we deal with expressiveness of well-structured transition systems in terms of finite words languages. For that purpose, we introduce the notion of *well-structured language*, and prove several of their properties. In particular, we provide three pumping lemmata that allow us to strictly separate the expressive powers of subclasses of extended Petri nets (in terms of finite words languages, this time).

We close the thesis by drawing a short conclusion in Chapter 9.

Chapter 2

Preliminaries

This first chapter recalls the basic notions that are studied along the thesis. After having fixed several mathematical notations, we discuss, in Section 2.2 various ordering notions and explain how *closed sets* (wrt to these orderings) can be finitely represented (in order to manipulate them algorithmically).

Then, in Section 2.3, we recall the notion of transition system and in particular, that of *well-structured* transition system (WSTS for short), topic of this thesis. We present several peculiar classes of WSTS that have been addressed in the literature:

- Monotonic extensions of Petri nets. We discuss the Self-Modifying Petri nets introduced in [Val78] and the Extended Petri nets [FGRVB06]. These models are useful to represent *counting abstractions* of parametrised systems [GS92, Van03].

- The Lossy Channel Systems [ABJ98] that have been introduced to model communication protocols over unreliable channels [AAB99].

- The Broadcast Protocols [EN98] that can model parametrised systems that communicate through *rendez-vous* and *broadcast* communications.

- The Timed Petri nets [ADMN04].

After this, we introduce in Section 2.4 and Section 2.5 several concepts that are relevant in the second part of this thesis, where we study *expressiveness properties* of WSTS. We recall several notions of *language theory* and introduce the notion of *labelled* WSTS.

We close the chapter by stating several decidability problems regarding behavioural and expressiveness properties of WSTS. In particular, we introduce the *coverability problem*, which is studied with more details in the first part of this thesis.

2.1 Mathematical notations

This section recalls some basic mathematical notations and conventions that will be used throughout this thesis.

- $\mathbb{N}$ denotes the set of natural numbers $\{0, 1, 2, \ldots\}$.
- $\mathbb{N}_0$ denotes the set of strictly positive natural numbers, i.e., $\mathbb{N} \setminus \{0\}$.
- $\mathbb{Z}$ denotes the set of integer numbers $\{0, 1, 2, \ldots\} \cup \{-1, -2, \ldots\}$.
- R.E. denotes the set of Recursively Enumerable languages [HMU01].
- For any set S, $|S| \in \mathbb{N} \cup \{+\infty\}$ denotes the cardinality of S, i.e., the number of elements in S.
- Given a set S, and a relation $\mathcal{R} \subseteq S \times S$, $\mathcal{R}^+$ denotes the transitive closure of $\mathcal{R}$. That is, $\mathcal{R}^+$ is the set of all the pairs (s_1, s_2) s.t. either (i) $(s_1, s_2) \in \mathcal{R}$; or (ii) there are $k \in \mathbb{N}_0$ elements $r_1, r_2, \ldots, r_k$ s.t. $(s_1, r_1) \in \mathcal{R}$, $(r_k, s_2) \in \mathcal{R}$ and, for any $1 \leq i < k$: $(r_i, r_{i+1}) \in \mathcal{R}$.
- Given a set S, and a relation $\mathcal{R} \subseteq S \times S$, $\mathcal{R}^*$ denotes the reflexive and transitive closure of $\mathcal{R}$. That is, $\mathcal{R}^* = \mathcal{R}^+ \cup \{(s, s) \mid s \in S\}$.

2.2 Handling closed sets

In this section, we introduce various notions of orderings (such as preorders, partial orders, well-quasi orders,...). Then, we define, in section 2.2.2 the notion of upward– and downward– closed sets (wrt to an ordering $\overline{\preceq}$). The special form these sets enjoy allows us to represent them in a finite and easily manipulable fashion. We explain how to obtain such a finite representation.

2.2.1 Orderings

Let us recall the notions of preorder (or quasiorder), partial order, total order, well-founded order and well-quasi order.

Definition 2.1 (Orderings) *Let S be a (possibly infinite) set. A relation $\lesssim \subseteq S \times S$ is*

a preorder (or quasiorder) *iff $\lesssim$ is reflexive and transitive, i.e., for any $s \in S$, $(s,s) \in \lesssim$ (reflexivity) and for any $s_1, s_2, s_3 \in S$: $(s_1, s_2) \in \lesssim$ and $(s_2, s_3) \in \lesssim$ implies that $(s_1, s_3) \in \lesssim$ (transitivity).*

a partial order *iff $\lesssim$ is a preorder and $\lesssim$ is antisymmetric, i.e., for any $s_1, s_2 \in S$: $(s_1, s_2) \in \lesssim$ and $(s_2, s_1) \in \lesssim$ implies that $s_1 = s_2$.*

a total order *iff $\lesssim$ is a partial order and for any $s_1, s_2 \in S$, either $(s_1, s_2) \in \lesssim$ or $(s_2, s_1) \in \lesssim$.* ■

In either cases, we say that $\lesssim$ is an ordering[1] (or order) on S. In this thesis, we will adopt the convention to write $s_1 \lesssim s_2$ to mean that the pair (s_1, s_2) is in the relation $\lesssim$. We write $s_1 \not\lesssim s_2$ when $(s_1, s_2) \notin \lesssim$. We also write $s_1 \lnsim s_2$ when $s_1 \lesssim s_2$ and $s_2 \not\lesssim s_1$. This holds for any order $\lesssim$ we will have to deal with. Remark that in the case of a partial order, for any $s_1, s_2 \in S$, we have $s_1 \lesssim s_2 \wedge s_2 \lesssim s_1$ implies $s_1 = s_2$. However, the reverse implication also holds by the reflexivity property. Hence, if $\lesssim$ is a partial order, $s_1 \lesssim s_2 \wedge s_2 \lesssim s_1$ iff $s_1 = s_2$. As a consequence $s_1 \lnsim s_2$ implies that $s_1 \neq s_2$.

Example 2.1

- *Let S be a set that contains at least two elements. The relation $\mathcal{R} = S \times S$ is a preorder on S, but not a partial order, because for any pair $s_1, s_2 \in S$ with $s_1 \neq s_2$, we have $s_1 \mathcal{R} s_2$ and $s_2 \mathcal{R} s_1$. Hence $\mathcal{R}$ is not antisymmetric.*
- *The ordering on pairs of natural numbers $\preccurlyeq_p \subseteq \mathbb{N}^2 \times \mathbb{N}^2$ s.t. $\langle i_1, i_2 \rangle \preccurlyeq_p \langle i'_1, i'_2 \rangle$ iff $i_1 \leq i'_1$ and $i_2 \leq i'_2$ is a partial order (because $\leq$ is a partial order – actually a total order – on $\mathbb{N}$).* ◊

In the sequel, we will often manipulate sets equipped with a dedicated order. Hence the following definition:

Definition 2.2 (Ordered Set) *A tuple $\langle S, \lesssim \rangle$ is an* ordered set *iff $\lesssim$ is a preorder on the set S.* ■

Before we consider special kinds of orders, we prove the following property that will often be used in the sequel:

[1] Remark that in many classical definitions, a preorder is *not* regarded as an order, because these definitions state than an order has to be antisymmetric. We have chosen to regard any reflexive and transitive relation as an order, as a means to obtain unified notations. Nevertheless, when necessary, we avoid ambiguity by explicitly using the terms 'preorder', 'quasiorder' or 'partial order'.

Lemma 2.1 *Let $\langle S, \preceq \rangle$ be an ordered set, and let s_1, s_2 and s_3 be three elements of S s.t.*

- *either $s_1 \prec s_2$ and $s_2 \preceq s_3$*
- *or $s_1 \preceq s_2$ and $s_2 \prec s_3$.*

Then, $s_1 \prec s_3$.

Proof. We provide the proof for the former case, i.e., $s_1 \prec s_2$ and $s_2 \preceq s_3$. The proof for the latter is similar.

By definition of $\prec$, $s_1 \prec s_2$ is equivalent to $s_1 \preceq s_2$ and $s_2 \not\preceq s_1$. Since $s_1 \preceq s_2$ and $s_2 \preceq s_3$, we have $s_1 \preceq s_3$ by transitivity of $\preceq$. It remains to show that $s_3 \not\preceq s_1$. We show this *per absurdum.* Let us assume that $s_3 \preceq s_1$. Since $s_2 \preceq s_3$ by hypothesis, we have $s_2 \preceq s_1$, by transitivity. Contradiction. Hence $s_1 \preceq s_3$ and $s_3 \not\preceq s_1$, which means that $s_1 \prec s_3$. □

Well-founded order Either of these orderings can enjoy the well-foundedness property, which states that the ordering cannot produce an infinite strictly decreasing sequence of elements:

Definition 2.3 (Well–Founded Order) *Let $\langle S, \preceq \rangle$ be an ordered set. S is* well-founded *iff there is no infinite sequence $s_1, s_2, \ldots, s_i \ldots$ s.t. for any $i \geq 1$: $s_i \in S$ and $s_{i+1} \prec s_i$.* ■

Example 2.2 *The ordering $\leq$ on $\mathbb{N}$ is well-founded. Indeed, for any $k \in \mathbb{N}$, the longest strictly decreasing sequence starting in k is $k, k-1, k-2, \ldots, 0$, which is necessarily finite. However $\leq$ on $\mathbb{Z}$ is not well-founded, because $0, -1, -2, \ldots$ forms an infinite strictly decreasing chain of elements.* ◊

Well-quasi orders This section introduces the notion of *well-quasi ordering*, which is a special kind of preorder:

Definition 2.4 (Well-quasi Ordering) *Let $\langle S, \preceq \rangle$ be an ordered set. Then, $\preceq$ is a* well-quasi ordering *(*WQO *for short) iff: for any infinite sequence $s_1, s_2, \ldots, s_i, \ldots$ s.t. $\forall i \geq 1 : s_i \in S$, there are two positions $k \in \mathbb{N}_0$ and $\ell \in \mathbb{N}_0$ s.t. $k < \ell$ and $s_k \preceq s_\ell$.* ■

Remark that a direct consequence of this definition is that any WQO is also well-founded (see Definition 2.3).

Example 2.3 *The total ordering $\leq \subseteq \mathbb{N}\times\mathbb{N}$ (classical ordering on the natural numbers) is a* WQO. *Indeed, suppose it is not the case. Then by definition 2.4, it is possible to build an infinite sequence $i_1, i_2, \ldots, i_j, \ldots$ of elements of $\mathbb{N}$ s.t. each element i_k of the sequence is $\not\geq$ than all the elements $i_1, i_2, \ldots, i_{k-1}$. Since $\leq$ is a total order, $i \not\geq j$ is equivalent to $i < j$. Hence, the chain $i_1, i_2, \ldots, i_j, \ldots$ is an infinite strictly decreasing sequence of natural numbers, which is not possible, because $\leq$ is well-founded (see Example 2.2). Contradiction.*

On the other hand, the ordering $\leq \subseteq \mathbb{Z}\times\mathbb{Z}$ is not *a* WQO, *because it is not well-founded.* ◊

Remark however that any well-founded order is not necessarily a WQO, as shown by the following example:

Example 2.4 *Let $\langle S, \overline{\mathcal{R}}\rangle$ be an ordered set s.t. $S = \{s_1, s_2, \ldots, s_i, \ldots\}$ is infinite, and let $\overline{\mathcal{R}} = \{(s, s) \mid s \in S\}$. Clearly, $\overline{\mathcal{R}}$ is both reflexive and transitive. Moreover, it is also well-founded since, for any $s \in S$, there is no $s' \in S$ s.t. $s\overline{\mathcal{R}}s'$ but $\neg(s'\overline{\mathcal{R}}s)$. Hence, any strictly decreasing sequence is necessarily of length 1. Thus, $\overline{\mathcal{R}}$ is a well-founded preorder.*

However, $\overline{\mathcal{R}}$ is not a WQO *because the infinite sequence $s_1, s_2, \ldots s_i, \ldots$ is such that there are no positions k and ℓ s.t. $k < \ell$ and $s_k\overline{\mathcal{R}}s_\ell$.* ◊

Finally, let us finish the discussion of WQO by providing the following Lemma, stating that, for any infinite sequence $s_1, s_2, \ldots, s_i, \ldots$ of elements from some set S equipped with a WQO $\overline{\leq}$, one can extract, from that sequence, an infinite subsequence that is increasing wrt to $\overline{\leq}$:

Lemma 2.2 *Let $\langle S, \overline{\leq}\rangle$ be an ordered set where $\overline{\leq}$ is a* WQO. *Let $s_1, s_2, \ldots, s_i, \ldots$ be an infinite sequence of elements s.t. for any $i \geq 1$: $s_i \in S$. Then, there exists a function $\rho : \mathbb{N}_0 \mapsto \mathbb{N}_0$ s.t. for any $i \geq 1$: $\rho(i) < \rho(i+1)$ and $s_{\rho(i)}\overline{\leq}s_{\rho(i+1)}$.*

Proof. Per absurdum. Let us assume there exists an ordered set $\langle S, \overline{\leq}\rangle$ s.t. $\overline{\leq}$ is a WQO, and there exists an infinite sequence $s_1, s_2, \ldots, s_i, \ldots$ of elements from S that does not admit any infinite $\overline{\leq}$-increasing subsequence. Let us consider any element s_i from that sequence and let us build a maximal $\overline{\leq}$-increasing sequence starting in s_i, i.e., a sequence $s_{\delta(1)}, s_{\delta(2)}, \ldots, s_{\delta(k)}$ s.t. $s_{\delta(1)} = s_i$, for any $1 \leq j < k$: $s_{\delta(j)}\overline{\leq}s_{\delta(j+1)}$ and for any $\ell > \delta(k)$: $s_{\delta(k)}\overline{\not\leq}s_\ell$. Remark that, by hypothesis, that sequence is necessarily finite. Thus, we have just shown that one can associate, to any element s_i, a position $\pi(i) \geq i$ s.t. for any $j > \pi(i)$, $s_{\pi(i)}\overline{\not\leq}s_j$ (simply take $\pi(i) = \delta(k)$). That is, for any $i \geq 1$, we are ensured that all the elements of the sequence appearing (strictly) after position $\pi(i)$ are not $\overline{\geq}$ than $s_{\pi(i)}$.

The function π allows us to identify infinitely many positions in the sequence $s_1, s_2, \ldots s_i, \ldots$ that are $\overline{\not\leq}$ to all their successors in the sequence. Indeed, by definition $s_{\pi(1)}$ is $\overline{\not\leq}$ to any s_j with $j > \pi(1)$. In particular, $\pi(\pi(1)+1) > \pi(1)$ is a position s.t. $s_{\pi(1)}\overline{\not\leq}s_{\pi(\pi(1)+1)}$ and $s_{\pi(\pi(1)+1)}\overline{\not\leq}s_j$ for any $j > \pi(\pi(1)+1)$. That reasoning can be repeated from position $\pi(\pi(1)+1)$ by considering $\pi(\pi(\pi(1)+1)+1)$ and so on. More precisely, we build the function $\tau : \mathbb{N} \mapsto \mathbb{N}$, defined recursively as follows:

$$\begin{aligned} \tau(0) &= 0 \\ \forall i \geq 1 : \tau(i) &= \pi(\tau(i-1)+1) \end{aligned}$$

First, remark that τ is a strictly increasing function: for any $i \geq 0$: $\tau(i) < \tau(i+1)$. Indeed, for any $i \geq 1$: $\tau(i) = \pi(\tau(i-1)+1) \geq \tau(i-1)+1 > \tau(i-1)$, by definition of τ and π. Hence, we have:

$$\begin{array}{lll} & \forall i \geq 1 : \forall j > \pi(i) : s_{\pi(i)}\overline{\not\leq}s_j & \\ \Rightarrow & \forall k \geq 1 : \forall j > \pi(\tau(k-1)+1) : s_{\pi(\tau(k-1)+1)}\overline{\not\leq}s_j & \forall k \geq 1 : \tau(k-1)+1 \geq 1 \\ \Rightarrow & \forall k \geq 1 : \forall j > \tau(k) : s_{\tau(k)}\overline{\not\leq}s_j & \text{Def. of } \tau \\ \Rightarrow & \forall k > 1 : \forall \ell \text{ s.t. } \tau(\ell) \geq \tau(k) : s_{\tau(k)}\overline{\not\leq}s_{\tau(\ell)} & \\ \Rightarrow & \forall k > 1 : \forall \ell > k : s_{\tau(k)}\overline{\not\leq}s_{\tau(\ell)} & \tau \text{ is strictly } \leq\text{-increasing} \end{array}$$

We conclude that the sequence $s_{\tau(1)}, s_{\tau(2)}, \ldots, s_{\tau(i)}, \ldots$ is an infinite sequence of elements of S for which one cannot find two positions k and ℓ with $k < \ell$ and $s_k\overline{\leq}s_\ell$. Hence, $\overline{\leq}$ is not a WQO, by Definition 2.4. Contradiction. □

Canonical set Let us close this section on orderings by the definition of canonical set. A set is canonical wrt to an ordering $\overline{\leq}$ iff it does not contains two distinct $\overline{\leq}$–comparable elements:

Definition 2.5 (Canonical Set) *Let $\langle S, \overline{\leq}\rangle$ be an ordered set. A set $S' \subseteq S$ is* canonical *(wrt $\overline{\leq}$) iff for any s_1, $s_2 \in S'$: $s_1 \neq s_2$ implies $s_1\overline{\not\leq}s_2$.* ■

Minimal and maximal elements Given an ordered set $\langle S, \overline{\leq}\rangle$ and $S' \subseteq S$:

1. The set of $\overline{\leq}$*–minimal elements* $\mathsf{Min}^{\overline{\leq}}(S')$ is $\{s \in S' \mid \nexists s' \in S' : s'\overline{<}s\}$.

2. The set of $\overline{\leq}$*–maximal elements* $\mathsf{Max}^{\overline{\leq}}(S')$ is $\{s \in S' \mid \nexists s' \in S' : s'\overline{<}s\}$.

Remark that these sets are unique, by definition. However, they might not be canonical, nor finite, as shown by the following example:

Example 2.5 *Let $\langle S, \mathcal{R}\rangle$ be an ordered set, where $\mathcal{R} = S \times S$, and let $S' \subseteq S$ be s.t. $|S'| \geq 2$. Thus, S' is not a canonical set (by definition of $\mathcal{R}$, all the elements are comparable to each other). Moreover, there is, in S, no pair of elements s and s' s.t. $s\overline{\leq}s'$ and $s'\overline{\not\leq}s$ (which is equivalent to $s\overline{<}s'$). Thus, $\mathsf{Min}^{\overline{\leq}}(S') = S' = \mathsf{Max}^{\overline{\leq}}(S')$. In the case where S' is infinite, $\mathsf{Min}^{\overline{\leq}}(S')$ and $\mathsf{Max}^{\overline{\leq}}(S')$ are both infinite too.* ◊

However, when $\overline{\leq}$ is a partial order, $\mathsf{Min}^{\overline{\leq}}(S')$ and $\mathsf{Max}^{\overline{\leq}}(S')$ *are* canonical:

Lemma 2.3 *Let $\langle S, \overline{\leq} \rangle$ be an ordered set, and let $S' \subseteq S$. If $\overline{\leq}$ is antisymmetric, then $\mathsf{Min}^{\overline{\leq}}(S')$ and $\mathsf{Max}^{\overline{\leq}}(S')$ are canonical sets.*

Proof. Per absurdum. Let us consider $\mathsf{Min}^{\overline{\leq}}(S')$, and let us suppose that it is not canonical. Hence, there are s and s' in $\mathsf{Min}^{\overline{\leq}}(S')$ s.t. $s \neq s'$ and $s \overline{\leq} s'$. Since $\overline{\leq}$ is antisymmetric, this implies that $s \overline{<} s'$. Hence, s' does not belong to $\mathsf{Min}^{\overline{\leq}}(S')$. Contradiction.

The same reasoning holds on $\mathsf{Max}^{\overline{\leq}}(S')$. □

2.2.2 Upward– and downward–closed sets

This section introduces special kinds of sets which are closed wrt to an ordering $\overline{\leq}$. We study in particular upward- and downward-closed sets.

A set U is *upward-closed* with respect to an ordering $\overline{\leq}$ iff for any element $u \in U$, all the elements that are larger (wrt $\overline{\leq}$) than u are in U too. The definition of *downward-closed* set is symmetrical:

Definition 2.6 (Upward-Closed Set) *Let $\langle S, \overline{\leq} \rangle$ be an ordered set. The set $U \subseteq S$ is a* $\overline{\leq}$-upward-closed set *iff for any $u \in U$: for any $u' \in S$: $u \overline{\leq} u'$ implies that $u' \in U$.* ■

Definition 2.7 (Downward-Closed Set) *Let $\langle S, \overline{\leq} \rangle$ be an ordered set. $D \subseteq S$ is a* $\overline{\leq}$-downward-closed set *iff for any $d \in D$: for any $d' \in S$: $d' \overline{\leq} d$ implies that $d' \in D$.* ■

In the literature, a $\overline{\leq}$-upward-closed set $U \subseteq S$ is sometimes called an *ideal in S*. When the ordering $\overline{\leq}$ is clear from the context, we sometimes write *upward-closed* and *downward-closed* instead of $\overline{\leq}$-upward-closed and $\overline{\leq}$-downward-closed.

Given a set S', one can consider its upward– or downward–closure, defined as follows:

Definition 2.8 (Upward- and Downward-Closure) *Let $\langle S, \overline{\leq} \rangle$ be an ordered set, and let $S' \subseteq S$. Then:*

1. *the* upward-closure of S', *noted $\uparrow(S')$, is the set $\{s \in S \mid \exists s' \in S' : s' \overline{\leq} s\}$.*
2. *the* downward-closure of S', *noted $\downarrow(S')$, is the set $\{s \in S \mid \exists s' \in S' : s \overline{\leq} s'\}$.* ■

Remark that, for any subset S' of S, $\uparrow(S')$ and $\downarrow(S')$ are both unique, by definition. Finally, we can state the definition of *generator* of an upward– or downward– closed set:

Definition 2.9 (Generator) *Let $\langle S, \overline{\leq} \rangle$ be an ordered set, U be a $\overline{\leq}$-upward-closed set of S, D be a $\overline{\leq}$-downward-closed set of S, and $G \subseteq S$. Then G is an* upward generator *of U iff $\uparrow(G) = U$, and G is a* downward generator *of D iff $\downarrow(G) = D$.* ■

When the context permits it, we sometimes refer to *upward* and *downward generators* simply as *generators.*

Finite representations of upward-closed sets Remark that, since S can be infinite, upward- and downward-closed sets can be infinite too. Thus, in order to be able to manipulate such sets in an algorithmic way, we must have some kind of finite and effective representation for them at our disposal.

We first address the upward-closed sets. Our presentation of the way to obtain a suitable representation of an upward-closed set follows that of [ACJT96]. The conditions we want to enforce to have a *suitable* representation are as follows:

Definition 2.10 (Minimal elements of an upward-closed set) *Let $\langle S, \overline{\leq} \rangle$ be an ordered set, and let $U \subseteq S$ be a $\overline{\leq}$-upward-closed set. Then, $\mathsf{UGen}(U)$ is a set of elements of S s.t.:*

(U_1) $\mathsf{UGen}(U) \subseteq U$;

(U_2) $\mathsf{UGen}(U)$ *is a* generator *of U;*

(U_3) $\mathsf{UGen}(U)$ *is* canonical. ■

Thus, $\mathsf{UGen}(U)$ is a minimal generator of U. Let us illustrate Definition 2.10 by several examples:

Example 2.6

1. *Let us consider the ordered set $\langle \mathbb{Z}, \leq \rangle$. Clearly, $\mathbb{Z}$ is $\leq$-upward-closed. However, there does not exist any set that satisfies the definition of $\mathsf{UGen}(U)$. Indeed, since $\leq$ is a total relation, any canonical subset of $\mathbb{Z}$ is either $\emptyset$ or a singleton. None of them represents $\mathbb{Z}$ because $\uparrow(\emptyset) = \emptyset$ and for any $z \in \mathbb{Z}$ we have $\mathbb{Z} \setminus \uparrow(\{z\}) = \{z-1, z-2, \ldots\} \neq \emptyset$. Hence, $\{z\}$ does not represent $\mathbb{Z}$.*

2. *Let us consider the ordered set $\langle \mathbb{N}, = \rangle$. Remark that $=$ is well-founded. Indeed, there is no infinite sequence $s_1, s_2, \ldots, s_i, \ldots$ of natural numbers s.t. for any $i \geq 1$: $s_i = s_{i+1}$ and $s_{i+1} \neq s_i$.*

   In this case, $\mathsf{UGen}\left(\mathbb{N}\right) = \mathbb{N}$*, because all the pairs of distinct elements i and j in $\mathbb{N}$ are $=$-incomparable. However, $\mathbb{N}$ is not finite.*

3. *Let us consider the ordered set $\langle S, \mathcal{R} \rangle$ where S is an infinite set and $\mathcal{R} = S \times S$. Remark that $\mathcal{R}$ is a* WQO*, because any element $s \in S$ is $\overline{\leq}$-comparable to any elements $s' \in S$. However, $\mathcal{R}$ is not antisymmetric.*

   Clearly, S is $\mathcal{R}$-upward-closed. However, for any element $s \in S$, $\{s\}$ is a (finite) canonical generator of S. Hence, $\mathsf{UGen}\left(S\right)$ *is not unique.*

4. *Let us consider the ordered set $\langle \mathbb{N}, \leq \rangle$, and $U = \{c \mid 5 \leq c\}$. Then,* $\mathsf{UGen}\left(U\right) = \{5\}$ *is unique, canonical and finite. Moreover,* $\mathsf{UGen}\left(U\right) = \mathsf{Min}^{\leq}\left(U\right)$*. Remark that, in the present case, $\leq$ is both a* WQO *and a total order.* ◊

The previous examples have shown that $\mathsf{UGen}\left(U\right)$ does not always exist, is not always unique and is sometimes not finite. The following results establish when it is so. First, the well-foundedness property is necessary to ensure that a *canonical generator* exists

Lemma 2.4 ([ACJT96]) *Let $\left\langle S, \overline{\leq} \right\rangle$ be an ordered set s.t. $\overline{\leq}$ is well-founded, and let $U \subseteq S$ be a $\overline{\leq}$-upward-closed set. Then, there exists a set $A \subseteq U$ s.t.:*

1. *A is canonical;*
2. *A is a generator of U.*

However, this lemma does not guarantee that the generator is finite, nor that it is unique (as we have seen in points 2 and 3 of Example 2.6). Moreover, the Lemma does not characterise the set A. Let us show that such a set can be obtained from $\mathsf{Min}^{\overline{\leq}}\left(U\right)$. For that purpose, we first show that when $\overline{\leq}$ is well-founded, $\mathsf{Min}^{\overline{\leq}}\left(U\right)$ is a generator of U:

Lemma 2.5 *Let $\left\langle S, \overline{\leq} \right\rangle$ be an ordered set s.t. $\overline{\leq}$ is well-founded, and let $U \subseteq S$ be a $\overline{\leq}$-upward-closed set. Then,* $\mathsf{Min}^{\overline{\leq}}\left(U\right)$ *is a generator of U.*

Proof. In the case where $U = \emptyset$, we have $\mathsf{Min}^{\overline{\leq}}\left(U\right) = \emptyset$, which is clearly a generator for U. Otherwise, let u be an element of U. Let $u_1, u_2, \ldots, u_n$ ($n \geq 1$) be a sequence of elements of U s.t. $u_1 = u$, for any $1 \leq i < n$: $u_{i+1} \overline{<} u_i$ and there is no $u' \in U$ s.t. $u' \overline{<} u_n$. Remark that the sequence is finite because $\overline{\leq}$ is well-founded. Since there is no $u' \in U$ s.t. $u' \overline{<} u_n$, we have $u_n \in \mathsf{Min}^{\overline{\leq}}\left(U\right)$. Moreover, $u_n \overline{\leq} u$ (remark that it might be the case that $u_n = u_1 = u$. In that case, $u_n \overline{\leq} u$ because $\overline{\leq}$ is reflexive). Since this holds for any element $u \in U$, we conclude that for any element $u \in U$, there is $u_n \in \mathsf{Min}^{\preceq}(U)$ s.t. $u_n \preceq u$. Hence, $U \subseteq \uparrow\Big(\mathsf{Min}^{\preceq}(U)\Big)$. Finally, since by definition $\mathsf{Min}^{\preceq}(U) \subseteq U$, we have: $\uparrow\Big(\mathsf{Min}^{\preceq}(U)\Big) \subseteq \uparrow(U) = U$. We conclude that $\uparrow\Big(\mathsf{Min}^{\preceq}(U)\Big) = U$. □

Thus, $\mathsf{Min}^{\preceq}(U)$ is a generator of U when $\preceq$ is well-founded. As we have seen, it is not guaranteed to be canonical. Let us show how we can extract from $\mathsf{Min}^{\preceq}(U)$, a canonical set A s.t. $\uparrow(A) = \uparrow\Big(\mathsf{Min}^{\preceq}(U)\Big)$.

Let $\sim \subseteq \mathsf{Min}^{\preceq}(U) \times \mathsf{Min}^{\preceq}(U)$ be the equivalence defined as follows: $\sim = \{(s_1, s_2) \mid s_1 \preceq s_2 \wedge s_2 \preceq s_1\}$. Let $Q = \{Q_1, Q_2, \ldots, Q_i, \ldots\}$ be the (possibly infinite) set of equivalence classes of $\sim$. Let us associate to every Q_i an element $r_i \in Q_i$, that will serve as an unique representative of Q_i. Let $R = \{r_1, r_2, \ldots, r_i, \ldots\}$. Remark that $R \subseteq \mathsf{Min}^{\preceq}(U)$. Clearly, for every i, we have $Q_i \subseteq \uparrow(r_i)$. Finally, let $f : \mathsf{Min}^{\preceq}(U) \mapsto R$ be a function that associates, to every elements $s \in \mathsf{Min}^{\preceq}(U)$ the representative r_i of the equivalence class Q_i that contains s, i.e., $\forall s \in \mathsf{Min}^{\preceq}(U) : f(s) = r_i$ iff $s \in Q_i$. Thus:

$$\forall i : \forall s \in Q_i \quad : \quad Q_i \subseteq \uparrow(f(s)) \tag{2.1}$$

However, since U is upward-closed and since, for any $s \in \mathsf{Min}^{\preceq}(U)$, $f(s) \in \mathsf{Min}^{\preceq}(U)$, we have:

$$\forall s \in \mathsf{Min}^{\preceq}(U) \quad : \quad \uparrow(f(s)) \subseteq \uparrow\Big(\mathsf{Min}^{\preceq}(U)\Big) \tag{2.2}$$

Let $A = \{f(s) \mid s \in \mathsf{Min}^{\preceq}(U)\}$, and let us show that it is indeed a canonical generator of U:

Lemma 2.6 *For any $\preceq$-upward-closed set U, the set $A = \{f(s) \mid s \in \mathsf{Min}^{\preceq}(U)\}$ is a canonical generator of U.*

Proof. Let us first show that A is canonical. By definition, there is no pair of elements s_1 and s_2 in $\mathsf{Min}^{\preceq}(U)$ s.t. $s_1 \preceq s_2$ and $s_1 \not\preceq s_2$. Hence, for every s_1, s_2 in $\mathsf{Min}^{\preceq}(U)$ s.t. $s_1 \neq s_2$: $s_1 \preceq s_2$ implies $s_2 \preceq s_1$ and $s_1 \preceq s_2$. Thus, by definition of $\sim$, and since $A \subseteq \mathsf{Min}^{\preceq}(U)$: for every s_1, s_2 in A s.t. $s_1 \neq s_2$: either $s_1 \not\preceq s_2$ and $s_2 \not\preceq s_1$ or $s_1 \sim s_2$. However, by definition of A, there can't be two distinct elements s_1 and s_2 in A s.t. $s_1 \sim s_2$. We conclude that for any s_1, s_2 in A s.t. $s_1 \neq s_2$: $s_1 \not\preceq s_2$ and $s_2 \not\preceq s_1$. Hence, A is canonical.

Let us show that $\uparrow(A) = \uparrow\Big(\mathsf{Min}^{\preceq}(U)\Big)$. By definition:

$$\uparrow(A) = \bigcup_{s \in \mathsf{Min}^{\preceq}(U)} \uparrow(f(s))$$

Thus, $\uparrow(A) \subseteq \uparrow\Big(\mathsf{Min}^{\overline{\leq}}(U)\Big)$, by (2.2). Moreover let s be an element of $\mathsf{Min}^{\overline{\leq}}(U)$, and i be s.t. $s \in Q_i$. By (2.1) and definition of A, we have: $\{s\} \subseteq Q_i \subseteq \uparrow(f(s)) \subseteq \uparrow(A)$. Since this holds for any $s \in \mathsf{Min}^{\overline{\leq}}(U)$, we obtain $\mathsf{Min}^{\overline{\leq}}(U) \subseteq \uparrow(A)$. We conclude that $\uparrow\Big(\mathsf{Min}^{\overline{\leq}}(U)\Big) = \uparrow(A)$, and, by Lemma 2.5, $\uparrow\Big(\mathsf{Min}^{\overline{\leq}}(U)\Big) = U$. □

Remark that the outcome of this 'canonisation' construction may not be unique, because it depends on the representative r_i we have chosen for every class Q_i. By selecting other r_i's, we obtain a different set A that has the same properties. We will see in the sequel under which conditions the set $\mathsf{UGen}(U)$ is unique.

Let us now assume that $\overline{\leq}$ is a well-quasi order. In that case, any canonical subset of S has to be finite, as stated by the next lemma:

Lemma 2.7 ([ACJT96]) *Let $\langle S, \overline{\leq}\rangle$ be an ordered set where $\overline{\leq}$ is a* WQO*. Then, for every $S' \subseteq S$: if S' is canonical, then S' is finite.*

Thus, since any WQO is well-founded and by Definition 2.10:

Corollary 2.1 *Let $\langle S, \overline{\leq}\rangle$ be an ordered set s.t. $\overline{\leq}$ is a* WQO*, and let $U \subseteq S$ be a $\overline{\leq}$-upward-closed set. Then, there exists $A \subseteq U$ s.t.:*

1. *A is canonical;*
2. *A is a generator of U;*
3. *A is finite.*

Finally, let us show that when the WQO we consider is a *partial order* (that is, it is an antisymmetric WQO), then each upward-closed set has an *unique* finite canonical generator. Remark that the WQO introduced in the literature dealing with the verification of infinite-state systems are usually antisymmetric (see sections 2.3.3, 2.3.4 and 2.3.5).

Lemma 2.8 *Let $\langle S, \overline{\leq}\rangle$ be an ordered set s.t. $\overline{\leq}$ is an antisymmetric* WQO*, and let $U \subseteq S$ be a $\overline{\leq}$-upward-closed set. Then, there exists an unique finite and canonical generator $A \subseteq U$ of U.*

Proof. By corollary 2.1, there exists A that respects points 1 through 3. Let us show that this set is unique.

Per absurdum. Suppose there are two sets $S_1 \neq S_2$ that satisfy Definition 2.10. Thus, by U_2: $S_1 \subseteq U$ and for any $s_1 \in S_1$, there must exist $s_2 \in S_2$ s.t. $s_2 \overline{\leq} s_1$. We can hold the same reasoning on S_2 and obtain that for any $s'_2 \in S_2$, there is $s'_1 \in S_1$ such that $s'_1 \overline{\leq} s'_2$.

Without loss of generality, let us assume that there exists $s_1 \in S_1 \setminus S_2$. Since $s_1 \in S_1$ but $s_1 \notin S_2$, and since $\overline{\leq}$ is antisymmetric, there is $s_2 \in S_2$ s.t. $s_2 \overline{<} s_1$. But since $s_2 \in S_2$, there is $s_3 \in S_1$ s.t. $s_3 \overline{\leq} s_2 \overline{<} s_1$. Thus, $s_3 \overline{<} s_1$ by Lemma 2.1. Hence, there are two different elements s_1 and s_3 in S_1 which are comparable, and S_1 is thus not canonical. Contradiction. □

Not surprisingly, in that case, for any $\overline{\leq}$-upward-closed set U, the set $\mathsf{UGen}\,(U)$ is exactly $\mathsf{Min}^{\overline{\leq}}\,(U)$:

Lemma 2.9 *Let $\langle S, \overline{\leq} \rangle$ be an ordered set s.t. $\overline{\leq}$ is an antisymmetric* WQO*, and let $U \subseteq S$ be a $\overline{\leq}$-upward-closed set. Then,* $\mathsf{UGen}\,(U)$*, exists, is a finite set and is equal to* $\mathsf{Min}^{\overline{\leq}}\,(U)$*.*

Proof. By Lemma 2.5, $\mathsf{Min}^{\overline{\leq}}\,(U)$ is a generator of U. By Lemma 2.3, and since $\overline{\leq}$ is antisymmetric, $\mathsf{Min}^{\overline{\leq}}\,(U)$ is also canonical. Finally, since $\overline{\leq}$ is a WQO, $\mathsf{Min}^{\overline{\leq}}\,(U)$ is finite, by Lemma 2.7. Hence, $\mathsf{Min}^{\overline{\leq}}\,(U)$ is a finite canonical generator of U. By Lemma 2.8, $\mathsf{UGen}\,(U)$ exists and is the only set that respects these conditions. Hence, $\mathsf{UGen}\,(U) = \mathsf{Min}^{\overline{\leq}}\,(U)$. □

Thus, in that case, any $\leq$-upward-closed set can be *effectively represented* by its finite set of minimal elements. On the other hand, $\overline{\leq}$-downward-closed sets are more difficult to represent effectively, because they may contain infinite increasing sequences of elements.

Finite representations of downward-closed sets To obtain a finite representation of $\overline{\leq}$-downward-closed sets, we must use well-chosen limit elements $\ell \notin S$ that will serve as representation for the $\overline{\leq}$-downward-closures of infinite increasing chains of elements. Thus, we introduce the notion of *adequate* domain of limits. Remark that in the following definition, we restrict ourselves to the case where the considered ordering $\overline{\leq}$ is a WQO.

Definition 2.11 (Adequate domain of limits) *Let $\langle S, \overline{\leq} \rangle$ be a an ordered set where $\overline{\leq}$ is a* WQO*, and L be a set of elements disjoint from S, the tuple $\langle L, \sqsubseteq, \gamma \rangle$ is called an* adequate domain of limits *for $\langle S, \overline{\leq} \rangle$ if the following conditions are satisfied:*

(L_1) representation mapping*: $\gamma : L \cup S \mapsto 2^S$ associates to each element in $L \cup S$ a $\leq$-downward-closed set $D \subseteq S$. Furthermore, for any $s \in S$, we impose that $\gamma(s) = {\downarrow}(s)$. In the following, γ is extended to sets $\mathcal{S} \subseteq L \cup S$ in the natural way: $\gamma(\mathcal{S}) = \cup_{s \in \mathcal{S}} \gamma(s)$;*

(L_2) top element*: there exists a special element $\top \in L$ such that $\gamma(\top) = S$;*

(L_3) precision order*: the elements of $L \cup S$ are ordered by the quasi-order $\sqsubseteq$, defined as follows: $d_1 \sqsubseteq d_2$ if and only if $\gamma(d_1) \subseteq \gamma(d_2)$;*

(L_4) completeness*: for any $\leq$-downward-closed set $D \subseteq S$, there exists a finite set $D' \subseteq L \cup S$ such that $\gamma(D') = D$.* ■

Remark 2.1 *Remark that we could have omitted the $\top$ element in this definition, since point (L_4) guarantees that* any *downward-closed set of S (that is, S included), can be finitely represented by elements of $L \cup S$. However, this simplifies our presentation of the* Expand, Enlarge and Check *algorithm in Chapter 4.*

Remark further that the definition of $\sqsubseteq$ follows directly from the γ. In other words, $d_1 \sqsubseteq d_2$ can be regarded as a shorthand notation for $\gamma(d_1) \subseteq \gamma(d_2)$.

Examples of adequate domain of limits that are applicable to represent downward-closed sets of configurations for classical models of computation (such as Petri nets, or Lossy channel systems) are provided in Section 3.1.

Remark that, unlike the upward-closed sets case, considering an antisymmetric WQO is not sufficient to guarantee the unicity of the representations of *downward-closed sets*. Indeed, $\overline{\leq}$ being antisymmetric does not necessarily imply that $\sqsubseteq$ is antisymmetric.Suppose, for instance, that there are two limit elements ℓ_1 and ℓ_2 s.t. $\gamma(\ell_1) = \gamma(\ell_2)$. This will be the case, for instance, when considering downward-closed regular expressions (suitable to represent downward-closed sets of configurations of Lossy channel systems) that are *not in normal form* (see Section 3.1.2).

Remark that when we consider a *finite $\overline{\leq}$-downward-closed set* D, limit elements are not necessary anymore since no infinite increasing sequences of elements can appear. In that case, $\mathsf{Max}^{\overline{\leq}}(D)$ is a finite generator of D:

Lemma 2.10 *Let $\left\langle S, \overline{\leq} \right\rangle$ be an ordered set and let $D \subseteq S$ be a finite $\overline{\leq}$-downward-closed set. Then, $\mathsf{Max}^{\overline{\leq}}(D)$ is a finite generator of D*

Proof. Since D is finite and $\mathsf{Max}^{\overline{\leq}}(D) \subseteq D$, by definition, $\mathsf{Max}^{\overline{\leq}}(D)$ is necessarily finite. Let d be an element of D, and let $d_1, d_2, \ldots$ be a maximal (non-extendable) sequence of $\overline{<}$-increasing elements of D starting in d, i.e.,: $d_1 = d$ and for any $i \geq 1$: $d_i \overline{<} d_{i+1}$. Let us first show, *per absurdum*, that such a sequence has to be finite. If the sequence is infinite, there are two positions k and ℓ s.t. $d_k = d_\ell$ because D is finite. Moreover $\ell - k > 1$, because otherwise, we would have $d_k \overline{<} d_k$, which contradicts the reflexivity property of $\overline{\leq}$. Since $\overline{\leq}$ is transitive, we have $d_k \overline{\leq} d_{\ell-1}$. However, since $d_{\ell-1} \overline{<} d_\ell = d_k$, we have $d_k \overline{\not\leq} d_{\ell-1}$. Contradiction. We conclude that the sequence $d_1, \ldots, d_n$ is finite.

Thus, we can associate, to any element $d \in D$, an element $d_n \in D$ s.t. $d \overline{\leq} d_n$ and s.t. there is no $d' \in D$ s.t. $d_n \overline{<} d$, because the sequence is maximal. This implies that $d_n \in \mathsf{Max}^{\overline{\leq}}(D)$. Hence, $\downarrow\left(\mathsf{Max}^{\overline{\leq}}(D)\right) = D$ □

Thus, $\mathsf{Max}^{\preceq}(D)$ is a finite generator for any finite downward-closed set D. Since this set is finite, it is possible to extract from $\mathsf{Max}^{\preceq}(D)$ at least one canonical set that has the same downward-closure.

Moreover, when the WQO considered is a partial order, the set $\mathsf{Max}^{\preceq}(D)$ is a finite, canonical and unique representation of D:

Lemma 2.11 *Let $\langle S, \preceq \rangle$ be an ordered set where $\preceq$ is an antisymmetric* WQO*, and let $D \subseteq S$ be a* finite *$\preceq$-downward-closed set. Then,* $\mathsf{Max}^{\preceq}(D)$ *is:*

1. *a generator of D;*
2. *finite;*
3. *canonical;*
4. *unique: there is no finite canonical set $D' \subseteq S$ s.t. $D' \neq D$ and D' is a generator of D.*

Proof. By Lemma 2.10, $\mathsf{Max}^{\preceq}(D)$ is a finite generator of D. Since $\preceq$ is antisymmetric, $\mathsf{Max}^{\preceq}(D)$ is also canonical, by Lemma 2.3.

Let us show *per absurdum* that $\mathsf{Max}^{\preceq}(D)$ is unique. Assume there is another set D' that is a finite canonical generator of D. Without loss of generality, there is $d \in \mathsf{Max}^{\preceq}(D) \setminus D'$. Thus, $d \in D$. Since D' is a generator of D, there is d' in D' s.t. $d \preceq d'$. Moreover $d' \neq d$, because $d \notin D'$. Hence, $d \prec d'$ because $\preceq$ is antisymmetric. Since $d' \in D' \subseteq D$, there is $d'' \in \mathsf{Max}^{\preceq}(D)$ s.t. $d' \preceq d''$. Hence, there are two distinct elements d and d'' in $\mathsf{Max}^{\preceq}(D)$ s.t. $d \prec d''$. Hence $\mathsf{Max}^{\preceq}(D)$ is not canonical. Contradiction. □

2.3 Well-structured Transition Systems

The models of computations that are considered in this thesis are called *Well-Structured Transition Systems*, or WSTS for short. A WSTS is essentially a transition system whose set of configurations is possibly infinite but ordered with respect to a well-quasi ordering. Let us first introduce the notion of *transition system*, preparatory to the definition of WSTS.

2.3.1 Transition systems

A transition system is a (possibly infinite) set of configurations equipped with a transition relation. The set of configurations usually represents the *states* of the system that is modelled. One of these configurations is regarded as the *initial configuration* of the system, i.e., the state in which the system is at the beginning of any computation. The transition relation expresses the dynamics of the system by indicating, given the current configuration of the system, what are the possible configurations of the system at the next step of computation. The next definition states this more formally.

Definition 2.12 (TRANSITION SYSTEM) *A* transition system *is a tuple* $\mathcal{S}=\langle C, c_0, \Rightarrow\rangle$ *such that:*

1. *C is a possibly infinite set of configurations;*
2. *$c_0 \in C$ is the* initial configuration*;*
3. *$\Rightarrow \subseteq C \times C$ is the transition relation.* ■

We adopt the following convention: we note $c_1 \Rightarrow c_2$ whenever c_1 and c_2 are two configurations of a transition system $\mathcal{S} = \langle C, c_0, \Rightarrow\rangle$ s.t. $(c_1, c_2) \in\Rightarrow$. We also use the notation $c_1 \Rightarrow^* c_2$ instead of $(c_1, c_2) \in\Rightarrow^*$.

Dynamics of a transition system The following definitions will be useful in order to precisely define the dynamics of transition systems:

Definition 2.13 (DYNAMICS OF TRANSITION SYSTEMS) *Given a transition system $\mathcal{S} = \langle C, c_0, \Rightarrow\rangle$, a quasiorder $\overline{\leq}$ s.t. $\langle C, \overline{\leq}\rangle$ is an ordered set, and a configuration c:*

1. $\mathsf{Post}(c)$ *denotes the set $\{c' \mid c \Rightarrow c'\}$ of one-step successors of c;*
2. $\mathsf{Pre}(c)$ *denotes the set $\{c' \mid c' \Rightarrow c\}$ of one-step predecessors of c;*
3. $\mathsf{PreUp}^{\overline{\leq}}(c)$ *denotes the set of all configurations whose one-step successors by $\Rightarrow$ are larger (w.r.t. $\overline{\leq}$) than c i.e.,* $\mathsf{PreUp}^{\overline{\leq}}(c) = \{c' \mid \exists c'' : c' \Rightarrow c'' \wedge c \overline{\leq} c''\}$.
4. $\mathsf{Post}^*(c)$ *denotes the set $\{c' \mid c \Rightarrow^* c'\}$ of successors of c;*
5. $\mathsf{Pre}^*(c)$ *denotes the set $\{c' \mid c' \Rightarrow^* c\}$ of predecessors of c;*
6. $\mathsf{Reach}(\mathcal{S})$ *denotes the set* $\mathsf{Post}^*(c_0)$. ■

In addition, when c' is a configuration in $\mathsf{Post}^*(c)$, we say that c' is *reachable* from c, and when c' is in $\mathsf{Reach}(\mathcal{S})$, we say that c' is reachable in $\mathcal{S}$, or simply that c' is reachable, when the transition system considered is clear from the context. We also extend all these operators (but Reach) to sets $C' \subseteq C$ in the natural way. For instance $\mathsf{Pre}(C') = \cup_{c' \in C'} \mathsf{Pre}(c')$, and so forth.

The semantics of a transition system is a set of *executions*, which are sequences of configurations where each pair of consecutive configurations satisfies the transition relation:

Definition 2.14 (EXECUTION) *Given a transition system $\mathcal{S} = \langle C, c_0, \Rightarrow\rangle$, an* execution *is:*

- *either a finite sequence $c_1, c_2, \ldots, c_\ell$ s.t. for any $1 \leq i < \ell$: $c_i \Rightarrow c_{i+1}$;*
- *or an infinite sequence $c_1, c_2, \ldots, c_i, \ldots$ s.t. for any $i \geq 1$: $c_i \Rightarrow c_{i+1}$.*

When $c_1 = c_0$, the execution is initialised. ■

When speaking about an execution of a transition system $\mathcal{S}$, we refer to an initialised execution, unless stated otherwise.

In the sequel, we will often assume that the transition systems we consider always have the ability to progress, whatever the state they are in. Such transition systems are called *deadlock-free*:

Definition 2.15 (DEADLOCK-FREENESS) *Let $\mathcal{S} = \langle C, c_0, \Rightarrow\rangle$ be a transition system, and let $\mathsf{Dead}(\mathcal{S})$ be its set of* deadlock configurations*, i.e.,*

$$\mathsf{Dead}(\mathcal{S}) = \{c \in C \mid \mathsf{Post}(c) = \emptyset\}$$

Then, $\mathcal{S}$ is deadlock-free *iff $\mathsf{Dead}(\mathcal{S}) = \emptyset$.* ■

Remark that any transition system can always be turned into a deadlock free transition systems with the same reachable configurations, by adding a *self-loop* on all configurations. That is, we consider the new transition relation $\Rightarrow' = \Rightarrow \cup \big\{(c, c) \mid c \in C\big\}$.

2.3.2 Well-structured Transition Systems

We are now equipped to define the central notion of this thesis: the *Well-Structured Transition Systems.*

Definition 2.16 (WELL-STRUCTURED TRANSITION SYSTEM) *A* Well-Structured Transition System *(*WSTS *for short) is a tuple $\mathcal{S} = \big\langle C, c_0, \Rightarrow, \overline{\leq}\big\rangle$ s.t.:*

- *$\langle C, c_0, \Rightarrow\rangle$ is a transition system;*
- *$\big\langle C, \overline{\leq}\big\rangle$ is an ordered set;*
- *$\overline{\leq}$ is a WQO;*
- *$\Rightarrow$ is $\overline{\leq}$-monotonic, that is: for any $c_1, c_2, c_3 \in C$ s.t.: $c_1 \Rightarrow c_2$ and $c_1 \overline{\leq} c_3$, there exists $c_4 \in C$ with $c_3 \Rightarrow^* c_4$ and $c_2 \overline{\leq} c_4$.* ■

A WSTS is thus a transition system equipped with a WQO ranging on its set of configurations. As a consequence, the definitions of Post, Pre, Post^*, Pre^*, and PreUp, as well as the notions of executions and reachable configurations given for a (plain) transition system at Definitions 2.13 and 2.14, extend naturally to the case of WSTS.

Example 2.7 *Let us consider* $\mathcal{S} = \langle \mathbb{N} \times \mathbb{N} \times \mathbb{N}, \langle 0, 1, 0 \rangle, \Rightarrow, \preccurlyeq \rangle$ *s.t.:*

- $\langle p_1, p_2, p_3 \rangle \preccurlyeq \langle q_1, q_2, q_3 \rangle$ *iff* $p_1 \leq q_1$ *and* $p_2 \leq q_2$ *and* $p_3 \leq q_3$ *and*
- $\langle p_1, p_2, p_3 \rangle \Rightarrow \langle q_1, q_2, q_3 \rangle$ *iff one of the following holds:*
  - $q_1 = p_1 + 1$ *and* $q_2 = p_2$ *and* $q_3 = p_3$ *or*
  - $p_1 \geq 1$ *and* $p_2 \geq 1$ *and* $q_1 = p_1 - 1$ *and* $q_2 = p_2 - 1$ *and* $q_3 = p_3 + 1$ *or*
  - $p_3 \geq 1$ *and* $q_1 = p_1 + 1$ *and* $q_2 = p_2 + 2$ *and* $q_3 = p_3 - 1$.

$\mathcal{S}$ *is a* WSTS. *The proof that* $\preccurlyeq$ *is a* WQO *can be found in the sequel. It is not difficult to see that such a system is* $\preccurlyeq$*-monotonic. Indeed, the conditions under which* $\langle p_1, p_2, p_3 \rangle \Rightarrow \langle q_1, q_2, q_3 \rangle$ *are of the form* $\wedge_i p_i \geq k_i$*, where the* k_i*'s are natural constants. Hence, if some configuration* $\langle p_1, p_2, p_3 \rangle$ *satisfies such a condition, then any configuration* $\langle p'_1, p'_2, p'_3 \rangle \succcurlyeq \langle p_1, p_2, p_3 \rangle$ *satisfies it too. Moreover, the 'effect' of the transitions consist in adding or subtracting natural constants from some coordinates, hence it is constant. Thus, there is a configuration* $\langle q'_1, q'_2, q'_3 \rangle$ *s.t.* $\langle p'_1, p'_2, p'_3 \rangle \Rightarrow \langle q'_1, q'_2, q'_3 \rangle$ *and* $\langle q_1, q_2, q_3 \rangle \preccurlyeq \langle q'_1, q'_2, q'_3 \rangle$. $\Diamond$

Motivation of the definition of WSTS The definition of WSTS is very general and may seem rather remote from the models of computation that have proved to be useful in practice when modelling computer systems. Nevertheless, we show in the remaining parts of this section that several well-known models of computation are WSTS. These models are:

1. The monotonic extensions of Petri nets, such as the Self-modifying Petri nets introduced in [Val78], the Petri nets with transfer arcs [Cia94], the Petri nets with non-blocking arcs [RVB04], and so forth... (see section 2.3.3).
2. The Lossy Channel Systems [AJ93, ABJ98, AAB99], which are sets of finite automata that communicate through lossy FIFO channels (see section 2.3.4).
3. The Broadcast Protocols [EN98], which are sets of finite automata that communicate through *rendez-vous* and broadcast synchronisations (see section 2.3.5).
4. The timed Petri nets, as defined by Abdulla *et al.* [ADMN04] (see section 2.3.5).

2.3.3 Monotonic Extensions of Petri nets

In his PhD thesis [Pet62], Karl Adam Petri has introduced a model of computation which is suitable to express the behaviour of concurrent systems. This model has been coined *Petri net* as a tribute to his inventor. Since then, a considerable amount of work has been devoted to the study of Petri nets, some of them proposing various *extensions* of the seminal definition.

A particularly general extension of the Petri nets is due to Valk and is called *Self-Modifying Petri Nets* [Val78] (SMPN for short). In this thesis, we will restrict our attention to a sub-class of SMPN: the *strongly monotonic* SMPN. This restriction is motivated by the fact that the Petri nets, as well as several interesting extensions of Petri nets that have been introduced in other works (such as: the Petri nets with transfer arcs, with non-blocking arcs, *etc.*) are (in some sense) equivalent to some strongly monotonic SMPN. Moreover, the *strong monotonicity* property of these SMPN will be important when introducing the *Expand, Enlarge and Check* algorithm in Chapter 4.

We open this section with the definitions of SMPN and strongly monotonic SMPN. Then, we introduce the class of *Extended Petri nets*[2] and discuss its relation to the class of strongly monotonic SMPN.

Self-Modifying Petri nets An SMPN is made up of a set of *places*, which are intuitively regarded as resources. The availability of each resource is represented by *putting* some tokens inside the places. The amount of tokens inside a place indicates the quantity of resource available. A function expressing how many tokens are assigned to each place is called a *marking*. The dynamic part of the system is expressed by the transitions, which *consume* (remove) some tokens in certain places and *produce* tokens in other places. The exact effect of each transition depends on the current marking:

Definition 2.17 ([Val78]) (SELF-MODIFYING PETRI NETS) *A* Self-Modifying Petri net*, SMPN for short, is a tuple $\mathcal{N} = \langle P, T, D^-, D^+, \mathbf{m}_0 \rangle$ s.t.:*

- $P = \{p_1, \dots, p_{|P|}\}$ *is a finite set of places. A* marking *is a function* $\mathbf{m} : P \mapsto \mathbb{N}$ *that assigns a natural value to each place. In the following, markings are also seen as tuples in* $\mathbb{N}^{|P|}$ *where the ith component is the value assigned to place* p_i. *Given a set of places* $\{p_1, p_2, \dots, p_k\}$, *we denote by* $\mathbf{m}\big(\{p_1, p_2, \dots, p_k\}\big)$ *the value* $\sum_{1 \le i \le k} \mathbf{m}(p_i)$;
- $T = \{t_1, \dots, t_{|T|}\}$ *is a finite set of transitions with* $T \cap P = \emptyset$;
- *For any* $1 \le i \le |T|$ *and any* $1 \le j \le |P|$, $D^-_{ij} : \mathbb{N}^{|P|} \mapsto \mathbb{N}$ *and* $D^+_{ij} : \mathbb{N}^{|P|} \mapsto \mathbb{N}$ *describe respectively the input and output effect of transition* t_i *on place* p_j.

[2]which encompass the Petri nets, the Petri nets with transfer arcs and the Petri nets with non-blocking arcs

Namely, D^-_{ij} and D^+_{ij} are functions of the marking $\mathbf{m}$ of the form[3] $\alpha + \sum_{k=1}^{|P|} \beta_k \cdot \mathbf{m}(p_k)$ *where $\alpha \in \mathbb{N}$ and $\beta_k \in \mathbb{N}$ for all $1 \leq k \leq |P|$;*

- $\mathbf{m}_0$ *is the initial marking of $\mathcal{N}$.* ■

A very widespread graphical representation of SMPN is adopted in this thesis. Each place $p \in P$ is represented by a circle (sometimes labelled by the name of the place). Each transition $t \in T$ is represented by a filled rectangle (that can be labelled by the name of the transition). Arrows are drawn from places to transitions and from transitions to places, in order to represent the effect of the transitions. More precisely, if $D^-_{ij} \neq 0$, then an arrow labelled by D^-_{ij} is drawn from p_j to t_i. Similarly, when $D^+_{ij} \neq 0$, an arrow labelled by D^+_{ij} is drawn from t_i to p_j. When the label of the arrow should be 1, we sometimes omit it on the figure. Finally, a marking $\mathbf{m}$ can also be represented in a graphical fashion, by drawing $\mathbf{m}(p)$ black tokens inside each place p. The two following examples should make this clear.

Example 2.8 *Let us consider the* SMPN $\mathcal{N}_\mu = \langle P, T, D^-, D^+, \mathbf{m}_0 \rangle$ *s.t.:*

- $P = \{p_1, p_2, p_3\}$;
- $T = \{t_1, t_2, t_3\}$;
- $D^- = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$;
- $D^+ = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$;
- $\mathbf{m}_0 = \langle 0, 1, 0 \rangle$.

In this definition, we have represented the functions D^+_{ij} and D^-_{ij} as matrices: the element in line i, column j of the matrix D^- (resp. D^+) is the value of function D^-_{ij} (D^+_{ij}). Thus, in the present case, all the β_k factors are equal to 0. $\mathcal{N}_\mu$ is graphically represented at Figure 2.1. ◊

Example 2.9 *Let us consider the* SMPN $\mathcal{N}_{ns} = \langle P, T, D^-, D^+, \mathbf{m}_0 \rangle$ *s.t.:*

- $P = \{p_1, p_2\}$;
- $T = \{t_1, t_2\}$;

[3]Naturally, terms whose coefficient are null might be omitted.

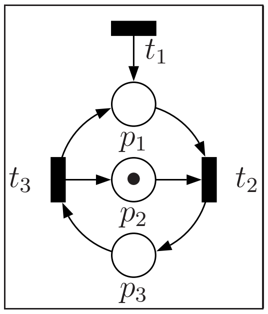


Figure 2.1: The SMPN $\mathcal{N}_\mu$.

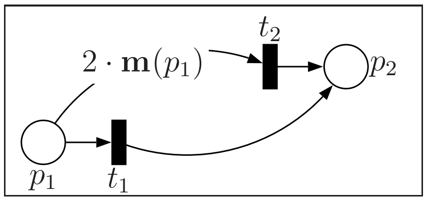


Figure 2.2: The SMPN $\mathcal{N}_{ns}$.

- $D^-_{11}(\mathbf{m}) = 1$, $D^-_{21}(\mathbf{m}) = 2 \cdot \mathbf{m}(p_1)$ *and* $D^-_{12}(\mathbf{m}) = D^-_{22}(\mathbf{m}) = 0$;
- $D^+_{22}(\mathbf{m}) = D^+_{12}(\mathbf{m}) = 1$ *and* $D^+_{11}(\mathbf{m}) = D^+_{21}(\mathbf{m}) = 0$;
- $\mathbf{m}_0 = \langle 0, 0 \rangle$.

It is graphically represented at Figure 2.2. ◊

Dynamics of SMPN We have already sketched the way an SMPN can evolve, by letting its transitions move tokens from one place to another. Let us define this more precisely.

Definition 2.18 (Enabled transition, effect of a transition) *Given an* SMPN $\mathcal{N} = \langle P, T, D^-, D^+, \mathbf{m}_0 \rangle$, *and a marking* $\mathbf{m}$ *of the places of* $\mathcal{N}$, *a transition* t_i *is* firable *(or* enabled*) from a marking* $\mathbf{m}$ *if* $\mathbf{m}(p_j) \geq D^-_{ij}(\mathbf{m})$ *for all* $p_j \in P$. *This is denoted by* $\mathbf{m} \xrightarrow{t_i}$. Firing *an enabled* t_i *from* $\mathbf{m}$ *leads to a marking* $\mathbf{m}' \in \mathbb{N}^{|P|}$. *This is noted* $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$, *and* $\mathbf{m}'$ *is computed as follows. First, we compute* $\mathbf{m}''$, *s.t. for any* $p_j \in P : \mathbf{m}''(p_j) = \mathbf{m}(p_j) - D^-_{ij}(\mathbf{m})$. *Then, we let* $\mathbf{m}'$ *be s.t. for any* $p_j \in P : \mathbf{m}'(p_j) = \mathbf{m}''(p_j) + D^+_{ij}(\mathbf{m})$. ■

Remark that the two steps in the computation of $\mathbf{m}'$ can be swapped when we manipulate (plain) markings of SMPN. However, the order of these steps will become relevant when we will manipulate extended markings (see Section 3.1.1)

From Definition 2.18, it is easy to see that an SMPN $\mathcal{N} = \langle P, T, D^-, D^+, \mathbf{m}_0\rangle$ naturally defines a transition system $\mathcal{S}_{\mathcal{N}} = \left\langle \mathbb{N}^{|P|}, \mathbf{m}_0, \Rightarrow\right\rangle$, where $\Rightarrow$ is such that we have $\mathbf{m}_1 \Rightarrow \mathbf{m}_2$, if and only if there exists $t_i \in T$ such that t_i is firable from $\mathbf{m}_1$ and $\mathbf{m}_1 \xrightarrow{t_i} \mathbf{m}_2$.

We adopt the following notational conventions throughout this thesis. Let $\sigma = t_1 t_2 \ldots t_n$ be a (possibly empty) sequence of n transitions (hence, if σ is empty, we have $n = 0$). We write $\mathbf{m} \xrightarrow{\sigma} \mathbf{m}'$ to mean that there are $\mathbf{m}_1, \ldots, \mathbf{m}_{n+1}$ such that $\mathbf{m}_1 = \mathbf{m}$, $\mathbf{m}_{n+1} = \mathbf{m}'$ and $\mathbf{m}_1 \xrightarrow{t_1} \mathbf{m}_2 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \mathbf{m}_{n+1}$. We sometimes write $\mathbf{m} \xrightarrow{*} \mathbf{m}'$ to mean that there exists a sequence of transitions σ such that $\mathbf{m} \xrightarrow{\sigma} \mathbf{m}'$. We also write $\mathbf{m} \rightarrow \mathbf{m}'$ to mean that there exists a transition t s.t. $\mathbf{m} \xrightarrow{t} \mathbf{m}'$. Moreover, we sometimes confuse an SMPN $\mathcal{N}$ with its associated transition system $\mathcal{S}_{\mathcal{N}}$, and write, for instance $\mathsf{Reach}(\mathcal{N})$ instead of $\mathsf{Reach}(\mathcal{S}_{\mathcal{N}})$.

Example 2.10 *In the SMPN $\mathcal{N}_\mu$ of Example 2.8, transition t_1 is always enabled and if $\mathbf{m} \xrightarrow{t_1} \mathbf{m}'$, then $\mathbf{m}'(p_1) = \mathbf{m}(p_1) + 1$, $\mathbf{m}'(p_2) = \mathbf{m}(p_2)$ and $\mathbf{m}'(p_3) = \mathbf{m}(p_3)$. Transition t_2 is enabled only if there is at least one token in p_1 and at least one token in p_2. t_3 is enabled only if there is at least one token in p_3. The set $\mathsf{Reach}(N_\mu)$ is $\{\langle i,1,0\rangle \mid i \geq 1\} \cup \{\langle i,0,1\rangle \mid i \geq 1\}$. Finally, let us illustrate the notation $\mathbf{m} \xrightarrow{\sigma} \mathbf{m}'$, by remarking that, for instance, $\langle 1,1,0\rangle \xrightarrow{t_2t_3t_2t_3t_2} \langle 0,0,1\rangle$.*

In the SMPN $\mathcal{N}_{ns}$ of Example 2.9, transition t_2 is enabled in $\mathbf{m}$ iff $\mathbf{m}(p_1) = 0$. Indeed, if $\mathbf{m}(p_1) > 1$, the condition $\mathbf{m}(p_1) \geq 2 \cdot \mathbf{m}(p_1)$ is false. ◊

We have just shown that any SMPN $\mathcal{N}$ defines a transition system $\mathcal{S}_{\mathcal{N}}$. However, this does not mean that $\mathcal{S}_{\mathcal{N}}$ is a WSTS. In order to identify which SMPN define a WSTS, we need a WQO to compare markings.

A WQO for the markings The WQO that we use to compare two markings is $\preccurlyeq$, defined as follows:

Definition 2.19 (The WQO $\preccurlyeq$) *Let P be a set of places of an SMPN. Then, the ordering $\preccurlyeq \subseteq \mathbb{N}^{|P|} \times \mathbb{N}^{|P|}$ is s.t. for any pair of markings $\mathbf{m}_1$ and $\mathbf{m}_2$:*

$$\mathbf{m}_1 \preccurlyeq \mathbf{m}_2 \textit{ iff for any } 1 \leq i \leq |P| : \mathbf{m}_1(p_i) \leq \mathbf{m}_2(p_i)$$

■

Remark that $\preccurlyeq$ is, by definition, a partial order. We write $\mathbf{m}_1 \prec \mathbf{m}_2$ iff $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$ but $\mathbf{m}_2 \not\preccurlyeq \mathbf{m}_1$. As usual,$\mathbf{m}_2 \succcurlyeq \mathbf{m}_1$ iff $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$ and $\mathbf{m}_2 \succ \mathbf{m}_1$ iff $\mathbf{m}_1 \prec \mathbf{m}_2$. Since $\preccurlyeq$ is a partial order, either $\mathbf{m}_1 \prec \mathbf{m}_2$ or $\mathbf{m}_1 \succ \mathbf{m}_2$ both imply that $\mathbf{m}_1 \neq \mathbf{m}_2$.

Let us show that this ordering is indeed a WQO. For that purpose, we first prove Lemma 2.2. It allows us to deduce that $\preccurlyeq$ is a WQO.

Lemma 2.12 *Let P be a set of places of an* SMPN*. Let $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_i, \ldots$ be an infinite sequence of markings ranging over P. Then there exists a strictly increasing function $\rho : \mathbb{N} \mapsto \mathbb{N}$ s.t. for any $i \geq 1$: $\mathbf{m}_{\rho(i)} \preccurlyeq \mathbf{m}_{\rho(i+1)}$.*

Proof. Let us consider the sequence $\mathbf{m}_1(p_1), \mathbf{m}_2(p_1), \ldots, \mathbf{m}_i(p_1), \ldots$ Since this is an infinite sequence of natural number and since $\leq$ is a WQO on $\mathbb{N}$, there exists, by Lemma 2.2, a function ρ_1 s.t. the sequence $\mathbf{m}_{\rho_1(1)}(p_1), \mathbf{m}_{\rho_1(2)}(p_1), \ldots, \mathbf{m}_{\rho_1(i)}(p_1), \ldots$ is an infinite $\leq$-increasing sequence, i.e., for any $i \geq 1$: $\mathbf{m}_{\rho_1(i)}(p_1) \leq \mathbf{m}_{\rho_1(i+1)}(p_1)$. Hence, we obtain an infinite sequence of markings $S_1 = \mathbf{m}_{\rho_1(1)}, \mathbf{m}_{\rho_1(2)}, \ldots, \mathbf{m}_{\rho_1(i)}, \ldots$ that is increasing on the first coordinate. Starting from that sequence, one can repeat the same reasoning on place p_2 an obtain a function ρ_2 s.t. the sequence $S_2 = \mathbf{m}_{\rho_2(1)}, \mathbf{m}_{\rho_2(2)}, \ldots, \mathbf{m}_{\rho_2(i)}, \ldots$ is increasing on the second and the first coordinate (since S_2 is a subsequence of S_1). By repeating this construction for any place of P, we eventually obtain a function $\rho_{|P|}$. That function allows us to define the sequence $S_{|P|} = \mathbf{m}_{\rho_{|P|}(1)}, \mathbf{m}_{\rho_{|P|}(2)}, \ldots, \mathbf{m}_{\rho_{|P|}(i)}, \ldots$ s.t. for any $i \geq 1 : \mathbf{m}_{\rho_{|P|}(i)} \preccurlyeq \mathbf{m}_{\rho_{|P|}(i+1)}$. We let $\rho = \rho_{|P|}$ to obtain the lemma. □

From the previous lemma, we obtain a proof that $\preccurlyeq$ is a WQO:

Proposition 2.1 *$\preccurlyeq$ is a* WQO*.*

Proof. Let $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_i, \ldots$ be an infinite sequence of markings ranging over the same set of places P, and let us show that there are two positions k and ℓ s.t. $k < \ell$ and $\mathbf{m}_k \preccurlyeq \mathbf{m}_\ell$. By Lemma 2.12, there exists a strictly increasing function $\rho : \mathbb{N} \mapsto \mathbb{N}$ s.t. for any $i \geq 1$: $\mathbf{m}_{\rho(i)} \preccurlyeq \mathbf{m}_{\rho(i+1)}$. We obtain the lemma by letting, for instance, $k = \rho(1)$ and $\ell = \rho(2)$. □

Monotonic and strongly monotonic SMPN Remark that not every SMPN defines a transition system whose transition relation is $\preccurlyeq$-monotonic. For instance, consider the SMPN of Example 2.9, *without transition t_1*. The transition relation $\Rightarrow$ of the transition system that corresponds to this SMPN is not $\preccurlyeq$-monotonic. Indeed, we have seen in Example 2.10, that t_2 is enabled in $\mathbf{m}$ when $\mathbf{m}(p_1) = 0$, but not when $\mathbf{m}(p_1) \geq 1$. Since we have suppressed the transition t_1, there is no transition enabled when $\mathbf{m}(p_1) \geq 1$. Hence $\Rightarrow = \{(\langle 0, 0\rangle, \langle 0, 1\rangle)\}$, which is clearly not monotonic. Thus, we need the following definition:

Definition 2.20 (Monotonic SMPN) *An* SMPN $\mathcal{N} = \langle P, T, D^-, D^+, \mathbf{m}_0\rangle$ *is* monotonic *iff, for any $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_3$ s.t. $\mathbf{m}_1 \rightarrow \mathbf{m}_2$ and $\mathbf{m}_1 \preccurlyeq \mathbf{m}_3$, there exists $\mathbf{m}_4$ s.t. $\mathbf{m}_3 \xrightarrow{*} \mathbf{m}_4$ and $\mathbf{m}_2 \preccurlyeq \mathbf{m}_4$.* ■

Obviously, the transition system that corresponds to a monotonic SMPN has a $\preccurlyeq$-monotonic transition relation.

Remark that, in this definition, we impose no restriction on the sequence of transitions σ s.t. $\mathbf{m}_3 \xrightarrow{\sigma} \mathbf{m}_4$. In particular, it could be the case that $\mathbf{m}_1 \xrightarrow{t} \mathbf{m}_2$ for some transition t, but that $\mathbf{m}_3 \xrightarrow{t}$ does not hold. For instance, consider the SMPN $\mathcal{N}_{ns}$ of Example 2.9. It is clearly monotonic. But if we let $\mathbf{m}_1 = \langle 0, 0\rangle$, $\mathbf{m}_2 = \langle 0, 1\rangle$ and $\mathbf{m}_3 = \langle 1, 0\rangle$, then $\mathbf{m}_1 \xrightarrow{t_2} \mathbf{m}_2$ and $\mathbf{m}_1 \preccurlyeq \mathbf{m}_3$, but t_2 is not enabled in $\mathbf{m}_3$. However, t_1 is and $\mathbf{m}_3 \xrightarrow{t_1} \langle 0, 1\rangle \succcurlyeq \mathbf{m}_2$. This motivates the following more restrictive definition:

Definition 2.21 (Strongly Monotonic SMPN) *$\mathcal{N} = \langle P, T, D^-, D^+, \mathbf{m}_0\rangle$ is a* strongly monotonic SMPN *iff $\mathcal{N}$ is an* SMPN *and for any $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_3$ s.t. there exists $t \in T$ with $\mathbf{m}_1 \xrightarrow{t} \mathbf{m}_2$ and $\mathbf{m}_1 \preccurlyeq \mathbf{m}_3$: there exists $\mathbf{m}_4$ s.t. $\mathbf{m}_3 \xrightarrow{t} \mathbf{m}_4$ and $\mathbf{m}_2 \preccurlyeq \mathbf{m}_4$.* ■

Clearly, any strongly monotonic SMPN is monotonic. The SMPN of Example 2.9 is monotonic but not strongly monotonic, whereas the SMPN of Example 2.8 is strongly monotonic. We will see in the sequel that several classes of extended Petri nets studied in the literature (Petri nets, Petri nets with transfer arcs and Petri nets with non-blocking arcs) are strongly monotonic SMPN or are (in some sense) equivalent to strongly monotonic SMPN.

It is possible to syntactically identify the class of strongly monotonic SMPN, as stated by the following lemma. We say that a transition t is *unfirable*, whenever there exists no marking $\mathbf{m}$ such that t is enabled in $\mathbf{m}$. In this thesis, we assume that all the SMPN we consider do not contain unfirable transitions[4].

Lemma 2.13 *Given an* SMPN *$\mathcal{N} = \langle P, T, D^-, D^+, \mathbf{m}_0\rangle$ without unfirable transitions, $\mathcal{N}$ is strongly monotonic if and only if for all $t_i \in T, p_j \in P$: either $D^-_{ij} = \alpha$ with $\alpha \in \mathbb{N}$ or $D^-_{ij} = \mathbf{m}(p_j)$.*

Proof. We prove the two directions of the iff independently.

1. *If $\mathcal{N}$ is strongly monotonic then $t_i \in T, p_j \in P : D^-_{ij} = \alpha$ with $\alpha \in \mathbb{N}$ or $D^-_{ij} = \mathbf{m}(p_j)$.* We proceed *per absurdum*. Suppose that it is not the case, that is $\mathcal{N}$ is $\preccurlyeq$-strongly monotonic and there exist $t_i \in T, p_j \in P$ such that D^-_{ij} is not of the form α with $\alpha \in \mathbb{N}$ or $\mathbf{m}(p_j)$. Let $D^-_{ij} = \sum_{p_k \in P} \beta_k \cdot \mathbf{m}(p_k) + \alpha$. We consider three cases:

   (a) $\beta_j > 1$. In that case, any marking $\mathbf{m}$ from which t_i is firable must satisfy $\mathbf{m}(p_j) = 0$. Let $\mathbf{m}'$ be a marking s.t. $\mathbf{m} \preccurlyeq \mathbf{m}'$ and $\mathbf{m}'(p_j) \geq 1$. Clearly, t_i is not firable from $\mathbf{m}'$ although $\mathbf{m}' \succcurlyeq \mathbf{m}$. Thus, $\mathcal{N}$ is not (strongly) monotonic. Contradiction.

[4]Remark that t_i is unfirable iff the system of linear inequations $\mathbf{m}(p_j) \geq D_i j^-(\mathbf{m})$ (for $j = 1, 2, \ldots, |P|$) admits no solution in $\mathbb{N}^{|P|}$ (where the variables are $\mathbf{m}(p_1)$, $\mathbf{m}(p_2)$,..., $\mathbf{m}(p_{|P|})$).

(b) ($\beta_j = 1$ and $\alpha > 0$). In this case, t_i is unfirable. Contradiction.

(c) $\beta_j = 0$ or ($\beta_j = 1$ and $\alpha = 0$). Since D^-_{ij} is not of the form α or $\mathbf{m}(p_j)$, there is $k' \neq j$ such that $\beta_{k'} > 0$. By hypothesis, t_i is firable from at least one marking $\mathbf{m}$. Let us construct the marking $\mathbf{m}'$ as follows: $\forall p_k \neq p_{k'} \in P$: $\mathbf{m}'(p_k) = \mathbf{m}(p_k)$, and $\mathbf{m}'(p_{k'}) = \mathbf{m}(p_{k'}) + \mathbf{m}(p_j) + 1$. By construction, $\mathbf{m} \preccurlyeq \mathbf{m}'$ but t_i is not firable from $\mathbf{m}'$. Indeed, for t_i to be firable we should have $\mathbf{m}'(p_j) = \mathbf{m}(p_j) \geq D^-_{ij}(\mathbf{m}') \geq \beta_{k'} \cdot \big(\mathbf{m}(p_{k'}) + \mathbf{m}(p_j) + 1\big)$. Since $\beta_{k'} > 0$, this is not possible. We conclude that $\mathcal{N}$ is not $\preccurlyeq$-strongly monotonic. Contradiction.

2. *If for any $t_i \in T$, for any $p_j \in P$: $D^-_{ij} = \alpha$ with $\alpha \in \mathbb{N}$ or $D^-_{ij} = \mathbf{m}(p_j)$ then $\mathcal{N}$ is $\preccurlyeq$-strongly monotonic.* We proceed *per absurdum* again. Suppose that $\mathcal{N}$ is not $\preccurlyeq$-strongly monotonic but for all $t_i \in T, p_j \in P : D^-_{ij} = \alpha$ with $\alpha \in \mathbb{N}$ or $D^-_{ij} = \mathbf{m}(p_j)$. Hence there are three markings $\mathbf{m}_1, \mathbf{m}_2$ and $\mathbf{m}_3$ and a transition t_i such that $\mathbf{m}_1 \xrightarrow{t_i} \mathbf{m}_2$, $\mathbf{m}_1 \preccurlyeq \mathbf{m}_3$ and there does not exist a marking $\mathbf{m}_4$ such that $\mathbf{m}_3 \xrightarrow{t_i} \mathbf{m}_4$ and $\mathbf{m}_2 \preccurlyeq \mathbf{m}_4$.

   Since $\mathbf{m}_1 \preccurlyeq \mathbf{m}_3$ and $\mathbf{m}_1(p_j) \geq D^-_{ij}(\mathbf{m}_1)$ for all $p_j \in P$, $\mathbf{m}_3(p_j) \geq D^-_{ij}(\mathbf{m}_3)$ for all $p_j \in P$. As a consequence, t_i is firable from $\mathbf{m}_3$. Suppose that $\mathbf{m}_3 \xrightarrow{t_i} \mathbf{m}_4$.

   Let $\mathbf{m}'_k$ ($k \in \{1, 2\}$) be such that $\mathbf{m}'_k(p_j) = \mathbf{m}_k(p_j) - D^-_{ij}(\mathbf{m}_k)$ for all $p_j \in P$. Since $\mathbf{m}_1 \preccurlyeq \mathbf{m}_3$, $\mathbf{m}'_1 \preccurlyeq \mathbf{m}'_3$. Moreover, we have that $D^+_{ij}(\mathbf{m}_1) \leq D^+_{ij}(\mathbf{m}_3)$ for all j such that $1 \leq j \leq |P|$. Since $\mathbf{m}_2(p_j) = \mathbf{m}'_1(p_j) + D^+_{ij}(\mathbf{m}_1)$ and $\mathbf{m}_4(p_j) = \mathbf{m}'_3(p_j) + D^+_{ij}(\mathbf{m}_3)$ for all $p_j \in P$, we conclude that $\mathbf{m}_2 \preccurlyeq \mathbf{m}_4$. Contradiction.

□

SMPN are WSTS Our interest in strongly monotonic SMPN is of course highly motivated by the fact that any strongly monotonic SMPN defines a transition system that is a WSTS when considering the WQO $\preccurlyeq$.

Proposition 2.2 *Let $\mathcal{N} = \langle P, T, D^+, D^-, \mathbf{m}_0 \rangle$ be a strongly monotonic SMPN and let $\mathcal{S}_{\mathcal{N}} = \langle S, s_0, \Rightarrow \rangle$ be the transition system that corresponds to $\mathcal{N}$. Then, $\langle S, s_0, \Rightarrow, \preccurlyeq \rangle$ is a WSTS.*

Proof. By hypothesis, the transition relation of $\mathcal{N}$ is monotonic. Moreover, $\preccurlyeq$ is a WQO by Proposition 2.1. Hence, $\langle S, s_0, \Rightarrow, \preccurlyeq \rangle$ is a WSTS by Definition 2.16. □

Motivation of the SMPN model Let us devote a few lines to explain why (extensions of) Petri nets are such an interesting and widespread model in the field of verification of concurrent systems. For that purpose, we recall briefly the notion of *counting abstraction*, through the following example.

Example 2.11 *Let us consider the following excerpt of pseudo-code. It describes a single process, and we will consider all the possible concurrent systems that one can obtain by executing one or several copies of this process at the same time.*

```
Shared Semaphore s ;

Process P {
   while(true) {
      s.try() ;
      Critical section ;
      s.release() ;
   }
}
```

Each process has to execute a critical section*, i.e., a sequence of instructions that cannot be executed by two different processes at the same time. For that reason, a* mutual exclusion *policy has to be enforced. This is obtained thanks to introduction of a* semaphore `s`*, which is shared by the different copies of the process. Initially, the semaphore is in the state 1. When a process executes the* `s.try()` *command, the semaphore goes to state 0. If a process tries to execute* `s.try()` *while the semaphore is in state 0, the process is blocked: it has to make another call to* `try` *later in order to try again to get access to the critical section. The effect of the* `s.release()` *command is to change the state of the semaphore to 1, hence allowing other processes to get access to the critical section thanks to a successful* `try`*.*

*A natural question that arises about this system is: '*is the mutual exclusion policy enforced ?*' or, '*Is there an execution of the whole system that reaches a state where two processes are in the critical section at the same time ?*' Of course, we want to decide this automatically, and for any number of processes in the system. This analysis can easily be done by devising an abstract model of the system, which is called a* counting abstraction*. Roughly speaking, a counting abstraction is a new (infinite) system whose states retain* how many *processes are in each state of the original system (plus the value of the shared variable). For instance, if we only retain the states '*outside critical section*' (state 1) and '*inside critical section*' (state 2) for the processes, a counting abstraction of our mutual exclusion system would be a transition system whose states are vectors of three natural numbers v, where $v[i]$ ($i = 1, 2$) counts how many processes are in state i, and $v[3]$ encodes the value of* `s`*. Such an abstraction is perfectly described by an* SMPN*. As a matter of fact the* SMPN $\mathcal{N}_\mu$ *of Example 2.8 is the counting* *abstraction we have just described. Places p_1 and p_3 count respectively how many processes are outside and inside the critical section, and place p_2 encodes the value of* `s`*. Remark that transition t_1 can 'create' tokens (processes) in place p_1 at any time, which allows us to consider, in a single abstraction, all the possible number of processes in the system*[5]*. Our question then becomes '*Is there $\mathbf{m} \in \mathsf{Reach}(\mathcal{N}_\mu)$ s.t. $\mathbf{m}(p_3) \geq 2$?*' We will see in the sequel that this question is an instance of the* Coverability Problem*, which is decidable on a large subclass of* SMPN *(which the present example belongs to).* ◊

Since the notion of *counting abstraction* is quite outside the topic of this thesis, we will not discuss it into further details. We refer the interested reader to [GS92] for the basic ideas, and to [Van03, DRVB02, DB01, Del00] for practical applications of this concept.

Extended Petri nets In the previous example, we have considered processes that somehow communicate through the semaphore only. In practice, other means of communications are offered by the programming languages one can use to implement a concurrent system. For instance, in the JAVA programming language, two keywords `notify` and `notifyAll` are available. Their respective meaning is, roughly speaking, to send a message to one non-deterministically chosen process or to all the processes (broadcast) from a given subset of the active processes (see [Lea00, OW99, Gra97b, Fla97] for more details). Similar communication procedures exists in other common programming languages. These communication procedures can be modelled in an SMPN, but their syntax (in particular the D^+ and D^- matrices) are not very intuitive for that purpose. On the other hand, the Petri nets with transfer arcs and the Petri nets with non-blocking arcs we are about to introduce have a syntax which simplify the modelling of the aforementioned communication procedures. Moreover, these classes have been extensively studied in the literature as in [Pet81, Cia94, Van03, RVB04, FGRVB06, GRVB06c].

Following the presentation of [FGRVB06], we have grouped these two classes, along with the (plain) Petri nets [Pet62] and the Petri nets with reset arcs, into the class of Extended Petri Nets (EPN for short), as stated by the next definition.

[5]Remark that, in the present case, we do not need to consider the *destruction* of processes, although this behaviour can easily be added to our model, simply by adding a transition that consumes only one token in p_1 and produces no token. Indeed, we are interested in coverability properties and the system is monotonic. Hence, if the set of *bad states* is reachable by an execution of the system that involves the destruction of some processes, it is also reachable by an execution along which no processes are destroyed. On the other hand, the destruction of processes does not allow *more processes* to enter the critical section.

Definition 2.22 ([FGRVB06]) (Extended Petri nets) *An* Extended Petri Net *(*EPN*) $\mathcal{N}$ is a tuple $\langle P, T, \mathbf{m}_0 \rangle$, where:*

- *P is a finite set $\{p_1, p_2, \ldots, p_n\}$ of places;*
- *T is a finite set of transitions. Each transition is of the form $\langle I, O, s, d, b \rangle$, where I and $O : P \mapsto \mathbb{N}$ are multi-sets of input and output places respectively. By convention, $O(p)$ (resp. $I(p)$) denotes the number of occurrences of p in O (resp. I). $s, d \in P \uplus \{\bot\}$ are the source and the destination places respectively of a* special arc*, and $b \in \mathbb{N} \cup \{+\infty\}$ is the bound associated to the special arc;*
- *$T \cap P = \emptyset$;*
- *$\mathbf{m}_0$ is the initial marking.*

Let us partition T into T_r and T_e such that $T = T_r \uplus T_e$. Without loss of generality, we assume that for each transition $\langle I, O, s, d, b \rangle \in T$, either $b = 0$ and $s = \bot = d$ (regular Petri transitions, grouped into T_r); or $b > 0$, $s \neq d$, $s \neq \bot$ and $d \neq \bot$ (extended transitions, grouped into T_e). We identify several non-disjoint classes of EPN*, depending on T_e:*

1. Petri nets *(*PN *for short): an* EPN *is a* PN *iff $T_e = \emptyset$;*
2. Petri nets with non-blocking arcs *(*PN+NBA*): an* EPN *is a* PN+NBA *iff for any $t = \langle I, O, s, d, b \rangle$ in T_e: $b = 1$;*
3. Petri nets with transfer arcs *(*PN+T*): an* EPN *is a* PN+T *iff for any $t = \langle I, O, s, d, b \rangle$ in T_e: $b = +\infty$;*
4. Petri nets with reset arcs *(*PN+R*): an* EPN *is a* PN+R *iff there exists one place $p_T \in P$, called the* trash can*, s.t., for any $t = \langle I, O, s, d, b \rangle$ in T: $I(p_T) = O(p_T) = 0$, and $t \in T_e$ implies that $d = p_T$, $s \neq p_T$, and $b = +\infty$. Remark that any* PN+R *is also a* PN+T*.* ■

The graphical convention we adopt to depict EPN is very similar to that for SMPN. Places are graphically depicted by circles; transitions by filled rectangles. For any transition $t = \langle I, O, s, d, b \rangle$, we draw an arrow from any place $p \in I$ to transition t and from t to any place $p \in O$. When $I(p)$ (resp. $O(p)$) is strictly greater than 1, we label the corresponding arrow by $I(p)$ ($O(p)$). For a PN+NBA, we draw a dotted arrow from s to t and from t to d (provided that $s, d \neq \bot$). For every PN+T that is not a PN+R, we draw a thick grey arrow from s to t and from t to d (provided that $s, d \neq \bot$). For a PN+R, the *trash can* is usually not depicted since no token can ever escape from it. For any transition t of the form $\langle I, O, s, p_T, +\infty \rangle$ of a PN+R, we simply draw an edge bearing a cross from s to t.

Let us now define the dynamics of an EPN. As in the case of SMPN, transitions can be enabled or not in some markings, and, when they are enabled, they can fire and modify the marking. This is stated by the following definition:

Definition 2.23 (Enabled transitions and firing) *Given an extended Petri net $\mathcal{N} = \langle P, T, \Sigma, \mathbf{m}_0 \rangle$, and a marking $\mathbf{m}$ of $\mathcal{N}$, a transition $t = \langle I, O, s, d, b \rangle$ is said to be* enabled in $\mathbf{m}$ *(notation: $\mathbf{m} \xrightarrow{t}$) iff $\forall p \in P : \mathbf{m}(p) \geq I(p)$. An enabled transition $t = \langle I, O, s, d, b \rangle$ can* occur *or* fire*, which deterministically transforms the marking $\mathbf{m}$ into a new marking $\mathbf{m}'$ (we denote this by $\mathbf{m} \xrightarrow{t} \mathbf{m}'$). $\mathbf{m}'$ is computed as follows:*

1. *First compute $\mathbf{m}_1$ such that: $\forall p \in P : \mathbf{m}_1(p) = \mathbf{m}(p) - I(p)$.*
2. *Then compute $\mathbf{m}_2$ as follows. If $s = d = \bot$, then $\mathbf{m}_2 = \mathbf{m}_1$. Otherwise:*
$$\mathbf{m}_2(s) = \begin{cases} 0 & \text{if } \mathbf{m}_1(s) \leq b \\ \mathbf{m}_1(s) - b & \text{otherwise} \end{cases}$$
and
$$\mathbf{m}_2(d) = \begin{cases} \mathbf{m}_1(d) + \mathbf{m}_1(s) & \text{if } \mathbf{m}_1(s) \leq b \\ \mathbf{m}_1(d) + b & \text{otherwise} \end{cases}$$
and
$$\forall p \in P \setminus \{d, s\} : \mathbf{m}_2(p) = \mathbf{m}_1(p)$$
3. *Finally, compute $\mathbf{m}'$, such that $\forall p \in O : \mathbf{m}'(p) = \mathbf{m}_2(p) + O(p)$.* ■

The following example should clarify these notions.

Example 2.12 *Figure 2.3 presents a transition $t = \langle I, O, s, d, +\infty \rangle$ equipped with a transfer arc. I and O are such that : $I(p_1) = I(s) = 1$, $I(p_2) = I(d) = 0$, $O(p_2) = 1$ and $O(p_1) = O(s) = O(d) = 0$.*

The successive steps to compute the effect of the firing of t are shown. Namely, (a) presents a marking $\mathbf{m}$ before the firing of t; (b) presents the marking $\mathbf{m}_1$ obtained by removing $I(p)$ tokens in every place p; (c) presents $\mathbf{m}_2$ obtained from $\mathbf{m}_1$ by transferring to d the two tokens present in s; and (d) presents the resulting marking $\mathbf{m}'$ obtained after producing $O(p)$ tokens in every place p.

If t had been equipped with a non-blocking arc (hence $t = \langle I, O, s, d, 1 \rangle$), only one token would have been transferred from s to d at step (c). In both cases, t would have been firable, even if $\mathbf{m}_1(s)$ had been 0. ◊

A direct consequence of Definition 2.23, is that EPN are (strongly) monotonic, as sated by the following lemma (taken from [RVB04]):

Lemma 2.14 ([RVB04]) *Let $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}'_1$ be three markings of an EPN, such that $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$ and $\mathbf{m}_1 \xrightarrow{t} \mathbf{m}'_1$ for some transition t of the EPN. Then, there exists $\mathbf{m}'_2$ such that $\mathbf{m}_2 \xrightarrow{t} \mathbf{m}'_2$ and $\mathbf{m}'_1 \preccurlyeq \mathbf{m}'_2$.*

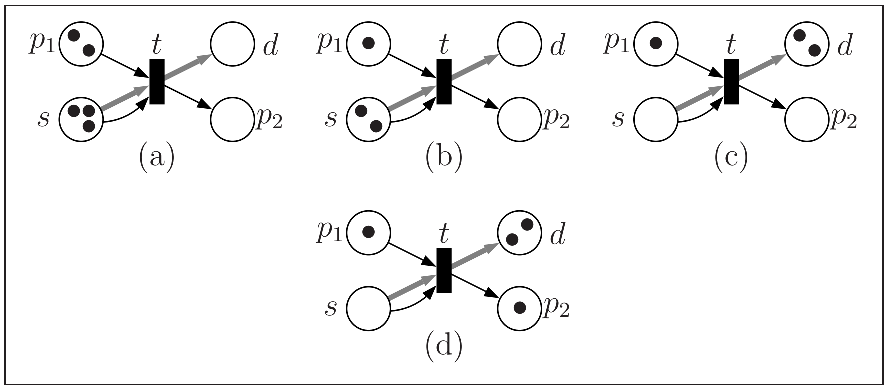


Figure 2.3: The four steps to compute the effect of a transfer arc

EPN are WSTS As in the case of strongly monotonic SMPN, EPN define transitions systems that are WSTS when we consider the ordering $\preccurlyeq$, as stated by the following proposition:

Proposition 2.3 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ be an* EPN *and let $\mathcal{S}_\mathcal{C} = \langle S, s_0, \Rightarrow \rangle$ be the transition system that corresponds to $\mathcal{N}$. Then, $\langle S, s_0, \Rightarrow, \preccurlyeq \rangle$ is a* WSTS.

Proof. By Lemma 2.14, $\Rightarrow$ is $\preccurlyeq$-monotonic. Moreover, $\preccurlyeq$ is a WQO by Proposition 2.1. Hence, $\langle S, s_0, \Rightarrow, \preccurlyeq \rangle$ is a WSTS by Definition 2.16. □

Comparison of SMPN and EPN Let us close this introduction on EPN by discussing briefly the relationship between the SMPN and the PN, PN+NBA, PN+T and PN+R. We first consider the PN, PN+T and PN+R:

1. Let $\mathcal{N} = \langle P, \{t_1, \ldots, t_n\}, \mathbf{m}_0 \rangle$ be a PN, and let us show how to build a corresponding SMPN $\mathcal{N}' = \langle P, T', D^-, D^+, \mathbf{m}_0 \rangle$. For any transition $t_i = \langle I, O, \bot, \bot, 0 \rangle \in T$, we create one and only one transition t'_i in T' where, for any $1 \leq j \leq |P|$, $D^-_{ij} = I(p_j)$ and $D^+_{ij} = O(p_j)$.

2. Let $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ be a PN+T that respects the following condition[6]:

$$\forall t = \langle I, O, s, d, b \rangle \in T_e \quad : \quad O(d) \geq I(s) \tag{2.3}$$

Let us build a corresponding SMPN $\mathcal{N}' = \langle P, T', D^-, D^+, \mathbf{m}_0 \rangle$. For any transition $t_i \in T_r$, we create in T' one and only one transition t'_i thanks to the same method

[6]In several works such as [Cia94], Petri nets with transfer arcs have a slightly different definition that ensures that for any $t = \langle I, O, s, d, b \rangle \in T_e$: $O(d) = I(s) = 0$. Remark that this condition implies (2.3)

as in the case of PN. For any transition $t_i = \langle I, O, s, d, +\infty\rangle$ in T_e we create one and only one transition t'_i in T' where:

$$\forall 1 \leq j \leq |P| : D^-_{ij}(\mathbf{m}) = \begin{cases} I(p_j) & \text{if } p_j \neq s \\ \mathbf{m}(p_j) & \text{if } p_j = s \end{cases}$$

and:

$$\forall 1 \leq j \leq |P| : D^+_{ij}(\mathbf{m}) = \begin{cases} O(p_j) & \text{if } p_j \neq d \\ \mathbf{m}(s) - I(s) + O(d) & \text{if } p_j = d \end{cases}$$

Remark that the definition of D^+ follows from the semantics of EPN transitions: the transfer of the tokens in p_j occurs *after* the $I(p_j)$ tokens have been removed from p_j.

Remark further that if we consider a PN+T $\mathcal{N}$ with a transition $t_i = \langle I, O, s, d, b\rangle$ s.t. $I(s) > O(d)$, we have (assuming that $d = p_j$): $D^+_{ij}(\mathbf{m}) = \mathbf{m}(s) + \alpha$ with $\alpha < 0$. Thus, $\alpha \notin \mathbb{N}$, which does not fit into the definition of SMPN (see Definition 2.17).

3. By definition, any PN+R is a PN+T, hence, any PN+R that respects (2.3) also admits a corresponding SMPN.

Let $\mathcal{N}$ be a PN, a PN+T that respects (2.3) or a PN+R that respects (2.3), and let $\mathcal{N}'$ be its corresponding SMPN (built as described above). Let σ be a sequence of transitions of $\mathcal{N}$, and let σ' be the sequence of transitions of $\mathcal{N}'$ obtained by replacing, in σ, each transition t_i by its corresponding t'_i. Then, for any marking $\mathbf{m}$, it is not difficult to see that $\mathbf{m} \xrightarrow{\sigma}$ iff $\mathbf{m} \xrightarrow{\sigma'}$ and that $\mathbf{m} \xrightarrow{\sigma} \mathbf{m}'$ iff $\mathbf{m} \xrightarrow{\sigma'} \mathbf{m}'$. This implies, e.g. that $\mathsf{Reach}(\mathcal{N}) = \mathsf{Reach}(\mathcal{N}')$.

Finally, remark that, by Lemma 2.13 the SMPN corresponding to a PN, a PN+T or a PN+R is a strongly monotonic SMPN (when it exists). This is a further motivation for the study of this peculiar class.

Unfortunately, the class of PN+NBA does not form a syntactic sub-class of SMPN. This is due to the fact that the effect of a non-blocking arc cannot be expressed by a function of the form $\alpha + \sum_i \beta_i \cdot \mathbf{m}(p_i)$. For instance, if we consider the transition $t_i = \langle I, O, p_j, p_k, 1\rangle$, then D^-_{ij} should be of the form $I(p_j) + \min\{\mathbf{m}(p_j), 1\}$. Clearly, the min function is problematic here.

Remark that one could think to build an SMPN in which two transitions t_1 and t_2 correspond to a single extended transition $t = \langle I, O, p_j, p_k, 1\rangle$ of a PN+NBA, where t_1 simulates the effect of t when the non-blocking arc has no effect ($D^-_{1j} = I(p_j)$ and $D^+_{1k} = O(p_k)$), and t_2 when the non-blocking arc has an effect ($D^-_{2j} = I(p_j) + 1$ and $D^+_{2k} = O(p_k) + 1$). However, that solution is not completely satisfactory because nothing prevents t_1 from firing in a marking $\mathbf{m}$ with $\mathbf{m}(p_j) > I(p_j)$. In this case, it is

t_2 that should fire, because the non-blocking arc of the PN+NBA would have an effect in such a marking $\mathbf{m}$.

Nevertheless, we will see in the sequel (see Section 5.4.3) that, given a PN+NBA, one can build an SMPN that has some properties in common with the original PN+NBA (actually, the same coverability properties). These properties will be enough for the applications of PN+NBA we are interested in. The same holds for the PN+T and the PN+R that do not enforce (2.3).

2.3.4 Lossy Channel Systems

Let us now turn our attention to the Lossy Channel Systems (or LCS for short). The study of this model from the 'computer-aided verification' point of view is due mainly to Abdulla and Bouajjani [AAB99, ABJ98]. Intuitively, an LCS is a finite set of finite automata that *communicate* through a finite number of unbounded FIFO channels. These channels have the characteristic to be *lossy*, that is, they can spontaneously lose messages at any time (however the FIFO property is always preserved). In the case of LCS, *communicating* through FIFO channels means that each transition of one of the automata is labelled by a function that assigns one operation to each channel. This operation tells how the channel has to be modified when the transition occurs. The possible operations are (for a character a taken from some alphabet Σ):

- ?a, which means *consume a letter* a *in the channel*;
- !a, which means *produce a letter* a *in the channel*;
- nop which means *do not modify the channel*.

The following definition states this more formally:

Definition 2.24 ([ABJ98]) (Lossy Channel System) *A* Lossy Channel System*,* LCS *for short, is a tuple $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ where*

- *$Q = \{q_1, \ldots, q_n\}$ is a finite set of locations;*
- *$q_0 \in Q$ is an initial location;*
- *F is a finite set of channels;*
- *Σ is a finite alphabet;*
- *$T \subseteq Q \times Op \times Q$ is a transition relation where*

$$Op : F \mapsto \bigcup_{a\in\Sigma}\{?a, !a\} \cup \{\mathsf{nop}\}$$

■

Dynamics of LCS An LCS defines a transition system with infinitely many states, since the channels are not bounded. A state of an LCS is made up of a tuple that indicates the current location and the content of each channel, as stated by the next definition.

Definition 2.25 ([ABJ98]) (State and set of states of an LCS) *Given an* LCS $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$*, a* state *of* $\mathcal{C}$ *is a pair* $\langle q, W\rangle$ *where* $q \in Q$ *and* $W : F \mapsto \Sigma^*$ *assigns a content to each channel. Moreover,* $\mathsf{States}\left(\mathcal{C}\right) = \{\langle q, W\rangle \mid q \in Q \wedge W : F \mapsto \Sigma^*\}$ *denotes the set of states of* $\mathcal{C}$*.* ■

For any word $w \in \Sigma^*$, let $\mathsf{Subword}\left(w\right)$ denotes the set of sub-words of w. That is, if $w = a_1 a_2 \cdots a_n$ (for $n \geq 1$), then $\mathsf{Subword}\left(w\right)$ is:

$$\left\{ a_{\rho(1)}a_{\rho(2)}\cdots a_{\rho(k)} \;\middle|\; \begin{array}{l} 1 \leq k \leq n \wedge \\ \rho : \{1,\dots k\} \mapsto \{1,\dots n\} \wedge \\ \forall 1 \leq i < k : \rho(i) < \rho(i+1) \end{array} \right\} \cup \{\varepsilon\}$$

Moreover, $\mathsf{Subword}\left(\varepsilon\right) = \{\varepsilon\}$. We extend $\mathsf{Subword}$ to functions $W : F \mapsto \Sigma$. In that case, $\mathsf{Subword}\left(W\right)$ denotes the set of functions $W' : F \mapsto \Sigma^*$ such that $\forall f \in F : W'(f) \in \mathsf{Subword}\left(W(f)\right)$.

Equipped with this notion, we can define the transition relation of a LCS. As in the case of SMPN or EPN, we first identify in which states transitions are *firable*. Intuitively, the only restriction on the firing of a transition is in the case where an operation of the form $?\mathtt{a}$ has to be applied on a channel f. Obviously, in this case, f must contain at least one $\mathtt{a}$. Remark that the character $\mathtt{a}$ does not need to be at the first position in the channel. For instance, consider the channel whose content[7] is $\mathtt{cbad}$. It is actually possible to read an $\mathtt{a}$ from that channel under the condition that the $\mathtt{c}$ and $\mathtt{b}$ characters are lost (which can always occur).

Then, we explain how a firable transition can *occur* and hence modify the current state of the LCS. As far as the channels are concerned, that effect is computed in three steps, which correspond respectively to (i) losing some characters, (ii) the actual effect of the transition (with the obvious semantics) and (iii) losing some characters again. The following definition states this in a formal way:

Definition 2.26 ([ABJ98]) (Firable transition of LCS and effects) *Given an* LCS $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$*, a transition* $t = \langle q_1, Op, q_2\rangle \in T_i$ *is* firable *from state* $\langle q, W\rangle$ *if* $q_1 = q$ *and for all* $f \in F : Op(f) = ?a$ *implies that* $a \in \mathsf{Subword}\left(W(f)\right)$*.*

Firing t from $\langle q, W\rangle$ *leads non-deterministically to any state* $\langle q_2, W'\rangle$ *s.t. there are* $\overline{W} \in \mathsf{Subword}\left(W\right)$*,* $\overline{W'}$ *with* $W' \in \mathsf{Subword}\left(\overline{W'}\right)$ *and, for any* $f \in F$ *:*

- $Op(f) = ?a$ *implies that* $\overline{W}(f) = a \cdot \overline{W'}(f)$ *and*

[7]We assume that the rightmost character is the last one that has been written to the channel.

- *$Op(f) = !a$ implies that $\overline{W'}(f) = \overline{W}(f) \cdot a$ and*
- *$Op(f) = \mathsf{nop}$ implies that $\overline{W}(f) = \overline{W'}(f)$*

Such a transition is noted $\langle q, W\rangle \xrightarrow{t} \langle q_2, W'\rangle$. ■

A consequence of Definition 2.26 is that any LCS $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ defines a transition system $\mathcal{S}_{\mathcal{C}} = \langle \mathsf{States}\,(\mathcal{C})\,, s_0, \Rightarrow\rangle$ where:

- $s_0 = \langle q_0, W_i\rangle$ with $W_i(f) = \varepsilon$ for all $f \in F$;
- for any $s_1, s_2 \in \mathsf{States}\,(\mathcal{C}) : s_1 \Rightarrow s_2$ if and only if $\exists t \in T$ with $s_1 \xrightarrow{t} s_2$.

In the sequel we often confuse an LCS $\mathcal{C}$ with its corresponding transition system $\mathcal{S}_{\mathcal{C}}$, and write, for instance $\mathsf{Reach}\,(\mathcal{C})$ instead of $\mathsf{Reach}\,(\mathcal{S}_{\mathcal{C}})$.

Remark 2.2 *In the general setting of [ABJ98], an* LCS *is made up of* several *finite automata. In our case, we have restricted ourselves to the case where there is only one finite automaton. As a matter of fact, this is not a real restriction. In the definition of [ABJ98], an* LCS *is of the form*

$$\mathcal{C}' = \left\langle \{Q^1, \dots Q^n\}, \{q_i^1, \dots, q_i^n\}\right\rangle, F, \Sigma, \{T^1, \dots, T^n\}$$

where each triple $\left\langle Q^j, q_i^j, T^j\right\rangle$ corresponds to a single automaton. The set of states of such a system is

$$\left\{\left\langle (q_1, \dots, q_n), W\right\rangle \mid W : F \mapsto \Sigma^* \bigwedge \wedge_{1 \leq i \leq n} q_i \in Q_i\right\}$$

The semantics the system is an interleaving semantics (one automaton can progress at a time). Then, it is not difficult to see that the LCS $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ *where:*

- *$Q = Q^1 \times Q^2 \times \cdots \times Q^n$;*
- *$q_0 = (q_0^1, q_0^2, \dots, q_0^n)$;*
- $T = \left\{ \left((q_1, q_2, \dots, q_n), (q_1', q_2', \dots, q_n')\right) \middle| \exists 1 \leq i \leq n : \exists Op : \begin{array}{c} (q_i, Op, q_i') \in T_i \\ \textit{and} \\ \bigwedge_{j \neq i} q_j = q_j' \end{array} \right\}$

is equivalent to $\mathcal{C}'$ in the sense that any sequence of transitions $t_1 t_2 \dots t_k$ firable in $\mathcal{C}$ is firable in $\mathcal{C}'$ and leads to the same states (and vice-versa). This implies in particular that $\mathsf{Reach}\,(\mathcal{C}) = \mathsf{Reach}\,(\mathcal{C}')$.

Moreover, this remark still holds if we add synchronisation labels *on the transitions of the automata that compose $\mathcal{C}'$. In that case, several automata can progress at the* *same time, according to the synchronisation labels. One just has to adapt the definition of the transition relation in order to take the labels into account.*

Thus our choice to keep a single automaton is not problematic as far as reachability properties are concerned. On the other hand, this restriction makes the presentation easier.

Before introducing the necessary notions to define WSTS from LCS, and in order to clarify the previous definitions, we introduce an example of LCS that should also serve as a practical motivation of this model.

Example 2.13 *The example we are about to discuss is a model of the* alternating bit protocol *[ABJ98]. In this communication protocol, two entities, called the* sender *and the* receiver *exchange messages through two unreliable communication channels A and M. The sender sends, to the receiver, messages that are numbered alternatively by 0 and 1 on the channel M. Each time the receiver receives a message numbered by i $(i = 0, 1)$, it sends to the receiver an* acknowledgement *labelled by i through channel A. The purpose of the numbering of messages and acknowledgements is to allow both entities to detect losses of messages, and proceed to the appropriate retransmissions if necessary. For instance the sender first sends a message numbered by 0. The receiver waits for such a message and thus discards any message numbered by 1. When the receiver has received a message numbered by 0, it sends an acknowledgement numbered by 0. The receiver acts symmetrically, that is, once it has sent a message numbered by 0, it waits for an acknowledgement numbered by 0 and, up to then, discards all the acknowledgements numbered by 1. Then the protocol continues with a message and an acknowledgement numbered by 1, then by 0 again and so forth (hence the* alternating bit *name).*

In order to model this protocol by an LCS*, we first devise two finite automata that correspond respectively to the receiver and the sender, and whose transitions are labelled by operations (in the* LCS *sense) on the two channels A and M. These automata are presented in Figure 2.4. Remark that in this model, we have abstracted away the actual content of the messages. We only distinguish between two types of messages: those labelled by 0 (denoted by character* `m0`*) and those labelled by 1 (denoted by* `m1`*). The messages* `a0` *and* `a1` *are of course the respective acknowledgements.*

Following Remark 2.2, there is an LCS $\mathcal{C}_{\mathsf{abp}} = \langle Q, q_0, F, \Sigma, T\rangle$ *whose set of states is* $Q = \{q^s_0, q^s_1, q^s_2, q^s_3\} \times \{q^r_0, q^r_1, q^r_2, q^r_3\}$, *initial state is* $q_0 = (q^s_0, q^r_0)$, *set of channels is* $F = \{A, M\}$, *alphabet is* $\Sigma = \{\mathtt{m0}, \mathtt{m1}, \mathtt{a0}, \mathtt{a1}\}$ *and transition relation T is s.t.* $\big((q^s_i, q^r_j), Op, (q^s_k, q^r_\ell)\big) \in T$ *iff (i) either $j = \ell$ and there is a transition in the sender that goes from q^s_i to q^s_k and is labelled*[8] *by the operations given by Op, or (ii) $i = k$ and there is a transition of the receiver that goes from q^r_j to q^r_ℓ and is labelled by the operations given by Op.*

[8]The `nop` operations have been omitted in the figure.

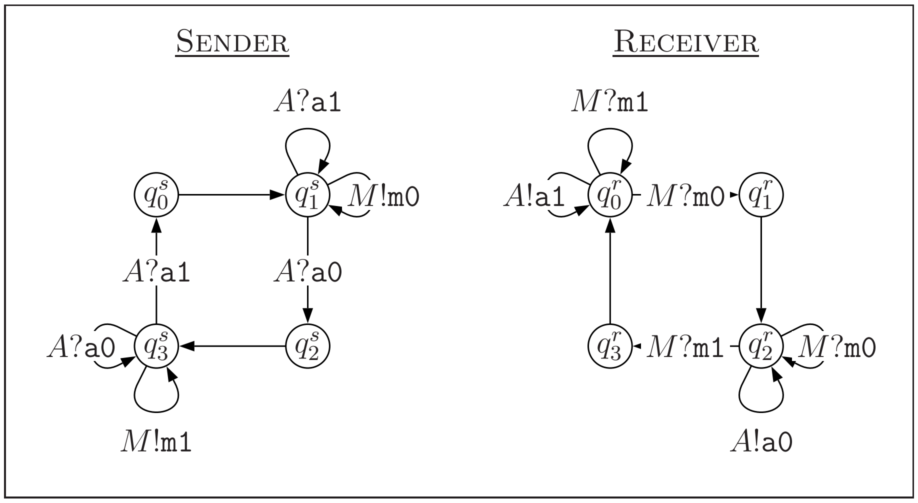


Figure 2.4: The two automata that make up the LCS model of the alternating bit protocol.

Although quite coarse, this model of the alternating bit protocol still allows to check for some behavioural properties. For instance, when a message is sent *by the sender, the following message should not be sent before the first one has been actually* received *by the receiver (because that would mean that the acknowledgement mechanism didn't work properly). Let us label*[9] *some of the transition of the two automata of Figure 2.4 in order to identify when these operations occur. We assume that the transitions from q_0^s to q_1^s and that from q_2^s to q_3^s correspond to a send, and label them by* snd*. Symmetrically, we identify the transitions from q_1^r to q_2^r and from q_3^r to q_4^r with the receive operation, and label them by* rcv*. Then it is not difficult to see that we can check the property that we have stated informally above by synchronising, on the labels* snd *and* rcv*, the two automata of Figure 2.4 with the observer of Figure 2.5. Then we can check that the resulting* LCS *can never reach a state that corresponds to the observer being in location* Err*.* ◊

A WQO for LCS States of LCS can be compared by an ordering $\precsim$ that we define now. Roughly speaking $s_1 \precsim s_2$ iff s_1 and s_2 correspond to the same location of the automaton and the contents of the channels in s_1 are subwords of the contents of the same channels in s_2. The following definition states this more formally.

[9]These labels are completely separated from the labelling by channel operations and will serve for synchronisation purpose with an observer.

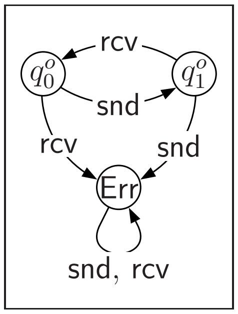


Figure 2.5: An observer for the LCS of Figure 2.4

Definition 2.27 ([ABJ98]) (The WQO $\precsim$) *Let $\mathcal{C}$ be an* LCS*. Then, the ordering $\precsim \subseteq$ States$(\mathcal{C}) \times$ States$(\mathcal{C})$ is s.t. for any pair of states $s = \langle q, W\rangle$ and $s' = \langle q', W'\rangle$:*

$$s \precsim s' \text{ iff } q = q' \text{ and for any } f \in F : W(f) \in \mathsf{Subword}(W'(f))$$

■

It is well-known that $\precsim$ is a WQO:

Proposition 2.4 ([ABJ98]) *$\precsim$ is a* WQO*.*

LCS are WSTS Here again, the main motivation for the study of the LCS model is that they naturally define transition systems which are WSTS when considering $\precsim$ as WQO as stated by the following proposition:

Proposition 2.5 ([ABJ98]) *Let $\mathcal{C}$ be an* LCS *and let $\mathcal{S}_{\mathcal{C}} = \langle S, s_0, \Rightarrow\rangle$ be the transition system that corresponds to $\mathcal{C}$. Then, $\langle S, s_0, \Rightarrow, \precsim\rangle$ is a* WSTS*.*

Because of the lossiness of LCS, this WSTS enjoys the peculiar property that if a configuration c is reachable in the WSTS, then, any configuration $c' \precsim c$ is reachable too:

Lemma 2.15 *Let $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ be an* LCS *and let $\mathcal{S}_{\mathcal{C}} = \langle S, s_0, \Rightarrow, \precsim\rangle$ be the* WSTS *that corresponds to $\mathcal{C}$. Then, for every $c \in$ Reach$(\mathcal{S}_{\mathcal{C}})$, for every $c' \precsim c$: $c' \in$ Reach$(\mathcal{S}_{\mathcal{C}})$.*

Proof. Follows directly from Definition 2.26 and Definition 2.27. □

2.3.5 Other classes of WSTS

Other classes of WSTS have been introduced and studied in the literature. This section briefly recalls two of them. A more complete list of WSTS that have been identified in the literature can be found in [FS01].

Broadcast protocols In this section, we recall the model of *broadcast protocols* (BP for short), which have been introduced in [EN98]. A BP describes a system made up of an unbounded number of processes which evolve according to one or several finite automata. Each state of the automaton corresponds to a state of the process, and the transitions are of three different types:

Unguarded transition Such transitions are simply labelled by an action a and can be taken at any time by any process whose current state is the originating state of the transition.

Rendez-vous These transitions describe a one-to-one blocking synchronisation. The synchronisation is made on the same action a. One of the processes is the *sender*: it fires a transition labelled by $a!$. The other process is the *receiver*: it fires a transition labelled by $a?$.

Broadcast These transitions describe a one-to-many synchronisation, on a common action a. The *sender* process fires a transition labelled by $a!!$. At that point, all the other processes fire the $a??$–labelled transition that originates in their current state (it is assumed that such a transition always exists, i.e., any process is always willing to receive a broadcast, possibly for doing nothing).

The formal definition (taken from [EFM99]) is as follows:

Definition 2.28 ([EFM99]) (Broadcast Protocol) *A* broadcast protocol *(or* BP*, for short) is a triple $\mathcal{B} = \langle S, L, R\rangle$ where:*

- *S is a finite set of states;*
- *L is a finite set of labels that is partitioned into:*
  - *A set Σ_ℓ of local labels*
  - *A set $\Sigma_r \times \{?\} \cup \Sigma_r \times \{!\}$ of input and output Rendez-vous*
  - *A set $\Sigma_b \times \{??\} \cup \Sigma_b \times \{!!\}$ of input and output broadcasts.*
- *$R \subseteq S \times L \times S$ is a set of transitions that satisfies: $\forall a \in \Sigma_b : \forall s \in S : \exists s' \in S : (s, a??, s') \in R$.* ■

In the sequel, we denote by Σ the set $\Sigma_\ell \cup \Sigma_b \cup \Sigma_r$, and assume that $\Sigma \cap \{!, !!, ?, ??\} = \emptyset$. We note $s \xrightarrow{x} s'$ to mean that $(s, x, s') \in R$, for any states $s, s' \in S$ and any label $x \in L$. In the sequel, we restrict ourselves to the broadcast protocols that satisfy the three additional constraints:

1. For any $s \in S$, for any $a \in \Sigma_b$, there is exactly one state $s' \in S$ s.t. $s \xrightarrow{a??} s'$
2. For any $a \in \Sigma_b$, there is one and only one $(s, a!!, s') \in R$.

3. For any $a \in \Sigma_r$, there is one and only one $(s, a!, s') \in R$ and one and only one $(s, a?, s') \in R$.

These constraints are actually not restrictive. Given a BP $\mathcal{B}$, one can always translate it into a BP $\mathcal{B}'$ that has the same coverability properties and enforces the three points above. See [EFM99] for the details.

A BP naturally defines a WSTS which corresponds to the *counting abstraction* of the BP: each configuration of the corresponding WSTS is a tuple of natural numbers that counts how many processes are in each state of the BP, and the transitions modify the states according to the semantics of the BP that we have sketched above.

More precisely, let $\mathcal{B} = \langle S, L, R\rangle$ be a BP with $S = \{s_0, \ldots, s_n\}$. In order to represent a possibly unbounded number of processes, the configurations of the WSTS $\mathcal{S}_{\mathcal{B}} = \langle C, c_0, \Rightarrow, \preccurlyeq_b\rangle$ associated to $\mathcal{B}$ are tuples of $\mathbb{N} \cup \{\omega\}$, where ω represents 'any number of processes'. The operations + and - and the ordering $\leq$ on natural numbers are extended to handle ω as follows: $\omega + \omega = \omega$; $\omega - \omega = 0$ and for any $c \in \mathbb{N}$: $\omega + c = \omega - c = \omega$; $c \leq \omega$; $\omega \not\leq c$.

Let $\mathbf{v}_0 \in (\mathbb{N} \cup \{\omega\})^n$ be a tuple that describes the initial configuration of the system: for any $1 \leq i \leq n$, there are initially $\mathbf{v}_0(i)$ processes in state s_i. Let us denote, for any $1 \leq i \leq n$, by $\mathbf{u}_i$ the vector of natural numbers s.t. $\mathbf{u}_i(i) = 1$ and for any $1 \leq j \leq n$: $j \neq i$ implies $\mathbf{u}_i(j) = 0$. The WSTS $\mathcal{S}_{\mathcal{B}} = \langle C, c_0, \Rightarrow, \preccurlyeq_b\rangle$ is defined as follows:

1. $C = (\mathbb{N} \cup \{\omega\})^n$;

2. $c_0 = \mathbf{v}_0$;

3. $\Rightarrow$ is s.t. $(c, c') \in\Rightarrow$ iff one of the following holds:

   - there is $1 \leq i \leq n$, $a \in \Sigma_\ell$ and $s_j \in S$: s.t. $c(i) \geq 1$, $s_i \xrightarrow{a} s_j$, $c' = c - \mathbf{u}_i + \mathbf{u}_j$.
   - there are $1 \leq i, j \leq n$, $a \in \Sigma_r$ and $s_k, s_\ell \in S$: s.t. $i \neq j$ implies $(c(i) \geq 1$ and $c(j) \geq 1)$, $i = j$ implies $c(i) \geq 2$, $s_i \xrightarrow{!a} s_k$, $s_j \xrightarrow{?a} s_\ell$ and $c' = c - \mathbf{u}_i - \mathbf{u}_j + \mathbf{u}_k + \mathbf{u}_\ell$.
   - there is $1 \leq i \leq n$, $a \in \Sigma_b$ and $s_j \in S$ s.t. $c(i) \geq 1$, $s_i \xrightarrow{!!a} s_j$ and $c' = c_3$, where c_3 is computed as follows:
     (a) $c_1 = c - \mathbf{u}_i$
     (b) for any $1 \leq k \leq n$, $c_2(k) = \sum_{\ell \in pred(k,a)} c_1(\ell)$, where $pred(k, a)$ is the set $\{\ell \mid s_\ell \xrightarrow{??a} s_k\}$
     (c) $c_3 = c_2 + \mathbf{u}_j$.

4. for any $c_1, c_2 \in C$: $c_1 \preccurlyeq_b c_2$ iff for any $1 \leq i \leq n$: $c_1(i) \leq c_2(i)$.

The proof that $\mathcal{S}_{\mathcal{B}}$ is indeed a WSTS can be found, for instance, in [EFM99].

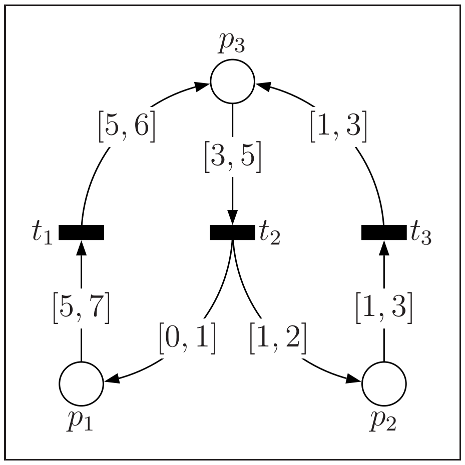


Figure 2.6: An example of TPN taken from [ADMN04].

Timed Petri nets Let us finally mention the model of timed Petri nets (TPN for short) as a class of WSTS that has been well covered by the literature [AN01, AN02, ADMN04]. A TPN has a structure similar to a PN: it is a bipartite graph whose nodes are called places and transitions, and places can be marked by (an unbounded number of) tokens. Unlike PN, each token has an age, which is a positive real number [10]. The in– and out–edges of the transitions bear intervals of the form $\langle a, b\rangle$ where $a \in \mathbb{N}$, $b \in \mathbb{N} \cup \{\infty\}$, $\langle$ is either '(' or '[' and $\rangle$ is either ')' or ']' in the case where $b \neq \infty$. As usual, (and) mean that the interval is open on the corresponding bound; [and] mean that it is closed. A transition t is enabled if, for any input place p, there is a token whose age is inside the interval labelling the arc (p, t). In that case, t can fire, consume tokens with the right ages in the input places and produce in each output place p' a fresh token whose age is non-deterministically chosen in the interval that labels the arc (t, p'). Remark thus that a TPN induces two dimensions of infinity: the unbounded number of tokens, and the unbounded value of the ages of the tokens.

In [ADMN04], the authors define a *region construction* (similar to that for Timed automata [AD94]). The (potentially infinite) transition system whose configurations are the regions and whose transitions correspond to the semantics of the TPN, lifted to the regions, is shown to be a WSTS (for a WQO defined in the paper).

Since we do not deal with TPN in this thesis, we do not describe it with further details. The interested reader is referred to the aforementioned publications for more details.

[10]Timed Petri nets should not be confused with Time Petri nets [Mer74, PZ91] where the ages are assigned to the transitions. Remark that other notions of Timed Petri nets exist [Ram74]

2.4 Language Theory

The second part of this thesis deals with the expressiveness of well-structured transition systems. This section briefly recalls relevant notions of language theory. Most of them are classical and are present in every introductory book on language theory, such as [Sal73], [HMU01] or [Gin75].

2.4.1 Words and languages

The very first notion is that of word. We distinguish two kinds of words: the (finite) words and the infinite words (or ω-words).

Definition 2.29 (Words) *Given a finite set Σ called an* alphabet *and whose elements are called* letters *or* characters*, a* (finite) word *w on Σ is either a finite sequence $a_1a_2\dots a_n$ of letters in Σ or the empty word ε. An* ω-word *on Σ is an infinite sequence $a_1a_2\cdots a_j\cdots$ of letters in Σ.*

We denote by $|w|$ the length of w:

$$|w| = \begin{cases} 0 & \text{if } w = \varepsilon \\ +\infty & \text{if } w \text{ is and } \omega\text{-word} \\ n & \text{if } w = a_1a_2\cdots a_n \end{cases}$$

■

An ω-word is sometimes called a word when the context makes it clear that it is infinite. From this definition follows directly the notion of *language*:

Definition 2.30 (Languages) *Given an alphabet Σ, a (finite words)* language *L on Σ is a (possibly infinite) set of finite words on Σ.*

*An ω-*language *L on Σ is a (possibly infinite) set of ω–words on Σ.* ■

When the context permits it, we sometime refer to ω-languages simply as languages. We adopt the widespread notation Σ^* to denote the language containing any finite word on the alphabet Σ.

2.4.2 Operations on words and languages

Let us begin with the operation of *concatenation* of two words. It is defined as follows:

Definition 2.31 (Word concatenation) *Let w_1 be a (finite) word on Σ and w_2 be a word or an ω-word on Σ. Then, $w_1 \cdot w_2$, called the* concatenation of w_1 and w_2*, is* *defined as follows:*

$$w_1 \cdot w_2 = \begin{cases} w_1 w_2 & \textit{if } w_1 \neq \varepsilon \textit{ and } w_2 \neq \varepsilon \\ w_1 & \textit{if } w_1 \neq \varepsilon \textit{ and } w_2 = \varepsilon \\ w_2 & \textit{if } w_1 = \varepsilon \textit{ and } w_2 \neq \varepsilon \\ \varepsilon & \textit{if } w_1 = w_2 = \varepsilon \end{cases}$$

■

Let us now introduce several operations on finite words and finite words languages:

1. Let $w = w_1 w_2 \cdot w_n$ be a finite word. Then, the *mirror of* w, noted w^R, is the word $w_n w_{n-1} \cdots w_1$.

2. If L_1 and L_2 are two finite words languages on Σ, then $L_1 \cdot L_2$ denotes the concatenation of L_1 and L_2 and is such that

$$L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$$

3. Another common operation on a finite words language L consists to apply an *homomorphism* which can be seen as a function that substitutes some letters for others in every word of L. Given a finite alphabet Σ, a *homomorphism* is a function $h : \Sigma^* \mapsto \Sigma^*$ s.t. $\forall w_1, w_2 \in \Sigma^* : h(w_1 \cdot w_2) = h(w_1) \cdot h(w_2)$. The inverse of h is the function $h^{-1} : \Sigma^* \mapsto 2^{\Sigma^*}$ such that $h^{-1}(w) = \{w' \mid h(w') = w\}$. If L is a language on Σ, then $h(L) = \{h(w) \mid w \in L\}$ and $h^{-1}(L) = \cup_{w \in L} h^{-1}(w)$.

   Remark that, by this definition, any homomorphism h must satisfy $h(\varepsilon) = \varepsilon$. Indeed, let $w \neq \varepsilon$ be a word. We have $w = w \cdot \varepsilon$. Hence, $h(w) = h(w \cdot \varepsilon) = h(w) \cdot h(\varepsilon)$, which implies that $h(\varepsilon) = \varepsilon$. Moreover, still by this definition, any homomorphism h can be completely characterised by a function $f_h : \Sigma \mapsto \Sigma^*$ s.t. for any $a \in \Sigma$, $f_h(a) = h(a)$. Indeed, let $w = a_1 a_2 \cdots a_n$ be a non-empty word (with $a_1, a_2, \ldots, a_n \in \Sigma$). Then, $h(w) = h(a_1 \cdot a_2 \cdots a_n) = h(a_1) \cdot h(a_2) \cdots h(a_n) = f_h(a_1) \cdot f_h(a_2) \cdots f_h(a_n)$. In the sequel, we sometimes confuse the homomorphism h with its characteristic function f_h.

4. The iteration of a finite words language L is the language $L^+ = \{w_1 \cdot \ldots \cdot w_n \mid n \geq 1 \wedge \forall 1 \leq i \leq n : w_i \in L\}$. Similarly, L^* (the Kleene closure of L) is the language $L^+ \cup \{\varepsilon\}$.

5. The complement of a finite words language L on the alphabet Σ is the (finite words) language $\Sigma^* \setminus L$.

2.4.3 Regular and Context-free languages

Several classes of languages of interest have been identified in the literature (see [HMU01]). Let us recall two of them, that will be relevant in the sequel. The former is that of *regular languages*. It is well-known that this class corresponds the class of languages recognised by *finite state automata* or *regular expressions* [HMU01].

Definition 2.32 (Regular Language) *Let Σ be a finite alphabet. Then, the set of all the regular languages on Σ is defined inductively as follows:*

- *$\emptyset$ is a regular language ;*
- *$\{\varepsilon\}$ is a regular language ;*
- *for any $a \in \Sigma$, $\{a\}$ is a regular language ;*
- *if L_1 and L_2 are two regular languages, then, $L_1 \cdot L_2$, $L_1 \cup L_2$ and L_1^* are all regular languages.* ■

The former class is that of *context-free languages*. A language is *context-free* iff it is recognised by some context-free grammar:

Definition 2.33 (Context-Free Grammar) *Let Σ be a finite alphabet. A* context-free grammar *on Σ is a tuple $\langle V, P, S, \Sigma\rangle$, where:*

1. *V is a finite set of* variables *s.t. $V \cap \Sigma = \emptyset$;*
2. *$P \subseteq V \times (V \cup \Sigma)^*$ is a finite set of* production rules*. It is usual to denote the elements (T, w) of P by $T \rightarrow w$.*
3. *$S \in V$ is the* start symbol *of the grammar.* ■

To each (context-free) grammar G, we associate a *derivation rule* $\Rightarrow_G$, defined as follows:

Definition 2.34 (Derivation Rule of a Grammar) *Let $G = \langle V, P, S, \Sigma\rangle$ be a (context-free) grammar. Then, $\Rightarrow_G \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ is s.t. $w_1 \Rightarrow_G w_2$ iff:*

$$\exists T \in V : \exists v_1, v_2 \in (V \cup \Sigma)^* : \exists T \rightarrow w \in P : w_1 = v_1 T v_2 \wedge w_2 = v_1 w v_2$$

■

Thanks to this definition, we obtain the characterisation of context-free languages:

Definition 2.35 (Context-Free Language) *Let Σ be a finite alphabet. A language L on Σ is a* context-free language *(*CFL *for short) iff there exists a context-free grammar $G = \langle V, P, S, \Sigma\rangle$ s.t. $L = \{w \mid S \Rightarrow_G^* w\}$.* ■

It is well-known that the class of context-free languages also corresponds to the class of languages recognised by pushdown automata [HMU01].

2.4.4 Closure properties

Let us consider a set (family) S of languages. A natural question about S asks whether S is closed under the various operations that we have introduced above, as well as under classical set theory operations such as union or intersection. Certain families of languages, called *full abstract families of languages* (of full AFL for short) have been identified and vastly studied in the literature (in particular by Ginsburg, see [Gin75]). A set of languages is a full AFL iff it is closed under certain operations as stated by the following definition:

Definition 2.36 ([Gin75, Sal73]) (Abstract Family of Languages) *A full* abstract family of languages *(full* AFL *for short) is a set of languages closed under* (*i*) *union,* (*ii*) *concatenation,* (*iii*) *intersection with regular languages,* (*iv*) *iteration,* (*v*) *homomorphism and* (*vi*) *inverse homomorphism.* ■

When the family of languages considered is closed under intersection too, one speaks about a *full* AFL *closed under intersection.*

2.4.5 Property of context-free languages

We finish this section by recalling the well-known *pumping lemma* on *context-free languages*:

Lemma 2.16 ([HMU01]) *Let L be a context-free language on Σ. Then, there exists $n \in \mathbb{N}$ s.t. for any $z \in L$ with $|z| \geq n$, there are u, v, w, x, y in Σ^* s.t.:*

1. *$u \cdot v \cdot w \cdot x \cdot y = z$ and;*
2. *$|v \cdot w \cdot x| \leq n$ and;*
3. *$|v \cdot x| \geq 1$ and;*
4. *for any $i \geq 0$: $u \cdot v^i \cdot w \cdot x^i \cdot y \in L$.*

This closes our short survey of language theory. Let us now explain how we can use WSTS to define languages.

2.5 Expressiveness of WSTS

In many cases, a language L can be defined by means of some associated computational device that receives a word w in input and replies 'yes' or 'no' whether $w \in L$ or not. For instance, any regular language is defined by a finite state automaton; context-free languages correspond to pushdown automata; and so forth. Similarly, the expressive power of several models of computation, such as Petri nets, has been explored. In the case of Petri nets, this is done by associating to the Petri net a labelling function of the transitions, and by defining a notion of *accepted word*.

We follow the same principles in order to study the expressive power of WSTS. First of all, we explain how we *label* transitions of WSTS, and define the notion of *accepted language*. Additionally, we show how the syntax of EPN can be adapted in order to obtain *labelled* EPN, and a corresponding labelled WSTS.

2.5.1 Labelled WSTS

To define the notion of *accepted language* in the case of a WSTS, we have to label the transitions of the underlying transition system. Thus, we adapt the definitions of *transition system* and *well-structured transition system* as follows:

Definition 2.37 (Labelled transition system) *A* labelled transition system *is a tuple $\mathcal{S} = \langle C, c_0, \Sigma, \Rightarrow \rangle$ such that:*

1. *C is a possibly infinite set of configurations;*
2. *$c_0 \in C$ is the* initial configuration*;*
3. *Σ is a finite alphabet that does not contain ε;*
4. *$\Rightarrow \subseteq C \times \big(\Sigma \cup \{\varepsilon\}\big) \times C$ is the transition relation.* ■

As we did in the case of transition systems, we adopt several conventional notations. We write $c_1 \overset{a}{\Rightarrow} c_2$ instead of $(c_1, a, c_2) \in \Rightarrow$. When the character labelling the transition is not relevant, we might omit it and write $c_1 \Rightarrow c_2$ to mean that there exists $a \in \Sigma \cup \{\varepsilon\}$ s.t. $c_1 \overset{a}{\Rightarrow} c_2$.

For any finite word w, we also write $c \overset{w}{\Rightarrow}^* c'$ to mean that w can be accepted along a finite path from c to c', as stated by the following definition:

Definition 2.38 (Extension of $\Rightarrow$) *Let $\mathcal{S} = \langle C, c_0, \Sigma, \Rightarrow \rangle$ be a labelled transition system, let c and c' be two configurations from C, and let $w \in \Sigma^*$ be a finite word. Then, $c \overset{w}{\Rightarrow}^* c'$ iff:*

- *either $w = \varepsilon$ and $c = c'$;*
- *or $\exists n \geq 1$: $\exists c_1, \ldots, c_{n+1} \in C$: $\exists a_1, \ldots a_n \in \Sigma \cup \{\varepsilon\}$ s.t. $c_1 = c$, $c_{n+1} = c'$, $w = a_1 \cdot a_2 \cdots a_n$ and for any $1 \leq i \leq n$: $c_i \overset{a_i}{\Rightarrow} c_{i+1}$.* ■

Remark that, according to this definition, $c \overset{\varepsilon}{\Rightarrow}^* c'$ means either that $c = c'$ or that $c \overset{\varepsilon}{\Rightarrow} c'$ or that there are finitely many configurations d_1, d_2, …, d_k s.t. $c \overset{\varepsilon}{\Rightarrow} d_1 \overset{\varepsilon}{\Rightarrow} d_2 \overset{\varepsilon}{\Rightarrow} \cdots \overset{\varepsilon}{\Rightarrow} d_k \overset{\varepsilon}{\Rightarrow} c'$. Remark further that, for any pair of configurations c_1 and c_2, and any character a, $c_1 \overset{a}{\Rightarrow} c_2$ implies $c_1 \overset{a}{\Rightarrow}^* c_2$, but the reverse implication does not hold.

Finally, since we can associated to any labelled transition system $\langle C, c_0, \Sigma, \Rightarrow\rangle$ a transition system $\langle C, c_0, \{(c, c') \mid \exists a : (c, a, c') \in \Rightarrow\}\rangle$, all the notions such as Post, Pre, Reach, and so on, can be applied to a labelled transition system.

Labelled WSTS Let us now consider the definition of *labelled* WSTS:

Definition 2.39 (Labelled Well-Structured Transition System) *A* Labelled Well-Structured Transition System *is a tuple* $\mathcal{S} = \left\langle C, c_0, \Sigma, \Rightarrow, \overline{\leq}\right\rangle$ *s.t.*

- $\langle C, c_0, \Sigma, \Rightarrow\rangle$ *is a labelled transition system;*
- $\Rightarrow$ *is* $\overline{\leq}$*-monotonic, that is: for any* $c_1, c_2, c_3 \in C$ *s.t. there is* $a \in \Sigma \cup \{\varepsilon\}$ *with* $c_1 \overset{a}{\Rightarrow} c_2$ *and* $c_1 \overline{\leq} c_3$*, there exists* $c_4 \in C$ *with* $c_3 \overset{a}{\Rightarrow}^* c_4$ *and* $c_2 \overline{\leq} c_4$*.*
- $\left\langle C, \overline{\leq}\right\rangle$ *is an ordered set where* $\overline{\leq}$ *is a* WQO*.* ■

Here again, we can associate a (non-labelled) WSTS to any WSTS, as in the case of (labelled) transition systems. For that reason, we often denote a labelled WSTS simply by WSTS. The context should make clear whether the labels are relevant or not.

Labelled EPN The same ideas can be applied to the EPN case:

Definition 2.40 (Labelled Extended Petri nets) *A* Labelled Extended Petri Net $\mathcal{N}$ *is a tuple* $\langle P, T, \Sigma, \mathbf{m}_0\rangle$*, where:*

- P *is a finite set* $\{p_1, p_2, \ldots, p_n\}$ *of places;*
- T *is a finite set of transitions. Each transition is of the form* $\langle I, O, s, d, b, \lambda\rangle$*, where* I *and* $O : P \mapsto \mathbb{N}$ *are multi-sets of input and output places respectively.* $s, d \in P \cup \{\bot\}$ *are the source and the destination places respectively of a* special arc*,* $b \in \mathbb{N} \cup \{+\infty\}$ *is the bound associated to the special arc and* $\lambda \in \Sigma \cup \{\varepsilon\}$ *is the label of the transition.* ■

As in the cases of transition systems and WSTS, we see a labelled EPN as an EPN whose transitions have been labelled by characters of Σ or by ε. Hence, the classification into PN, PN+NBA, PN+T and PN+R of Definition 2.22 is adopted in the present case too. As a consequence, we sometimes write 'EPN' instead of 'labelled EPN' and the effects of (sequences of) transitions are computed the same way (see Definition 2.23). We also re-use the notation $\mathbf{m}_1 \xrightarrow{\sigma} \mathbf{m}_2$ (where σ is a finite sequence of transitions) to mean that σ is enabled in $\mathbf{m}_1$ and that firing it leads to $\mathbf{m}_2$.

A labelled EPN $\mathcal{N} = \langle P, T, \Sigma, \mathbf{m}_0 \rangle$ naturally defines $\mathcal{C}_{\mathcal{N}} = \left\langle \mathbb{N}^{|P|}, \mathbf{m}_0, \Sigma, \Rightarrow, \preccurlyeq \right\rangle$, a labelled WSTS s.t. $\mathbf{m}_1 \overset{a}{\Rightarrow} \mathbf{m}_2$ iff there exists $t = \langle I, O, s, d, b, \lambda \rangle \in T$ with $\lambda = a$ and $\mathbf{m}_1 \xrightarrow{t} \mathbf{m}_2$.

Finally, we define a function Λ which labels the sequences of transitions of an EPN, as follows:

Definition 2.41 (Label of a Sequence of Transitions) *Let σ be a finite sequence of transitions.*

- *If σ is the empty sequence, then $\Lambda(\sigma) = \varepsilon$.*
- *If $\sigma = t$, where $t = \langle I, O, s, d, b, \lambda \rangle$ is a single transition, then $\Lambda(\sigma) = \lambda$.*
- *If σ is the finite sequence $t_1 t_2 \dots t_k$ for $k > 1$, then $\Lambda(\sigma) = \Lambda(t_1 \dots t_{k-1}) \cdot \Lambda(t_k)$.*
- *If σ is the infinite sequence $t_1 t_2 \dots$, then $\Lambda(\sigma) = \lim_{i \to +\infty} \Lambda(t_1 \dots t_i)$.* ■

Remark that this definition is sound even in the case where σ is infinite since any infinite sequence of transitions admits one and only one prefix of length i (for any $i \geq 0$). Remark further that it $\Lambda(\sigma)$ might be finite even for an infinite sequence σ. This happens when σ contains only finitely many occurrences of transitions that are labelled by a character of Σ (and the others are labelled by ε).

By abuse of notation we also write $\mathbf{m} \xrightarrow{x} \mathbf{m}'$ to mean that there exists a *finite* sequence of transitions σ such that $\Lambda(\sigma) = x$ and $\mathbf{m} \xrightarrow{\sigma} \mathbf{m}'$, and $\mathbf{m} \xrightarrow{x'}$ to mean that we can fire the *infinite* sequence of transitions σ' (with $\Lambda(\sigma') = x'$) from $\mathbf{m}$. Remark that if $\mathcal{N}$ is a labelled EPN and $\mathcal{C}_{\mathcal{N}} = \langle C, c_0, \Sigma, \Rightarrow, \preccurlyeq \rangle$ is its corresponding labelled WSTS, then for any pair of markings $\mathbf{m}_1$ and $\mathbf{m}_2$ and any word w, $\mathbf{m}_1 \xrightarrow{w} \mathbf{m}_2$ iff $\mathbf{m}_1 \overset{w}{\Rightarrow}{}^* \mathbf{m}_2$. The same holds in the case of an ω-word w: $\mathbf{m} \xrightarrow{w}$ iff there is an infinite execution of $\mathcal{C}_{\mathcal{N}}$ that starts in $\mathbf{m}$ and is labelled by w.

We are now ready to introduce two notions of *language of* WSTS: in terms of *finite* and *infinite* words. This closely follows a classical approach that has been adopted for many other models of computations such as finite automata [HMU01].

2.5.2 Accepted finite words language

In order to define the set of *finite* words that are accepted by a WSTS, one has to fix an acceptance condition that takes the form of a set of *accepting configurations*. A word w is accepted by the WSTS iff w labels some initialised execution of the WSTS that ends in a configuration belonging to the accepting set of configurations:

Definition 2.42 (Language of a WSTS) *Given a* WSTS $\mathcal{S} = \langle C, c_0, \Sigma, \Rightarrow, \overline{\leq} \rangle$, *and a set $C' \subseteq C$ of* accepting configurations, *the* language *of $\mathcal{S}$ (for the set C' of accepting configurations), noted $L(\mathcal{S}, C')$, is the set of all the finite words w such that $c_0 \overset{w}{\Rightarrow}{}^* c$ for some $c \in C'$.* ■

By imposing some well-chosen restrictions about the set of accepting configurations, one can obtain different classes of languages. In the restricted case of PN, this approach has already been followed in classical works of the literature such as [Pet81], [Sal85] or [Jan86]. These are the classes we will consider:

Definition 2.43 (Classes of Languages of WSTS) *Let W be a (finite or infinite) set of* WSTS. *Then:*

- $L^L(W) = \{L(\mathcal{S}, C') \mid \mathcal{S} \in W$ *and C' is finite*$\}$ *is the set of all the languages that are defined by a* WSTS $\mathcal{S}$ *in S with a* finite set *of accepting configurations.*
- $L^T(W) = \{L(\mathcal{S}, \mathsf{Dead}\,(\mathcal{S})) \mid \mathcal{S} \in W\}$ *is the set of all the languages that are defined by a* WSTS $\mathcal{S}$ *in S and where the accepting configurations are all the* deadlock *configurations of $\mathcal{S}$.*
- $L^G(W) = \{L(\mathcal{S}, C') \mid \mathcal{S} \in W$ *and C' is upward-closed*$\}$ *is the set of all the languages that are defined by a* WSTS $\mathcal{S}$ *in S with an* upward-closed *set of accepting configurations.*
- $L^P(W) = \{L(\mathcal{S}, C) \mid \mathcal{S} = \langle C, c_0, \Sigma, \Rightarrow, \overline{\leq} \rangle \in W\}$ *is the set of all the languages that are defined by a* WSTS $\mathcal{S}$ *in W and where* any *configuration is accepting.*

Moreover, the classes $L^L_{\not\varepsilon}(W)$, $L^T_{\not\varepsilon}(W)$, $L^G_{\not\varepsilon}(W)$ and $L^P_{\not\varepsilon}(W)$ are defined the same way with the additional restriction that the WSTS $\mathcal{S}$ *that recognises the language has no ε-labelled transitions.* ■

Remark that for any set W of WSTS, $L^L_{\not\varepsilon}(W) \subseteq L^L(W)$, $L^T_{\not\varepsilon}(W) \subseteq L^T(W)$, $L^G_{\not\varepsilon}(W) \subseteq L^G(W)$ and $L^P_{\not\varepsilon}(W) \subseteq L^P(W)$. Moreover, $L^P(W) \subseteq L^G(W)$ and $L^P_{\not\varepsilon}(W) \subseteq L^G_{\not\varepsilon}(W)$, since the set of all the configurations is upward-closed. Hence, we will seldom discuss the classes L^P and $L^P_{\not\varepsilon}(W)$ in the sequel. Most of the results that hold on L^G (resp. $L^G_{\not\varepsilon}(W)$) will be easy to adapt to the class L^P ($L^P_{\not\varepsilon}(W)$). We also abuse the notation by writing, for instance $L^G(\mathsf{PN})$ to denote the class of languages that are accepted by some WSTS $\mathcal{C}_{\mathcal{N}}$ that corresponds to a PN $\mathcal{N}$. Not surprisingly, these different classes of languages enjoy different properties, as we will see in the sequel.

2.5.3 Accepted ω-language

Let us now consider under which condition an *infinite* word is accepted by a WSTS. In the present case, there is no need for a set of accepting configurations since the executions are never ending[11]. Remark that we distinguish two cases, depending on the fact that we allow ε-labelled transitions to fire when accepting a word or not.

Definition 2.44 (ω-LANGUAGE OF A WSTS) *Let $\mathcal{S} = \langle C, c_0, \Sigma, \Rightarrow, \leq \rangle$ be a WSTS. An ω-word w on Σ is* accepted *by $\mathcal{S}$ iff there exists an infinite execution $c_0, c_1, \ldots, c_j, \ldots$ of $\mathcal{C}$ and an infinite sequence $a_1, a_2, \ldots, a_j, \ldots$ of elements of $\Sigma \cup \{\varepsilon\}$ s.t. for any $i \geq 0$: $c_i \overset{a_{i+1}}{\Rightarrow} c_{i+1}$ and $w = a_1 \cdot a_2 \cdots a_j \cdots$*

The ω-language *$L^{\omega}(\mathcal{S})$ of $\mathcal{S}$ is the set of the ω-words that are accepted by $\mathcal{S}$.* ■

As in the case of finite words languages, we can define classes of ω-languages:

Definition 2.45 (CLASSES OF ω-LANGUAGES OF WSTS) *Let W be a (finite or infinite) set of WSTS. Then:*

1. *$L^{\omega}(W) = \{L^{\omega}(\mathcal{S}) \mid \mathcal{S} \in W\}$.*
2. *$L^{\omega}_{\not\varepsilon}(W) = \{L^{\omega}(\mathcal{S}) \mid \mathcal{S} \in W$ and $\mathcal{S}$ has no ε–transition$\}$.* ■

In the present case too, we abuse the notations and write L^{ω}(PN), L^{ω}(PN+T), and so forth.

2.6 Decidability problems on WSTS

In the previous sections, we have thoroughly defined the notion of WSTS and shown how this theoretical model can be of practical interest, by providing several examples of widely studied models that are (labelled) WSTS. However, these models wouldn't be really interesting if only trivial properties could be decided on them. In the present section, we present several decision problems on WSTS, or their subclasses. Decidability results for these problems will be provided in the sequel.

2.6.1 Behavioural properties

The reachability problem The reachability problem asks a question of the form '*is it guaranteed that a given situation never occurs ?*'. It can thus be used to check safety properties:

[11]We do not consider here acceptance conditions such as Büchi conditions, which ask that a run visits infinitely often some states of the systems to be accepting.

Problem 1 (RPWsts: The reachability problem for WSTS)

- **Instance:** *A* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ *and a configuration* $c \in C$*.*
- **Question:** *Does* $c \in \mathsf{Reach}(\mathcal{S})$ *?* ■

The coverability problem That problem is closely related to the reachability problem and is of major interest, as far as WSTS are concerned. For that reason, it has been widely studied in previous works [KM69, ACJT96, ABJ98, AAB99, AN01, ADMN04, DR00, DRVB02, RVB04, FS01, FRSB02, EN98, EFM99, DFS98].

The coverability problem asks whether a given upward-closed set of configurations is reachable in a given WSTS. Many safety properties can thus be reduced to the coverability problem too. For instance, the property we have introduced at Example 2.8, can be proved by solving an instance of the coverability problem.

Problem 2 (CPWsts: The coverability problem for WSTS)

- **Instance:** *A* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ *and a* $\overline{\leq}$*-upward-closed set* $U \subseteq C$*.*
- **Question:** *Does* $\mathsf{Reach}(\mathcal{S}) \cap U \neq \emptyset$ *?* ■

Let us provide an example of instance of CPWsts:

Example 2.14 *The* WSTS *that is defined by the Petri net of Example 2.8, together with the upward-closed set* $\{\mathbf{m} \mid \langle 0, 2, 0 \rangle \preccurlyeq \mathbf{m}\}$ *forms an instance of* CPWsts*. The mutual exclusion of the system is violated if and only if the answer is positive.* ◊

We will see in Section 3.2 that this problem is decidable on the class of EWSTS, and we shall introduce in Chapter 4 and Chapter 5 a new solution to this problem.

The unbounded computation problem That problem asks whether there is an infinite computation in the WSTS:

Problem 3 (UCWsts: The unbounded computation for WSTS)

- **Instance:** *A* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$.
- **Question:** *Does* $\mathcal{S}$ *admit an infinite initialised execution ?* ■

Problems specific to EPN More specific decidability results have been obtained when considering the restricted class of EPN. Three problems of interest are: the place boundedness, the quasi-liveness and the boundedness.

Problem 4 (PBEpn: The place-boundedness for EPN)

- **Instance:** *An* EPN $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$*, a place* $p \in P$
- **Question:** *Is there* $n \in \mathbb{N}$*, such that for all* $\mathbf{m} \in \mathsf{Reach}(\mathcal{N})$*:* $\mathbf{m}(p) \leq n$ *?* ■

Problem 5 (QLEpn: The quasi-liveness problem for EPN)

- **Instance:** *An* EPN $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ *and a transition* $t \in T$
- **Question:** *Is there* $\mathbf{m} \in \mathsf{Reach}(\mathcal{N})$ *such that* $\mathbf{m} \xrightarrow{t}$ *?* ■

Remark that this problem can be reduced to the coverability problem. Indeed, let $t = \langle I, O, s, d, b \rangle$ be a transition of an EPN. Then t is firable from any marking $\mathbf{m}$ s.t. for any place p: $\mathbf{m}(p) \geq I(p)$. Thus, there exists a $\mathbf{m} \in \mathsf{Reach}(\mathcal{N})$ with $\mathbf{m} \xrightarrow{t}$ iff the upward-closed set $U = \{\mathbf{m} \mid \forall p : \mathbf{m}(p) \geq I(p)\}$ has a non-empty intersection with $\mathsf{Reach}(\mathcal{N})$.

Problem 6 (BoundEpn: The boundedness for EPN)

- **Instance:** *An* EPN $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$*.*
- **Question:** *Is* $\mathsf{Reach}(\mathcal{N})$ *finite ?* ■

Decidability results for these problems will be discussed in Section 3.3

2.6.2 Expressiveness properties

Emptiness The emptiness problem asks whether the (finite words) language of a WSTS is empty.

Problem 7 (EmptyWsts: The emptiness problem for WSTS)

- **Instance:** *A labelled* WSTS $\mathcal{S} = \left\langle C, c_0, \Sigma, \Rightarrow, \overline{\preceq} \right\rangle$ *and a set* $C' \subseteq C$ *of accepting configurations.*
- **Question:** *Does* $L(\mathcal{S}, C') = \emptyset$ *?* ■

Remark that a similar notion can be defined on ω-language of WSTS. However, we do not address it in this thesis.

Universality The universality problem – defined here again in terms of finite words language – asks whether the WSTS can accept *any finite word* definable on its alphabet Σ:

Problem 8 (UnivEWsts: The universality problem for WSTS)

- **Instance:** *A labelled* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq}, \Sigma \right\rangle$ *and a set* $C' \subseteq C$ *of accepting configurations.*
- **Question:** *Does* $L(\mathcal{S}, C') = \Sigma^*$ *?* ■

LTL The action–based LTL is a logic that is interpreted over ω–words. Given a finite alphabet Σ, a formula ϕ of action–based LTL is syntactically defined by the following BNF rule:

$$\phi ::= a \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \bigcirc\phi_1 \mid \Box\phi_1 \mid \diamond\phi_1 \mid \phi_1\mathcal{U}\phi_2$$

where $a \in \Sigma$ and ϕ_1 and ϕ_2 are action–based LTL formula.

Let $w = w_1 w_2 \cdots w_i \cdots$ be an ω-word on Σ and let ϕ be an action–based LTL formula. For any $i \geq 1$, let $s(w,i)$ be the infinite suffix $w_i w_{i+1} w_{i+2} \cdots$ of w. We say that w *satisfies* ϕ (written $w \models \phi$) iff one of the following holds:

1. $\phi = a$ implies $w_1 = a$;
2. $\phi = \neg\phi_1$ implies that $w \not\models \phi'$;
3. $\phi = \phi_1 \vee \phi_2$ implies that either $w \models \phi_1$ or $w \models \phi_2$;
4. $\phi = \bigcirc\phi_1$ implies that $s(w,2) \models \phi_1$;
5. $\phi = \Box\phi_1$ implies that for any $j \geq 1$: $s(w,j) \models \phi_1$;
6. $\phi = \diamond\phi_1$ implies that $w \models \neg\Box\neg\phi_1$;
7. $\phi = \phi_1\mathcal{U}\phi_2$ implies that there exists $k \geq 1$ s.t. for any $1 \leq j < k$: $s(w,j) \models \phi_1$ and for any $\ell \geq k$: $s(w,\ell) \models \phi_2$.

Given an ω-language L on Σ, and an action–based LTL formula ϕ, we say that L *satisfies* ϕ (noted $L \models \phi$) iff for every word $w \in L$, $w \models \phi$. Hence, we define the LTL satisfiability problem as:

Problem 9 (LTLSatis: The action–based LTL satisfiability problem for WSTS)

- **Instance:** *A labelled* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq}, \Sigma \right\rangle$, *and an action–based LTL formula (on* Σ*).*
- **Question:** *Does* $L^\omega(\mathcal{S}) \models \phi$ *?* ■

Chapter 3

State of the art

In the previous chapter, we have introduced the main concepts that will be studied in this thesis. We have shown how to represent and manipulate upward– and downward–closed sets (thanks to an adequate domain of limits, in the case of downward–closed sets); we have stated several decision problems on the models of computation (WSTS, SMPN, EPN, LCS) that we want to study; and we have defined their respective notions of accepted languages.

In the present chapter, we review several results of the literature regarding these three topics. More precisely:

1. In section 3.1, we present adequate domains of limits that are suitable to represent downward–closed sets of configurations in the case of EPN, SMPN and LCS. This will be important for the algorithms we discuss in Part I of this thesis.
2. In Section 3.2 and Section 3.3, we recall decidability results on WSTS and some of their subclasses. More precisely, in Section 3.2, we provide a short survey of the various results obtained regarding the *coverability problem* on WSTS. We recall the general *backward algorithm* of [ACJT96, FS01], as well as the Karp&Miller algorithm [KM69], the only complete forward algorithm to decide the coverability problem on the class PN only. Then, we discuss several attempts to extend the Karp&Miller procedure to other classes of WSTS. Unfortunately, they have all produced semi-algorithms.

   In section 3.3, we address the other decidability problems. The emphasis on the coverability problem is motivated by the fact that the first part of this thesis is devoted to this problem.
3. In Section 3.4 we present several results that have been obtained regarding the respective expressive powers of PN, PN+T, PN+NBA and PN+R mainly (to the best of our knowledge, the general case of the expressiveness of WSTS has not been addressed yet).

3.1 Adequate domains of limits

We open the chapter by recalling domains of limits that are suitable to represent downward-closed sets of tuples of naturals and downward-closed sets of configurations of LCS.

3.1.1 An adequate domain of limits for $\langle \mathbb{N}^k, \preccurlyeq \rangle$

The domain of limits we are about to introduce is suitable to finitely represent sets of tuples of naturals that are downward-closed. This will be useful mainly in the analysis of Petri nets and their monotonic extensions. We will consider ω-markings, which are tuples over $\mathbb{N} \cup \{\omega\}$. The limit elements will be ω-markings with at least one ω (in order to avoid that the set of limits intersects with the set of configurations). Intuitively, the ω symbol represents 'any natural value'. Thus, for instance, the tuple $\langle 1, 1, \omega \rangle$ will be used to represent all the tuples of naturals $\langle i_1, i_2, i_3 \rangle$ s.t. $i_1 \leq 1$, $i_2 \leq 1$.

The idea of using ω-markings to represent downward-closed sets of tuple of naturals is certainly not new. It appears already in the algorithm that computes a finite representation of $\downarrow(\mathsf{Reach}\,(\mathcal{N}))$, introduced by Karp and Miller in [KM69] (we discuss this algorithm in Section 3.2.4).

We first define precisely the domain of limits and then show that it is adequate for $\langle \mathbb{N}^k, \preccurlyeq \rangle$. Then, we discuss an extension of the transition relation of strongly monotonic SMPN to handle extended markings. This extension will be useful to compute the most precise over-approximation of the successors of a given ω-marking, that is representable by means of an ω-marking.

We consider the domain of limits $\langle \mathcal{L}, \preccurlyeq_e, \gamma \rangle$ where:

1. $\mathcal{L} = (\mathbb{N} \cup \{\omega\})^k \setminus \mathbb{N}^k$;
2. $\preccurlyeq_e \subseteq (\mathbb{N} \cup \{\omega\})^k \times (\mathbb{N} \cup \{\omega\})^k$ is such that $\langle m_1, \ldots, m_k \rangle \preccurlyeq_e \langle m'_1, \ldots, m'_k \rangle$ if and only if $\forall 1 \leq i \leq k : m_i \leq m'_i$ where $c < \omega$ for all $c \in \mathbb{N}$;
3. γ is defined as: $\gamma\,(\mathbf{m}) = \{\mathbf{m}' \in \mathbb{N}^k \mid \mathbf{m}' \preccurlyeq_e \mathbf{m}\}$.

In the following, tuples in $\mathcal{L}$ are called *ω-markings*. We also note $\mathbf{m}_1 \prec_e \mathbf{m}_2$ when $\mathbf{m}_1 \preccurlyeq_e \mathbf{m}_2$ but $\mathbf{m}_2 \not\preccurlyeq_e \mathbf{m}_1$. Notice that in the present case, the $\top$ element (with $\gamma(\top) = \mathbb{N}^k$) is the ω-marking that assigns ω to all the places. The following property holds on ω-markings:

Proposition 3.1 *Given an ω-marking $\mathbf{m}$ and a finite set $S = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n\}$ of ω-markings, the following property holds: $\gamma\,(\mathbf{m}) \subseteq \gamma\,(S)$ if and only if there exists $1 \leq i \leq n$ s.t. $\mathbf{m} \preccurlyeq_e \mathbf{m}_i$.*

Proof. In the case where there exists $1 \leq i \leq n$ s.t. $\mathbf{m} \preccurlyeq_e \mathbf{m}_i$, we have $\gamma(\mathbf{m}) \subseteq \gamma(\mathbf{m}_i)$, hence $\gamma(\mathbf{m}) \subseteq \gamma(S)$.

Let us show, *per absurdum* that $\gamma(\mathbf{m}) \subseteq \gamma(S)$ implies that there is $1 \leq i \leq n$ s.t. $\mathbf{m} \preccurlyeq_e \mathbf{m}_i$. Thus, let us assume it is not the case, that is, $\gamma(\mathbf{m}) \subseteq \gamma(S)$ but there is no $1 \leq i \leq n$ s.t. $\mathbf{m} \preccurlyeq_e \mathbf{m}_i$. Thus, there are two indices k and ℓ s.t. $k \neq \ell$ and s.t. there are two markings $\overline{\mathbf{m}}_k, \overline{\mathbf{m}}_l \in \gamma(\mathbf{m})$ with: $\overline{\mathbf{m}}_k \in \gamma(\mathbf{m}_k) \setminus \cup_{i \neq k}\gamma(\mathbf{m}_i)$ and $\overline{\mathbf{m}}_\ell \in \gamma(\mathbf{m}_\ell) \setminus \cup_{i \neq \ell}\gamma(\mathbf{m}_i)$. This means in particular that there is no $1 \leq j \leq n$ s.t. $\overline{\mathbf{m}}_k \in \gamma(\mathbf{m}_j)$ and $\overline{\mathbf{m}}_\ell \in \gamma(\mathbf{m}_j)$.

Since both $\overline{\mathbf{m}}_k$ and $\overline{\mathbf{m}}_\ell$ are in $\gamma(\mathbf{m})$, we have $\overline{\mathbf{m}}_k \preccurlyeq_e \mathbf{m}$ and $\overline{\mathbf{m}}_\ell \preccurlyeq_e \mathbf{m}$, by definition of γ. Thus, for any place p, $\mathbf{m}(p) \geq \overline{\mathbf{m}}_k(p)$ and $\mathbf{m}(p) \geq \overline{\mathbf{m}}_\ell(p)$. Hence, the marking $\overline{\mathbf{m}}$ s.t. for any place p, $\overline{\mathbf{m}}(p) = \max\{\overline{\mathbf{m}}_k(p), \overline{\mathbf{m}}_\ell(p)\}$ is s.t. $\overline{\mathbf{m}} \preccurlyeq_e \mathbf{m}$ which means that $\overline{\mathbf{m}} \in \gamma(\mathbf{m}) \subseteq \gamma(S)$. Thus, by definition of γ, there is $1 \leq j \leq n$ s.t. $\overline{\mathbf{m}} \preccurlyeq_e \mathbf{m}_j$. Since $\preccurlyeq_e$ is transitive, we have $\overline{\mathbf{m}}_k \preccurlyeq_e \mathbf{m}_j$ and $\overline{\mathbf{m}}_\ell \preccurlyeq_e \mathbf{m}_j$. We conclude that there is $1 \leq j \leq n$ s.t. $\overline{\mathbf{m}}_\ell \in \gamma(\mathbf{m}_j)$ and $\overline{\mathbf{m}}_k \in \gamma(\mathbf{m}_j)$. Contradiction. □

It is also useful to remark that any downward-closed set in this domain can be uniquely and finitely represented by a set of ω-markings, as stated by the next lemma:

Lemma 3.1 *For any $\preccurlyeq$-downward-closed set D in $\mathbb{N}^k$ there exists a set $\mathcal{D} \subseteq \big(\mathbb{N} \cup \{\omega\}\big)^k$ which:*

1. *is a* generator *of D: $\gamma(\mathcal{D}) = D$;*
2. *is* canonical*: for any $\mathbf{m}_1, \mathbf{m}_2 \in \mathcal{D}$, $\mathbf{m}_1 \neq \mathbf{m}_2$ implies that $\mathbf{m}_1 \not\preccurlyeq_e \mathbf{m}_2$;*
3. *is* finite*;*
4. *is* unique*, i.e., there is no $\mathcal{D}' \subseteq \big(\mathbb{N} \cup \{\omega\}\big)^k$ s.t. $\mathcal{D}' \neq \mathcal{D}$ and $\gamma(\mathcal{D}') = \gamma(\mathcal{D}) = D$.*

Proof. Let D be a $\preccurlyeq$–downward–closed set of $\mathbb{N}^k$. In the case where $D = \mathbb{N}^k$, we let $\mathcal{D} = \langle \omega, \omega, \ldots, \omega \rangle$. In the case where $D = \emptyset$, we let $\mathcal{D} = \emptyset$. In both cases, it is clear that $\mathcal{D}$ is a finite and canonical representation of D.

In the case where D is neither $\emptyset$ nor $\mathbb{N}^k$, we let $U = \{\mathbf{m} \mid \mathbf{m} \notin D\}$. Let us show, *per absurdum* that U is $\overline{\leq}$–upward–closed. Let us consider $\mathbf{m}_1 \in U$, and let us assume that there exists $\mathbf{m}_2 \succcurlyeq \mathbf{m}_1$ with $\mathbf{m}_2 \notin U$. The latter implies that $\mathbf{m}_2 \in D$. Since D is $\preccurlyeq$–downward–closed, any $\mathbf{m}_3 \preccurlyeq \mathbf{m}_2$ is in D. Hence, $\mathbf{m}_1$ is in D. Thus, $\mathbf{m}_1$ is not in U. Contradiction.

By Lemma 2.1, U is representable by a finite set $\mathsf{UGen}(U) = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n\}$. Thus, U is exactly the set of markings $\mathbf{m}$ s.t.

$$\bigvee_{i=1}^{n} \bigwedge_{j=1}^{k} \mathbf{m}_i(p_j) \leq \mathbf{m}(p_j)$$

Hence, since $D = \{\mathbf{m} \mid \mathbf{m} \notin U\}$, D is exactly the set of markings $\mathbf{m}$ s.t.:

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k} \mathbf{m}_i(p_j) > \mathbf{m}(p_j)$$

By transforming this formula in disjunctive normal form[1], we obtain a new equivalent formula of the form:

$$\Phi = \bigvee_{j=1}^{k^n} \bigwedge_{i=1}^{n} \varphi_{i,j}$$

where each $\varphi_{i,j}$ is of the form $c > \mathbf{m}(p_m)$, $c \in \mathbb{N}$ and $1 \le m \le k$. Let Ψ_j denote, for every $1 \le j \le k^n$, the conjunction $\wedge_{i=1}^{n} \varphi_{i,j}$.

The formula Φ can be simplified by applying the following transformations successively:

1. If there is $1 \le j \le k^n$ and $1 \le i \le n$ s.t. $\varphi_{i,j}$ is of the form $0 > \mathbf{m}(p)$, we suppress Ψ_j from Φ, since $\mathbf{m}(p) \in \mathbb{N}$.

2. For any remaining $1 \le j \le k^n$ s.t. there are $1 \le m \le k$, $x, y \in \mathbb{N}$ and $1 \le \ell_1 < \ell_2 \le n$ with $\varphi_{\ell_1,j} = x > \mathbf{m}(p_m)$ and $\varphi_{\ell_2,j} = y > \mathbf{m}(p_m)$, we replace Ψ_j by:

$$\bigwedge_{i=1}^{\ell_1-1} \varphi_{i,j} \wedge \bigwedge_{i=\ell_1+1}^{\ell_2-1} \varphi_{i,j} \wedge \bigwedge_{i=\ell_2+1}^{n} \varphi_{i,j} \wedge K > \mathbf{m}(p_m)$$

   where $K = \min\{x, y\}$.

It is not difficult to see that the resulting formula Φ' denotes the same set of markings, i.e., D, than Φ.

Finally, for each remaining $1 \le j \le k^n$, replace Φ_j by the formula $\mathbf{m} \preccurlyeq_e \mathbf{m}'_j$, where, for any $1 \le i \le k$, $\mathbf{m}'_j = \min\{y \mid y > \mathbf{m}(p_i) \text{ is a conjunct of } \Psi_j\} - 1$ (with the convention that $\min \emptyset = \omega$ and $\omega - 1 = \omega$).

It is thus clear that, for any marking $\mathbf{m}$, $\mathbf{m}$ satisfies Ψ'_j iff $\mathbf{m} \preccurlyeq_e \mathbf{m}'_j$. Since Φ' contains finitely many Ψ'_j, the set $\{\mathbf{m}'_j \mid 1 \le j \le c\}$ is a finite representation of D. Since this set is finite, we can extract from it a canonical set $\mathcal{D} = \{\mathbf{m}'_j \mid 1 \le j \le c \wedge \nexists 1 \le i \le c : \mathbf{m}'_j \prec_e \mathbf{m}'_i\}$.

We finish the proof by showing that there exists a unique canonical set $\mathcal{D}$ such that $\gamma(\mathcal{D}) = D$. This can be done *per absurdum*: suppose there is another canonical set $\mathcal{D}'$ of markings that represents D. Without loss of generality, that means that there exists $\mathbf{m} \in \mathcal{D}$ such that $\mathbf{m} \notin \mathcal{D}'$. Since $\forall \mathbf{m}' \in \gamma(\mathbf{m}) : \mathbf{m}' \in D$ and $\gamma(\mathcal{D}') = D$ by hypothesis, we have following Proposition 3.1 that there exists $\mathbf{m}' \in \mathcal{D}' : \mathbf{m} \prec_e \mathbf{m}'$.

[1]Through successive applications of the following distributive rule: $\phi \wedge (\psi_1 \vee \psi_2) = (\phi \wedge \psi_1) \vee (\phi \wedge \psi_2)$

Since $\gamma(\mathbf{m}) \subset \gamma(\mathbf{m}')$ and $\gamma(\mathcal{D}) = D$, we conclude, by Proposition 3.1, that there exists $\mathbf{m}'' \in \mathcal{D} : \mathbf{m} \prec_e \mathbf{m}''$. Hence, $\mathcal{D}$ is not canonical, which is a contradiction. $\square$

Example 3.1 *Let us provide an example of the construction used in the proof of Lemma 3.1. For that purpose, we consider the downward-closed set of 2-tuples of natural numbers* $D = \{\mathbf{m} \mid (\mathbf{m}(p_1) \leq 1 \wedge \mathbf{m}(p_2) \leq 1) \vee \mathbf{m}(p_1) \leq 0\}$*. Its complement is the upward-closed set U characterised by the following disjunctive formula:*

$$\begin{array}{c}\big(\mathbf{m}(p_1) \geq 2 \quad \wedge \quad \mathbf{m}(p_2) \geq 0\big)\\ \vee \\ \big(\mathbf{m}(p_1) \geq 1 \quad \wedge \quad \mathbf{m}(p_2) \geq 2\big)\end{array}$$

Thus D, is exactly the set of markings $\mathbf{m}$ *s.t.:*

$$\begin{array}{c}\big(\mathbf{m}(p_1) < 2 \quad \vee \quad \mathbf{m}(p_2) < 0\big)\\ \wedge \\ \big(\mathbf{m}(p_1) < 1 \quad \vee \quad \mathbf{m}(p_2) < 2\big)\end{array}$$

This formula is equivalent to the the following formula in disjunctive normal form:

$$\begin{array}{c}\big(\mathbf{m}(p_1) < 2 \quad \wedge \quad \mathbf{m}(p_1) < 1\big)\\ \vee \\ \big(\mathbf{m}(p_2) < 0 \quad \wedge \quad \mathbf{m}(p_1) < 1\big)\\ \vee \\ \big(\mathbf{m}(p_1) < 2 \quad \wedge \quad \mathbf{m}(p_2) < 2\big)\\ \vee \\ \big(\mathbf{m}(p_2) < 0 \quad \wedge \quad \mathbf{m}(p_2) < 2\big)\end{array}$$

We can suppress from this formula the second and the fourth disjuncts, because they contain $\mathbf{m}(p_2) < 0$*. Moreover, the first disjunct is equivalent to* $\mathbf{m}(p_1) < 1$*. Hence, we obtain:*

$$\big(\mathbf{m}(p_1) < 1\big) \vee \big(\mathbf{m}(p_1) < 2 \wedge \mathbf{m}(p_2) < 2\big)$$

We finish the construction by associating one ω-marking to each disjunct. For the former disjunct, it is $\langle 0, \omega\rangle$, and for the latter it is $\langle 1, 1\rangle$. Thus:

$$D = \gamma\left(\{\, \langle 0, \omega\rangle \,, \langle 1, 1\rangle \,\}\right)$$

$\Diamond$

As a direct consequence of Lemma 3.1 and of the definition of γ, we obtain:

Theorem 3.1 *$\langle \mathcal{L}, \preccurlyeq_e, \gamma \rangle$ is an adequate domain of limits for $\langle \mathbb{N}^k, \preccurlyeq \rangle$.*

Proof. We prove that $\langle \mathcal{L}, \preccurlyeq_e, \gamma \rangle$ enforces the four points of Definition 2.11:

(L_1) *representation mapping*: follows directly from the definition of γ;

(L_2) *top element*: $\top = \langle \omega, \omega, \ldots, \omega \rangle$;

(L_3) *precision order*: follows directly from the definitions of γ and $\preccurlyeq_e$;

(L_4) *completeness*: follows from Lemma 3.1.

$\square$

Remark that Lemma 3.1 also states that each downward–closed set of markings can be *uniquely* represented by a finite set of ω-markings (which is not guaranteed in general by the definition of adequate domain of limits).

Extension of the transition relation Given a strongly monotonic SMPN, it is possible to extend its transition relation in order to handle directly ω-markings. More precisely, given an ω-marking $\mathbf{m}$ and a transition t, we would like to compute the unique ω-marking $\mathbf{m}'$ s.t. $\gamma(\mathbf{m}')$ is the downward-closure of the set of successors of $\gamma(\mathbf{m})$ by t.

Let $\mathcal{N}$ be a $\preccurlyeq$-strongly monotonic SMPN. We extend the underlying transition relation from markings to ω-markings[2]. For example, let us assume that for some place j and some transition i, we have $D^-_{ij}(\mathbf{m}) = \mathbf{m}(p_j)$, $D^+_{ij} = 5$, and $D^-_{ik}(\mathbf{m}) = D^+_{ik}(\mathbf{m}) = 0$ for any $k \neq j$. Let us consider an extended marking $\mathbf{m}$ s.t. $\mathbf{m}(p_j) = \omega$, and let us compute $\mathbf{m}'$ s.t. $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$. According to the definition of $\rightarrow$, we first compute $\mathbf{m}''$, which is s.t. $\mathbf{m}''(p_j) = \mathbf{m}(p_j) - D^-_{ij}(\mathbf{m}) = \omega - \omega = 0$, and $\mathbf{m}''(p_k) = \mathbf{m}(p_k)$, for any $k \neq j$. Then, we obtain $\mathbf{m}'$, by letting $\mathbf{m}'(p_j) = \mathbf{m}''(p_j) + D^+_{ij}(\mathbf{m}) = 0 + 5 = 5$, and $\mathbf{m}'(p_k) = \mathbf{m}''(p_k) = \mathbf{m}(p_k)$, for $k \neq j$.

In the sequel, we adopt the following notation. Let $\mathcal{N}$ be an SMPN, let $\mathbf{m}$ be an ω-marking of $\mathcal{N}$ and let t_i be a transition of $\mathcal{N}$. Let $\mathbf{m}'$ be s.t. $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$. Then we let $\mathsf{Post}(\mathbf{m}, t_i) = \mathbf{m}'$. That notation is extended to sets M of ω-markings in the natural way: $\mathsf{Post}(M, t_i) = \cup_{\mathbf{m} \in M} \mathsf{Post}(\mathbf{m}, t_i)$.

Let us show that the way we have extended the transition relation is well-suited in the following sense. Let $\mathbf{m}$ and $\mathbf{m}'$ be two (extended) markings such that $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$ for some transition t_i. Then $\gamma(\mathbf{m}')$ is the most precise $\preccurlyeq_e$-downward-closed over-approximation for $\mathsf{Post}(\gamma(\mathbf{m}), t_i)$.

[2]Here again, we assume that $\omega + \omega = \omega$, $\omega - \omega = 0$, $c \cdot \omega = \omega$ for all $c \in \mathbb{N} \setminus \{0\}$, $0 \cdot \omega = 0$, $\omega + c = \omega$ for all $c \in \mathbb{Z}$

Lemma 3.2 *Let $\mathcal{N} = \langle P, T, D^+, D^-, \mathbf{m}_0 \rangle$ be a $\preccurlyeq$-strongly monotonic* SMPN *and let $\mathbf{m}, \mathbf{m}'$ be two ω-markings. If $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$ for some $t_i \in T$, then $\gamma(\mathbf{m}')$ has the two following properties:*

1. **Covering***: $\mathsf{Post}(\gamma(\mathbf{m}), t_i) \subseteq \gamma(\mathbf{m}')$;*

2. **Preciseness***: there is no finite set $S \subseteq \left(\mathbb{N} \cup \{\omega\}\right)^{|P|}$ such that $\mathsf{Post}(\gamma(\mathbf{m}), t_i) \subseteq \gamma(S) \subset \gamma(\mathbf{m}')$.*

Proof. **(Covering)** Suppose that the covering property is not verified. In this case, there exist four (possibly extended) markings $\mathbf{m}, \mathbf{m}', \mathbf{n}$ and $\mathbf{n}'$, and a transition $t_i \in T$ such that $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$, $\mathbf{n} \xrightarrow{t_i} \mathbf{n}'$, $\mathbf{n} \in \gamma(\mathbf{m})$ and $\mathbf{n}' \notin \gamma(\mathbf{m}')$. Hence, there exists $p_j \in P$ such that $\mathbf{n}'(p_j) > \mathbf{m}'(p_j)$.

Following Lemma 2.13[3], $D^-_{ij}(\mathbf{m})$ is either $\alpha' \in \mathbb{N}$, or $\mathbf{m}(p_j)$. Since $D^+_{ij}(\mathbf{m})$ is of the form $\sum_{p_k \in P} \beta''_k \cdot \mathbf{m}(p_k) + \alpha''$, with $\alpha'' \in \mathbb{N}$ and $\beta''_k \in \mathbb{N}$ for any $1 \leq k \leq |P|$, we deduce that the effect $D^+_{ij}(\mathbf{m}) - D^-_{ij}(\mathbf{m})$ of a transition t_i on place p_j for a marking $\mathbf{m}$, may be of two forms. Either $D^+_{ij}(\mathbf{m}) - D^-_{ij}(\mathbf{m}) = \sum_{p_k \in P} \beta_k \cdot \mathbf{m}(p_k) + \alpha$ or $D^+_{ij}(\mathbf{m}) - D^-_{ij}(\mathbf{m}) = \sum_{p_k \in P} \beta_k \cdot \mathbf{m}(p_k) + \alpha - \mathbf{m}(p_j)$ with $\beta_k \in \mathbb{N}$ for all k and $\alpha \in \mathbb{Z}$. Hence, either $\mathbf{n}'(p_j) = \mathbf{n}(p_j) + \sum_{p_k \in P} \beta_k \cdot \mathbf{n}(p_k) + \alpha$ and $\mathbf{m}'(p_j) = \mathbf{m}(p_j) + \sum_{p_k \in P} \beta_k \cdot \mathbf{n}(p_k) + \alpha$, or $\mathbf{n}'(p_j) = \sum_{p_k \in P} \beta_k \cdot \mathbf{n}(p_k) + \alpha$ and $\mathbf{m}'(p_j) = \sum_{p_k \in P} \beta_k \cdot \mathbf{m}(p_k) + \alpha$. In both cases, since $\mathbf{n} \in \gamma(\mathbf{m})$, $\mathbf{n}(p_k) \leq \mathbf{m}(p_k)$ for all $p_k \in P$, hence $\sum_{p_k \in P} \beta_k \cdot \mathbf{n}(p_k) + \alpha \leq \sum_{p_k \in P} \beta_k \cdot \mathbf{m}(p_k) + \alpha$. We conclude that $\mathbf{n}'(p_j) \leq \mathbf{m}'(p_j)$ and we obtain a contradiction.

(Preciseness) In order to establish the preciseness property, we prove that if $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$, then any marking $\mathbf{n}' \in \gamma(\mathbf{m}')$ is covered by a marking $\overline{\mathbf{n}}' \in \mathsf{Post}(\gamma(\mathbf{m}), t_i)$. This clearly implies that the set $\gamma(\mathbf{m}')$ is the minimal $\preccurlyeq_e$-downward-closed set that contains $\mathsf{Post}(\gamma(\mathbf{m}), t_i)$, since for any $\preccurlyeq_e$-downward-closed set $D \subset \gamma(\mathbf{m}')$, there exists at least one marking $\mathbf{n} \in \mathsf{Post}(\gamma(\mathbf{m}), t_i)$ that is not in D. We proceed by showing how to build $\overline{\mathbf{n}}'$, for any $\mathbf{n}' \in \gamma(\mathbf{m}')$.

For that purpose, we first define the marking $\overline{\mathbf{n}}$, from $\mathbf{n}'$, as follows. Let c be an natural constant s.t.

$$c > \max\{|\alpha_1|, \ldots, |\alpha_{|P|}|\} + \max\{\mathbf{n}'(p_k) \mid 1 \leq k \leq |P|\}$$

where α_j is the constant term in $D^+_{ij} - D^-_{ij}$. Then:

$$\forall 1 \leq j \leq |P| : \overline{\mathbf{n}}(p_j) = \begin{cases} \mathbf{m}(p_j) & \text{if } \mathbf{m}(p_j) \in \mathbb{N} \\ c & \text{otherwise} \end{cases}$$

Remark that, by construction, $\overline{\mathbf{n}} \in \gamma(\mathbf{m})$, and that t_i is firable from $\overline{\mathbf{n}}$. We let $\overline{\mathbf{n}}'$ be s.t. $\overline{\mathbf{n}} \xrightarrow{t_i} \overline{\mathbf{n}}'$. Hence, $\overline{\mathbf{n}}' \in \mathsf{Post}(\gamma(\mathbf{m}), t_i)$. Thus, from the covering property, $\overline{\mathbf{n}}' \in \gamma(\mathbf{m}')$.

[3]Remember that we have assumed that all the SMPN we consider in this thesis do not contain unfirable transitions.

Let us show that $\overline{\mathbf{n}}' \succcurlyeq \mathbf{n}'$. For that purpose, we show that, for any $1 \leq j \leq |P|$, $\mathbf{n}'(p_j) \leq \overline{\mathbf{n}}'(p_j)$. Remember that $\overline{\mathbf{n}}'$ has been obtained by firing t_i from $\overline{\mathbf{n}}$. According to Lemma 2.13 again, the effect of t_i might be of two form:

1. Either $D^+_{ij}(\mathbf{m}) - D^-_{ij}(\mathbf{m}) = \sum_{p_k \in P} \beta_k \cdot \mathbf{m}(p_k) + \alpha_j - \mathbf{m}(p_j)$ with $\beta_k \in \mathbb{N}$ for all k and $\alpha_j \in \mathbb{N}$. In that case, we have, for any $p_j \in P$:

$$\begin{aligned}\overline{\mathbf{n}}'(p_j) &= \overline{\mathbf{n}}(p_j) + \sum_{k=1}^{|P|} \beta_k \cdot \overline{\mathbf{n}}(p_k) + \alpha_j - \overline{\mathbf{n}}(p_j) \\ &= \sum_{k=1}^{|P|} \beta_k \cdot \overline{\mathbf{n}}(p_k) + \alpha_j\end{aligned}$$

Here again, we have to consider two cases:

(a) In the case where, for any $1 \leq k \leq |P|$, $\beta_k \neq 0$ implies that $\mathbf{m}(p_k) \neq \omega$, we have:

$$\begin{aligned}\sum_{k=1}^{|P|} \beta_k \cdot \overline{\mathbf{n}}(p_k) + \alpha_j &= \sum_{k=1}^{|P|} \beta_k \cdot \mathbf{m}(p_k) + \alpha_j \\ &= \mathbf{m}(p_j) + \sum_{k=1}^{|P|} \beta_k \cdot \mathbf{m}(p_k) + \alpha_j - \mathbf{m}(p_j) \\ &= \mathbf{m}'(p_j)\end{aligned}$$

because $\mathbf{m}'$ has been obtained by applying the effect of t_i on $\mathbf{m}$. Hence, $\overline{\mathbf{n}}'(p_j) = \mathbf{m}'(p_j)$. Since, $\mathbf{n}' \in \gamma(\mathbf{m}')$, we have $\mathbf{n}'(p_j) \leq \mathbf{m}'(p_j) = \overline{\mathbf{n}}'(p_j)$.

(b) In the case where there is $1 \leq k \leq |P|$ s.t. $\beta_k > 0$ and $\mathbf{m}(p_k) = \omega$, then the term $\beta_k \cdot \overline{\mathbf{n}}(p_k)$ of the sum is non-null and equal to $\beta_k \cdot c$, by definition of $\overline{\mathbf{n}}$. Hence, since the other terms of the sum, as well as α_j, are positive, we have:

$$\begin{aligned}\sum_{k=1}^{|P|} \beta_k \cdot \overline{\mathbf{n}}(p_k) + \alpha_j &\geq c \\ &\geq \max\{\mathbf{n}'(p_k) \mid p_k \in P\} \\ &\geq \mathbf{n}'(p_j)\end{aligned}$$

2. Or, $D^+_{ij}(\mathbf{m}) - D^-_{ij}(\mathbf{m}) = \sum_{p_k \in P} \beta_k \cdot \mathbf{m}(p_k) + \alpha_j$ with $\beta_k \in \mathbb{N}$ for all k and $\alpha_j \in \mathbb{Z}$. In that case, we have, for any $p_j \in P$:

$$\overline{\mathbf{n}}'(p_j) = \overline{\mathbf{n}}(p_j) + \sum_{p_k \in P} \beta_k \cdot \overline{\mathbf{n}}(p_k) + \alpha_j$$

We consider two cases:

(a) In the case where $\mathbf{m}(p_j) \neq \omega$ and, for any $1 \leq k \leq |P|$, $\beta_k \neq 0$ implies that $\mathbf{m}(p_k) \neq \omega$, we have:

$$\begin{aligned} \overline{\mathbf{n}}(p_j) + \sum_{p_k \in P} \beta_k \cdot \overline{\mathbf{n}}(p_k) + \alpha_j &= \mathbf{m}(p_j) + \sum_{p_k \in P} \beta_k \cdot \mathbf{m}(p_k) + \alpha_j \\ &= \mathbf{m}'(p_j) \end{aligned}$$

We conclude, as we did in (1a), that $\mathbf{n}(p_j) \leq \overline{\mathbf{n}}'(p_j)$.

(b) Otherwise, either $\overline{\mathbf{n}}(p_j) = c$ or there exists $1 \leq k \leq |P|$ s.t. the k-th term of the sum is non-null and equal to $\beta_k \cdot c$. In both cases, we have:

$$\begin{aligned} & \overline{\mathbf{n}}(p_j) + \sum_{p_k \in P} \beta_k \cdot \overline{\mathbf{n}}(p_k) + \alpha_j \\ \geq\ & c + \alpha_j \\ =\ & \max\{\mathbf{n}'(p_k) \mid p_k \in P\} + \max\{|\alpha_1|, |\alpha_2|, \ldots, |\alpha_{|P|}|\} + \alpha_j \\ \geq\ & \max\{\mathbf{n}'(p_k) \mid p_k \in P\} \\ \geq\ & \mathbf{n}'(p_j) \end{aligned}$$

□

The two conditions that we have just proved (successor covering and preciseness) will actually be necessary for the 'Expand, Enlarge and Check' algorithm (for the coverability problem of WSTS) that we will introduce in Chapter 4. We can obtain a stronger results, saying that the marking $\mathbf{m}'$ that we compute thanks to the extended transition relation, as successor by transition t_i of $\mathbf{m}'$, represents *exactly* $\gamma\left(\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)\right)$ (remark that, in general, $\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)$ is not downward-closed):

Proposition 3.2 *Let $\mathcal{N} = \langle P, T, D^+, D^-, \mathbf{m}_0\rangle$ be a $\preccurlyeq$-strongly monotonic* SMPN, *let $t_i \in T$ be a transition and let $\mathbf{m}$ and $\mathbf{m}'$ be two ω-markings s.t. $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$. Then $\gamma\left(\mathbf{m}'\right) = \gamma\left(\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)\right)$.*

Proof. Per absurdum. Let us assume it is not the case, i.e., $\gamma\left(\mathbf{m}'\right) \neq \gamma\left(\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)\right)$. By Lemma 3.2, $\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right) \subseteq \gamma\left(\mathbf{m}'\right)$ and there is no finite subset S of $\left(\mathbb{N} \cup \{\omega\}\right)^{|P|}$ s.t. $\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right) \subseteq \gamma\left(S\right) \subset \gamma\left(\mathbf{m}'\right)$.

Since $\gamma\left(\mathbf{m}'\right)$ is $\preccurlyeq$-downward-closed, we have: $\gamma\left(\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)\right) \subseteq \gamma\left(\mathbf{m}'\right)$. Thus, $\gamma\left(\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)\right) \neq \gamma\left(\mathbf{m}'\right)$ implies that $\gamma\left(\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)\right) \subset \gamma\left(\mathbf{m}'\right)$. By Lemma 3.1, there exists a finite subset S of $\left(\mathbb{N} \cup \{\omega\}\right)^{|P|}$ s.t. $\gamma\left(S\right) = \gamma\left(\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)\right)$. Hence, $S \subseteq \left(\mathbb{N} \cup \{\omega\}\right)^{|P|}$ is a finite set s.t.: $\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right) \subseteq \gamma\left(\mathsf{Post}\left(\gamma\left(\mathbf{m}\right), t_i\right)\right) \subseteq \gamma\left(S\right) \subset \gamma\left(\mathbf{m}'\right)$. Contradiction. □

Remark that a similar extension could be defined for EPN. We do not need it in the sequel, so we omit it here.

3.1.2 An adequate domain of limits for $\langle \mathsf{States}(\mathcal{C}), \precsim \rangle$

Let us show that it is also possible to represent any downward–closed set of configurations of a LCS $\mathcal{C}$ by a finite set of elements that are algorithmically manipulable. By definition of $\precsim$, a set of configurations S is $\precsim$–downward–closed iff for every configuration $\langle q, W\rangle$ in S, all the configurations of the form $\langle q, W'\rangle$, where $W'(f)$ is a subword of $W(f)$, for every channel F, are also in S. Thus, the main difficulty in representing downward–closed sets of configurations consists in devising a representation for *downward–closed languages* (with respect to word inclusion).

Such a finite representation has been devised by Abdulla, Bouajjani and Jonsson, and presented in [ABJ98] (most of the results we present in this section come from this work. Remark however that we sometimes use a slightly different terminology). In this work, they introduce the *simple regular expressions* (or sre for short), which are a special case of regular expressions, suitable to represent any downward–closed set of words. Thanks to the sre, and according to the definition of $\precsim$, it is then easy to define a domain of limits, whose elements represent downward-closed sets of configurations, simply by combining a location of the LCS with one sre per channel.

We recall the definition of sre, and their normal form. Then, we recall how to symbolically test entailment between two sre. Finally, we show how to define an *adequate domain of limits* for LCS, thanks to sre.

Simple regular expression In order to define sre, we first have to state the definition of $\precsim$-Downward-Closed Regular Expression. These expressions are special cases of *regular expressions* (in the classical sense, see [HMU01]). Hence, they will be used to define regular languages, that have the characteristic to be $\precsim$-downward-closed.

Definition 3.1 ([ABJ98]) (Downward-Closed Regular Expression)
Given a finite alphabet Σ, a downward-closed regular expression *(*dc–re *for short) on Σ is an expression of one of the following forms:*

- *either $(a_1 + \ldots + a_n)^*$ where $n \geq 1$, for any $1 \leq i \leq n$: $a_i \in \Sigma$, and for any i, j s.t. $1 \leq i \leq n$, $1 \leq j \leq n$: $i \neq j$ implies that $a_i \neq a_j$. Such an expression is called a $*$–*dc–re*;*
- *or $(a + \varepsilon)$ where a is a character of Σ. Such an expression is called an ε–*dc–re*;*
- *or ε.*

Given a dc–re *d, we denote by $\alpha(d)$ the alphabet of d. That is:*

$$\alpha(d) = \begin{cases} \emptyset & \text{if } d = \varepsilon \\ \{a\} & \text{if } d = (a+\varepsilon) \\ \{a_1, a_2, \ldots a_n\} & \text{if } d = (a_1 + a_2 + \cdots + a_n)^* \end{cases}$$

■

Remark that in [ABJ98], dc–re are called *atomic expressions*. Given an alphabet Σ, we denote by $L(\Sigma)$ the set of all dc–re on Σ. Based on this definition, we can introduce the *simple regular expressions*[4] (or sre for short), which are finite concatenations of dc–re:

Definition 3.2 ([ABJ98]) (Simple Regular Expression) *Given a finite alphabet Σ, a* simple regular expression *(*sre *for short) on Σ is either a* dc–re *on Σ, or an expression of the form $d_1 \cdot \ldots \cdot d_n$ where $n \geq 2$ and for any $1 \leq i \leq n$: $d_i \neq \varepsilon$ is a* dc–re *on Σ, or $\emptyset$.*

The size $|s|$ of a sre *s is defined as:*

$$|s| = \begin{cases} 0 & \textit{if } s = \varepsilon \textit{ or } s = \emptyset \\ n & \textit{if } s = d_1 \cdot d_2 \cdots d_n \textit{ with } n \geq 1 \end{cases}$$

■

As any regular expression, an sre defines a (regular) language, which is denoted by $[\![s]\!]$:

Definition 3.3 (Denotation of an sre) *Let s be an* sre*. Then, $[\![s]\!]$ is the language generated by s, defined inductively as follows:*

- $[\![\emptyset]\!] = \emptyset$*;*
- $[\![\varepsilon]\!] = \{\varepsilon\}$*;*
- *For any $a \in \Sigma$: $[\![(a + \varepsilon)]\!] = \{a, \varepsilon\}$*
- *For any $\{a_1, a_2, \ldots, a_n\} \subseteq \Sigma$:*

$$[\![(a_1 + a_2 + \cdots + a_n)^*]\!] \\ = \\ \big\{\varepsilon\big\} \cup \big\{w_1 w_2 \ldots w_k \mid k \in \mathbb{N}_0 \wedge \forall 1 \leq i \leq k : w_i \in \{a_1, a_2, \ldots a_n\}\big\}$$

- $[\![d \cdot s]\!] = [\![d]\!] \cdot [\![s]\!]$. ■

Given a set S of sre, $[\![S]\!]$ is the set $\cup_{s \in S} [\![s]\!]$.

In the sequel, we denote by $L(\Sigma)^*$ the set of any sre on the alphabet Σ. As a consequence of the definition, we obtain:

Lemma 3.3 ([ABJ98]) *For any* sre *s, $[\![s]\!]$ is a $\precsim$-downward-closed language.*

[4]In [ABJ98], such expressions are called *products* and sre are defined as disjunctions of such products. Since we do not need the disjunction, and in order to keep the forthcoming discussion simple, we restrict our definition of sre to the case where only one product is present in the disjunction.

On the other hand, these expressions are suitable to represent any downward–closed language, as stated by the following Theorem:

Theorem 3.2 ([ABJ98]) *Let Σ be a finite alphabet and let L be a language on Σ s.t. for any $w \in L$, for any $w' \in \mathsf{Subword}(w)$: $w' \in L$, i.e., L is downward–closed (wrt to word inclusion). Then, there exists a finite set of* sre *S on Σ s.t. $[\![S]\!] = L$.*

Remark that this Theorem does not guarantee that the (set of) sre representing L is unique. However, a notion of *normal form* exists for sre and is defined as follows:

Definition 3.4 (Normal form sre) *Any* dc–re *and $\emptyset$ are* normal form sre*. An* sre *$d_1 \cdot d_2 \cdots d_n$ is in* normal form *if and only if for each $1 \leq i < n$: $[\![d_i \cdot d_{i+1}]\!] \not\subseteq [\![d_i]\!]$ and $[\![d_i \cdot d_{i+1}]\!] \not\subseteq [\![d_{i+1}]\!]$.* ■

The following lemma states that for any sre, there exists a unique sre in normal form that has the same denotation, and which is easily computable:

Lemma 3.4 ([ABJ98]) *For any* sre *s there is an unique* sre *in normal form s' s.t. $[\![s]\!] = [\![s']\!]$. Furthermore, s' can be derived from s in linear time.*

Example 3.2 *Let us consider the four* sre*:*

1. $s_1 = (\mathtt{a})^* \cdot (\mathtt{a} + \varepsilon) \cdot (\mathtt{a} + \mathtt{b})^* \cdot (\mathtt{c} + \varepsilon)$
2. $s_2 = (\mathtt{a} + \varepsilon) \cdot (\mathtt{a} + \varepsilon) \cdot (\mathtt{a} + \mathtt{b})^* \cdot (\mathtt{b} + \varepsilon) \cdot (\mathtt{c} + \varepsilon)$
3. $s_3 = (\mathtt{a})^* \cdot (\mathtt{a} + \mathtt{b})^* \cdot (\mathtt{c} + \varepsilon)$
4. $s_4 = (\mathtt{a} + \mathtt{b})^* \cdot (\mathtt{c} + \varepsilon)$

It is not difficult to see that:

$$[\![s_1]\!] = [\![s_2]\!] = [\![s_3]\!] = [\![s_4]\!]$$
$$=$$
$$\Big\{ w \cdot \mathtt{c}, w \mid w = \varepsilon \vee \big(w = w_1 \cdots w_n \wedge \forall 1 \leq i \leq n = w_i \in \{\mathtt{a}, \mathtt{b}\} \big) \Big\}$$

However, s_4 only is in normal form. ♢

In the sequel, we use sre in order to represent (downward-closed) sets of contents of LCS channels. For that purpose, the empty language $\emptyset$ is not necessary: the empty channel will be represented by ε (hence, the only possible content of the channel is the empty word). As a consequence, we will not consider the empty sre $\emptyset$ anymore, from now on. Remark however that the algorithms and results we are about to present can be extended to the $\emptyset$ case (see for instance [ABJ98]).

Entailment An algorithm to test entailment between two sre is introduced in [ABJ98] too. The first step to devise such an algorithm consists in defining a procedure to compute entailment between dc–re. This procedure can be deduced from the following result (from [ABJ98]), that holds for any pair of dc–re d_1 and d_2.

Lemma 3.5 ([ABJ98]) *Let d_1 and d_2 be two* dc–re*. Then, the following holds:*

$$\begin{gathered} [\![d_1]\!] \subseteq [\![d_2]\!] \\ \textit{iff} \\ d_1 = \varepsilon \\ \textit{or} \\ d_2 \textit{ is a } *\text{–dc–re} \textit{ and } \alpha(d_1) \subseteq \alpha(d_2) \\ \textit{or} \\ d_1 \textit{ is an } \varepsilon\text{–dc–re} \textit{ and } d_2 = d_1 \end{gathered}$$

It is then easy to define an algorithm that tests the entailment between two dc–re. Algorithm 3.1 (from [ABJ98]) realises this (ignore the comment for the moment). Remark that it is a *symbolic* algorithm in the sense that it works by directly manipulating the sre s and s' instead of $[\![s]\!]$ and $[\![s']\!]$.

Algorithm 3.1: An algorithm to test entailment between two sre.

```
Data: Two sre $s = d_1 \cdot d_2 \cdots d_n$ and $s' = d'_1 \cdot d'_2 \cdots d'_k$
Result: true if $[\![s]\!] \subseteq [\![s']\!]$, false otherwise
Boolean TestEntailSre($d_1$, $d_2$)
begin
    $i \leftarrow 1$ ;
    $j \leftarrow 1$ ;
    while $i \leq n$ and $j \leq k$ do
        if $[\![d_i]\!] \subseteq [\![d'_j]\!]$ then
            // $\rho(i) \leftarrow j$ ;
1           $i \leftarrow i + 1$ ;
2           if $d'_j$ is an $\varepsilon$–dc–re then
                $j \leftarrow j + 1$ ;
        else
            $j \leftarrow j + 1$ ;
    if $i = n + 1$ then return(true) ;
    else return(false) ;
end
```

As a consequence we obtain the following lemma:

Lemma 3.6 ([ABJ98]) *Entailment between* sre *can be checked in linear time.*

If we consider Algorithm 3.1 with care, we remark that it establishes a correspondence between the dc–re of s and those of s': each dc–re d that makes up s is associated to a dc–re d' of s' s.t. $[\![d]\!] \subseteq [\![d']\!]$. As a consequence, $[\![s]\!] \subseteq [\![s']\!]$ iff there exists a function ρ that establishes this correspondence, as stated by the following Lemma:

Lemma 3.7 *For any pair of* sre $s = d_1 \cdot d_2 \cdots d_n$ *and* $s' = d'_1 \cdot d'_2 \cdots d'_k$*:* $[\![s]\!] \subseteq [\![s']\!]$ *iff there exists a non-decreasing function* $\rho : \{1 \ldots n\} \mapsto \{1, \ldots k\}$ *s.t.:*

1. *for any* $1 \leq i \leq n$*:* $[\![d_i]\!] \subseteq [\![d'_{\rho(i)}]\!]$*;*
2. *for any* $1 \leq i < n$*:* $\rho(i) = \rho(i+1)$ *implies that* $d'_{\rho(i)}$ *is a* $*$–dc–re*.*

Proof. Let us modify Algorithm 3.1 to let it build the function ρ. This is done by inserting an instruction that lets $\rho(i) = j$ *before* line 1. It is easy to see that this modified version of Algorithm 3.1 respects the following invariant:

$$\begin{gathered}\forall 1 \leq \ell \leq i-2 : \rho(\ell) \leq \rho(\ell+1) \\ \wedge \\ \forall 1 \leq \ell \leq i-1 : [\![d_\ell]\!] \subseteq [\![d'_{\rho(\ell)}]\!] \\ \wedge \\ \forall 1 \leq \ell \leq i-2 : \rho(\ell) = \rho(\ell+1) \text{ implies } d'_{\rho(\ell)} \text{ is a } *\text{–dc–re}\end{gathered}$$

In particular, the third point follows from the test of line 2.

We obtain the Lemma by combining the invariant and the fact that, at the end of the algorithm, $i = n+1$ iff $[\![s]\!] \subseteq [\![s']\!]$. □

We are thus able to test entailment between two sre. The entailment between an sre s and a set S of sre can be tested easily, according to the following lemma:

Lemma 3.8 ([ABJ98]) *Let* Σ *be a finite alphabet. Let* $s \in L(\Sigma)^*$ *be an* sre *on* Σ *and let* $S \subseteq L(\Sigma)^*$ *be a finite set of* sre*. Then* $[\![s]\!] \subseteq [\![S]\!]$ *iff there exists* $s' \in S$ *s.t.* $[\![s]\!] \subseteq [\![s']\!]$*.*

As a consequence, given two sets S and S' of sre, we have $[\![S]\!] \subseteq [\![S']\!]$ iff for any $s \in S$, there is $s' \in S'$ s.t. $[\![s]\!] \subseteq [\![s']\!]$.

Adequate domain of limits We are now ready to define a domain of limits for $\langle \mathsf{States}(\mathcal{C}), \precsim \rangle$, where $\mathcal{C} = \langle Q, q_0, F, \Sigma, T \rangle$ is any LCS. According to Definition 2.11, this boils down to defining a suitable set $\mathcal{L}(\Sigma, Q)$ of limit elements (containing a $\top$ element), together with an ordering ranging over $\mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$, and a function γ that associates a downward closed set to each element of $\mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$, such that any downward-closed set is finitely representable:

- The set of limits is the set $\mathcal{L}(\Sigma, Q) =$

  $$\{\langle q, E\rangle \mid q \in Q, E : F \mapsto L(\Sigma)^* \text{ assigns an } \mathsf{sre} \text{ to each channel}^5\} \cup \{\top\}$$

  We extend the notation $[\![\ ]\!]$ to elements in $\mathcal{L}(\Sigma, Q)$: for $\langle q, E\rangle \in \mathcal{L}(\Sigma, Q)$: $[\![\langle q, E\rangle]\!]$ denotes the set of pairs $\langle q, W\rangle \in \mathsf{States}(\mathcal{C})$ such that, for any $f \in F$: $W(f) \in [\![E(f)]\!]$.

- The function $\gamma : \mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q) \mapsto 2^{\mathsf{States}(\mathcal{C})}$ is such that

  1. for all $\langle q, W\rangle \in \mathsf{States}(\mathcal{C}) : \gamma(\langle q, W\rangle) = \{\langle q, W'\rangle \mid \langle q, W'\rangle \precsim \langle q, W\rangle\}$;
  2. $\gamma(\top) = \{\langle q, W\rangle \mid q \in Q, W(f) \in \Sigma^* \text{ for all } f \in F\}$;
  3. for all $\langle q, E\rangle \in \mathcal{L}(\Sigma, Q) \setminus \{\top\} : \gamma(\langle q, E\rangle) = [\![\langle q, E\rangle]\!]$.

- The ordering $\overline{\sqsubseteq} : (\mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)) \times (\mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q))$ is defined as follows: $c_1 \overline{\sqsubseteq} c_2$ if and only if $\gamma(c_1) \subseteq \gamma(c_2)$.

Remark that $\overline{\sqsubseteq}$ is a quasiorder, and not a partial order. Indeed, there can be several limit elements that represent the same downward-closed set (see Example 3.2 for instance).

Remark 3.1 *In the present case, the* $\top$ *element is equivalent to a finite set of elements of the form* $\langle q, E\rangle$ *s.t.* $q \in Q, E : F \mapsto L(\Sigma)^*$*. Indeed, assume we are considering an* LCS $\mathcal{C}$ *with alphabet* $\Sigma = \{a_1, a_2, \ldots, a_n\}$ *and set of states* Q*, and let* d_Σ *be the* sre $(a_1 + a_2 + \cdots + a_n)^*$*. Then, let*

$$S_\top = \{\langle q, E\rangle \mid q \in Q \text{ and } \forall f \in F : E(f) = d_\Sigma\}$$

It is not difficult to see that $\gamma(S_\top) = \mathsf{States}(\mathcal{C}) = \gamma(\top)$*. Moreover,* $S_\top$ *is finite because there are finitely many states in* Q*.*

With these definitions, we obtain an *adequate domain of limits* for $\langle \mathsf{States}(\mathcal{C}), \precsim\rangle$, as stated by the following theorem:

Theorem 3.3 $\left\langle \mathcal{L}(\Sigma, Q), \overline{\sqsubseteq}, \gamma \right\rangle$ *is an adequate domain of limits for* $\langle \mathsf{States}(\mathcal{C}), \precsim\rangle$.

Proof. We establish this result by proving the four points of Definition 2.11:

(L_1) It is easy to show that for any $\langle q, E\rangle \in \mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$, $\gamma(\langle q, E\rangle)$ is $\precsim$-downward-closed (see [ABJ98]);

(L_2) the element $\top$ is such that $\gamma(\top)$ is equal to $\mathsf{States}(\mathcal{C})$;

[5] We also require that E does not assign ε to all the channels because we require in Definition 2.11 that the set of limits is disjoint from $\mathsf{States}(\mathcal{C})$.

(L_3) by definition $c_1 \overline{\sqsubseteq} c_2$ if and only if $\gamma(c_1) \subseteq \gamma(c_2)$ for all $c_1, c_2 \in \mathsf{States}\,(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$;

(L_4) from Theorem 3.2 we deduce that if $S \subseteq \mathsf{States}\,(\mathcal{C})$ is $\precsim$-downward-closed, then there exists $S' \subseteq \mathsf{States}\,(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$ such that S' is finite and $\gamma(S') = S$.

□

Entailment between limit elements The definition of $\overline{\sqsubseteq}$, together with Algorithm 3.1, provides us a way to test, given two limit elements d_1 and d_2, whether $\gamma\,(d_1) \subseteq \gamma\,(d_2)$. It remains to show how to test, given a limit element d and a set of limit elements D, whether $\gamma\,(d) \subseteq \gamma\,(D)$. It turns out that the entailment holds iff there is $d' \in D$ s.t. $\gamma\,(d) \subseteq \gamma\,(d')$, as shown by the following Lemma:

Lemma 3.9 *Let $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ be a LCS and let $\left\langle \mathcal{L}(\Sigma, Q), \overline{\sqsubseteq}, \gamma \right\rangle$ be an adequate domain of limits for $\langle \mathsf{States}\,(\mathcal{C})\,, \precsim\rangle$. Let d be an element from $\mathcal{L}(\Sigma, Q) \setminus \{\top\}$ and let D be a finite subset of $\mathcal{L}(\Sigma, Q) \setminus \{\top\}$. Then, $\gamma\,(d) \subseteq \gamma\,(D)$ iff there exists $d' \in D$ s.t. $\gamma\,(d) \subseteq \gamma\,(d')$.*

Proof. In the case where there exists $d' \in D$ s.t. $\gamma\,(d) \subseteq \gamma\,(d')$, it is trivial that $\gamma\,(d) \subseteq \gamma\,(D)$.

The proof of the other direction is as follows. Let us suppose that $\gamma\,(d) \subseteq \gamma\,(D)$. Let us assume that $d = \langle q, E\rangle$ for some $q \in Q$, and let us assume, without loss of generality, that, for any $\langle q', E'\rangle \in D$: $q' = q$ (this is not restrictive, since the definition of $\precsim$ imposes that two configurations are comparable only if they are in the same state. Hence, $\gamma\,(\langle q', E'\rangle) \cap \gamma\,(\langle q, E\rangle) = \emptyset$ when $q' \neq q$). Let us finally assume that $F = \{f_1, f_2, \ldots, f_n\}$. We prove that there is $d' \in D'$ s.t. $\gamma\,(d) \subseteq \gamma\,(d')$ by induction on the number n of channels.

- **Base case:** $n = 1$. In that case, $\gamma\,(d) \subseteq \gamma\,(D)$ iff

$$[\![E(f_1)]\!] \subseteq \cup_{\langle q, E'\rangle \in D} [\![E'(f_1)]\!]$$

  by definition of γ. Remark that the expression on the left-hand side of the $\subseteq$ is a sre, and that the expression on the right-hand side is a finite set of sre. Hence, by Lemma 3.8, this holds iff there is $d' = \langle q, E'\rangle \in D$ s.t. $[\![E(f_1)]\!] \subseteq [\![E'(f_1)]\!]$. This means that $\gamma\,(d) \subseteq \gamma\,(d')$.

- **Inductive case** $n = k + 1$. For any $k \geq 1$, for any limit element $\langle q, E\rangle$ ranging on at least k channels, we let $\pi(\langle q, E\rangle\,, k) = \langle q, E'\rangle$ s.t. $E' : F^k \mapsto L(\Sigma)^*$ and for any $1 \leq i \leq k$: $E'(f_k) = E(f_k)$. That is, $\pi(\langle q, E\rangle\,, k)$ is a new limit element ranging on k channels, and whose sre correspond to those of $\langle q, E\rangle$ as far as the k first channels are concerned. We extend the function π to sets S of limit elements in the usual way: $\pi(S, k) = \{\pi(d', k) \mid d' \in S\}$.

Then, let us consider the set

$$D' = \left\{ d' \in D \mid \gamma(\pi(d,k)) \subseteq \gamma(\pi(d',k)) \right\}$$

Since $\gamma(d) \subseteq \gamma(D)$, we have $\gamma(\pi(d,k)) \subseteq \gamma(\pi(D,k))$. Hence, by induction hypothesis, there exists $d' \in \pi(D,k)$ s.t. $\gamma(\pi(d,k)) \subseteq \gamma(d')$. Otherwise stated, there is $d'' \in D$ s.t. $\gamma(\pi(d,k)) \subseteq \gamma(\pi(d'',k))$. Hence, the set D' is not empty. Remark that, by definition of D' and π, for any $d' = \langle q, E' \rangle \in D'$, for any $1 \leq i \leq k$: $[\![E(f_i)]\!] \subseteq [\![E'(f_i)]\!]$.

Let us show that $\gamma(d) \subseteq \gamma(D')$. This is proved *per absurdum*: we assume that $\gamma(d) \not\subseteq \gamma(D')$.

We first build two configurations c and c' as shown in the two following points:

1. Let $c = \langle q, W \rangle$ be a configuration in $\gamma(d) \backslash \gamma(D')$ (such a configuration has to exist since we have assumed that $\gamma(d) \not\subseteq \gamma(D')$). Since $c \notin \gamma(D')$, we have that, for any $d' = \langle q, E' \rangle \in D'$, $c \notin \gamma(d')$. Hence, for any $d' = \langle q, E' \rangle \in D'$, there exists $1 \leq i \leq k+1$ s.t. $W(f_i) \notin [\![E'(f_i)]\!]$. However, by definition of D', and since $c \in \gamma(D)$, for any $d' = \langle q, E' \rangle \in D'$, for any $1 \leq i \leq k$, $W(f_i) \in [\![E(f_i)]\!] \subseteq [\![E'(f_i)]\!]$. We conclude $W(f_{k+1}) \in [\![E(f_{k+1})]\!]$ (because $c \in \gamma(d)$), but that for any $d' = \langle q, E' \rangle \in D'$: $W(f_{k+1}) \notin [\![E'(f_{k+1})]\!]$.
2. By definition of D', we have: for any $d' \in D \setminus D'$: $\gamma(\pi(d,k)) \not\subseteq \gamma(\pi(d',k))$. Since we are dealing with limit elements ranging over k channels, the lemma holds, by induction hypothesis, and $\gamma(\pi(d,k)) \not\subseteq \gamma(\pi(D \setminus D',k))$. Let $c' = \langle q, W' \rangle$ be a configuration from $\gamma(\pi(d,k)) \setminus \gamma(\pi(D \setminus D',k))$.

From c and c', we build a new configuration $\overline{c} = \left\langle q, \overline{W} \right\rangle$, ranging over $k+1$ channels, as follows: $\overline{W}(f_{k+1}) = W(f_{k+1})$ and for any $1 \leq i \leq k$: $\overline{W}(f_i) = W'(f_i)$. By construction, $\overline{c} \notin \gamma(D')$ because the content of the channel f_{k+1} cannot be covered by any $d' \in D'$. Moreover, $\overline{c} \notin \gamma(D \setminus D')$, because $\pi(\overline{c},k) = c' \notin \pi((D \setminus D',k))$. Hence, $\overline{c} \notin \gamma(D)$. However, by construction again, $\overline{c} \in \gamma(d)$. Indeed, $\pi(\overline{c},k) = c'$ is in $\gamma(\pi(d,k))$, which means that for any $1 \leq i \leq k$, $\overline{W}(f_i) \in [\![E(f_i)]\!]$. Moreover, $\overline{W}(f_{k+1}) = W(f_{k+1}) \in [\![E(f_{k+1})]\!]$. Hence, the existence of $\overline{c} \in \gamma(d) \backslash \gamma(D)$ allows to conclude that $\gamma(d) \not\subseteq \gamma(D)$. Contradiction.

From the previous contradiction, we conclude that $\gamma(d) \subseteq \gamma(D')$. Hence, in particular, $[\![E(f_{k+1})]\!] \subseteq \cup_{\langle q,E' \rangle \in D'} [\![E'(f_{k+1})]\!]$. By Lemma 3.8, this implies that there exists $\langle q, E' \rangle \in D'$ s.t. $[\![E(f_{k+1})]\!] \subseteq [\![E'(f_{k+1})]\!]$. Moreover, since $\langle q, E' \rangle \in D'$, and by definition of D', we know that for any $1 \leq i \leq k$: $[\![E(f_i)]\!] \subseteq [\![E'(f_i)]\!]$. We conclude that for any $1 \leq i \leq k+1$: $[\![E(f_i)]\!] \subseteq [\![E'(f_i)]\!]$, which means that $\gamma(d) \subseteq \gamma(\langle q, E' \rangle)$, with $\langle q, E' \rangle \in D' \subseteq D$.

□

Entailment between sets of limits and configurations As a consequence of Lemma 3.9, if D_1 and D_2 are two finite subsets of $\mathcal{L}(\Sigma, Q) \setminus \{\top\}$, we have $\gamma(D_1) \subseteq \gamma(D_2)$ iff for any $d_1 \in D_1$, there is $d_2 \in D_2$ with $\gamma(d_1) \subseteq \gamma(d_2)$. Since, by Remark 3.1, the $\top$ element can be replaced by a finite subset of $\mathcal{L}(\Sigma, Q) \setminus \{\top\}$, one obtains a way to test whether $\gamma(D'_1) \subseteq \gamma(D'_2)$ for any finite subsets D'_1 and D'_2 of $\mathcal{L}(\Sigma, Q)$.

Finally, we handle elements of $\mathsf{States}(\mathcal{C})$, thanks to the following definition, that turns a configuration $c \in \mathsf{States}(\mathcal{C})$ into a limit element that has the same denotation:

Definition 3.5 (Limit corresponding to a configuration) *Let $c = \langle q, W\rangle$ be a configuration of an* LCS. *Then,* $\mathsf{limit}(c)$ *is the element $\langle q, E\rangle \in \mathcal{L}(\Sigma, Q)$ s.t.:*

$$\forall f \in F : E(f) = \begin{cases} (a_1 + \varepsilon) \cdot (a_2 + \varepsilon) \cdots (a_k + \varepsilon) & \text{if } W(f) = a_1 \cdot a_2 \cdots a_k \\ \varepsilon & \text{if } W(f) = \varepsilon \end{cases}$$

■

We extend the definition of limit to limit elements d: in that case, $\mathsf{limit}(d) = d$.

As a consequence, we have the following immediate Lemma (proof omitted):

Lemma 3.10 *For any $c \in \mathcal{L}(\Sigma, Q) \cup \mathsf{States}(\mathcal{C})$, $\gamma(c) = \gamma(\mathsf{limit}(c))$.*

Thanks to Lemma 3.10 and Remark 3.1, we conclude that given a finite subset D of $\mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$, we can build a finite subset D' of $\mathcal{L}(\Sigma, Q) \setminus \{\top\}$ s.t. $\gamma(D) = \gamma(D')$. Thus, given two finite subsets D_1 and D_2 of $\mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$, one can test whether $\gamma(D'_1) \subseteq \gamma(D'_2)$ by first building two finite subsets D'_1 and D'_2 of $\mathcal{L}(\Sigma, Q) \setminus \{\top\}$ corresponding to D_1 and D_2, and testing whether $\gamma(D_1) \subseteq \gamma(D_2)$, thanks to the aforementioned techniques.

Computation of Post Finally, it has been shown in [ABJ98] that Post is computable in a *symbolic way* on the elements of $\mathcal{L}(\Sigma, Q)$.

Lemma 3.11 ([ABJ98]) *Given an* LCS *$\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ with an adequate domain of limits $(\mathcal{L}(\Sigma, Q), \overline{\sqsubseteq}, \gamma)$, an element $c \in \mathcal{L}(\Sigma, Q) \cup \mathsf{States}(\mathcal{C})$, and a transition $t \in T$, it is possible to compute, in linear time (in the size*[6] *of c) an element $c' \in \mathcal{L}(\Sigma, Q) \cup \mathsf{States}(\mathcal{C})$ s.t. $[\![c']\!] = \{c_2 \mid \exists c_1 \in [\![c]\!] : c_1 \xrightarrow{t} c_2\}$.*

As a consequence (see also Remark 3.1) we can extend the function Post to elements of $\mathcal{L}(\Sigma, Q)$: for any $s \in \mathcal{L}(\Sigma, Q)$, $\mathsf{Post}(s)$ returns an element of $\mathcal{L}(\Sigma, Q)$ s.t. $[\![\mathsf{Post}(s)]\!] = \mathsf{Post}([\![s]\!])$.

[6]Naturally, the size of an element $c \in \mathsf{States}(\mathcal{C})$ is the sum of the lengths of the contents of the channels. The size of an element $c \in \mathcal{L}(\Sigma, Q)$, is the sum of the lengths of its sre.

3.2 The coverability Problem

This section recalls the state of the art regarding CPWsts, the *coverability problem* for WSTS. The algorithms to solve this problem are fixed point algorithms that can be classified into two categories, depending on the way the iterations are computed:

- *The forward approach*: Compute $\mathsf{Post}^*(c_0)$ and answer yes iff $U \cap \mathsf{Post}^*(c_0) \neq \emptyset$. This approach follows directly from the definition of CPWsts;
- *The backward approach*: Compute $\mathsf{Pre}^*(U)$ and answer yes iff $c_0 \in \mathsf{Pre}^*(U)$. This stems from the fact that $\mathsf{Pre}^*(U)$ is the set of all the configurations that can reach U in any number of steps of the transition relation.

As we will see in this section, CPWsts has been proved decidable in [ACJT96] thanks to a backward algorithm. As a consequence, we discuss backward algorithms before the forward ones. In section 3.2.1, we recall the general backward algorithm of [ACJT96]. Then, we recall a well-known *forward* algorithm, due to Karp and Miller, that decides coverability on the class PN (Section 3.2.4). That solution works by computing the *coverability set* of the PN, which is sufficient to decide the coverability problem (we discuss this set in Section 3.2.3). Unfortunately, that idea cannot be applied to other classes of WSTS: in Sections 3.2.5 and 3.2.6, we recall several forward *semi*-algorithms for the coverability problem on other classes of WSTS such as BP, LCS and TPN. All these algorithms, albeit quite efficient in practice, try to compute the coverability set of the system they analyse, and, therefore, offer no guarantee of termination.

3.2.1 A general backward algorithm for the CPWsts

Alain Finkel seems to be the first to have presented a solution to CPWsts for a rather general sub-class of WSTS in [Fin90]. However, at that time, the interest in WSTS was not as broad as it is today, and a solution to CPWsts, very similar to that of Finkel, has been independently published by Abdulla *et al* [ACJT96] some years later. This is the solution we are about to present now.

To be able to define effective procedures to solve the general coverability problem on WSTS, we will need to make several assumptions on the WSTS we manipulate. Indeed, the general definition of WSTS does not impose, for instance, that the WQO or the transition relation are decidable. Of course, if it is not the case, the WSTS will hardly be analysable by algorithmic procedures. Hence the following definition that states several constraints for a WSTS to be *effective*.

Definition 3.6 (EFFECTIVE WSTS) *An* effective well-structured transition system *(or* EWSTS *for short) is a tuple $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$ s.t.:*

- *$\mathcal{S}$ is a* WSTS*;*
- *given any pair of configurations c_1 and c_2 in C, one can decide whether $c_1 \Rightarrow c_2$ or not;*
- *given any pair of configurations c_1 and c_2 in C, one can decide whether $c_1 \overline{\leq} c_2$ or not;*
- *given any configuration $c \in C$, one can effectively compute (a finite representation of)* $\mathsf{PreUp}^{\overline{\leq}}(c)$. ■

The solution of [ACJT96] is a general one in the sense that it works for any EWSTS. Let us first explain the general ideas of this algorithm. Consider an EWSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$ and an upward-closed U. The algorithm of [ACJT96] is based on the *backward approach*: starting from U it iterates the operator $f(X) = \mathsf{Pre}(X) \cup U$ up to stabilisation. That is, it computes all the configurations that can reach U in any number of steps. It is important to remark that, since the systems we consider are *monotonic*, if some set S is upward-closed then $\mathsf{Pre}(S)$ is upward-closed too. Moreover, the union of two upward-closed sets is an upward-closed set. Thus, the algorithm always maintains an upward-closed set of configurations that can reach U.

In practice, since any upward-closed set U is potentially infinite, we represent it by means of $\mathsf{UGen}(U)$. As we manipulate WSTS, there always exists a finite a canonical set $\mathsf{UGen}(U)$ that represents U, by Corollary 2.1. This implies that we must be able to compute the Pre of an upward-closed set U, the union of two upward-closed sets U_1 and U_2, and the membership of a configuration c to an upward-closed set U in a symbolic fashion (by manipulating minimal elements of the upward-closed sets only). More precisely, given three upward–closed sets U, U_1 and U_2 respectively represented by finite generators G, G_1 and G_2, we need to devise:

- a $\mathsf{Pre}^{\#}$ operator s.t. $\mathsf{Pre}^{\#}(G)$ is a finite subset of $\mathsf{Pre}(U)$ with $\uparrow\!\left(\mathsf{Pre}^{\#}(G)\right) = \mathsf{Pre}(U)$. Thus, $\mathsf{Pre}^{\#}(G)$ must be a finite representation of $\mathsf{PreUp}^{\overline{\leq}}(G)$. Since G is finite, and since we consider effective WSTS, this operator exists, by Definition 3.6.
- a $\cup^{\#}$ operator s.t. $G_1 \cup^{\#} G_2$ is a finite subset of $U_1 \cup U_2$ s.t. $\uparrow\!\left(G_1 \cup^{\#} G_2\right) = U_1 \cup U_2$. We can simply let $\cup^{\#} = \cup$. It is possible to extract from $G_1 \cup^{\#} G_2$ a set $\mathsf{UGen}(U_1 \cup U_2)$ that is a finite and canonical generator of $U_1 \cup U_2$.
- an $\in^{\#}$ operator s.t. $c \in^{\#} G$ iff $c \in U$. By definition of Min, we can let $c \in^{\#} G$ iff there is $m \in G$ with $m \overline{\leq} c$.

The algorithm of [ACJT96] is presented in Algorithm 3.2. In this version of the algorithm, we have used the Min function in order to reduce the size of the sets we manipulate (this is correct by Lemma 2.5). This does not imply that we manipulate canonical sets, although this is feasible as we have seen in the above discussion (see also Lemma 2.6).

Algorithm 3.2: The algorithm of [ACJT96] to solve CPWsts.

```
Data: An EWSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ and a finite set $\mathsf{UGen}(U)$ of minimal
      elements of a $\overline{\leq}$-upward-closed set $U \subseteq C$.
Result: true if $\mathsf{Reach}(\mathcal{S}) \cap U \neq \emptyset$, false otherwise.
begin
    $F \leftarrow \mathsf{UGen}(U)$ ;
    $R \leftarrow \emptyset$ ;
    while there is $c \in F$ s.t. $c \notin^{\#} R$ do
        $R \leftarrow \mathsf{Min}^{\overline{\leq}}\left(R \cup^{\#} F\right)$ ;
        $F \leftarrow \mathsf{Min}^{\overline{\leq}}\left(\mathsf{Pre}^{\#}(F)\right)$ ;
    if $c_0 \in^{\#} R$ then return(true) ;
    else return(false) ;
end
```

Remark that one can obtain an on-the-fly algorithm by incorporating the test **if**$c_0 \in^{\#} R$ in the loop. The test of the **while** becomes: *$c_0 \notin^{\#} R$ and there is $c \in F$ s.t. $c \notin^{\#} R$.*

Theorem 3.4 ([ACJT96, FS01]) *For any* EWSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ *and any $\overline{\leq}$-upward-closed set $U \subseteq C$, Algorithm 3.2 always terminates. Moreover, it returns* `true` *iff the answer to* CPWsts *on $\mathcal{S}$ and U is positive.*

3.2.2 Practical improvements of the backward algorithm

Efficient techniques to obtain a practical implementation of the backward approach presented at Algorithm 3.2 have been studied by Delzanno, Raskin and Van Begin in [DR00, DRVB01, DRVB04], in the case of monotonic extensions of Petri nets. The two main techniques that they introduce in these papers are:

- The use of *invariants* to constrain the search and reduce the state space. An invariant of a PN $\mathcal{N}$ [Rei86] is a linear constraint on markings that provide an *over-approximation* of $\mathsf{Reach}(\mathcal{N})$. Hence, one can use invariants to *remove* from the frontier any marking $\mathbf{m}$ that does not satisfy all the invariants. The classical techniques to compute invariants of PN can be extended to monotonic extensions of Petri nets as shown in [STC96]. This optimisation is thus not restricted to the class PN.

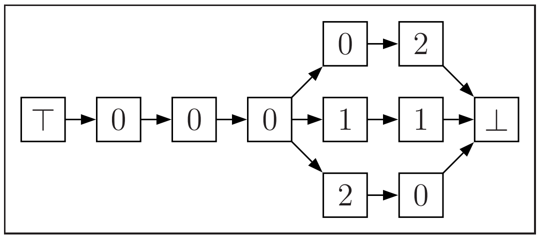


Figure 3.1: An example of CST that represents the upward-closed set U s.t. $\mathsf{UGen}\left(U\right) = \left\{\langle 0,0,0,0,2\rangle, \langle 0,0,0,1,1\rangle, \langle 0,0,0,2,0\rangle\right\}$

- The use of *an efficient symbolic data-structure* to store upward-closed sets of markings. This data structure is called *Covering Sharing Tree* [DR00] (CST for short) and is an extension of the Sharing Trees [Zam97, ZLC95]. The idea consists in representing an upward-closed set U by a finite set G_U that is a generator of U (not necessarily canonical). The markings $\mathbf{m} \in G_U$ are stored in a compact fashion: an acyclic graph having a top ($\top$) and a bottom ($\bot$) node is used. Each node v of the graph, different from $\top$ and $\bot$, is labelled by a natural number $\Lambda(v)$. Each path $\top, v_1, v_2, \ldots v_k, \bot$ from the top to the bottom node represents a marking $\langle \Lambda(v_1), \Lambda(v_2), \ldots, \Lambda(v_k)\rangle$. This allows to share the common prefixes and suffixes of the various markings in G_U. In the best cases, the size of the CST is logarithmic in the size of G_U. Figure 3.1 presents an example of CST. The CST on the Figure represents the upward-closed set $U = \{\mathbf{m} | \mathbf{m} \succcurlyeq \langle 0,0,0,0,2\rangle \vee \mathbf{m} \succcurlyeq \langle 0,0,0,1,1\rangle \vee \mathbf{m} \succcurlyeq \langle 0,0,0,2,0\rangle\}$.

  In [DRVB04], symbolic algorithms to manipulate CST are presented. By symbolic, we mean that the operations manipulate directly the CST instead of the set it represents. For instance, given two CST C_1 and C_2 representing respectively the sets U_1 and U_2, it is possible to compute a third CST that represents $U_1 \cup U_2$ by a direct manipulation of the structure of C_1 and C_2.

  CST can be used in Algorithm 3.2 to store the sets F and R, for instance.

The practical efficiency of these heuristics has been demonstrated on the class PN in [DRVB01], and on an extension of Petri nets, called multi-transfer nets[7] in [DRVB02]. In the latter paper, the authors analyse counting abstractions of multithread Java program and are able to analyse models with more than 40 places in a few seconds.

3.2.3 Forward algorithms for CPWsts

The design of a *forward algorithm* to decide CPWsts is harder than the backward case. Indeed, in a first attempt, one could think of an algorithm that is symmetrical to the backward one and computes the least fixed point of $f(X) = \mathsf{Post}\left(X\right) \cup \{\mathbf{m}_0\}$. This

[7]That class contains the class EPN.

would amount to compute the set of reachable states of the considered WSTS. However, this is not feasible since the set of reachable states is not computable in general for LCS [CFI96]. Even for PN, this task seems difficult because the set of reachable markings is not semilinear in general [HP79].

A more realistic approach to devise a *forward algorithm* to CPWsts consists in computing a *coverability set* of the WSTS. That set is a finite representation of the downward-closure of the reachable states (such a finite representation always exists when we have an adequate domain of limits at our disposal). It is well–known that the coverability set is sufficient to decide the coverability problem.

That line of research has been pursued for many years, since the introduction in 1969 by Karp&Miller of an algorithm that computes a *coverability set* for Petri nets (see Section 3.2.4). As we will see in Section 3.2.5 and Section 3.2.6, several attempts to extend the Karp & Miller algorithm to other classes of WSTS have been endeavoured. However, as far as we can tell, they have led to incomplete algorithms, i.e., without guarantee of termination. In chapter 4 of the present thesis, we introduce the first general *forward algorithm* to decide CPWsts.

In this section we recall the notion of *coverability set*. Sections 3.2.4, 3.2.5 and 3.2.6 present the `Karp&Miller` algorithm, as well as some of the semi-(forward) algorithms that have been presented in the literature.

Coverability set The notion of *coverability set* relies on the *covering set*:

Definition 3.7 (The Covering Set) *Let $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \preceq \right\rangle$ be a* WSTS*. The* covering set *of $\mathcal{S}$, noted* $\mathsf{Cover}\,(\mathcal{S})$*, is the set* $\downarrow\!(\mathsf{Reach}\,(S))$. ■

The following proposition states that the covering set is indeed suitable to decide the coverability problem.

Proposition 3.3 *For any* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \preceq \right\rangle$*, for any $\preceq$-upward-closed set $U \subseteq C$:* $\mathsf{Reach}\,(\mathcal{S}) \cap U = \emptyset$ *if and only if* $\mathsf{Cover}\,(\mathcal{S}) \cap U = \emptyset$*.*

Proof. We prove the two directions of the iff independently:

$\mathsf{Reach}\,(\mathcal{S}) \cap U \neq \emptyset \Rightarrow \mathsf{Cover}\,(\mathcal{S}) \cap U \neq \emptyset$**.** Stems from the fact that, by Definition 3.7, $\mathsf{Reach}\,(\mathcal{S}) \subseteq \mathsf{Cover}\,(\mathcal{S})$ for any WSTS $\mathcal{S}$.

$\mathsf{Cover}\,(\mathcal{S}) \cap U \neq \emptyset \Rightarrow \mathsf{Reach}\,(\mathcal{S}) \cap U \neq \emptyset$**.** Let c be a configuration in $\mathsf{Cover}\,(\mathcal{S}) \cap U$. By definition of $\mathsf{Cover}\,(\mathcal{S})$, there exists $c' \in \mathsf{Reach}\,(\mathcal{S})$ s.t. $c \preceq c'$. Since U is $\preceq$-upward-closed, c' is in U too. Hence $c' \in \mathsf{Reach}\,(\mathcal{S}) \cap U$ and thus, $\mathsf{Reach}\,(\mathcal{S}) \cap U \neq \emptyset$.

□

In general, $\mathsf{Cover}(\mathcal{S})$ might be infinite. Indeed, $\mathsf{Reach}(\mathcal{S})$ may be infinite in the case of WSTS and, by definition, $\mathsf{Reach}(\mathcal{S}) \subseteq \mathsf{Cover}(\mathcal{S})$ for any WSTS $\mathcal{S}$. However, by Definition 2.11, any downward-closed set of configurations can be finitely represented thanks to an adequate domain of limits. Such a finite representation is called a *coverability set*:

Definition 3.8 (Coverability Set) *Let $\mathcal{S} = \langle C, c_0, \Rightarrow, \preceq \rangle$ be a* WSTS *and let $\langle L, \sqsubseteq, \gamma \rangle$ be an adequate domain of limits to represent downward-closed sets of C. A finite set $\mathcal{CS} \subseteq L \cup C$ is called a* coverability set *of $\mathcal{S}$ iff $\gamma(\mathcal{CS}) = \mathsf{Cover}(\mathcal{S})$.* ■

Since L is an adequate domain of limits, there always exists a finite canonical set that respects Definition 3.8. Moreover, when it is *unique*, this set is called *the minimal coverability set* and is denoted by $\mathsf{CS}(\mathcal{S})$. By Lemma 3.1, it is the case for EPN and SMPN: their coverability sets can be finitely represented by an unique canonical set of ω-markings (hence called the *minimal coverability set*).

It is not difficult to see that the coverability problem on a WSTS $\mathcal{S}$ can be decided thanks to any coverability set of $\mathcal{S}$. Indeed, let $\langle S, \preceq \rangle$ be a well-quasi ordered set with adequate domain of limits $\langle L, \sqsubseteq, \gamma \rangle$, let $D \subseteq S$ be a $\preceq$-downward-closed set, and let $U \subseteq S$ be a $\preceq$-upward-closed set. Then $D \cap U \neq \emptyset$ iff there are $d \in \mathsf{DGen}(D)$ and $u \in \mathsf{UGen}(U)$ s.t. $u \sqsubseteq d$. Hence, given a WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \preceq \rangle$, a $\preceq$-upward-closed U represented by a finite generator G, and a coverability set $\mathcal{CS}$ of $\mathcal{S}$, one can effectively test whether $U \cap \mathsf{Cover}(\mathcal{S}) = \emptyset$ or not.

Let us now look into the Karp & Miller algorithm, that computes a coverability set for any PN.

3.2.4 The Karp&Miller algorithm

The Karp & Miller algorithm is a procedure to compute a coverability set of any PN. It relies on the computation of a *tree* whose nodes are labelled by ω-markings, and whose edges correspond to the firing of (sequences of) transitions. At the end of the algorithm, the labels of the tree's nodes form a (not necessarily minimal) coverability set of the Petri net . That algorithm also exploits acceleration techniques to avoid the computation of an infinite tree (remember that, in general, the number of reachable ω-markings of a Petri net is *infinite*).

Let us first define the notion of labelled tree as well as the acceleration function.

Definition 3.9 (Labelled Tree) *Given a set S, a S-labelled tree is a tuple $\mathcal{T} = \langle N, B, root, \Lambda \rangle$, where N is a set of nodes, $B \subseteq N \times N$ is a set of edges, $root \in N$ is the root node and $\Lambda : N \mapsto S$ is a labelling function of the nodes by elements of S. Given two nodes n and n' in N, we write respectively $B(n, n')$, $B^*(n, n')$ $B^+(n, n')$ instead of $(n, n') \in B$, $(n, n') \in B^*$, $(n, n') \in B^+$.*

Given two nodes n and n' s.t. $B(n, n')$, we say that n is the father *of n' and n' is a* child *of n. Given two nodes n and n' s.t. $B^*(n, n')$, we say that n is an* ancestor *of n' and n' is a descendant of n.*

The sets N and B respect the following conditions:

1. *Each node $n \in N$ different from root has one and only one father: $\forall n \in N \setminus \{root\}$: $|\{n' \mid B(n', n)\}| = 1$;*
2. *The node root has no father: $\nexists n \in N$ s.t. $B(n, root)$;*
3. *For every node $n \in N$, for every descendant n' of n: n' is not an ancestor of n: $\forall n, n' \in N$: $n \neq n' \wedge B^*(n, n')$ implies $\neg B^*(n', n)$.* ■

In the following, we will always consider $(\mathbb{N} \cup \{\omega\})^{|P|}$-labelled trees, that is, trees whose nodes are labelled by ω-markings ranging over the set of places P of the Petri net we deal with.

Let $\mathbf{m}_a$ and $\mathbf{m}$ be two ω-markings and σ be a sequence of transitions s.t. $\mathbf{m}_a \prec_e \mathbf{m}$ and $\mathbf{m}_a \xrightarrow{\sigma} \mathbf{m}$. Hence, the same sequence can be fired repeatedly from $\mathbf{m}$, by the monotonicity property of PN. Let P' denote the set of places that strictly increase along that sequence, i.e., $P' = \{p \mid \mathbf{m}_a(p) < \mathbf{m}(p)\}$. As the effect of a sequence of transitions is constant, the marking of places in P' can exceed any bound: for any place $p \in P'$, for any $k \geq 1$, there is a marking $\mathbf{m}'$ that is reachable in the PN, and such that $\mathbf{m}'(p) \geq k$. We can represent this by computing a ω-marking $\mathbf{m}_\omega$ s.t. for any $p \in P'$: $\mathbf{m}_\omega(p) = \omega$ and for any $p \notin P'$: $\mathbf{m}_\omega(p) = \mathbf{m}(p)$. This is called the *acceleration* of $\mathbf{m}$ w.r.t. $\mathbf{m}_a$. It can be generalised to any number of ancestor that are strictly smaller than $\mathbf{m}$, as presented in Algorithm 3.3.

The Accelerate function computes the *acceleration* of a marking $\mathbf{m}$ with respect to a finite set S of markings (these markings are assumed to label ancestors of the node labelled by $\mathbf{m}$). The result is a new ω-marking $\mathbf{m}_\omega$. Remark that in the case where no acceleration is possible, $\mathbf{m}_\omega = \mathbf{m}$. Remark also that this procedure is non-deterministic as we do not fix the order in which the ancestors have to be considered. However, the result of the function is unique whatever ordering is adopted.

We can now introduce the `Karp&Miller` algorithm. Given a PN whose initial ω-marking is $\mathbf{m}_0$, it builds a tree $\mathcal{T}$ as described in Algorithm 3.4. Roughly speaking, the `Karp&Miller` procedure can be seen as the computation of a reachability tree where accelerations are applied when computing the successors of each node, and branches are stopped as soon as they contain two different nodes with identical labels.

Initially, the tree consists of a single node $\mathbf{n}_0$ labelled by the initial marking $\mathbf{m}_0$ of the PN. The algorithm maintains a set *to_treat* of nodes waiting to be processed. Initially, *to_treat* contains $\mathbf{n}_0$ only. At each iteration of the main **while** loop, $\mathcal{T}$ is

Algorithm 3.3: The Accelerate function, *à la* Karp&Miller.

```
Data: A finite set $S$ of $\omega$-markings and an $\omega$-markings $\mathbf{m}$
Result: An $\omega$-marking, result of the acceleration of $\mathbf{m}$
Accelerate $(S, \mathbf{m})$
begin
    $\mathbf{m}_\omega \leftarrow \mathbf{m}$ ;
    foreach $\mathbf{m}' \in S$ s.t. $\mathbf{m}' \prec_e \mathbf{m}$ do
        foreach place $p$ s.t. $\mathbf{m}'(p) < \mathbf{m}(p)$ do
            $\mathbf{m}_\omega(p) \leftarrow \omega$ ;
    return($\mathbf{m}_\omega$) ;
end
```

potentially augmented. For that purpose, each node n in *to_treat* (which is guaranteed to be a leaf node) is treated. That treatment consists first in searching for an ancestor n' s.t. $\Lambda(n') = \Lambda(n)$. If such an ancestor exists, n doesn't have to be developed. Otherwise, all the successors of $\Lambda(n)$ by some PN transitions are computed and accelerated. The successors of n are created accordingly and added to *to_treat*. The algorithm terminates when *to_treat* is empty, i.e., as soon as there is no node to be treated left.

Algorithm 3.4: The Karp&Miller algorithm.

```
Data: A PN $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$
Result: A coverability set of $\mathcal{N}$
Karp&Miller
begin
    $\mathcal{T} \leftarrow \langle N, B, n_0, \Lambda \rangle$ where $N = \{n_0\}$, $B = \emptyset$ and $\Lambda(n_0) = \mathbf{m}_0$ ;
    $to\_treat \leftarrow \{n_0\}$ ;
    while $to\_treat \neq \emptyset$ do
        let $n$ be a node of $to\_treat$ ;
        $to\_treat \leftarrow to\_treat \setminus \{n\}$ ;
        if $\nexists \overline{n} : B^+(\overline{n}, n) \wedge \Lambda(\overline{n}) = \Lambda(n)$ then
            foreach $\mathbf{m} \in \mathsf{Post}(\Lambda(n))$ do
                $S \leftarrow \{\Lambda(n') \mid B^*(n', n)\}$ ;
                Let $n'$ be a new node s.t. $\Lambda(n') = \mathsf{Accelerate}(S, \mathbf{m})$ ;
                $N \leftarrow N \cup \{n'\}$ ;
                $B \leftarrow B \cup \{(n, n')\}$ ;
                $to\_treat \leftarrow to\_treat \cup \{n'\}$ ;
    return($\cup_{n \in N}\{\Lambda(n)\}$) ;
end
```

It is well-known that this algorithm terminates and is correct:

Theorem 3.5 ([KM69]) *For any* PN $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$*, the* `Karp&Miller` *procedure terminates. Upon termination,* $\{\Lambda(n) \,|\, n \in N\}$ *forms a coverability set of* $\mathcal{N}$*:*

$$\gamma\left(\{\Lambda(n) \,|\, n \in N\}\right) = \downarrow\!(\mathsf{Post}^*(\mathbf{m}_0))$$

Properties of the Karp&Miller tree The tree built by the `Karp& Miller` procedure has interesting properties that we will exploit mainly in Chapter 6. Let us discuss them now.

First, we adopt the following notation. Let $n \neq \mathit{root}$ be a node of some Karp&Miller tree. Hence, $\Lambda(n)$ has been obtained by calling $\mathsf{Accelerate}$ with parameters S and $\mathbf{m}$. In this case, we say that n has been obtained *by the acceleration of* $\mathbf{m}$ (with S). Remark that it might be the case that $\Lambda(n) = \mathbf{m}$. For any node $n \neq \mathit{root}$ of any Karp&Miller tree, we assume that the function $\mathsf{M}(n)$ returns the marking $\mathbf{m}$ s.t. n has been obtained by the acceleration of $\mathbf{m}$. Remark that, for any node $n \neq \mathit{root}$, $\mathsf{M}(n) \in \mathsf{Post}(n')$ where n' is the father of n.

The properties we are looking for can be found in the proof provided by Karp and Miller for their algorithm. It can be sketched as follows. Let $n \neq \mathit{root}$ be a node of the tree, and let us consider $\mathsf{M}(n)$. Clearly, there exists a (possibly empty) set of places that have been *accelerated* when building $\Lambda(n)$ from $\mathsf{M}(n)$. These are all the places p s.t. $\mathsf{M}(n)(p) \neq \omega$ and $\Lambda(n)(p) = \omega$. The proof of the algorithm works by exposing a sequence of transitions $\varsigma(n)$ associated to n that is (i) firable from $\mathbf{m}$, (ii) has a strictly positive effect on all the places that have been accelerated when building n and (iii) has a non-negative[8] effect on the other places that did not contain ω before the acceleration. Remark that we do not constrain the effect on the places that contained ω before the acceleration, because $\omega - c = \omega + c = \omega$ for any natural constant c. Thus, $\varsigma(n)$ can be seen as a witness for the correctness of the acceleration that has been performed when building n. In [KM69], a procedure to build such a sequence is given.

Let us state this more formally. The sequence $\varsigma(n)$ we are alluding to must satisfy the following definition. For any sequence of transitions σ, of a PN $\mathcal{N}$, and for any place p of $\mathcal{N}$ let $\sigma(p)$ denote the effect of σ on p.

Definition 3.10 (The $\varsigma(n)$ sequence) *Let* $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ *be a* PN *and let* $\mathcal{T} = \langle N, B, \mathit{root}, \Lambda \rangle$ *be its Karp&Miller tree. Then,* $\varsigma : N \mapsto T^*$ *is a function that associates a sequence of transitions to every node* n*, and is defined as follows:*

- *If* $n = \mathit{root}$*, then* $\varsigma(n)$ *returns the empty sequence.*
- *If there is no* $n' \in N$ *s.t.* $B^+(n', n)$ *and* $\Lambda(n') \preccurlyeq_e \Lambda(n)$ *(hence,* n *has not been obtained by an acceleration), then* $\varsigma(n)$ *returns the empty sequence.*

[8]This is a minimal condition to ensure that the acceleration is *sound*.

- *Otherwise, n has been obtained by the acceleration of* $\mathsf{M}(n)$. *Let* $P_a = \{p \in P \mid \Lambda(n)(p) = \omega$ *and* $\mathsf{M}(n)(p) \neq \omega\}$ *and let* $P_\omega = \{p \in P \mid \Lambda(n)(p) = \mathsf{M}(n)(p) = \omega\}$. *In that case,* $\varsigma(n)$ *returns one*[9] *of the finite non-empty sequences that respects:*

    1. *for any* $p \in P_a$: $\varsigma(n)(p) > 0$;
    2. *for any* $p \in P \setminus (P_a \cup P_\omega)$: $\varsigma(n)(p) \geq 0$;
    3. $\varsigma(n)$ *is firable from* $\mathsf{M}(n)$. ■

The existence of σ_n in the third case is guaranteed by the following lemma, that can be extracted from the main proof of the Algorithm 3.4, in [KM69]:

Lemma 3.12 ([KM69]) *Let* $\mathcal{N} = \langle P, T, \mathbf{m}_0\rangle$ *be a* PN *and let* $\mathcal{T} = \langle N, B, root, \Lambda\rangle$ *be its Karp&Miller tree. Let* $n \neq root$ *be a node of* $\mathcal{T}$. *Let* $P_a = \{p \in P \mid \Lambda(n)(p) = \omega$ *and* $\mathsf{M}(n)(p) \neq \omega\}$ *and* $P_\omega = \{p \in P \mid \Lambda(n)(p) = \mathsf{M}(n)(p) = \omega\}$. *Then, there exists a sequence of transitions* σ *of* $\mathcal{N}$ *s.t.:*

1. *for any* $p \in P_a$: $\sigma(p) > 0$;
2. *for any* $p \in P \setminus (P_a \cup P_\omega)$: $\sigma(p) \geq 0$;
3. σ *is firable from* $\mathsf{M}(n)$.

Remark 3.2 *The* `Karp&Miller` *procedure might compute a coverability set that is* not minimal*: upon termination, the set* $\{\Lambda(n) \mid n \in N\}$ *might contain two* ω*-markings* $\mathbf{m}_1$ *and* $\mathbf{m}_2$ *s.t.* $\mathbf{m}_1 \prec_e \mathbf{m}_2$. *However, this set is finite and it is thus easy to extract* $\preccurlyeq_e$*-maximal elements from it.*

The idea of keeping maximal elements only along the construction of the coverability tree to avoid to consider unnecessarily large intermediate trees, is at the basis of the minimal coverability tree algorithm *[Fin91] that we discuss in Chapter 6.*

3.2.5 An attempt to extend the Karp & Miller algorithm to broadcast protocols

In [EN98], Emerson and Namjoshi present a generalisation of the Karp & Miller procedure to the case of Ordered Labelled Transition Systems (OLTS for short). These models are basically WSTS extended with a labelling function of the transitions, as well as other information that are necessary to verify liveness properties (such as atomic propositions). Let us present a version of this procedure adapted to the case of BP (which form a special case of OLTS).

[9] Remark that there can be several sequences that satisfy these conditions. Since ς is a function it returns one and only one of these sequences for each node n.

The algorithm of [EN98] (henceforth referred to as the `EN` procedure) receives as input a WSTS $\mathcal{S}_{\mathcal{B}} = \langle C, c_0, \Rightarrow, \preccurlyeq \rangle$ that corresponds to a BP $\mathcal{B}$ and builds a graph $\langle V, E \rangle$ where $V \subseteq C$ is a set of vertices and $E \subseteq V \times V$ is a set of edges. At the end of the computation, V forms a coverability set of $\mathcal{S}_{\mathcal{B}}$, i.e., $\downarrow(V) = \downarrow(\mathsf{Reach}\,(\mathcal{S}_{\mathcal{B}}))$. The `EN` procedure is presented in Algorithm 3.5. Our presentation of the `EN` procedure is inspired from [EFM99].

The ideas at work in this algorithm are the same as in the Karp & Miller algorithm. Initially, the graph contains a single node c_0. The treatment of a node c consists in developing all its $\Rightarrow$-successors c'. In the case where there exists another node c'' in the graph s.t. $c'' \succcurlyeq_e c'$, an edge (c', c'') is added to the graph and c' is not further developed. If there exists a path from some node c_1 to c' with $c_1 \prec_e c'$, an acceleration[10] is applied, and the result of the acceleration is added to the graph and to the frontier. Otherwise, c' is added to the graph as a successor of c, and put into the frontier.

Remark that, since the `EN` procedure is based on the iteration of the Post operator, it is a *forward* procedure.

From [EN98], we have the following theorem:

Theorem 3.6 ([EN98]) *Let $\mathcal{B}$ be a* BP*, let $\mathcal{S}_{\mathcal{B}}$ be its associated* WSTS *and let $\langle V, E \rangle$ be the graph built by the* `EN` *procedure on $\mathcal{S}_{\mathcal{B}}$. Then, upon termination, V is a coverability set of $\mathcal{S}_{\mathcal{B}}$.*

Remark that the termination of Algorithm 3.5 is not proved in [EN98]. And indeed, one year later, Esparza, Finkel and Mayr have presented in [EFM99] an example of BP on which Algorithm 3.5 does not terminate. Hence, the `EN` procedure is only a semi-algorithm to decide the coverability problem on BP.

3.2.6 Other forward semi-algorithms

Several other works of the literature have presented *forward* algorithms to decide the coverability problem on various classes of WSTS. All these solutions are guaranteed to compute a coverability set of the considered WSTS when they terminate. However, they offer no guarantee of termination.

Timed Petri nets The team of Parosh Abdulla has presented in [ADMN04] a datastructure, called *Regions generators* to represent (possibly infinite) downward-closed sets of regions of timed Petri nets. They also introduce a method to accelerate transitions of timed Petri nets, i.e., to compute in one step the region generator r' that is obtained by the unbounded iteration of a given transition t from another region generator r. By means of this, they are able to analyse, in a forward fashion, non-trivial examples of timed Petri nets.

[10]The acceleration function is the same as in the Karp & Miller algorithm. Indeed, according to the semantics of BP, the configurations of $\mathcal{S}_{\mathcal{B}}$ are tuples on $\mathbb{N} \cup \{\omega\}$, i.e., ω-markings.

Algorithm 3.5: The `EN` procedure [EN98] to compute the coverability graph of Broadcast Protocols.

```
Data: A WSTS $\mathcal{S}_{\mathcal{B}} = \langle C, c_0, \Rightarrow, \preccurlyeq \rangle$ that corresponds to a BP $\mathcal{B}$
Result: A coverability set of $\mathcal{S}_{\mathcal{B}}$
begin
    $V \leftarrow \{c_0\}$ ;
    $E \leftarrow \emptyset$ ;
    $to\_treat \leftarrow \{c_0\}$;
    while $to\_treat \neq \emptyset$ do
        Choose and remove $c$ from $to\_treat$ ;
        foreach $c'$ s.t. $c \Rightarrow c'$ do
            if $\exists c'' \in V$ s.t. $c' \preccurlyeq_e c''$ then
                $E \leftarrow E \cup \{(c, c'')\}$ ;
            else if $\exists c_1, c_2, \ldots c_n \in V$ s.t. $c_1 \prec_e c'$ and $c_n = c$ and $\forall 1 \leq i < n$:
            $(c_i, c_{i+1}) \in E$ then
                $c_\omega \leftarrow$ Accelerate $(\{c_1, \ldots, c_n\}, c')$ ;
                $V \leftarrow V \cup \{c_\omega\}$ ;
                $E \leftarrow E \cup \{(c, c_\omega)\}$ ;
                $to\_treat \leftarrow to\_treat \cup \{c_\omega\}$;
            else
                $V \leftarrow V \cup \{c'\}$ ;
                $E \leftarrow E \cup \{(c, c')\}$ ;
                $to\_treat \leftarrow to\_treat \cup \{c'\}$;
    return($V$) ;
end
```

Lossy Channel Systems In [ABS01], Ahmed Bouajjani *et al* introduce the tool TReX. This tool is capable of performing the forward analysis of LCS, by accelerating some (sequences of) transitions. It uses the sre datastructure to represent the computed state space.

3.3 Decidability results for WSTS

In this section, we consider the problems that we have introduced in Section 3.3, and recall the decidability results that have been established in the literature regarding these problems. These results are summarised in Table 3.1.

3.3.1 Behavioural properties

Reachability Reachability is, in general, not decidable on WSTS: it is not decidable on PN+NBA nor on PN+T:

Theorem 3.7 ([RSVB03]) RPWSTS *is not decidable on the class* PN+NBA.

Theorem 3.8 ([DFS98]) RPWSTS *is not decidable on the class* PN+T.

However, it has been shown decidable on the class PN:

Theorem 3.9 ([May84]) RPWSTS *is decidable on the class* PN.

Quite surprisingly, it is also decidable on LCS. Indeed, it is equivalent to coverability because LCS are lossy, and coverability is decidable on LCS (see Section 3.2.1).

Proposition 3.4 RPWSTS *is decidable on the class* LCS.

Proof. Let $\mathcal{S}_\mathcal{C} = \left\langle C, c_0, \Rightarrow, \overline{\preceq} \right\rangle$ be the WSTS that corresponds to a given LCS $\mathcal{C}$, and let $c \in C$ be a configuration whom we want to test reachability. Let $U = \uparrow(\{c\})$ and let us show that $c \in \mathsf{Reach}\,(\mathcal{S}_\mathcal{C})$ iff $U \cap \mathsf{Reach}\,(\mathcal{S}_\mathcal{C}) \neq \emptyset$. Clearly, if $U \cap \mathsf{Reach}\,(\mathcal{S}_\mathcal{C}) = \emptyset$, then, in particular $c \notin \mathsf{Reach}\,(\mathcal{S}_\mathcal{C})$, because $c \in U$ ($\precsim$ is reflexive). On the other hand, if $U \cap \mathsf{Reach}\,(\mathcal{S}_\mathcal{C}) \neq \emptyset$, then, there exists $c' \in \mathsf{Reach}\,(\mathcal{S}_\mathcal{C})$ s.t. $c \precsim c'$. But this imply that $c \in \mathsf{Reach}\,(\mathcal{S}_\mathcal{C})$ by Lemma 2.15. □

Coverability We have seen in Section 3.2.1 that the coverability problem is decidable on the class EWSTS:

Theorem 3.10 ([ACJT96]) *The coverability problem is decidable for* EWSTS.

Since, on the other hand, reachability is not decidable in general on EWSTS, this provides a further motivation for the interest in the coverability problem.

Place-boundedness Place-boundedness is clearly decidable for the class PN since the minimal coverability set is computable for that class (thanks to the `Karp&Miller` procedure, for instance).

Theorem 3.11 PBEpn *is decidable when we consider* PN

The problem of place-boundedness for PN+T has been studied in [DFS98] by Dufourd, Finkel and Schnoebelen. One of the main results of the paper is that:

Theorem 3.12 ([DFS98]) PBEpn *is undecidable when we consider* PN+T*.*

Since every PN+R is a PN+T, we obtain:

Corollary 3.1 PBEpn *is undecidable when we consider* PN+R*.*

A similar result has been proved by Raskin and Van Begin:

Theorem 3.13 ([RVB04]) PBEpn *is undecidable when we consider* PN+NBA*.*

Coverability sets are not computable in general It is not difficult to see that, given a coverability set $\mathcal{CS}$ of an EPN $\mathcal{N}$, place p is unbounded in $\mathcal{N}$ iff there exists $\mathbf{m} \in \mathcal{CS}$ s.t. $\mathbf{m}(p) = \omega$. Hence, one cannot compute a coverability set for any class of WSTS on which PBEpn is not decidable:

Corollary 3.2 *There is no algorithm that computes a coverability set for the classes* PN+R*,* PN+T *and* PN+NBA*.*

One can show that, given a PN+T $\mathcal{N}$, it is possible to build a BP $\mathcal{B}$ that simulates $\mathcal{N}$, in the sense that a coverability set of $\mathcal{N}$ can be deduced from one of $\mathcal{B}$. Hence, a procedure to compute a coverability set of BP would provide us with a way to compute a coverability set of PN+T. Since the latter is not computable (Corollary 3.2), such a procedure cannot exist:

Corollary 3.3 *There is no algorithm that, given a* BP $\mathcal{B}$*, computes a coverability set of* $\mathcal{B}$*.*

As far as LCS are concerned, it has been shown in [CFI96], that the reachability set $\mathsf{Reach}(\mathcal{C})$ is not computable in general. However, because of the lossiness of LCS (Lemma 2.15): for any LCS $\mathcal{C}$, $\mathsf{Reach}(\mathcal{C}) = \mathsf{Cover}(\mathcal{C})$. Hence, a coverability set is not computable in general for LCS.

Theorem 3.14 ([CFI96]) *There is no algorithm that, given a* LCS $\mathcal{C}$*, computes a coverability set of* $\mathcal{C}$*.*

Corollary 3.3 and Theorem 3.14 explain why the algorithms to decide the coverability problems on BP (presented in [EN98]) and on LCS (in [ABJ98]) are semi-algorithms. Indeed, upon termination, these algorithms are guaranteed to compute a coverability set. Hence, they are necessarily incomplete.

The unbounded computation problem That problem is decidable by a very simple algorithm that can be sketched as follows. The algorithm [ACJT96] assumes that the WSTS is *recursively finitely branching*, i.e., that for any $c \in C$, the set $\mathsf{Post}(c)$ is computable and finite. The algorithm consists in unfolding the transition relation of $\mathcal{S}$ under the form of a tree rooted in c_0, and to stop the construction and answer 'yes' as soon as two configurations c and c' s.t. $c \in \mathsf{Post}^*(c_0)$, $c' \in \mathsf{Post}^*(c)$ and $c \overline{\leq} c'$ are met. If no such pair of configurations exists, all the executions of $\mathcal{S}$ are necessarily finite, by WQO property, and the construction of the tree eventually terminates. In that case, the algorithm answers 'no'.

Theorem 3.15 ([ACJT96]) UCWsts *is decidable for recursively finitely branching* WSTS*.*

Boundedness Remark that the decidability of PBEpn implies the decidability of BoundEpn. Indeed, for any EPN $\mathcal{N}$, $\mathsf{Reach}(\mathcal{N})$ is finite iff every place p of $\mathcal{N}$ is bounded. On the other hand, the decidability of BoundEpn does *not* imply the decidability of PBEpn: if some EPN $\mathcal{N}$ is bounded, then, each of its place is bounded. However, for $\mathsf{Reach}(\mathcal{N})$ to be infinite, it is sufficient that *one* place be unbounded. Hence, it might be the case that $\mathsf{Reach}(\mathcal{N})$ is infinite whereas some place p is bounded.

Unfortunately, place-boundedness is decidable in the case of PN (Theorem 3.11) but not in the case of PN+NBA (Theorem 3.13), PN+T (Theorem 3.12) and PN+R (Corollary 3.1). Thus, the above reasoning allows us to obtain the decidability of PBEpn for the class PN only. Remark that the decidability of boundedness can also be deduced from the fact that one can compute a coverability set of any PN. Indeed, a PN $\mathcal{N}$ is bounded iff for any $\mathbf{m} \in \mathsf{CS}(\mathcal{N})$, for any place p: $\mathbf{m}(p) \neq \omega$.

Theorem 3.16 BoundEpn *is decidable when we consider* PN*.*

As far as PN+T are concerned, the following results are to be found in [DFS98]:

Theorem 3.17 ([DFS98]) BoundEpn *is decidable when we consider* PN+T*.*

Since every PN+R is a PN+T, we obtain:

Corollary 3.4 BoundEpn *is decidable when we consider* PN+R*.*

Remark 3.3 *It is important to remark that, by Definition 2.22, any* PN+R *is a* PN+T*, hence the previous corollary. However, another extension of Petri nets by means of special arcs that reset the content of certain places has been routinely studied in the literature. It is the class of* Petri nets with reset arcs *[Cia94, DFS98].*

A Petri net with reset arcs is a PN *augmented with special transitions that, when fired, empty the content of a given place (see [Cia94] for the exact definition). Thus, the only difference between a Petri net with reset arcs and a* PN+R*, is that, in the case of a* PN+R*, the tokens that are taken away from the places that are reset do not disappear from the net. They are rather transferred to a* trash can *place p_{Tr} from which they can never escape.*

That difference is important, because, its has been proved in [DFS98] that boundedness becomes undecidable on this class. This might sound intriguing to the reader, who might think that Corollary 3.4 is in contradiction with the result of [DFS98]. Let us explain why it is not the case. Let $\mathcal{N}$ be a Petri net with reset arcs, and let us transform it into a PN+R *$\mathcal{N}'$ by adding to $\mathcal{N}$ a trashcan place p_{Tr}, and by replacing each reset arc of $\mathcal{N}$ by an extended transition whose source place is the place that the transition resets, and the destination is p_{Tr}. We obtain a* PN+R*. Since* BoundEpn *is decidable on that class, we can decide the boundedness of $\mathcal{N}'$. Unfortunately, this does not allow us to conclude whether $\mathcal{N}$ is bounded or not.*

Indeed, since every place of $\mathcal{N}$ is a place of $\mathcal{N}'$, the boundedness of $\mathcal{N}'$ implies that $\mathcal{N}$ is bounded. On the other hand, if $\mathcal{N}'$ is unbounded, it might be the case that p_{Tr} is the only unbounded place. Since PBEpn *is undecidable on* PN+R *and* PN+T*, we cannot determine whether it is the case or not. Hence, we cannot deduce anything about the boundedness of $\mathcal{N}$.*

Thus, the main difference between PN+R *and Petri nets with reset arcs is to be found in the fact that a 'reset' (actually a transfer) in the case of* PN+R *does* not modify *the total amount of tokens in the net, whereas a reset in a Petri net with reset arcs can decrease the total number of tokens of the net by an arbitrarily large amount of tokens.*

The case of PN+NBA has been addressed in [RVB04]:

Theorem 3.18 ([RVB04]) BoundEpn *is decidable when we consider* PN+NBA*.*

3.3.2 Expressiveness properties

Emptiness and Universality To the best of our knowledge, these problems have not yet been addressed in the literature for the general case of WSTS.

LTL The problem that asks whether the ω-language of a WSTS satisfies some formula of action-based LTL has been shown decidable by Esparza on the class PN [Esp94].

It is not decidable in general for WSTS, because it is not decidable on $\mathsf{PN{+}NBA}$, for instance [RSVB03].

Theorem 3.19 ([Esp94]) LTLSatis *is decidable on the class* PN.

Theorem 3.20 ([RSVB03]) LTLSatis *is undecidable on the class* $\mathsf{PN{+}NBA}$.

3.4 Expressive powers

The expressive power of WSTS in the general case has seldom been studied in the literature. All the results we are aware of concern the expressive powers of PN, $\mathsf{PN{+}T}$ and $\mathsf{PN{+}R}$, mostly in the finite words case, with a finite set of accepting states.

3.4.1 Results on PN

An important source of results regarding the expressive power of PN in the finite words case is [Pet81]. A first general result states some relationships that exist between the class $L^L(\mathsf{PN})$, $L^P(\mathsf{PN})$, $L^T(\mathsf{PN})$ and $L^G(\mathsf{PN})$ (see Definition 2.43):

Theorem 3.21 ([Pet81]) $L^T(\mathsf{PN}) = L^L(\mathsf{PN}) \supseteq L^G(\mathsf{PN}) \supseteq L^P(\mathsf{PN})$.

In the case where ε-labelled transitions are disallowed, we have:

Theorem 3.22 ([Pet81]) $L^T_{\not\varepsilon}(\mathsf{PN}) \subseteq L^L_{\not\varepsilon}(\mathsf{PN}) \supseteq L^G_{\not\varepsilon}(\mathsf{PN}) \supseteq L^P_{\not\varepsilon}(\mathsf{PN})$.

Remark that $L^T(\mathsf{PN}) = L^L(\mathsf{PN})$ and $L^G(\mathsf{PN}) \supseteq L^P(\mathsf{PN})$ follow from the definitions. The proof of $L^L(\mathsf{PN}) \supseteq L^G(\mathsf{PN})$ requires an easy construction that can be found in [Pet81].

Then, several closure properties are also stated, regarding the class $L^L(\mathsf{PN})$:

Theorem 3.23 ([Pet81]) $L^L(\mathsf{PN})$ *is closed under concatenation, union, intersection and homomorphism.*

$L^L(\mathsf{PN})$ *is not closed under iteration.*

Hence, $L^L(\mathsf{PN})$ is not a full AFL.

The membership of certain languages to various classes of PN language has also been studied in [Pet81]:

1. $\mathcal{L} = \{\mathtt{a}^n\mathtt{b}^n | n \geq 1\}$ is in $L^L(\mathsf{PN})$.

	Problem							
Model	RPWSTS	CPWSTS	UCWSTS	PBEPN	cov. set.	BOUNDEPN	QLEPN	LTLSATIS
EWSTS	U	D	D	irr.	NC	irr.	irr.	U
PN	D	D	D	D	C	D	D	D
PN+NBA	U	D	D	U	NC	D	D	U
PN+T	U	D	D	U	NC	D	D	?
PN+R	U	D	D	U	NC	D	D	?
LCS	D	D	D	irr.	NC	irr.	irr.	U

Table 3.1: Summary of the decidability results for various classes of WSTS. U = undecidable; D = decidable; C = computable; NC = not computable; irr. = irrelevant for this class of systems; ? = no result found in the literature.

2. The language of all palindromes on some alphabet Σ, $\mathcal{L}^R = \{w \cdot w^R \mid w \in \Sigma^*\}$ is not in $L^L(\mathsf{PN})$. As a consequence, it is not in $L^G(\mathsf{PN})$, $L^T(\mathsf{PN})$ nor $L^P(\mathsf{PN})$.

   The argument to show that $\mathcal{L}^R \not\in L^L(\mathsf{PN})$ is quite involved and can be summarised as follows. Let us assume that there exists a PN $\mathcal{N}$ and a finite set S s.t. $L(\mathcal{N}, S) = \mathcal{L}^R$. Then, since the number of places of $\mathcal{N}$ is finite, it is possible to find two words $x_1 \cdot x_1^R$ and $x_2 \cdot x_2^R$ in $\mathcal{L}^R$ that are *long enough* in the sense that the marking $\mathbf{m}_1$ reached after reading x_1 and the marking $\mathbf{m}_2$ reached after reading x_2 are equal. Hence, $\mathcal{N}$ accepts $x_1 \cdot x_2^R$ too. The proof of existence of these two words is quite involved. In Section 8.3.1, we exhibit a proof that $\mathcal{L}^R \not\in L^G(\mathsf{PN})$ which is much simpler thanks to the pumping Lemmata that we introduce in Chapter 8.

3. $\mathcal{L}_{cs} = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n | n \geq 1\}$ is in $L^L(\mathsf{PN})$.

These results are used to deduce the relationship between $L^L(\mathsf{PN})$ and some well-studied classes of the Chomsky hierarchy [HMU01]. It is not difficult to see that any non-deterministic finite automaton can be translated into a Petri net with the same language. However, it is well-known that $\mathcal{L}$ is not regular. Hence:

Theorem 3.24 ([Pet81]) *$\mathcal{R} \subset L^L(\mathsf{PN})$ where $\mathcal{R}$ is the class of regular languages.*

Since $\mathcal{L}$ and $\mathcal{L}^R$ are both context–free, we also have:

Theorem 3.25 ([Pet81]) *$L^L(\mathsf{PN})$ is incomparable to the class of context-free languages.*

Finally:

Theorem 3.26 ([Pet81]) *Any language in $L^L(\mathsf{PN})$ is context-sensitive.*

Remark that the class $L^G(\mathsf{PN})$ has not been studied as much as $L^L(\mathsf{PN})$. In [Pet81], Peterson limits himself to the study of the class $L^L(\mathsf{PN})$, and remarks that:

> [$L^L(\mathsf{PN})$] has been investigated in the literature [...] Some results have been obtained [...] for the prefix languages[11] [...] The [classes $L^G(\mathsf{PN})$ and $L^T(\mathsf{PN})$] have been defined but no work has been done on their development.

[11]That is, the class $L^P(\mathsf{PN})$.

3.4.2 Results on $\mathsf{PN+T}$ and $\mathsf{PN+R}$

As far as these Petri nets extensions are concerned, the main source of results is [Cia94], which provides a survey of the literature and completes the previous results when necessary. A first result holds for any kind of accepting sets, and follows from the definitions of PN, $\mathsf{PN+T}$ and $\mathsf{PN+R}$:

Theorem 3.27 ([Cia94]) *For any $X \in \{L, G, P, T\}$, we have:*

$$L^X(\mathsf{PN}) \subseteq L^X(\mathsf{PN+R}) \subseteq L^X(\mathsf{PN+T}) \subseteq \text{R.E.}$$

and

$$L^X_{\not\varepsilon}(\mathsf{PN}) \subseteq L^X_{\not\varepsilon}(\mathsf{PN+R}) \subseteq L^X_{\not\varepsilon}(\mathsf{PN+T}) \subseteq \text{R.E.}$$

Then, Ciardo focuses on the classes with L-type sets of accepting markings too:

Theorem 3.28 ([Cia94]) *The following holds:*

$$L^L(\mathsf{PN}) \subset L^R(\mathsf{PN}) = L^T(\mathsf{PN}) = \text{R.E.}$$

and

$$L^L_{\not\varepsilon}(\mathsf{PN}) \subset L^L_{\not\varepsilon}(\mathsf{PN+R}) \subseteq L^L_{\not\varepsilon}(\mathsf{PN+T}) \subset \text{R.E.}$$

Remark that it is not known whether there exists a language L that can be accepted by a $\mathsf{PN+T}$ without ε-transitions but not by a $\mathsf{PN+R}$ without ε-transitions. Ciardo conjectures that it is the case. We come back on this conjecture in Section 8.3.6.

3.4.3 Results on WSTS

The expressive power of WSTS has, to the best of our knowledge, not yet been studied *per se*. From the definition of classes $L^L(\mathsf{WSTS})$, $L^G(\mathsf{WSTS})$, $L^T(\mathsf{WSTS})$ and $L^P(\mathsf{WSTS})$, we have trivially:

Proposition 3.5

$$\begin{array}{rcl} L^T(\mathsf{WSTS}) & \subseteq & L^L(\mathsf{WSTS}) \\ L^P(\mathsf{WSTS}) & \subseteq & L^G(\mathsf{WSTS}) \\ L^T_{\not\varepsilon}(\mathsf{WSTS}) & \subseteq & L^L_{\not\varepsilon}(\mathsf{WSTS}) \\ L^P_{\not\varepsilon}(\mathsf{WSTS}) & \subseteq & L^G_{\not\varepsilon}(\mathsf{WSTS}) \end{array}$$

By the definition 3.6 of *effective* WSTS, there exists, for any EWSTS $\mathcal{S}$, a Turing machine $M_{\mathcal{S}}$ that computes the successors of any configuration of the WSTS. This also implies that there exists an encoding of the configurations of the WSTS in terms of a word that one can store on the ribbon of a Turing machine. Thus, the language of any EWSTS can be recognised by a Turing Machine (that uses $M_{\mathcal{S}}$ as a sub-procedure). Hence, $L^L(\mathsf{EWSTS}) \subseteq$ R.E.. Thus, we can deduce the following result from Theorem 3.28 and Proposition 3.5:

Proposition 3.6 $L^L(\mathsf{EWSTS}) = L^T(\mathsf{EWSTS}) =$ R.E.

Since many problems are undecidable on the class R.E., this result is a strong indication that other accepting conditions should be considered to obtain positive decidability results. This will be done in Chapter 8, where we will consider in particular the class $L^G(\mathsf{WSTS})$, and show that many interesting results can be obtained on it.

3.5 Discussion

It should now be clear that WSTS are interesting models of computations:

1. WSTS subsume the classes PN, strongly monotonic SMPN, LCS and TPN, that have been widely studied in the literature, and that have proved to be useful in the modelling of real computer systems (the interest of BP is more theoretical).

2. Adequate domains of limits have been identified[12] for PN, SMPN and LCS. This means that their upward– and downward–closed sets of configurations are algorithmically manipulable.

3. The coverability problem, to which many *safety properties* can be reduced, is *decidable* on the class EWSTS.

4. In the particular case of PN, the coverability set is computable, thanks to the Karp&Miller algorithm. This allows to decide other problems than the coverability problem.

5. WSTS in general, and EPN in particular, can be used to represent finite words languages or ω-languages, that are interesting in practice to represent traces of execution of computer systems.

On the other hand, we are now able to identify several noteworthy problems:

1. Although a general *backward* algorithm exists, no general *forward* algorithm is known to solve the coverability on the general class of WSTS. Such an algorithm would be interesting, both from the theoretical and from the practical point of view (as shown in [HKQ03] – see the introductory discussion of Chapter 4).

[12]We have not discussed an adequate domain of limits for TPN. It exists however. See [ADMN04], for instance.

2. The Karp&Miller algorithm is known to be rather inefficient in practice (see [Fin91], and the introduction of Chapter 6).

3. The expressive power of WSTS has seldom been addressed in the literature. The expressiveness of the classes PN, PN+NBA, PN+T and PN+R has been studied very sparsely when upward–closed sets of accepting markings are considered. Moreover, in this case, the relationship between the respective expressive powers of these classes of EPN are not clearly established.

We partly close these problems in the sequel of this thesis. In Chapter 4 and Chapter 5, we present a general forward algorithm to solve the coverability problem on WSTS. In Chapter 6, we discuss several improvements to the Karp&Miller procedure that have been proposed in the literature [Fin91, Lut95], and introduce a new solution to compute the coverability set of PN. In Chapter 7, we study the expressiveness of EPN in terms of ω-words. Finally, in Chapter 8, we consider the expressive power of WSTS, and more particularly EPN, on finite words.

Part I

Coverability properties

Chapter 4

Expand, Enlarge and Check

The present chapter introduces the *Expand, Enlarge and Check* algorithmic schema (EEC for short), a new solution to the coverability problem of WSTS. The need for a solution to CPWsts that is different from that of [ACJT96] (see Algorithm 3.2 in Section 3.2.1) can be motivated by the following observation, quoted from the introduction of the paper [HKQ03] by Henzinger, Kupferman and Qadeer:

> Forward state traversal has several obvious advantages over backward state traversal. First, for operational system models, successor states are often easier to compute than predecessor states. Second, only the reachable part of the state space is traversed. Third, optimisations such as on-the-fly [GPVW95] and partial-order [Pel94, God96] methods can be incorporated naturally.

As a way to emphasise this observation, the authors also report on experimental results that confirm the superiority of *forward exploration* on the backward one, from a practical point of view. The advantage of forward traversal in practice has also been observed in many other works such as [ADMN04, ABS01, FRSB02]. When they terminate, the forward semi-algorithms presented in these papers are usually more efficient than their backward counterpart. Thus, an algorithm that decides the coverability problem by relying on the Post operator only would be interesting, both from a theoretical and a practical point of view.

Unfortunately, the state of the art is quite remote from this situation, as far as CPWsts is concerned. The only general algorithm to decide CPWsts is that of [ACJT96], and relies on the Pre operator (it is a so-called *backward algorithm*). In the notable case of Petri nets, the `Karp&Miller` procedure is a *forward algorithm* that computes the coverability set, and allows thus to decide the coverability problem. Regrettably, as we have seen in Section 3.2.3, the idea of computing the coverability set to decide the coverability problem does not scale up to other classes of WSTS, because this set is, in general, not computable.

In this chapter, we introduce the first general forward algorithm to decide CPWsts (under some reasonable effectiveness requirements and assuming that there exists an adequate domain of limits[1]). Unlike its predecessors, this solution gives up the idea of computing the coverability set. It rather relies on the idea of *approximating* the WSTS.

In order to decide CPWsts, we claim that two types of approximations are necessary:

- In the case where the upward-closed set U is **reachable** in the WSTS $\mathcal{S}$, there exists a *finite execution* of $\mathcal{S}$ that ends up in U. That finite execution can be regarded as a very simple transition system that forms an *under-approximation* of $\mathcal{S}$, and is a finite witness of the reachability of U in $\mathcal{S}$. Remark that any under-approximation of $\mathcal{S}$ whose set of executions contains at least the execution alluded to is also suitable to prove that U is reachable.

- In the case where the upward-closed set U is **not reachable** in the WSTS $\mathcal{S}$, then, we know by Proposition 3.3 that $\mathsf{Cover}(\mathcal{S})$ is sufficient to prove this. However, any other downward-closed set O s.t. $\mathsf{Cover}(\mathcal{S}) \subseteq O$ and $O \cap U = \emptyset$ is also sufficient. Such a set O forms an *over-approximation* of (the covering set of) $\mathcal{S}$. Moreover, O is finitely representable and effectively manipulable, provided that we have an adequate domain of limits at our disposal, and under our effectiveness requirements (see Section 4.1).

We conclude that there always exists a finite object to prove that U is reachable or not in $\mathcal{S}$.

The EEC algorithm is essentially a procedure that *enumerates* a sequence of pairs of approximations (an under– and an over–approximation). At each step of the enumeration, the following tests are made:

1. is U reachable in the under-approximation ? If yes, U is reachable in $\mathcal{S}$.

2. is U unreachable in the over-approximation ? If yes, $\mathcal{S}$ can't reach U.

The sequence of approximations computed by EEC has the property that one of these tests is eventually true.

It remains to devise a procedure to compute these sequences of approximations. Given a WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\preceq} \right\rangle$ with adequate domain of limits $\langle L, \sqsubseteq, \gamma \rangle$, the computation of the under– and over–approximations relies on two functions: $\mathsf{Under}(\mathcal{S}, C')$ and $\mathsf{Over}(\mathcal{S}, C', L')$. The former receives $\mathcal{S}$ and a finite set $C' \subseteq C$ and returns a transition system whose behaviours are those of $\mathcal{S}$ when we restrict ourselves to the configurations in C'. The latter receives $\mathcal{S}$, a finite set $C' \subseteq C$ and a finite set $L' \subseteq L$,

[1]A recent work [GRVB06a] provides a way to automatically devise such an adequate domain of limits.

and returns an over-approximation of $\mathcal{S}$ whose behaviours are those of $\mathcal{S}$ as long as they stay inside the configurations of C', while the reachable configurations that do not belong to C' are over-approximated by elements of L'. Thus, in practice, one simply has to enumerate a sequence $C_0, C_1, \ldots, C_i, \ldots$ of finite sets of configurations and a sequence $L_0, L_1, \ldots, L_i$ of finite sets of limits, as sketched on Figure 4.1. How these sets are enumerated depends on the type of WSTS that is analysed. In Chapter 5, we apply EEC to the case of strongly monotonic SMPN and of LCS, and show that a simple prototype outperforms previous tools to decide CPWSTS on these models.

This chapter is organised as follows. In section 4.1, we recall some preliminaries such as And-Or graphs (that we use to represent the over-approximations). In Section 4.2, we explain how the under– and over–approximations are computed. Finally, in Section 4.3, we present the EEC algorithmic schema.

The content of this chapter is based on the articles [GRVB04] and [GRVB06b].

4.1 Preliminaries

This section states some preliminary definitions that will be used throughout the chapter. More precisely, we explain which WSTS our algorithm can handle, by providing *effectiveness requirements* on WSTS. We then define the notion of *And-Or graph*, a special kind of bipartite graph that we will use to *over-approximate* WSTS. We also define the associated *avoidability problem* on And-Or graph.

4.1.1 Effectiveness requirements

Let us first state several conditions that a WSTS has to satisfy to ensure the effectiveness and termination of the algorithms we are about to present.

Definition 4.1 (EFFECTIVE WSTS AND DOMAIN OF LIMITS)
A WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ *and an adequate domain of limits* $\langle L, \sqsubseteq, \gamma \rangle$ *are* effective *if the following conditions are satisfied:*

(E_1) *C and L are recursively enumerable;*

(E_2) *for any $c_1, c_2 \in C$, we can decide whether $c_1 \Rightarrow c_2$;*

(E_3) *for any $d \in L \cup C$ and for any finite subset $D \subseteq L \cup C$, we can decide whether* $\mathsf{Post}\,(\gamma(d)) \subseteq \gamma(D)$*;*

(E_4) *For any finite subsets $D_1, D_2 \subseteq L \cup C$, we can decide whether $\gamma(D_1) \subseteq \gamma(D_2)$.* ■

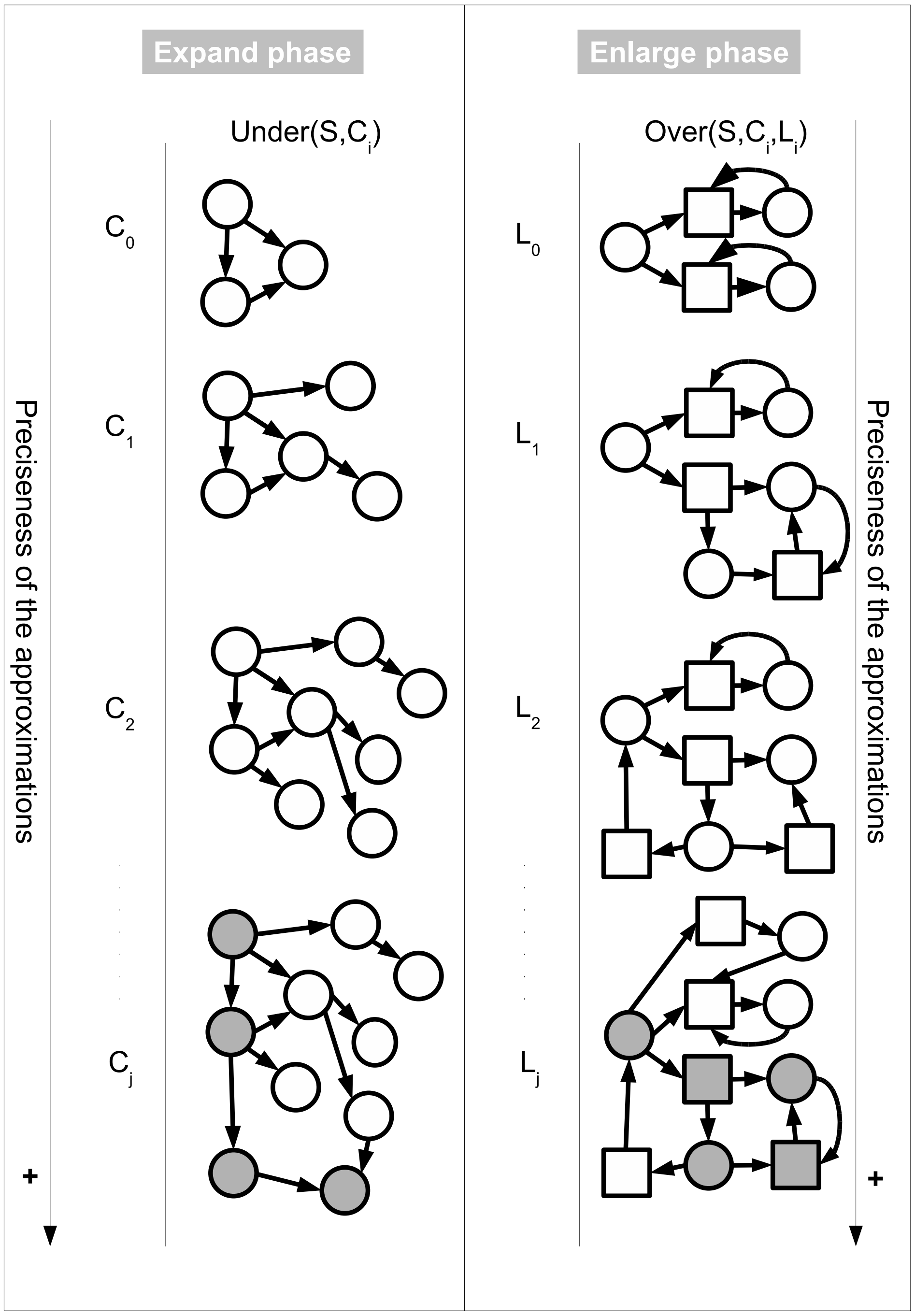


Figure 4.1: A graphical sketch of the EEC algorithmic schema. This figure illustrates the construction of the successive under- and over-approximations (during the *Expand* and *Enlarge* phases respectively). The *Check* takes place after each pair of approximation has been built and consists in testing whether there exists (*i*) an execution (nodes in grey in the last under-approximation) of the under-approximation that reaches U and (*ii*) an unfolding (nodes in grey in the last over-approximation) of the over-approximation that avoids U.

Remark that the domains proposed in the literature to handle forward analysis of WSTS, respect these conditions. For instance, in Section 3.1, we have introduced the domains of *ω- markings* and *simple regular expressions* to handle extensions of Petri nets and LCS respectively. From classical results of the literature [ABJ98] and from the results of Section 3.1, it is easy to deduce that conditions (E_1) through (E_4) hold on these two domains.

Finally, in the present chapter we request that the WSTS be *deadlock-free* (see Definition 2.15). That property will be assumed in the proofs of the EEC algorithmic schema. Remark that this is not restrictive in practice since one can always turn any WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$ in a deadlock-free WSTS $\mathcal{S}' = \left\langle C, c_0, \Rightarrow', \overline{\leq} \right\rangle$ that has the same reachability properties, by letting $\Rightarrow' = \Rightarrow \cup \{(c, c) \mid c \in C\}$. It is not difficult to see that $\mathsf{Reach}(\mathcal{S}) = \mathsf{Reach}(\mathcal{S}')$.

4.1.2 And-Or graphs and unfoldings

As stated before, the definition of *And-Or graphs* is central to our analysis of WSTS. An And-Or graph is a bipartite graph whose set of nodes is divided into so-called *Or* and *And* nodes. Such graphs have been routinely used in numerous applications such as the modelling of turn-based games for instance [AHK02] (an Or node represents the possible choices of the protagonist, which are thus controllable; and an And node represents the possible uncontrollable moves of the antagonist). In our setting, And-Or graphs will be used in a slightly different way. This is the definition:

Definition 4.2 (And-Or Graph) *An* And-Or graph *is a tuple $G = \langle V_A, V_O, v_i, \rightarrowtail \rangle$ where $V = V_A \cup V_O$ is the finite set of nodes (V_A is the set of 'And' nodes and V_O is the set of 'Or' nodes), $V_A \cap V_O = \emptyset$, $v_i \in V_O$ is the initial node, and $\rightarrowtail \subseteq (V_A \times V_O) \cup (V_O \times V_A)$ is the transition relation, such that for any $v \in V$, there exists $v' \in V$ with $(v, v') \in \rightarrowtail$.* ■

We adopt the following convention for the graphical representation of And-Or graphs: And nodes are represented by square nodes and Or-nodes are represented by round nodes.

A classical notion in the analysis of And-Or graphs is that of *unfolding*, which intuitively corresponds to the notion of path in a plain graph. When considering a plain graph that compactly represents the whole transition relation of a system, a path is simply an unfolding of the transition relation, that is, a possible execution of the system. Such a path represents the choices for the next action that has been taken in each node. In the case of *And-Or graphs*, all the successors of each And-node have to be taken into account. Hence, the structure we obtain by *unfolding* an And-Or graph is actually a tree (labelled by nodes of the And-Or graph), where each Or-node has exactly one child, and each And-node has as many children as it has successors in the And-Or graph. This is stated more precisely in the next definition:

Definition 4.3 (Compatible Unfolding) *A* compatible unfolding *of an And-Or graph $G = \langle V_A, V_O, v_i, \rightarrowtail\rangle$ is an infinite labelled tree $T_G = \langle N, root, B, \Lambda\rangle$ where: (i) N is the set of nodes of T_G, (ii) $root \in N$ is the root of T_G, (iii) $B \subseteq N \times N$ is the transition relation of T_G, (iv) $\Lambda : N \mapsto V_A \cup V_O$ is the labelling function of the nodes of T_G by nodes of G. Λ respects the three following compatibility conditions (Λ is extended to sets of nodes in the usual way):*

(C_1) $\Lambda(root) = v_i$;

(C_2) *for all $n \in N$ such that $\Lambda(n) \in V_A$, we have that:*

1. *for all nodes $v' \in V_O$ such that $\Lambda(n) \rightarrowtail v'$, there exists one and only one $n' \in N$ such that $B(n, n')$ and $\Lambda(n') = v'$, and ;*
2. *for all nodes $n' \in N$ such that $B(n, n')$, there exists $v' \in V_O$ such that $\Lambda(n) \rightarrowtail v'$ and $\Lambda(n') = v'$.*

(C_3) *for all $n \in N$ such that $\Lambda(n) \in V_O$, there exists one and only one $n' \in N$ such that $B(n, n')$, and $\Lambda(n) \rightarrowtail \Lambda(n')$.* ■

Since we see an unfolding as a possible evolution of a system represented by the And-Or graph, it is natural to define the *avoidability problem* in this setting. Quite naturally, one can avoid a set E in an And-Or graph G iff there exists an unfolding of G that does not intersect with E:

Problem 10 (AOGAvoid: The avoidability problem for And-Or graphs)

- **Instance:** *An And-Or graph $G = \langle V_A, V_O, v_i, \rightarrowtail\rangle$ and a set E.*
- **Question:** *Does there exist $T = \langle N, root, \Lambda, B\rangle$, a compatible unfolding of G, such that $\Lambda(N) \cap E = \emptyset$?* ■

When the answer is positive, we say that E is *avoidable* in G. Otherwise, we say that it is *unavoidable*.

The following result states that AOGAvoid is an easily decidable problem:

Theorem 4.1 ([Imm81]) AOGAvoid *is complete for PTIME.*

We are now ready to explain how to exploit these definitions to build *under* and *over-approximations* of a WSTS.

4.2 Under and Over-approximations

In the present section, we define two kinds of (parametrised) approximations of WSTS that will be used by 'Expand, Enlarge and Check'.

We first explain, in section 4.2.1, how to build an *under-approximation* of a given WSTS w.r.t. to a finite subset of reachable states $C' \subseteq C$. Intuitively, that approximation contains all the initialised computations of the WSTS that visit states of C' only. It allows us to decide the *positive instances* of the coverability problem since an upward-closed set of configurations is reachable in a WSTS $\mathcal{S}$ iff there exists a finite execution of $\mathcal{S}$ that reaches it.

In section 4.2.2, we show how to build an *over-approximation* of a given WSTS, w.r.t. a given finite set of reachable states $C' \subseteq C$ and a given finite set of limit elements $L' \subseteq L$. These abstractions are *And-Or graphs* whose nodes are annotated by $\overline{\leq}$-downward-closed sets of states of a WSTS. We show that any unfolding of this And-Or graph is able to *simulate* [Mil89] the behaviours of its associated WSTS (Proposition 4.3). Moreover, if the $\leq$-downward-closed sets that are used to annotate the And-Or graph are *precise enough* (in a sense that we make clear in Proposition 4.5), then the And-Or graph allows us to decide *negative instances* of the coverability problem.

4.2.1 The C'-Exact Partial Reachability Graph $\mathsf{Under}\,(\mathcal{S}, C')$

Given a WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \leq\rangle$ and a finite set $C' \subseteq C$ (with $c_0 \in C'$), let us show how to build the *C'-exact partial reachability graph* (C'-EPRG for short) $\mathsf{Under}\,(\mathcal{S}, C')$. It is an under-approximation of $\mathcal{S}$ (in the sense of Proposition 4.1). Let us first define precisely the notion of C'-EPRG:

Definition 4.4 (The Exact Partial Reachability Graph) *Given a* WSTS $\mathcal{S} = \big\langle C, c_0, \Rightarrow, \overline{\leq}\big\rangle$ *and a set* $C' \subseteq C$ *with* $c_0 \in C'$*, the* C'-EPRG *of* $\mathcal{S}$ *is the transition system* $\mathsf{Under}\,(\mathcal{S}, C') = \big\langle C', c_0, \big(\Rightarrow \cap (C' \times C')\big)\big\rangle$. ■

The following propositions state the usefulness of the C'-EPRG to decide the coverability problem. The first one states that these graphs are *adequate* in the sense that when a set U of configurations[2] is reachable in the C'-EPRG, it is also reachable in the corresponding WSTS.

Proposition 4.1 (Adequacy) *Given a* WSTS $\mathcal{S} = \big\langle C, c_0, \Rightarrow, \overline{\leq}\big\rangle$*, a finite set* $C' \subseteq C$ *with* $c_0 \in C'$ *and a set* $U \subseteq C$*:* ***If*** $\mathsf{Reach}\,(\mathsf{Under}\,(\mathcal{S}, C')) \cap U \neq \emptyset$ ***then*** $\mathsf{Reach}\,(\mathcal{S}) \cap U \neq \emptyset$.

Proof. If $\mathsf{Reach}\,(\mathsf{Under}\,(\mathcal{S}, C')) \cap U \neq \emptyset$, then there exists a finite execution $c_0, c_1, \ldots c_n$ of $\mathsf{Under}\,(\mathcal{S}, C')$ s.t. $c_n \in U$. However, by definition of $\mathsf{Under}\,(\mathcal{S}, C')$, that execution is also an execution of $\mathcal{S}$. Hence, u is reachable in $\mathcal{S}$. □

[2]The proposition holds for any subset U of C, whether it is $\overline{\leq}$-upward-closed or not.

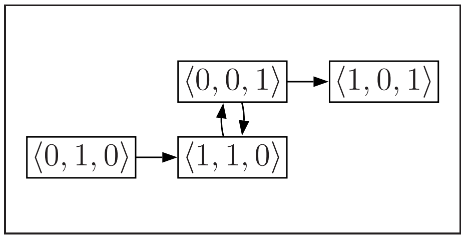


Figure 4.2: $\mathsf{Under}\left(\mathcal{S}_{\mathcal{N}_\mu}, C'\right)$ for $C' = \{\mathbf{m} \mid \forall p \in \{p_1, p_2, p_3\} : \mathbf{m}(p) \leq 1\}$.

The second proposition states the *completeness* of C'-EPRG for some sets $C' \subseteq C$: when a given set[3] U is actually reachable in a WSTS, there exists a set $C' \subseteq C$ that allows to prove the reachability of U thanks to the C'-EPRG.

Proposition 4.2 (Completeness) *Given a* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\preceq} \right\rangle$ *and a set* $U \subseteq C$*: **If*** $\mathsf{Reach}\left(\mathcal{S}\right) \cap U \neq \emptyset$ ***then*** *there exists a finite set* $C' \subseteq C$ *with* $c_0 \in C'$ *such that* $\mathsf{Reach}\left(\mathsf{Under}\left(\mathcal{S}, C'\right)\right) \cap U \neq \emptyset$.

Proof. Since U is reachable in $\mathcal{S}$ there exists a finite execution $c_0, c_1, \ldots c_n$ of $\mathcal{S}$ s.t. $c_n \in U$. Let $C' = \{c_0, \ldots, c_n\}$. Then, $c_0, c_1, \ldots c_n$ is also an execution of $\mathsf{Under}\left(\mathcal{S}, C'\right)$, and U is reachable in $\mathsf{Under}\left(\mathcal{S}, C'\right)$ (note that $c_0 \in C'$, by construction). □

Example 4.1 *Let us consider the Petri net* $\mathcal{N}_\mu$ *of Figure 2.1. Let* $\mathcal{S}_{\mathcal{N}_\mu}$ *be the* WSTS *that corresponds to* $\mathcal{N}_\mu$*. Let* $C' = \{\mathbf{m} \mid \forall p \in \{p_1, p_2, p_3\} : \mathbf{m}(p) \leq 1\}$*. Then,* $\mathsf{Under}\left(\mathcal{S}_{\mathcal{N}_\mu}, C'\right)$ *is the transition system depicted in Figure 4.2.*

Let $U_1 = \{\mathbf{m} \mid \mathbf{m}(p_3) \geq 1\}$ *and* $U_2 = \{\mathbf{m} \mid \mathbf{m}(p_1) \geq 5\}$*. Remark that* U_1 *and* U_2 *are both reachable in* $\mathcal{S}_{\mathcal{N}_\mu}$*, but that* U_1 *only is reachable in* $\mathsf{Under}\left(\mathcal{S}_{\mathcal{N}_\mu}, C'\right)$*.*

Clearly, $\mathcal{S}_{\mathcal{N}_\mu}$ *is an under-approximation of* $\mathcal{S}_{\mathcal{N}_\mu}$*. As a matter of fact, it corresponds to* $\mathcal{N}_\mu$ *restricted to a single process (see Example 2.11 for the explanation of the model).*

4.2.2 The And-Or Graph $\mathsf{Over}\left(\mathcal{S}, C', L'\right)$

Let us now show how to over-approximate a WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\preceq} \right\rangle$ by means of an And-Or graph. Just as the EPRG was parametrised by a finite set of *concrete* elements, this over-approximation relies upon $C' \subseteq C$, a finite set of concrete elements; and L', a finite set of limit elements. It has the form of an And-Or graph $\mathsf{Over}\left(\mathcal{S}, C', L'\right)$ whose unfoldings all simulate $\mathcal{S}$ (as shown later, in Proposition 4.3). Intuitively, $\mathsf{Over}\left(\mathcal{S}, C', L'\right)$ will represent the exact behaviours of $\mathcal{S}$ as long as these behaviours stay inside the set C'. When an execution of $\mathcal{S}$ reaches a state outside C', this execution is not precisely represented in $\mathsf{Over}\left(\mathcal{S}, C', L'\right)$ anymore. However,

[3]The proposition holds for any subset U of C, whether it is $\overline{\preceq}$-upward-closed or not.

all the configurations outside C' that will be reached during such an execution will be over-approximated (covered) by elements of L'.

The key observation that motivates the use of And-Or graphs in our algorithm relies on the fact that, when we consider a configuration c, s.t. $\mathsf{Post}(c) \not\subseteq C'$ (hence $\mathsf{Post}(c)$ must be over-approximated), there might be several subsets elements $E_1, E_2, \ldots E_n$ of $L' \cup C'$ that over-approximate $\mathsf{Post}(c)$ (that is, with $\gamma(\mathsf{Post}(c)) \subseteq \gamma(E_i)$) and that are incomparable: for any $i \neq j$, $\gamma(E_i) \not\subseteq \gamma(E_j)$. Hence, it is not possible to choose the *most precise* over-approximation of $\mathsf{Post}(c)$. We will represent that situation by an Or-node labelled by c, and whose successors are labelled by all the possible over-approximations $E_1, E_2, \ldots E_n$ of $\mathsf{Post}(c)$. All these over-approximations have to be taken into account because some of them might be too coarse to prove that the upward-closed set is not reachable, while some others might be precise enough (see Section 4.3.1 for further discussion on this topic). Remark also that for certain classes of WSTS, we are ensured that there will always be a *single* most precise (downward-closed) over-approximation to represent any subset of C'. We discuss this particular case in the sequel. Let us now state the precise definition of $\mathsf{Over}(\mathcal{S}, C', L')$:

Definition 4.5 (The over–approximation And-Or Graph) *Given a* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$*, an adequate domain of limits* $\langle L, \sqsubseteq, \gamma \rangle$ *for* $\langle C, \overline{\leq} \rangle$*, a finite subset* $C' \subseteq C$ *with* $c_0 \in C'$*, and a finite subset* $L' \subseteq L$ *with* $\top \in L'$*, the And-Or graph* $G = \langle V_A, V_O, v_i, \rightarrowtail \rangle$*, noted* $\mathsf{Over}(\mathcal{S}, C', L')$*, is defined as follows:*

(A_1) $V_O = L' \cup C'$;

(A_2) *And-nodes are non empty subsets of* $L' \cup C'$ *which contain* $\sqsubseteq$*-incomparable elements only:* $V_A = \{S \in 2^{L' \cup C'} \setminus \{\emptyset\} \mid \nexists d_1 \neq d_2 \in S : d_1 \sqsubseteq d_2\}$;

(A_3) $v_i = c_0$;

($\mathsf{A}_{4.1}$) *The successors of any And-node are Or nodes:* $(n_1, n_2) \in \rightarrowtail$ *with* $n_1 \in V_A, n_2 \in V_O$ *if and only if* $n_2 \in n_1$;

($\mathsf{A}_{4.2}$) *The successors of an Or-node* n *are all the most precise elements of* $L' \cup C'$ *that represent the set of successors of* $\gamma(n)$*, i.e., for any* $n_1 \in V_O, n_2 \in V_A$: $(n_1, n_2) \in \rightarrowtail$ *if and only if*

1. successor covering: $\mathsf{Post}(\gamma(n_1)) \subseteq \gamma(n_2)$ *and*
2. preciseness: $\neg \exists n \in V_A : \mathsf{Post}(\gamma(n_1)) \subseteq \gamma(n) \subset \gamma(n_2)$. ■

Notice that all the nodes of $\mathsf{Over}(\mathcal{S}, C', L')$ have at least one successor. Indeed, for all $n \in V_A$, since $n \neq \emptyset$ (following point $\mathsf{A}_{4.1}$ and point A_2 of Definition 4.5), n has at least one successor. Since, by point A_2 of Definition 4.5, And-nodes are subsets of $L' \cup C'$ that do not contain comparable elements, and since $\top \in L'$, with $\gamma(\top) = C$, by point L_2 of Definition 2.11, there exists an And node which is exactly $\{\top\}$. Hence, for any $n \in V_O$, we can always approximate the (non-empty) set of successors of $\gamma(n)$, and we are guaranteed that n will have at least one successor (point $\mathsf{A}_{4.2}$ of Definition 4.5).

Notice also that, under the hypothesis that the effectiveness requirement of Definition 4.1 are satisfied, $\mathsf{Over}\left(\mathcal{S}, C', L'\right)$ is effectively constructible for any finite subsets C' and L' of C and L respectively.

As stated before, we will use And-Or graphs to decide the negative instances of CPWsts. For that purpose, we need to define which node of the And-Or graph corresponds to the $\preceq$-upward-closed set U whose reachability we want to decide. Given a WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \preceq\right\rangle$, an associated And-Or graph $\mathsf{Over}\left(\mathcal{S}, L', C'\right) = \langle V_A, V_O, v_i, \rightarrowtail\rangle$, and an $\preceq$-upward-closed set of states $U \subseteq C$, we denote by $\mathsf{Nodes}\left(U\right)$ the set of nodes $v \in V_A \cup V_O$ such that $\gamma\left(v\right) \cap U \neq \emptyset$. That is, $\mathsf{Nodes}\left(U\right)$ is the set of nodes whose associated $\preceq$-downward-closed set of states intersects with U.

Example 4.2 *We refer the reader to Section 4.3.1 for an example of And-Or graph.*

Degenerated case If an And-Or graph is such that any Or-node has exactly one successor, the And-Or graph is said to be *degenerated*. In that case, the avoidability problem is equivalent to the (un)reachability problem in a plain graph. From the definition of $\mathsf{Over}\left(\mathcal{S}, C', L'\right)$, we can easily see that the And-Or graph will be degenerated if for any $d \in L' \cup C'$, there exists a *unique* minimal set $\gamma\left(D\right)$ such that $D \in V_A$ and $\mathsf{Post}\left(\gamma\left(d\right)\right) \subseteq \gamma\left(D\right)$. Thus, *degenerated* And-Or graphs will appear when we consider certain well-chosen domain of limits, that we can characterise. This is the purpose of the next definition:

Definition 4.6 (Perfect Pairs) *Given a* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \preceq\right\rangle$ *and an adequate domain of limits* $\langle L, \sqsubseteq, \gamma\rangle$ *for* $\left\langle C, \preceq\right\rangle$, *we say that a pair* $\langle C', L'\rangle$, *where* $C' \subseteq C$ *with* $c_0 \in C$ *and* $L' \subseteq L$ *with* $\top \in L'$, *is* perfect *if for any* $d \in L' \cup C'$, *there exists one and only one canonical set* $D \subseteq L' \cup C'$ *such that the following holds:*

1. $\mathsf{Post}\left(\gamma(d)\right) \subseteq \gamma(D)$ *and*

2. *there is no* $D' \subseteq L' \cup C'$ *with* $\mathsf{Post}\left(\gamma(d)\right) \subseteq \gamma(D') \subset \gamma(D)$. ■

The next lemma states that this characterisation is sufficient to obtain a degenerated And-Or graph:

Lemma 4.1 *Given a* WSTS $S = \left\langle C, c_0, \Rightarrow, \preceq\right\rangle$, *an adequate domain of limits* $\langle L, \sqsubseteq, \gamma\rangle$ *for* $\left\langle C, \preceq\right\rangle$, *a finite subset* $C' \subseteq C$ *with* $c_0 \in C'$, *and a finite subset* $L' \subseteq L$ *with* $\top \in L'$: **if** $\langle C', L'\rangle$ *is perfect,* **then** $\mathsf{Over}\left(\mathcal{S}, C', L'\right)$ *is a degenerated And-Or graph.*

Proof. Immediate from Definition 4.5 and Definition 4.6. $\square$

The following example shows that it is possible to devise pairs $\langle C', L' \rangle$ that are not perfect.

Example 4.3 *Let us consider a* LCS $\mathcal{C}$ *with set of states* Q*, alphabet* $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ *and only one channel. Let* $\mathcal{S}_{\mathcal{C}} = \langle \mathsf{States}(\mathcal{C}), c_0, \Rightarrow, \precsim \rangle$ *be its associated* WSTS *and* $\left\langle \mathcal{L}(\Sigma, Q), \overline{\sqsubseteq}, \gamma \right\rangle$ *be its adequate domain of limits. Let* $C' = \{c_0\}$ *and* L' *be the set of any limit element containing an* sre *on* Σ *of size* ≤ 2*, plus the* $\top$ *element. Let us consider* $d = \langle q, (\mathsf{a} + \varepsilon) \cdot (\mathsf{b} + \varepsilon) \rangle \in L'$*, for some state* $q \in Q$*. From* q*, let us assume that the only firable transition consists in adding a* c *to the channel and move to state* q'*. Hence,* $\mathsf{Post}(\gamma(d)) = \gamma(d')$ *where* $d' = \langle q', (\mathsf{a} + \varepsilon) \cdot (\mathsf{b} + \varepsilon) \cdot (\mathsf{c} + \varepsilon) \rangle$*. However,* $d' \notin L'$*, because it contains an* sre *of size 3.*

One can devise several canonical subsets of $C' \cup L'$ *that over-approximate* d' *as precisely as possible. This implies that* $\langle C', L' \rangle$ *does not form a perfect pair. These sets are* $D_1 = \left\{ \langle q', (\mathsf{a} + \mathsf{b})^* \cdot (\mathsf{c} + \varepsilon) \rangle \right\}$ *and* $D_2 = \left\{ \langle q', (\mathsf{a} + \varepsilon) \cdot (\mathsf{b} + \mathsf{c})^* \rangle \right\}$*. Remark that* $\gamma(D_1) \not\subseteq \gamma(D_2)$*, that* $\gamma(D_2) \not\subseteq \gamma(D_1)$*, and that both* D_1 *and* D_2 *contain* sre *in normal form.* $\Diamond$

Properties of $\mathsf{Over}(\mathcal{S}, C', L')$ Let us now prove some properties of $\mathsf{Over}(\mathcal{S}, C', L')$ that show how it is related to the coverability problem. More precisely, we first prove that, for any pair $\langle C', L' \rangle$ such that $c_0 \in C'$ and $\top \in L'$, this abstraction is *adequate* to decide negative instances of the coverability problem (Proposition 4.4). That is, we show that, for any upward-closed set U, if U is avoidable in $\mathsf{Over}(\mathcal{S}, C', L')$, then it is also avoidable in $\mathcal{S}$.

Then, we prove that, for some pair $\langle C', L' \rangle$, it is *complete* to decide negative instances (Proposition 4.5). This means that, if U is avoidable in $\mathcal{S}$ it is always possible to find a pair $\langle C', L' \rangle$ s.t. U is avoidable in $\mathsf{Over}(\mathcal{S}, C', L')$. In some sense, such a pair $\langle C', L' \rangle$ can be considered as a witness or a proof that U is avoidable in $\mathcal{S}$.

To establish those results, we first show that $\mathsf{Over}(\mathcal{S}, C', L')$ is a proper over-approximation of $\mathcal{S}$, in the sense that it can simulate $\mathcal{S}$ for any $\langle C', L' \rangle$ such that $c_0 \in C'$ and $\top \in L'$. By *simulating*, we mean that, for any execution $c_0, c_1, \ldots, c_n$ of $\mathcal{S}$, there exists an unfolding of $\mathsf{Over}(\mathcal{S}, C', L')$ that contains a path whose nodes cover $c_0, c_1, \ldots, c_n$. This is the purpose of the following proposition:

Proposition 4.3 (Simulation) *Given a* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \preceq \right\rangle$ *with an adequate domain of limits* $\langle L, \sqsubseteq, \gamma \rangle$ *for* $\left\langle C, \preceq \right\rangle$*, the following holds for any* $C' \subseteq C$ *with* $c_0 \in C'$ *and* $L' \subseteq L$ *with* $\top \in L'$*: for any execution* $c_0 c_1 \ldots c_k$ *of* $\mathcal{S}$ *and any unfolding* $T = \langle N, root, B, \Lambda \rangle$ *of* $\mathsf{Over}(\mathcal{S}, C', L')$ *there exists a path* $n_0 n_1 \ldots n_{2k}$ *of* T *with* $n_0 = root$ *and* $c_i \in \gamma(\Lambda(n_{2i}))$ *for any* $0 \leq i \leq k$*.*

Proof. Let $c_0, \dots, c_k$ be a path of $\mathcal{S}$. For any unfolding, we will show, by induction on the length k of the path in $\mathcal{S}$, that there exists a path $n_0 n_1 \dots n_{2k}$ of the unfolding such that $c_i \in \gamma(\Lambda(n_{2i}))$ for all i such that $0 \leq i \leq k$.
Base case: The base case is trivial since $\Lambda(root) = c_0$ following A_3 and C_1.
Induction step: Suppose that there exists a path $P = n_0, \dots, n_{2i}$ $(i < k)$ of the unfolding, such that $c_j \in \gamma(\Lambda(n_{2j}))$ for all j such that $1 \leq j \leq i$. Let us show that there exists a path $n_0 \dots n_{2(i+1)}$ of the unfolding, where $c_j \in \gamma(\Lambda(n_{2j}))$ for all j such that $1 \leq j \leq i+1$. Following C_3, the successor n of n_{2i} in the unfolding is s.t. $\Lambda(n) = v$, where $v = \{d_1, \dots, d_\ell\}$ is an And-node with $\Lambda(n_{2i}) \rightarrowtail v$. From point $\mathsf{A}_{4.2}$ of Definition 4.5, since $c_i \in \gamma(\Lambda(n_{2i}))$, and since $c_i \Rightarrow c_{i+1}$, there is $1 \leq j \leq \ell$ s.t. $c_{i+1} \in \gamma(d_j)$. Moreover, following $\mathsf{A}_{4.1}$ and C_2, v has a successor v' such that $\Lambda(v') = d_j$. Thus, $c_{i+1} \in \gamma(\Lambda(v'))$. We conclude that in the path P extended with the nodes v and v', each Or-node n_{2j} covers its corresponding c_j, i.e., $c_j \in \gamma(\Lambda(n_{2j}))$. $\square$

Intuitively, this proposition tells us that, for any over-approximation $\mathsf{Over}(\mathcal{S}, C', L')$, for any reachable configuration c of $\mathcal{S}$, there is, in any unfolding of $\mathsf{Over}(\mathcal{S}, C', L')$, an Or-node[4] n that covers c. Since this holds for any reachable configuration, we deduce that the (labels of the) set of Or-nodes of any unfolding forms an over-approximation of $\mathsf{Reach}(\mathcal{S})$. Hence the following corollary:

Corollary 4.1 *Let $\mathcal{S} = \langle C, c_0, \Rightarrow, \preceq \rangle$ be a* WSTS*. Let $\langle L, \sqsubseteq, \gamma \rangle$ be an adequate of limits for $\langle C, \preceq \rangle$. Let C' be a finite subset of C s.t. $c_0 \in C'$, and let L' be a finite subset of L s.t. $\top \in L'$. Then, for any unfolding $\mathcal{T} = \langle N, root, B, \Lambda \rangle$ of* $\mathsf{Over}(\mathcal{S}, C', L')$*, we have:* $\mathsf{Reach}(\mathcal{S}) \subseteq \gamma(\Lambda(N \cap V_O))$*, where V_O is the set of Or-nodes of* $\mathsf{Over}(\mathcal{S}, C', L')$*.*

Thus, any And-Or graph $\mathsf{Over}(\mathcal{S}, C', L')$ can be regarded as a compact way to encode several over-approximations of $\mathsf{Reach}(\mathcal{S})$, built upon the sets C' and L'. The usefulness of the And-Or graph for that purpose will be discussed in Section 4.3.1.

A direct consequence of this simulation property is given by Proposition 4.4: if some set[5] U of configurations is avoidable in an over-approximation of a WSTS $\mathcal{S}$, then $\mathcal{S}$ cannot reach U. This implies that And-Or graphs are *adequate* to decide the negative instances of the coverability problem.

Proposition 4.4 (Adequacy) *Given a* WSTS *$\mathcal{S} = \langle C, c_0, \Rightarrow, \preceq \rangle$, an adequate domain of limits $\langle L, \sqsubseteq, \gamma \rangle$ for $\langle C, \preceq \rangle$, and a set $U \subseteq C$, the following holds for any $C' \subseteq C$ with $c_0 \in C'$ and $L' \subseteq L$ with $\top \in L'$:* if $\mathsf{Nodes}(U)$ *is avoidable in* $\mathsf{Over}(\mathcal{S}, C', L')$*,* then $\mathsf{Reach}(\mathcal{S}) \cap U = \emptyset$*.*

Proof. Since $\mathsf{Nodes}(U)$ is avoidable in $\mathsf{Over}(\mathcal{S}, C', L')$, there exists an unfolding $\mathcal{T} = \langle N, root, B, \Lambda \rangle$ of $\mathsf{Over}(\mathcal{S}, C', L')$ s.t. $N \cap \mathsf{Nodes}(U) = \emptyset$. This means that $\gamma(N) \cap U = \emptyset$. By Corollary 4.1, $\mathsf{Reach}(\mathcal{S}) \subseteq \gamma(N \cap V_O) \subseteq \gamma(N)$. Hence, $\mathsf{Reach}(\mathcal{S}) \cap U = \emptyset$. $\square$

[4]Proposition 4.3 ensures that the configuration c will be covered by a node whose index is even in the corresponding path of the unfolding. Hence, this nodes is an Or-node.
[5]That set doesn't have to be upward-closed.

Finally, we prove a result of *completeness*. Intuitively, Proposition 4.5 says that, when the pair $\langle C', L'\rangle$ is *precise enough*, $\mathsf{Over}(\mathcal{S}, C', L')$ allows us to decide *negative instances* of the coverability problem. To prove that theorem, we first prove Lemma 4.2 that says that, if $L' \cup C'$ contains a coverability set and the $\overline{\leq}$-upward-closed set U of configurations is not reachable into the WSTS, then there exists an unfolding that does not intersect with U.

Lemma 4.2 *Given a* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq}\rangle$, *an adequate domain of limits* $\langle L, \sqsubseteq, \gamma\rangle$ *for* $\langle C, \overline{\leq}\rangle$, *an* $\overline{\leq}$*-upward-closed set* $U \subseteq C$, *the following holds for any* $C' \subseteq C$ *with* $c_0 \in C'$ *and* $L' \subseteq L$ *with* $\top \in L'$ *such that there exists a coverability set* $\mathcal{CS}$ *of* $\mathcal{S}$ *with* $\mathcal{CS} \subseteq L' \cup C'$: **if** $\mathsf{Reach}(\mathcal{S}) \cap U = \emptyset$ **then** *there exists an unfolding* $T = \langle N, root, B, \Lambda\rangle$ *of* $\mathsf{Over}(\mathcal{S}, C', L')$ *such that* $\forall n \in N : \gamma(\Lambda(n)) \cap U = \emptyset$.

Proof. We construct such an unfolding by induction, and use Proposition 3.3 to conclude. More precisely, we show how to compute an unfolding whose nodes n are such that $\gamma(\Lambda(n)) \subseteq \gamma(\mathcal{CS}) = \mathsf{Cover}(\mathcal{S})$. Following Proposition 3.3 and the fact that $\mathsf{Reach}(\mathcal{S}) \cap U = \emptyset$, that implies that $\gamma(\Lambda(n)) \cap U = \emptyset$ for all the nodes n of the unfolding.

Base case: Notice that $\Lambda(root) = c_0$ following C_1 (Definition 4.3) and A_3 (Definition 4.5), and $c_0 \in \gamma(\mathcal{CS})$ following Definition 3.7 and Definition 3.8. Moreover, $\mathsf{Post}(\gamma(c_0)) \subseteq \gamma(\mathcal{CS})$ because $\mathsf{Post}(\gamma(c_0)) \subseteq \gamma(\mathsf{Reach}(\mathcal{S}))$, by monotonicity, and because $\gamma(\mathsf{Reach}(\mathcal{S})) = \mathsf{Cover}(\mathcal{S}) = \gamma(\mathcal{CS})$.

Thus, $\mathcal{CS}$ covers the successors of v_i. Hence, following $\mathsf{A}_{4.2}$ (Definition 4.5), there exists $v \in V_A$ (the set of And-nodes) with $v_i \rightarrowtail v$ and $\gamma(v) \subseteq \gamma(\mathcal{CS})$ since v satisfies the preciseness property of $\mathsf{A}_{4.2}$ (Definition 4.5). We extend the unfolding by choosing such an And-node v and add one successor node n to *root* such that $\Lambda(n) = v$.

Induction step: Suppose that we can construct $2k$ layers of the unfolding such that for all the nodes n of the $2k$ first layers, $\gamma(n) \subseteq \gamma(\mathcal{CS})$. Let us show that we can construct $2k+2$ layers such that for all the nodes n of the $2k+2$ first layers, $\gamma(n) \subseteq \gamma(\mathcal{CS})$.

By induction hypothesis, all the And-nodes n in the $2k$-th layer are such that $\Lambda(n) = \{d_1, \ldots, d_\ell\}$ and $\gamma(\Lambda(n)) \subseteq \gamma(\mathcal{CS})$. Since, following $\mathsf{A}_{4.1}$ (Definition 4.5), all the successors nodes v of $\Lambda(n)$ in $\mathsf{Over}(\mathcal{S}, C', L')$ are such that $v \in \Lambda(n)$, we have that $\gamma(v) \subseteq \gamma(\mathcal{CS})$. We conclude, following C_2 (Definition 4.3), that all the Or-nodes n' of the $2k+1$-th layer are such that $\gamma(\Lambda(n')) \subseteq \gamma(\mathcal{CS})$.

For each node n of the $(2k+1)$-th layer, since $\mathcal{S}$ is monotonic and $\gamma(n) \subseteq \gamma(\mathcal{CS})$, we have that $\forall c \in \gamma(n), \forall c'$ s.t. $c \Rightarrow c'$, there exists $c'' \in \mathsf{Reach}(\mathcal{S}) : c \overline{\leq} c''$ and $c'' \Rightarrow^* c'''$ with $c' \overline{\leq} c'''$ and $c''' \in \gamma(\mathcal{CS})$. Since $\gamma(\mathcal{CS})$ is $\overline{\leq}$-downward-closed we obtain that $\mathsf{Post}(\gamma(n)) \subseteq \gamma(\mathcal{CS})$ for all the nodes n of the $2k+1$-th layer.

Hence, there exists following $A_{4.2}$ (Definition 4.5) an And-node v with $\gamma(v) \subseteq \gamma(\mathcal{CS})$ and $\Lambda(n) \rightarrowtail v$ since v satisfies the preciseness property of $\mathsf{A}_{4.2}$ (Definition 4.5) and $\mathcal{CS}$

covers the successors of $\gamma(\Lambda(n))$. So, we extend the unfolding by choosing such a node v and add one successor n' to n such that $\Lambda(n') = v$. That allows us to conclude that we can construct the $2k+2$-th first layers of the unfolding with the property that all the nodes n are such that $\gamma(\Lambda(n)) \subseteq \mathcal{CS}$. $\square$

We are now ready to prove our *completeness* theorem:

Proposition 4.5 (Completeness) *Given a* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \preceq \rangle$, *an adequate domain of limits* $\langle L, \sqsubseteq, \gamma \rangle$ *for* $\langle C, \preceq \rangle$, *a* $\preceq$*-upward-closed set* $U \subseteq C$ *and a coverability set* $\mathcal{CS}$ *of* $\mathcal{S}$, *the following holds for any* $C' \subseteq C$ *with* $c_0 \in C'$ *and* $L' \subseteq L$ *with* $\top \in L'$ *such that* $\mathcal{CS} \subseteq L' \cup C'$: if $\mathsf{Reach}(\mathcal{S}) \cap U = \emptyset$ then $\mathsf{Nodes}(U)$ *is avoidable in* $\mathsf{Over}(\mathcal{S}, C', L')$.

Proof. As $\mathsf{Reach}(\mathcal{S}) \cap U = \emptyset$, there exists, from Lemma 4.2, an unfolding $\mathcal{T} = \langle N, root, B, \Lambda \rangle$ s.t. for every node $n \in N$: $\gamma(n) \cap U = \emptyset$. Hence, for every node $n \in N$: $n \notin \mathsf{Nodes}(U)$. Thus, $\mathsf{Nodes}(U)$ is avoidable in $\mathsf{Over}(\mathcal{S}, C', L')$. $\square$

Remark that we do not request that $L' \cup C' = \mathcal{CS}$. Hence, we can prove that U is not reachable without having a coverability set at our disposal.

Construction of $\mathsf{Over}(\mathcal{S}, C', L')$ in practice In the next section, we present the EEC algorithm. One of the main steps of this algorithm involves the computation of an And-Or graph and the decision of the coverability problem on this And-Or graph. The step that consists in computing the And-Or graph deserves some discussion.

First of all, it is important to remark that, given two finite subsets C' and L' of C and L respectively, $\mathsf{Over}(\mathcal{S}, C', L')$ is always constructible. The difficult point is to compute the successors of the Or-nodes, that is, given an element $c \in L' \cup C'$, to compute all the most precise subsets $E_1, E_2, \ldots E_n$ of $L' \cup C'$ whose downward-closure covers $\mathsf{Post}(\gamma(c))$. This can be done naively be enumerating all the (finitely many) subsets of $L' \cup C'$, and keeping the most precise ones. Of course, a more straightforward algorithm would greatly improve the efficiency of EEC in practice.

Such an algorithm should receive the element c and compute directly the sets $E_1, E_2, \ldots E_n$, in a symbolic fashion. It would also depend on the domain we are dealing with. In sections 5.4 and 5.5, we apply EEC respectively to the class of strongly monotonic SMPN, and to that of LCS. In both cases, we provide an algorithm of the kind we have just alluded to, that directly computes representatives of the most precise over-approximations of the successors of a set of configurations.

4.3 The 'Expand, Enlarge and Check' algorithm

On the basis of the results presented in section 4.2, we now propose a new algorithmic schema to decide the coverability problem of effective WSTS (in the sense of Definition 4.1). This algorithmic schema is called 'Expand, Enlarge and Check' or EEC for short. The main ingredients of EEC have already been introduced in the previous section: propositions 4.1 and 4.2 state that any EPRG is a suitable under-approximation to decide positive instances of CPWsts, and that, if U is indeed reachable there exists an EPRG that is a witness of the reachability of U. Symmetrically, propositions 4.4 and 4.5 say that any And-Or graph is a suitable over-approximation of the set of reachable states, and that, if U is not reachable, one can always find an And-Or graph that is a proof for this. Moreover, these under- and over-approximations are parametrised by subsets of C and L, which are recursively enumerable, according to Definition 4.1. Thus, our algorithm works by iteratively constructing pairs of approximations (under and over-approximations) of the WSTS which become more and more precise. After a finite number of steps either a concrete trace to a *covering state* will be found (in the EPRG), or *precise enough abstraction* (under the form of an And-Or graph) will be computed to prove that no covering state can ever be reached. This informal statement is formalised in Theorem 4.2.

Let $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ be a WSTS with adequate domain of limits $\langle L, \sqsubseteq, \gamma \rangle$. Let $C_0, C_1, \ldots, C_n, \ldots$ be an infinite sequence of finite subsets C such that (i) $\forall i \geq 0 : C_i \subseteq C_{i+1}$, (ii) $\forall c \in \mathsf{Reach}(\mathcal{S}) : \exists i \geq 0 : c \in C_i$, and (iii) $c_0 \in C_0$. Let $L_0, L_1, \ldots, L_n, \ldots$ be a infinite sequence of finite sets of limits such that (i) $\forall i \geq 0 : L_i \subseteq L_{i+1}$, (ii) $\forall \ell \in L : \exists i \geq 0 : \ell \in L_i$ and (iii) $\top \in L_0$. In the sequel, these sequences are often called *adequate sequences* of sets of configurations (resp. limits). Those sequences of sets exist because C and L are recursively enumerable, by E_1. Remark that these conditions imply that, for any finite subset D of C (resp. $L \cup C$), there exists i (resp. j) such that $D \subseteq C_i$ (resp. $D \subseteq L_j \cup C_j$). This holds in particular for any coverability set $\mathcal{CS}$ of $\mathcal{S}$. The schema is given at Algorithm 4.1 and its proof of correctness is stated in Theorem 4.2.

Theorem 4.2 *For any* WSTS S *with adequate domain of limits* $\langle L, \sqsubseteq, \gamma \rangle$ *that are effective, for any* $\leq$*-upward-closed set* U *represented by* $\mathsf{UGen}(U)$*, for any adequate sequence* $C_0, C_1, \ldots, C_n, \ldots$ *of sets of configurations and any adequate sequence* $L_0, L_1, \ldots, L_n, \ldots$ *of sets of limits, Algorithm 4.1 terminates after a finite amount of time and returns 'Reachable' if* $\mathsf{Reach}(S) \cap U \neq \emptyset$*, 'Unreachable' otherwise.*

Proof. Let us first prove that the body of the main loop always terminates. In order to establish this, let us notice that C_i is finite for all $i \geq 0$, that the transition relation $\Rightarrow$ is decidable (following E_2) and that $\leq$ is decidable too. Hence we can test whether $\mathsf{Reach}(\mathsf{Under}(S, C_i)) \cap U \neq \emptyset$ for all $i \geq 0$. Then, let us remark that the And-Or graph, as well as $\mathsf{Nodes}(U)$ are both constructible, because of the effectiveness properties

Algorithm 4.1: The EEC schema of algorithm.

```
Data: a finite representation of a WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$ with the adequate
      limit domain $\langle L, \sqsubseteq, \gamma \rangle$ for $\langle C, \leq \rangle$ that are effective (Definition 4.1).
Data: a finite representation of an $\leq$-upward-closed set of states $U \subseteq C$.
Data: an adequate sequence $C_0, C_1, \ldots, C_n, \ldots$ of finite subsets of $C$.
Data: an adequate sequence $L_0, L_1, \ldots, L_n, \ldots$ of finite subsets of $L$.
begin
    $i \leftarrow 0$;
    while (true) do
        'Expand'
            Compute Under $(S, C_i)$;
        'Enlarge'
            Compute Over $(S, C_i, L_i)$;
        'Check'
            if Reach (Under $(S, C_i)) \cap U \neq \emptyset$ then
                return 'Reachable' ;
            else if Nodes $(U)$ is avoidable in Over $(S, C_i, L_i)$ then
                return 'Unreachable' ;
        $i \leftarrow i + 1$;
end
```

of Definition 4.1. Hence, we can effectively test whether $\mathsf{Nodes}(U)$ is avoidable in $\mathsf{Over}(S, C_i, L_i)$ (remember that the avoidability problem is PTIME-complete).

It remains to prove that the algorithm returns a correct answer after a finite number of iterations of the loop. First remark that, by construction, C_j is a finite subset of C containing c_0 and L_j is a finite subset of L containing $\top$, for any $j \geq 0$.

If $\mathsf{Reach}(S) \cap U \neq \emptyset$, $\mathsf{Nodes}(U)$ is not avoidable in $\mathsf{Over}(S, C_i, L_i)$ for all $i \geq 0$ (by Proposition 4.4). Moreover, following Proposition 4.2, and since $C_0, C_2, \ldots, C_n, \ldots$ is an enumeration of the finite subsets of C, there exists j such that $\mathsf{Reach}(\mathsf{Under}(S, C_j)) \cap U \neq \emptyset$. We conclude that Algorithm 4.1 returns ‘Reachable’ if $\mathsf{Reach}(S) \cap U \neq \emptyset$.

If $\mathsf{Reach}(S) \cap U = \emptyset$, then, following Proposition 4.1, $\mathsf{Reach}(\mathsf{Under}(S, C_i)) \cap U = \emptyset$ for all $i \geq 0$. Moreover, there exists $i \geq 0$ such that there exists a coverability set $\mathcal{CS}$ of $\mathcal{S}$ with $\mathcal{CS} \subseteq L_i \cup C_i$. Hence, from Lemma 4.2, $\mathsf{Nodes}(U)$ is avoidable in $\mathsf{Over}(S, C_i, L_i)$ and we conclude that Algorithm 4.1 returns ‘Unreachable’ if $\mathsf{Reach}(S) \cap U = \emptyset$. □

Note that Theorem 4.2, that states the adequation and completeness of our algorithmic schema (for the coverability problem of effective WSTS), is not in contradiction with the result of [DFS98] which establishes that there does not exist a procedure that always terminates and returns a coverability set for a large class of WSTS, including ours. Indeed, to establish the correctness of our algorithm, we only need to ensure that a coverability set will eventually be included in the sequence of C_i’s and L_i’s. Nevertheless, given a pair $\langle C_i, L_i \rangle$, it is not possible to establish algorithmically that this pair contains a coverability set. Furthermore, given a particular $\leq$-upward-closed set U, our algorithm may terminate before reaching a pair $\langle C_i, L_i \rangle$ that contains a coverability set, because the set U is reachable or because the abstraction constructed from a pair $\langle C_j, L_j \rangle$, with $j < i$, is sufficiently precise to prove that U is not reachable.

Moreover, the constraints on the sequence of L_i’s computed by Algorithm 4.1 may be relaxed. Indeed, those constraints ensure that the algorithm eventually considers a set of limits which allows to construct a graph that is precise enough to decide negative instances of the coverability problem. However, following Proposition 4.5, it is sufficient to ensure that there exists $i \geq 0$ such that $L_i \cup C_i$ contains a coverability set. Hence, only the limits of a coverability set must appear in the sequence of L_i’s.

4.3.1 Why we need And-Or graphs

The reader might wonder why we have used And-Or graphs to represent the over-approximations of WSTS, and not plain graphs, as it is often the case in other works. We have already sketched the argument in favour of And-Or graphs when introducing them: there is no guarantee that any set of configuration can be uniquely over-approximated by elements of $L' \cup C'$. Now that we have introduced the EEC algorithm, we can illustrate this argument by a concrete example.

Consider the WSTS $\mathcal{S}_{\mathbb{N}} = \langle \mathbb{N}, 0, \Rightarrow, \leq_p \rangle$, where:

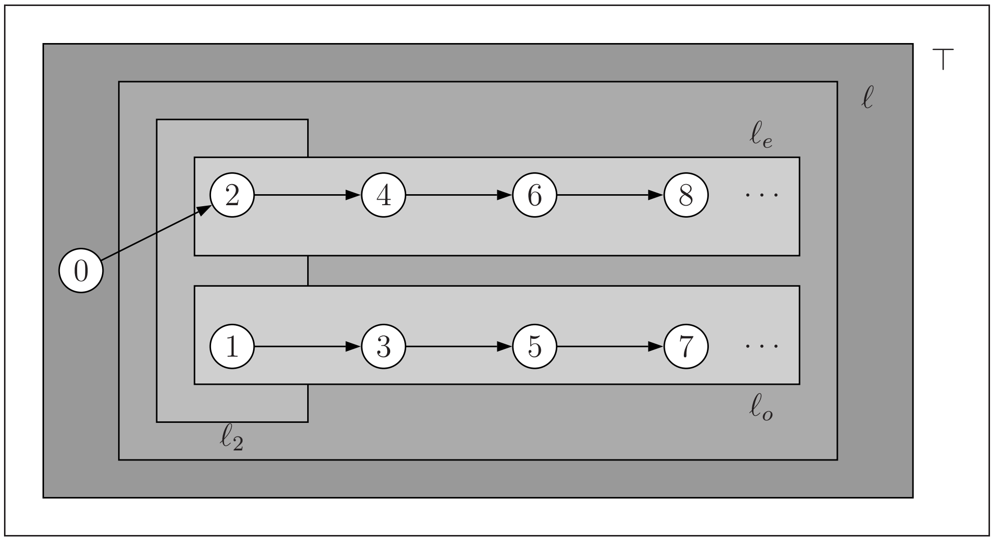


Figure 4.3: The configurations of $\mathcal{S}_{\mathbb{N}}$ and the limits that cover them.

- $\Rightarrow = \{(i, i+2) \mid i \geq 0\}$;
- $\leq_p = \{(i, i+2j) \mid i \geq 1, j \geq 0\}$.

Thus, the state space of this system contains two infinite ascending chains: $2, 4, 6, \ldots$ and $1, 3, 5, \ldots$ Remark that 0, the initial state, is incomparable to any other state and that only the ascending chain of even number is reachable.

Let us fix the adequate domain of limits for $\mathcal{S}_{\mathbb{N}}$ defined as: $L = \{\ell, \ell_e, \ell_o, \ell_2, \top\}$, where (Figure 4.3 depicts this):

1. $\gamma(\ell) = \mathbb{N} \setminus \{0\}$;
2. $\gamma(\ell_e) = \{2, 4, 6, \ldots\}$;
3. $\gamma(\ell_o) = \{1, 3, 5, \ldots\}$;
4. $\gamma(\ell_2) = \{1, 2\}$;
5. $\gamma(\top) = \{0, 1, 2 \ldots\}$.

Thus, the unique coverability set of the system is $\mathsf{CS}\,(\mathcal{S}_{\mathbb{N}}) = \{0, \ell_e\}$.

Let us now fix $C' = \{0\}$ and $L' = \{\ell, \ell_e, \ell_2, \top\}$, and let us build $\mathsf{Over}\,(\mathcal{S}_{\mathbb{N}}, C', L')$. Remark that $\mathsf{CS}\,(\mathcal{S}_{\mathbb{N}}) \subseteq L' \cup C'$. We obtain the And-Or graph of Figure 4.4 (where And-nodes are represented by rectangles and Or-nodes are represented by circles). Indeed, ℓ_e and ℓ_2 are two incomparable limits which are both suitable to cover the one-step successor of the initial configuration. However, while ℓ_e is sufficient to cover all the successors of 0, we need ℓ to over-approximate the successors of ℓ_2.

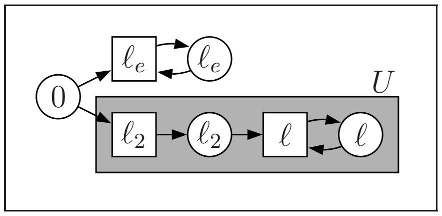


Figure 4.4: The reachable part (from v_i) of the And-Or graph obtained with C' and L'. The nodes in the grey box are in the upward-closed set U.

Finally, let us choose the $\leq_p$-upward-closed set of *bad* states $U = \{i \mid 1 \leq_p i\}$. Remark that the system is *safe* w.r.t. U, since only *even* natural numbers (which are all $\leq_p$-incomparable to 1) can be reached. But, due to the coarse over-approximation, one of the unfoldings of the And-Or graph intersects with U (see Figure 4.4). And this happens even though all the elements of the coverability set are present in $L' \cup C'$. Thus, one cannot thoroughly represent this over-approximation of the system thanks to a plain graph. Otherwise, one would have to *choose* the *right* successor of the initial node. At each step i of the algorithm, an exponential number of such plain graphs could have to be constructed, in order to test for all the possible choices. Such a procedure is clearly less efficient than the PTime algorithm that decides the avoidability on And-Or graphs. Remark that a procedure considering plain graphs would have to test whether U is avoidable in *all the graphs* it builds, until it finds a graph proving that the system is safe. Otherwise, it could never terminate: this happens, e.g. if the system is safe but the only abstractions the algorithm builds and explores are repeatedly too coarse.

4.4 Discussion

This section has introduced the 'Expand, Enlarge, and Check' algorithmic schema that decides the coverability problem on a very large class of WSTS. Examples of WSTS that this method can handle are strongly monotonic SMPN; Petri nets; Petri nets with transfer arcs, with reset arcs, with non-blocking arcs; lossy channel systems; timed Petri nets; broadcast protocols,...

Other applications of the EEC algorithm are possible and have already been studied by other researchers: it has been applied in the verification of *asynchronous programs* in [JM07], and has been cited by several papers and PhD. theses related to the verification of infinite state systems [Bin05, BH05, Bar06, GRVB06a].

In order to be exploited (on a class of WSTS that respects the effectiveness criteria of Definition 4.1), this algorithmic schema has to be instantiated. This involves that an *adequate domain of limits* must be available, because any downward-closed set of configurations has to be finitely representable. We have not provided an automatic way to devise such an adequate domain of limits. However, the recent paper [GRVB06a] presents a general framework to automatically obtain such an adequate representation of downward-closed sets.

In this chapter, we have purposely kept the discussion at a purely theoretical level. This leaves much room for practical improvements of the method. In particular, efficient methods to build the under- and over-approximations are welcome (and the sequences of C_i's and L_i's, which is related). The next chapter addresses some of these points.

Chapter 5

Practical applications of EEC

In the previous chapter, we have introduced the 'Expand, Enlarge, and Check' algorithm that decides the coverability problem on WSTS. As we have noticed in the discussion at the end of this chapter, several sub–procedures (such as the search for an error trace in the under–approximations, or the construction of the And–Or graph) of EEC have been presented at a very high level. In this chapter, we consider these problems with more details, and show how to obtain efficient procedures to solve them.

The two main improvements that are discussed in this chapter are:

1. We study finite WSTS and present an efficient algorithm to decide coverability of these systems. We motivate the interest in these finite WSTS by showing that the reachability problem in the under–approximations built during the 'Expand' phase can be reduced to deciding coverability in a finite WSTS. Thus, our efficient algorithm can be used to speed up this phase. We further show that the avoidability problem that has to be dealt with during the 'Enlarge' phase can, in some cases, be reduced to deciding coverability in a finite WSTS too.

2. We show how the over-approximations can be built in an efficient way when the WSTS considered are LCS.

The chapter is organised as follows. We begin with some preliminary notions such as *lossiness abstraction* and finite WSTS. Then, in Section 5.2, we show how finite WSTS can be exploited in EEC, and provide in Section 5.3 an efficient algorithm to decide coverability on these systems. We exploit this in the next two sections, by instantiating the EEC schema of algorithm to the case of strongly monotonic SMPN (in Section 5.4) and to the case of LCS (in Section 5.5). In the case of LCS, we explain how the construction of the over-approximations can be improved.

The content of this chapter is based on the articles [GRVB05], [GRVB04] and [GRVB06b]. Section 5.3 and Section 5.4.4 contain unpublished material.

5.1 Preliminaries

This first section introduces concepts that are necessary in this chapter.

5.1.1 Simple monotonicity

In our definition of WSTS, we have considered a very weak notion of monotonicity: for every c_1, c_2 and c_3 s.t. $c_1 \Rightarrow c_2$ and $c_1 \overline{\leq} c_3$, there exists c_4 s.t. $c_3 \Rightarrow^* c_4$ and $c_2 \overline{\leq} c_4$. However, all the practical models of WSTS that we have considered (EPN, strongly monotonic SMPN, LCS, ...), have the guarantee that there exists c_4 s.t. $c_3 \Rightarrow c_4$ and $c_2 \overline{\leq} c_4$. That (stronger) definition allows to obtain nice properties (that we will exploit in the sequel) such as: $c_1 \overline{\leq} c_2$ implies $\downarrow(\mathsf{Post}(c_1)) \subseteq \downarrow(\mathsf{Post}(c_2))$.

This stronger version of monotonicity is called *simple monotonicity*, and we obtain the following definition:

Definition 5.1 (SIMPLY MONOTONIC WSTS) *A* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ *is said to be* simply monotonic *iff, for every c_1, c_2 and c_3 in C s.t. $c_1 \Rightarrow c_2$ and $c_1 \overline{\leq} c_3$, there exists $c_4 \in C$ with $c_3 \Rightarrow c_4$ and $c_2 \overline{\leq} c_4$.* ■

In this chapter, we will assume that all the WSTS we consider are *simply monotonic*. Hence, we will write simply 'WSTS' instead of 'simply monotonic WSTS', 'monotonicity' instead of 'simple monotonicity', and so forth... In practice, this new definition is not restrictive since EPN, strongly monotonic SMPN and LCS are simply monotonic.

5.1.2 Lossiness abstraction

Let us first consider the notion of *lossy* WSTS. An LCS is said to be *lossy* because messages can be lost by the FIFO channels that ensure the communications between the automata. From the point of view of the ordering $\precsim$, it means that if a configuration c' is reachable from a configuration c, then, all the configurations c'' s.t. $c'' \precsim c'$ are reachable from c too. It is clear that not every WSTS enjoys this property. However, given a WSTS $\mathcal{S}$, one can consider its *lossy* counterpart $\mathsf{lossy}(\mathcal{S})$, defined as follows:

Definition 5.2 (LOSSY WSTS) *The* lossy version *of a* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ *is the* WSTS $\mathsf{lossy}(\mathcal{S}) = \langle C, c_0, \Rightarrow_\ell, \overline{\leq} \rangle$ *where:*

$$\Rightarrow_\ell = \left\{ (c, c') \mid \exists c'' \in C : c \Rightarrow c'' \wedge c' \overline{\leq} c'' \right\}$$

A lossy WSTS *is a* WSTS $\mathcal{S}$ *s.t.* $\mathcal{S} = \mathsf{lossy}(\mathcal{S})$. ■

Remark that $\Rightarrow_\ell$ is still $\overline{\leq}$-monotonic. Hence, $\mathsf{lossy}(\mathcal{S})$ is indeed a WSTS.

Since the transition relation of $\mathsf{lossy}(\mathcal{S})$ is defined as an *extension* of $\Rightarrow$, $\mathsf{Reach}\left(\mathcal{S}\right) \subseteq \mathsf{Reach}\left(\mathsf{lossy}(\mathcal{S})\right)$, for any WSTS $\mathcal{S}$. However, $\mathsf{Reach}\left(\mathcal{S}\right) \supseteq \mathsf{Reach}\left(\mathsf{lossy}(\mathcal{S})\right)$ does not hold in general. Nevertheless, by definition of $\Rightarrow_\ell$, for any $c \in \mathsf{Reach}\left(\mathsf{lossy}(\mathcal{S})\right) \setminus \mathsf{Reach}\left(\mathcal{S}\right)$, there exists $c' \in \mathsf{Reach}\left(\mathcal{S}\right)$ with $c \overline{\leq} c'$:

Lemma 5.1 *For any* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$*: for any* $c \in \mathsf{Reach}\left(\mathsf{lossy}(\mathcal{S})\right)$*, there exists* $\overline{c} \in \mathsf{Reach}\left(\mathcal{S}\right)$ *s.t.* $c \overline{\leq} \overline{c}$*.*

Proof. Let us assume that $\mathsf{lossy}(\mathcal{S}) = \left\langle C, c_0, \Rightarrow_\ell, \overline{\leq} \right\rangle$ and let us c be a configuration of $\mathsf{Reach}\left(\mathsf{lossy}(\mathcal{S})\right)$. Let $c_0, c_1, \ldots, c_n$ be a finite execution of $\mathsf{lossy}(\mathcal{S})$ s.t. $c_n = c$. Let us prove, by induction on n that, for any $1 \leq i \leq n$, there exists $\overline{c}_i \in \mathsf{Reach}\left(\mathcal{S}\right)$ s.t. $c_i \overline{\leq} \overline{c}_i$.

Base case: $n = 0$**.** In this case, $\overline{c}_0 = c_0$. The lemma holds because $c_0 \overline{\leq} c_0$.

Inductive case: $n = k + 1$**.** By induction hypothesis, there is $\overline{c}_k \in \mathsf{Reach}\left(\mathcal{S}\right)$ s.t. $c_k \overline{\leq} \overline{c}_k$. Let c_{k+1} be s.t. $c_k \Rightarrow_\ell c_{k+1}$. By definition of $\Rightarrow_\ell$, there is c'_{k+1} s.t $c_k \Rightarrow c'_{k+1}$ and $c_{k+1} \overline{\leq} c'_{k+1}$. By monotonicity of $\Rightarrow$, there is $\overline{c}_{k+1}$ s.t. $\overline{c}_k \Rightarrow \overline{c}_{k+1}$ and $c'_{k+1} \overline{\leq} \overline{c}_{k+1}$. Hence, $c_{k+1} \overline{\leq} \overline{c}_{k+1}$.

□

Hence the following corollary:

Corollary 5.1 *For any* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$*:*

1. $\mathsf{Cover}\left(\mathcal{S}\right) = \mathsf{Cover}\left(\mathsf{lossy}(\mathcal{S})\right)$*;*
2. *for any* $\overline{\leq}$*-upward closed set* $U \subseteq C$*:* $\mathsf{Reach}\left(\mathcal{S}\right) \cap U = \emptyset$ *iff* $\mathsf{Cover}\left(\mathsf{lossy}(\mathcal{S})\right) \cap U = \emptyset$*.*

As a consequence, when we want to decide the coverability problem on a WSTS $\mathcal{S}$, we can always consider its lossy version $\mathsf{lossy}(\mathcal{S})$. Thanks to this result, we will be able to apply the algorithm of section 5.3 to any finite under-approximations built by EEC, and therefore improve the practical efficiency of the 'Expand' phase (as well as the 'Enlarge' phase when the over-approximations are *degenerated And-Or graphs*).

5.1.3 Finite WSTS

As we will see in the next section, one can obtain an efficient algorithm to decide the coverability problem on *finite* WSTS by exploiting the monotonicity property to reduce the space of configurations that have to be explored. The next definition states precisely what is a finite WSTS:

$$\begin{tikzcd}
& \boxed{\langle 0,0,1\rangle} \arrow[r] \arrow[d, shift left] & \boxed{\langle 1,0,1\rangle} \arrow[r] \arrow[d, shift left] & \boxed{\langle 2,0,1\rangle} \\
\boxed{\langle 0,1,0\rangle} \arrow[r] & \boxed{\langle 1,1,0\rangle} \arrow[r] \arrow[u, shift left] & \boxed{\langle 2,1,0\rangle} \arrow[u, shift left] &
\end{tikzcd}$$

Figure 5.1: $\mathsf{Under}\left(\mathcal{S}_{\mathcal{N}_\mu}, C'\right)$ for $C' = \{\mathbf{m} \mid \forall p \in \{p_1, p_2, p_3\} : \mathbf{m}(p) \leq 2\}$. Only the nodes that are reachable from the initial node $v_i = \langle 0,1,0\rangle$ have been drawn on the figure.

Definition 5.3 (Finite WSTS) *A* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq}\right\rangle$ *is* finite *iff* C *is finite.* ■

Remark that a finite WSTS can admit infinite executions. In particular, if the WSTS is deadlock free, every maximal execution of the WSTS is infinite.

5.2 Application of finite WSTS to EEC

We will present, in section 5.3, an efficient algorithm that decides the coverability problem on finite WSTS. In order to motivate the usefulness of this algorithm, we first show how it can be used in the framework of EEC. We show that, thanks to corollary 5.1, finite WSTS can always be used during the 'Expand' phase. Moreover, in the case where the And-Or graph is degenerated, the 'Enlarge' phase consists in deciding coverability on a finite WSTS too.

'Expand' phase Let us first consider the 'Expand' phase. Unfortunately, the under-approximations built by EEC are not, in general, finite WSTS, as shown by the following example.

Example 5.1 *Consider the Petri net $\mathcal{N}_\mu$ introduced in Example 2.8 (Figure 2.1). Figure 5.1 represents* $\mathsf{Under}\left(\mathcal{S}_{\mathcal{N}_\mu}, C'\right)$ *for* $C' = \{\mathbf{m} \mid \forall p \in \{p_1, p_2, p_3\} : \mathbf{m}_p \leq 2\}$*. The transition system* $\langle C', \langle 0,1,0\rangle, \Rightarrow\rangle$ *(where* $\Rightarrow$ *is the transition relation of* $\mathsf{Under}\left(\mathcal{S}_{\mathcal{N}_\mu}, C'\right)$*) is* not $\preccurlyeq$-monotonic *because if we let* $c_1 = \langle 1,0,1\rangle$*,* $c_2 = \langle 2,0,1\rangle$ *and* $c_3 = \langle 2,0,1\rangle$*, we have clearly* $c_1 \Rightarrow c_2$ *and* $c_1 \preccurlyeq c_3$*, but there is no* c_4 *s.t.* $c_2 \preccurlyeq c_4$ *and* $c_3 \Rightarrow c_4$*.*

However, if the WSTS $\mathcal{S}$ under analysis is *lossy*, then $\mathsf{Under}\left(\mathcal{S}, C'\right)$ is guaranteed to be a WSTS too, as shown in the following proposition. Remark that this restriction is not problematic in practice. Indeed, by Corollary 5.1, we can analyse $\mathsf{lossy}(\mathcal{S})$ instead of $\mathcal{S}$, as far as CPWsts is concerned.

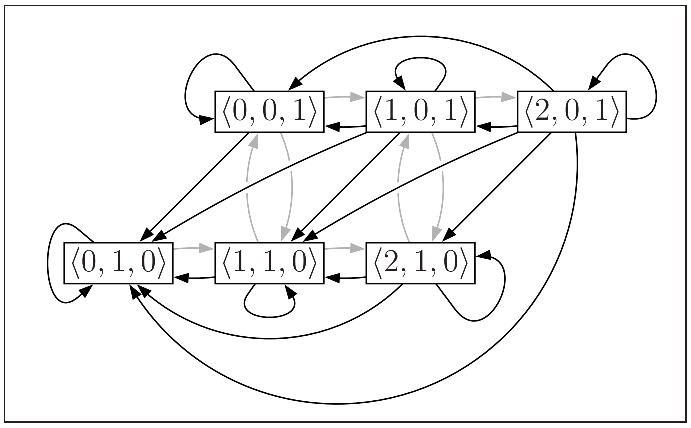


Figure 5.2: $\mathsf{Under}\left(\mathsf{lossy}(\mathcal{S}_{\mathcal{N}_\mu}), C'\right)$ for $C' = \{\mathbf{m} \mid \forall p \in \{p_1, p_2, p_3\} : \mathbf{m}(p) \leq 2\}$. Only the nodes that are reachable from the initial node $v_i = \langle 0, 1, 0\rangle$ have been drawn on the figure. The arrows corresponding to the transition relation of $\mathsf{Under}\left(\mathcal{S}_{\mathcal{N}_\mu}, C'\right)$ have been drawn in grey.

Proposition 5.1 *Given a* lossy WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq}\right\rangle$, *and a finite* $C' \subseteq C$: $\mathsf{Under}\,(\mathcal{S}, C')$ *is a finite* WSTS.

Proof. According to Definition 2.16, it is sufficient to show that the transition relation of $\mathsf{Under}\,(\mathcal{S}, C')$ is $\overline{\leq}$-monotonic. Let $\Rightarrow'$ denote this transition relation. Let c_1, c_2 and c_3 be three configurations of C' s.t. $c_1 \Rightarrow c_2$ and $c_1 \overline{\leq} c_3$. Since $\mathcal{S}$ is monotonic, and since $c \Rightarrow' c'$ implies $c \Rightarrow c'$, there exists $c_4 \in C$ s.t. $c_2 \Rightarrow c_4$ and $c_3 \overline{\leq} c_4$. Since $\mathcal{S}$ is lossy, we have $c_2 \Rightarrow c_3$ too. As c_2 and c_3 are both in C', we have $c_2 \Rightarrow' c_3$, by definition of $\Rightarrow'$. Thus, $\Rightarrow'$ is $\overline{\leq}$-monotonic because $c_3 \overline{\leq} c_3$. □

Example 5.2 *Let us consider once again the Petri net $\mathcal{N}_\mu$ introduced in Example 2.8 (Figure 2.1). Figure 5.2 shows* $\mathsf{Under}\left(\mathsf{lossy}(\mathcal{S}_{\mathcal{N}_\mu}), C'\right)$ *for* $C' = \{\mathbf{m} \mid \forall p \in \{p_1, p_2, p_3\} : \mathbf{m}(p) \leq 2\}$. *Remark that, unlike Example 5.1, we have this time considered the lossy version of $\mathcal{S}_{\mathcal{N}_\mu}$. It is not difficult to see that $\Rightarrow$ is now (simply) $\overline{\leq}$-monotonic. Hence* $\mathsf{Under}\left(\mathsf{lossy}(\mathcal{S}_{\mathcal{N}_\mu}), C'\right)$ *is a finite* WSTS.

'Enlarge' phase Let us now discuss the relevance of finite WSTS during the 'Enlarge' phase. Let $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq}\right\rangle$ be a WSTS with adequate domain of limits $\langle L, \sqsubseteq, \gamma\rangle$. Let $C'' \subseteq C$ and $L'' \subseteq L$ be two finite sets s.t. $c_0 \in C''$ and $\top \in L''$. Clearly, not every over-approximation $\mathsf{Over}\,(\mathcal{S}, C'', L'')$ is a finite WSTS, since over-approximations are, in general, (non-degenerated) And-Or graphs. However, when $\mathsf{Over}\,(\mathcal{S}, C'', L'')$ is a *degenerated And-Or graph*, the avoidability problem of U in $\mathsf{Over}\,(\mathcal{S}, C'', L'')$ can be reduced to the coverability problem of a finite WSTS $\mathsf{OverWSTS}\,(\mathcal{S}, C'', L'')$, defined hereunder.

Given an element $c \in L'' \cup C''$ let $\mathsf{mps}(c)$ be the most precise set of elements of $L'' \cup C''$ that covers $\mathsf{Post}\,(\gamma(c))$. That is, $\mathsf{mps}(c) \subseteq L'' \cup C''$ is s.t. $\mathsf{Post}\,(\gamma(c)) \subseteq \gamma(\mathsf{mps}(c))$ and, there is no $D \subseteq L'' \cup C''$ s.t. $\mathsf{Post}\,(\gamma(c)) \subseteq \gamma(D) \subset \gamma(\mathsf{mps}(c))$. If $\langle C'', L'' \rangle$ is a perfect pair, the set $\mathsf{mps}(c)$ is unique for any $c \in L'' \cup C''$, by Definition 4.6. Moreover, it always exists, since we impose that $\top \in L''$ in the EEC algorithm, and $\mathsf{Over}\,(\mathcal{S}, C'', L'')$ is degenerated by Lemma 4.1.

We can now define $\mathsf{OverWSTS}\,(\mathcal{S}, C'', L'') = \left\langle C', c_0', \Rightarrow', \preceq' \right\rangle$:

- $C' = L'' \cup C''$;
- $c_0' = c_0$;
- for any c_1, c_2 in C': $c_1 \Rightarrow' c_2$ iff $c_2 \in \mathsf{mps}(c_1)$;
- $\preceq' = \sqsubseteq \cap (C' \times C')$.

Let us show that $\mathsf{OverWSTS}\,(\mathcal{S}, C'', L'')$ is indeed a finite WSTS:

Lemma 5.2 *Let $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \preceq \right\rangle$ be a* WSTS *and let $\langle L, \sqsubseteq, \gamma \rangle$ be an adequate domain of limits for $\left\langle C, \preceq \right\rangle$. Let C'' be a finite subset of C s.t. $c_0 \in C''$, and let L'' be a finite subset of L s.t. $\top \in L''$. If $\langle C'', L'' \rangle$ is a perfect pair, then* $\mathsf{OverWSTS}\,(\mathcal{S}, C'', L'') = \left\langle C', c_0', \Rightarrow', \preceq' \right\rangle$ *is a finite, simply monotonic* WSTS.

Proof. Clearly, $\langle C', c_0', \Rightarrow' \rangle$ is a finite transition system, and $\preceq'$ is a WQO. Let us show that $\Rightarrow'$ is (simply) $\preceq'$-monotonic. Let c_1, c_2 and c_3 be three configurations in C' s.t. $c_1 \Rightarrow' c_2$ and $c_1 \preceq' c_3$, and let us show that there exists $c_4 \in C'$ s.t. $c_3 \Rightarrow' c_4$ and $c_2 \preceq' c_4$.

Since $c_1 \preceq' c_3$, we have $c_1 \sqsubseteq c_3$ (by definition of $\preceq'$) and thus $\gamma(c_1) \subseteq \gamma(c_3)$ (by definition of $\sqsubseteq$). Hence, since $\Rightarrow$ is monotonic, $\mathsf{Post}\,(\gamma(c_1)) \subseteq \mathsf{Post}\,(\gamma(c_3))$. By definition of mps, $\mathsf{Post}\,(\gamma(c_3)) \subseteq \gamma(\mathsf{mps}\,(c_3))$. Thus, $\mathsf{Post}\,(\gamma(c_1)) \subseteq \gamma(\mathsf{mps}\,(c_3))$. By definition of mps, this implies that $\gamma(\mathsf{mps}\,(c_1)) \subseteq \gamma(\mathsf{mps}\,(c_3))$. Hence, for any $c \in \mathsf{mps}(c_1)$, there is $c' \in \mathsf{mps}(c_3)$ s.t. $c \sqsubseteq c'$, which is equivalent to $c \preceq' c'$. However, $c_1 \Rightarrow' c_2$ implies that $c_2 \in \mathsf{mps}(c_1)$. Hence there is $c_4 \in \mathsf{mps}(c_3)$ s.t. $c_2 \preceq' c_4$. Finally, since $c_4 \in \mathsf{mps}(c_3)$, we have $c_3 \Rightarrow' c_4$, by definition of $\Rightarrow'$. □

Remark that $\mathsf{OverWSTS}\,(\mathcal{S}, C'', L'')$ corresponds to the (plain) graph that is naturally associated to the degenerated And-Or graph $\mathsf{Over}\,(\mathcal{S}, C'', L'')$. The set of configurations of $\mathsf{OverWSTS}\,(\mathcal{S}, C'', L'')$ is the set of Or-nodes of $\mathsf{Over}\,(\mathcal{S}, C'', L'')$. The And-nodes have been omitted since they are redundant in the case of a degenerated And-Or graph. The transition relation has been adapted accordingly. That is, for any $c_1, c_2 \in C'$, we have $c_1 \Rightarrow' c_2$ in $\mathsf{OverWSTS}\,(\mathcal{S}, C'', L'')$ iff there are two Or-nodes $v_1 = c_1$ and $v_2 = c_2$ and an And-node $v_3 = \mathsf{mps}(c_1)$ in the And-Or graph, such that $v_1 \Rightarrow'' v_3 \Rightarrow'' v_2$, where $\Rightarrow''$ denotes the transition relation of the And-Or graph. As a consequence, testing for the avoidability of U in the And-Or graph is equivalent to an instance of the coverability problem, where the transition system is $\mathsf{OverWSTS}\left(\mathcal{S}, C'', L''\right)$, and the upward-closed set is $U' = \{d \in L'' \cup C'' \mid \gamma(d) \cap U \neq \emptyset\}$. That is, it is the set of the elements of $L'' \cup C''$ whose associated downward-closed set (wrt to γ) has a non-empty intersection with U. Remark that for any element $d \in U'$, any $d' \in L'' \cup C''$ s.t. $d \sqsubseteq d'$ is in U' too, because $d \sqsubseteq d'$ implies that $\gamma\left(d\right) \subseteq \gamma\left(d'\right)$. Thus, U' is indeed upward-closed wrt to $\sqsubseteq$. Hence, it is also $\overline{\leq}'$-upward-closed, by definition of $\overline{\leq}'$. Thus, we obtain the following proposition:

Proposition 5.2 *Let $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq}\right\rangle$ be a* WSTS *with adequate domain of limits $\langle L, \sqsubseteq, \gamma\rangle$. Let C'' be a finite subset of C s.t. $c_0 \in C''$, and let L'' be a finite subset of L s.t. $\top \in L''$. Let $\langle C'', L''\rangle$ be a perfect pair. Let $U \subseteq C$ be a $\overline{\leq}$-upward-closed set. Then, U is avoidable in* $\mathsf{Over}\left(\mathcal{S}, C'', L''\right)$ *iff* $\gamma\left(\mathsf{OverWSTS}\left(\mathcal{S}, C'', L''\right)\right) \cap U = \emptyset$.

Proof. Let $\mathsf{OverWSTS}\left(\mathcal{S}, C'', L''\right) = \left\langle C', c_0', \Rightarrow', \overline{\leq}'\right\rangle$. Since $\langle C'', L''\rangle$ is a perfect pair, $\mathsf{Over}\left(\mathcal{S}, C'', L''\right) = \langle V_A, V_O, v_i, \Rightarrow''\rangle$ is degenerated, by Lemma 4.1. Hence, it has a single[1] unfolding $\langle N, root, B, \Lambda\rangle$ that intersects with U iff U cannot be avoided in $\mathsf{Over}\left(\mathcal{S}, C'', L''\right)$. Remark that for every And-node $n \subseteq C'' \cup L''$ in any unfolding, for every $d \in n$, there is an Or-node $n' = d$ in the unfolding (see Definition 4.5 and Definition 4.3). Hence, And-nodes are redundant when checking for the intersection of the unfolding with a given upward-closed set. Thus:

$$\gamma\left(\bigcup_{n\in N} \Lambda\left(n\right)\right) \cap U = \emptyset \quad \text{iff} \quad \gamma\left(\bigcup_{n\in N\cap V_O} \Lambda\left(n\right)\right) \cap U = \emptyset \tag{5.1}$$

Let V' be the set of every Or-node of $\mathsf{Over}\left(\mathcal{S}, C'', L''\right)$ that are reachable from its initial node v_i. Thus, by construction, V' is equal to the set $\mathsf{Reach}\left(\mathsf{OverWSTS}\left(\mathcal{S}, C'', L''\right)\right)$. Hence, we have:

$$\begin{array}{lll}
 & U \text{ is avoidable in } \mathsf{Over}\left(\mathcal{S}, C'', L''\right) & \\
\Leftrightarrow & \forall n \in N : \gamma(n) \cap U = \emptyset & \text{Def. of avoidability} \\
\Leftrightarrow & \forall v \in V' : \gamma(v) \cap U = \emptyset & \text{By (5.1)} \\
\Leftrightarrow & \forall c \in \mathsf{Reach}\left(\mathsf{OverWSTS}\left(\mathcal{S}, C'', L''\right)\right) : \gamma(c) \cap U = \emptyset & \\
 & & V' = \mathsf{Reach}\left(\mathsf{OverWSTS}\left(\mathcal{S}, C'', L''\right)\right) \\
\Leftrightarrow & \gamma\left(\mathsf{Reach}\left(\mathsf{OverWSTS}\left(\mathcal{S}, C'', L''\right)\right)\right) \cap U = \emptyset & \text{Def. of U'}
\end{array}$$

□

It should now be clear that an efficient procedure to compute $\gamma\left(\mathsf{Reach}\left(\mathcal{S}\right)\right)$, for any finite WSTS $\mathcal{S}$, is highly desirable in order to improve the practical efficiency of EEC. We discuss such a procedure in the following section.

[1]Up to labelled tree isomorphism.

5.3 An efficient procedure to decide coverability on finite WSTS

This section presents an efficient procedure to decide the coverability problem when dealing with *finite* WSTS, i.e., WSTS with a finite number of reachable configurations. The algorithm exploits the *monotonicity* property of WSTS in order to reduce the number of elements that have to be treated. More precisely, the algorithm computes a sequence of *downward-closed sets* of elements that converges to the coverability set of the WSTS (which is sufficient to decide coverability as well as other problems, see Section 2.6). These downward-closed sets are represented and manipulated by means of sets of maximal elements, which are usually smaller than the sets they represent. Since we are dealing with finite WSTS, there is no need for *limit elements*, because no infinite increasing sequence of reachable elements will appear (see Lemma 2.10).

Let us consider a finite WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$ and the function $F(X) = \mathsf{Post}\,(X) \cup \{c_0\}$ that is defined for any set $X \subseteq C$. Since F is a monotonic function for $\subseteq$, the least fixed point $\mu X.F(X)$ of F exists, is unique, and is equal to $\mathsf{Post}^*\,(c_0)$, by Knaster–Tarski's theorem.

Let $X_0, X_1, \ldots X_i, \ldots$ be the sequence of subsets of C defined as follows:

Definition 5.4 (The FinCov sequence) *Let $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$ be a WSTS. Then, $\mathsf{FinCov}\,(\mathcal{S})$ is the sequence $X_0, X_1, \ldots, X_i, \ldots$ defined as follows:*

1. *$X_0 = \{c_0\}$;*

2. *for any $i \geq 1$: $X_{i+1} = \mathsf{Post}\,(X_i) \cup \{c_0\}$.* ■

Clearly, that sequence eventually converges to $\mathsf{Post}^*\,(c_0)$ for any finite WSTS. Thus, the sequence of X_i's we have just defined provides us a practical way to compute $\mathsf{Post}^*\,(c_0)$, and hence, to decide the coverability problem. It is presented at Algorithm 5.1. From the discussion above, we have:

Proposition 5.3 *Let $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq} \right\rangle$ be a finite WSTS and let $X_0, X_1, \ldots, X_i, \ldots$ be its covering sequence $\mathsf{FinCov}\,(\mathcal{S})$. Then, there exists $k \in \mathbb{N}$ s.t.:*

- *for any $1 \leq i < k$: $X_i \subset X_{i+1}$;*

- *$X_k = X_{k+1} = \mathsf{Post}^*\,(c_0)$.*

An immediate corollary of this proposition is:

Corollary 5.2 *For any finite* WSTS $\mathcal{S}$ *and any upward-closed set* U*, Algorithm 5.1 terminates and answers 'Reachable' iff* $U \cap \mathsf{Reach}(\mathcal{S}) \neq \emptyset$.

Algorithm 5.1: A simple fixed-point algorithm to decide the coverability problem on finite WSTS.

```
Data: A WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$
Data: A (finite) generator $\mathsf{UGen}(U)$ of a $\overline{\leq}$-upward-closed $U \subseteq C$
Result: Reachable iff $U \cap \mathsf{Reach}(\mathcal{S}) \neq \emptyset$
$X_0 \leftarrow \{c_0\}$ ;
$i \leftarrow 0$ ;
repeat
    $i \leftarrow i + 1$ ;
    $X_i \leftarrow \mathsf{Post}(X_{i-1}) \cup \{c_0\}$ ;
until $X_i = X_{i-1}$ ;
if $\exists c \in U, c' \in X_i : c \overline{\leq} c'$ then return 'Reachable' ;
else return 'Unreachable' ;
```

Let us show how this algorithm can be improved by keeping sets of $\overline{\leq}$-maximal elements only. First, remember that, by Proposition 3.3, the covering set $\mathsf{Cover}(\mathcal{S}) = {\downarrow}(\mathsf{Reach}(\mathcal{S}))$ is suitable to decide the coverability problem, because $\mathsf{Reach}(\mathcal{S}) \cap U = \emptyset$ iff $\mathsf{Cover}(\mathcal{S}) \cap U = \emptyset$. Since $\mathcal{S}$ is finite, $\mathsf{Cover}(\mathcal{S})$ is finite too, because $\mathsf{Cover}(\mathcal{S}) \subseteq C$, by definition. Hence, by Lemma 2.10, it is representable by $\mathsf{Max}^{\overline{\leq}}(\mathsf{Cover}(\mathcal{S}))$.

The idea of our improved algorithm consists in building a sequence of $\overline{\leq}$-downward-closed sets that eventually converges to $\mathsf{Cover}(\mathcal{S})$. Each downward-closed set is represented by a set of $\overline{\leq}$–maximal elements. Thus, in practice, the algorithm computes the sequence, $\overline{X}_0, \overline{X}_1, \ldots \overline{X}_n$ (defined hereunder) of subsets of C s.t. for any i, ${\downarrow}(X_i) = {\downarrow}(\overline{X}_i)$. Each set $\overline{X}_i$ is obtained from $\overline{X}_{i-1}$ by computing the Post operation and keeping $\overline{\leq}$-maximal elements only. Since the sets of maximal elements are typically smaller than the sets they represent, the practical performance of the algorithm is improved (see Section 5.4.5 and Section 5.5.5). This is the definition of the new sequence:

Definition 5.5 (The FinCovMax sequence) *Let* $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq} \rangle$ *be a* WSTS*. Then,* $\mathsf{FinCovMax}(\mathcal{S})$ *is the sequence* $\overline{X}_0, \overline{X}_1, \ldots, \overline{X}_i, \ldots$ *defined as follows:*

1. $\overline{X}_0 = \{c_0\}$;

2. *for any* $i \geq 1$: $\overline{X}_{i+1} = \mathsf{Max}^{\overline{\leq}}\left(\mathsf{Post}\left(\overline{X}_i\right) \cup \{c_0\}\right)$. ■

Let us show that $\mathsf{FinCovMax}(\mathcal{S})$ corresponds to $\mathsf{FinCov}(\mathcal{S})$ as far as the downward-closure is concerned. We first prove the following auxiliary Lemma:

Lemma 5.3 *Let $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq}\right\rangle$ be a* WSTS *and let $A \subseteq C$ and $B \subseteq C$ be s.t. $\downarrow(A) = \downarrow(B)$. Then $\downarrow(\mathsf{Post}(A)) = \downarrow(\mathsf{Post}(B))$.*

Proof. Let a be an element of A since $\downarrow(A) = \downarrow(B)$, there exists $b \in B$ s.t. $a \overline{\leq} b$. By monotony of WSTS, for any $a' \in \mathsf{Post}(a)$, there exists $b' \in \mathsf{Post}(b)$ s.t. $a' \overline{\leq} b'$. Hence $\downarrow(\mathsf{Post}(A)) \subseteq \downarrow(\mathsf{Post}(B))$. By a symmetrical reasoning, we prove that $\downarrow(\mathsf{Post}(B)) \subseteq \downarrow(\mathsf{Post}(A))$. Hence, $\downarrow(\mathsf{Post}(A)) = \downarrow(\mathsf{Post}(B))$. □

We can now show the correctness and adequacy of $\mathsf{FinCovMax}(\mathcal{S})$:

Proposition 5.4 *For any* WSTS $\mathcal{S} = \left\langle C, c_0, \Rightarrow, \overline{\leq}\right\rangle$ *s.t.:*

$$\mathsf{FinCov}(\mathcal{S}) = X_0, X_1, \ldots, X_i, \ldots$$

and

$$\mathsf{FinCovMax}(\mathcal{S}) = \overline{X}_0, \overline{X}_1, \ldots, \overline{X}_i, \ldots$$

the following holds:

$$\forall i \geq 1 : \downarrow(X_i) = \downarrow\left(\overline{X}_i\right)$$

Proof. The proof is by induction on i.

Base case $i = 0$: trivial.

Inductive case $i = k+1$: by induction hypothesis $\downarrow(X_k) = \downarrow\left(\overline{X}_k\right)$. Thus:

$$\begin{array}{lll}
 & \downarrow(X_k) = \downarrow\left(\overline{X}_k\right) & \\
\Rightarrow & \downarrow(\mathsf{Post}(X_k)) = \downarrow\left(\mathsf{Post}\left(\overline{X}_k\right)\right) & \text{by Lemma 5.3} \\
\Rightarrow & \downarrow(\mathsf{Post}(X_k)) \cup \downarrow(\{c_0\}) = \downarrow\left(\mathsf{Post}\left(\overline{X}_k\right)\right) \cup \downarrow(\{c_0\}) & \\
\Rightarrow & \downarrow(\mathsf{Post}(X_k) \cup \{c_0\}) = \downarrow\left(\mathsf{Post}\left(\overline{X}_k\right) \cup \{c_0\}\right) & \\
\Rightarrow & \downarrow(\mathsf{Post}(X_k) \cup \{c_0\}) = \downarrow\left(\mathsf{Max}^{\overline{\leq}}\left(\mathsf{Post}\left(\overline{X}_k\right) \cup \{c_0\}\right)\right) & \text{Def. of } \mathsf{Max} \\
\Rightarrow & \downarrow(X_{k+1}) = \downarrow\left(\overline{X}_{k+1}\right) &
\end{array}$$

□

Thus, Algorithm 5.1 can be improved in practice by keeping maximal elements only during the computation. This amounts to replace the line:

$$X_i \leftarrow \mathsf{Post}(X_{i-1}) \cup \{c_0\}$$

by:

$$X_i \leftarrow \mathsf{Max}^{\overline{\leq}}(\mathsf{Post}(X_{i-1}) \cup \{c_0\})$$

in Algorithm 5.1. As we will see in the following sections, this solution improves the practical efficiency of EEC.

5.4 Application to strongly monotonic SMPN

This section shows how the EEC algorithmic schema can be instantiated to the class of strongly monotonic SMPN. First of all, we rely on the adequate domain of limits $\langle \mathcal{L}, \preccurlyeq_e, \gamma \rangle$ for $\langle \mathbb{N}^k, \preccurlyeq \rangle$ that has been defined in section 3.1.1 (see Theorem 3.1). Then, remember that the extension of the transition relation, defined in Section 3.1.1, is exact (in the sense of Proposition 3.2). As a consequence, we show in Section 5.4.1, that we have an effective domain of limits. Then we define in Section 5.4.2 the sets C_i's and L_i's that EEC may consider to build the under– and over–approximations. It turns out that in the case of SMPN, And-Or graphs are *degenerated*. Thus, under– and over–approximations can be represented by finite WSTS, whose coverability problem can be decided efficiently thanks to the algorithm of Section 5.3. In Section 5.4.3, we rely on all these results to instantiate the EEC algorithmic schema to the case of strongly monotonic SMPN.

5.4.1 Domain of limits

Thanks to the discussion of the previous section, we can show that our domain of limits $\langle \mathcal{L}, \preccurlyeq_e, \gamma \rangle$ meets the effectiveness requirement of Definition 4.1.

Theorem 5.1 *Any $\preccurlyeq$-strongly monotonic SMPN $\mathcal{N}$ with the adequate domain of limits $\langle \mathcal{L}, \preccurlyeq_e, \gamma \rangle$ are effective.*

Proof. Points E_1, E_2 and E_4 are obvious. E_3 stems from the fact that $\mathsf{Post}(\gamma(\mathbf{m})) \subseteq \gamma(M)$ iff for any $\mathbf{m}'$ with $\mathbf{m} \to \mathbf{m}'$, there exists $\mathbf{m}'' \in M$ with $\mathbf{m}' \preccurlyeq_e \mathbf{m}''$, by Lemma 3.2 (remark that there are finitely many $\mathbf{m}'$ s.t. $\mathbf{m} \to \mathbf{m}'$). □

5.4.2 Construction of the C_i's and L_i's

The following definition explains how we construct the C_i's and L_i's.

Definition 5.6 (Sequences of C_i and L_i for SMPN) *For any $i \geq 0$:*

- $C_i = \{0, \ldots, i\}^k \cup \{\mathbf{m}_0\}$, *i.e. C_i is the set of markings where each place is bounded by i (plus the initial marking);*
- $L_i = \{\mathbf{m} \in \{0, \ldots i, \omega\}^k \mid \mathbf{m} \notin \mathbb{N}^k\}$, *i.e., L_i is the set of all the ω-markings $\mathbf{m}$ s.t. there is $p \in P$ with $\mathbf{m}(p) = \omega$, and all the places p' s.t. $\mathbf{m}(p') \neq \omega$ contain at most i tokens.* ■

It is easy to see that:

1. for all $i \geq 0 : C_i \subset C_{i+1}$ and $L_i \subset L_{i+1}$;
2. for any $\mathbf{m} \in \mathbb{N}^k$, there exists $i \in \mathbb{N}$ such that for all $j \geq i : \mathbf{m} \in C_j$;
3. for any $\mathbf{m} \in \mathcal{L}$, there exists $i \in \mathbb{N}$ such that for all $j \geq i : \mathbf{m} \in L_j$;
4. $\mathbf{m}_0 \in C_0$;
5. $\top = \langle \omega, \ldots, \omega \rangle \in L_0$.

Degenerated And-Or graph Let us show that in the present case, one obtains a *degenerated* And-Or graph. We establish this result by showing, following Lemma 4.1, that the pairs $\langle C_i, L_i \rangle$ are *perfect* pairs. For this purpose, we first introduce the function $\mathsf{Bound}\,(\mathbf{m}, k)$ and establish an auxiliary lemma about this function. Given an ω-marking $\mathbf{m}$ over set of places P and $k \in \mathbb{N}$, we define $\mathsf{Bound}\,(\mathbf{m}, k) : \big(\mathbb{N} \cup \{\omega\}\big)^{|P|} \times \mathbb{N} \mapsto \big(\mathbb{N} \cup \{\omega\}\big)^{|P|}$ as follows:

$$\forall p_i \in P : \mathsf{Bound}\,(\mathbf{m}, k)\,(p_i) = \begin{cases} \mathbf{m}(p_i) & \text{if } \mathbf{m}(p_i) \leq k \\ \omega & \text{otherwise} \end{cases}$$

Given that definition, it should be clear, that, for any ω-marking $\mathbf{m}$, $\mathbf{m} \preccurlyeq_e \mathsf{Bound}\,(\mathbf{m}, i)$. Moreover, $\big\{\mathsf{Bound}\,(\mathbf{m}, i)\,\big\}$ is the most precise subset of $L_i \cup C_i$ to over-approximate of $\mathbf{m}$, as shown by the following Lemma:

Lemma 5.4 *Given any $i \in \mathbb{N}$, let C_i and L_i be constructed following Definition 5.6 and let $\mathbf{m}$ be an ω-marking. There does not exist a finite set $S \subseteq L_i \cup C_i$ such that $\gamma(\mathbf{m}) \subseteq \gamma(S)$ and $\gamma(\mathsf{Bound}\,(\mathbf{m}, i)) \not\subseteq \gamma(S)$.*

Proof. Per absurdum. Let us assume there exists $S \subseteq L_i \cup C_i$ s.t. $\gamma\,(\mathbf{m}) \subseteq \gamma\,(S)$ and $\gamma\,(\mathsf{Bound}\,(\mathbf{m}, i)) \not\subseteq \gamma\,(S)$. Remark that S is finite since $L_i \cup C_i$ is finite by hypothesis. Since $\gamma\,(\mathbf{m}) \subseteq \gamma\,(S)$, there is $\mathbf{m}' \in S$ s.t. $\gamma\,(\mathbf{m}) \subseteq \gamma\,(\mathbf{m}')$, by Proposition 3.1. Moreover, $\mathbf{m}' \in L_i \cup C_i$, because $S \subseteq L_i \cup C_i$. Let us show that $\gamma\,(\mathsf{Bound}\,(\mathbf{m}, i)) \subseteq \gamma\,(\mathbf{m}')$. For that purpose, we show that, for any place p, $\mathsf{Bound}\,(\mathbf{m}, i)\,(p) \leq \mathbf{m}'(p)$. First, observe that, since $\gamma\,(\mathbf{m}) \subseteq \gamma\,(\mathbf{m}')$, we have $\mathbf{m}(p) \leq \mathbf{m}'(p)$, for any place p, by definition of γ. Then, we consider several cases:

1. If $\mathbf{m}(p) = \omega$, then, $\mathbf{m}'(p) = \omega$ because $\mathbf{m}'(p) \geq \mathbf{m}(p)$. On the other hand, by definition, $\mathsf{Bound}\,(\mathbf{m}, i)\,(p) = \omega$. Hence, $\mathsf{Bound}\,(\mathbf{m}, i)\,(p) = \mathbf{m}'(p)$.
2. If $i + 1 \leq \mathbf{m}(p) < \omega$, then, $\mathbf{m}'(p) = \omega$, because, $\mathbf{m}'(p) \geq \mathbf{m}(p)$ and $\mathbf{m}'(p) \in \{1, 2, \ldots, i, \omega\}$, since $\mathbf{m}' \in L_i \cup C_i$. Moreover, $\mathsf{Bound}\,(\mathbf{m}, i)\,(p) = \omega$ by definition. Hence, $\mathsf{Bound}\,(\mathbf{m}, i)\,(p) = \mathbf{m}'(p)$.
3. If $0 \leq \mathbf{m}(p) \leq i$, then $\mathsf{Bound}\,(\mathbf{m}, i)\,(p) = \mathbf{m}(p)$. Hence, $\mathsf{Bound}\,(\mathbf{m}, i)\,(p) = \mathbf{m}(p) \leq \mathbf{m}'(p)$.

In any case, $\mathsf{Bound}(\mathbf{m}, i)(p) \leq \mathbf{m}'(p)$. Since this holds for any place p, we have $\mathsf{Bound}(\mathbf{m}, i) \preccurlyeq_e \mathbf{m}'$. Since $\mathbf{m}' \in S$, we conclude that $\gamma(\mathsf{Bound}(\mathbf{m}, i)) \subseteq \gamma(S)$. Contradiction. □

Let us now extend the definition of Bound to sets of ω-markings. For any $i \geq 1$, for any $S \subseteq (\mathbb{N} \cup \{\omega\})^{|P|}$, $\mathsf{Bound}(S, i) = \{\mathsf{Bound}(\mathbf{m}, i) \mid \mathbf{m} \in S\}$. As a consequence of Lemma 5.4, we obtain the following lemma, that states that $\gamma(\mathsf{Bound}(D, i))$ is the most precise downward-closed set of markings that is representable in $L_i \cup C_i$ and that covers D.

Lemma 5.5 *Given any $i \in \mathbb{N}$, let C_i and L_i be constructed following Definition 5.6. Let D be a finite set of ω-markings. Then:*

1. $\gamma(D) \subseteq \gamma(\mathsf{Bound}(D, i))$ *and*
2. *there is no $S \subseteq L_i \cup C_i$ s.t. $\gamma(D) \subseteq \gamma(S)$ and $\gamma(\mathsf{Bound}(D, i)) \not\subseteq \gamma(S)$.*

Proof. If $D = \emptyset$, then $\mathsf{Bound}(D, i) = \emptyset$ for any $i \geq 1$ and the Lemma is trivial. Otherwise, we prove the two points separately.

$\gamma(D) \subseteq \gamma(\mathsf{Bound}(D, i))$ follows from the fact that, for any $\mathbf{m} \in D$, $\mathsf{Bound}(\mathbf{m}, i) \in \mathsf{Bound}(D, i)$ (by definition of Bound on sets of ω-markings) and $\mathbf{m} \preccurlyeq_e \mathsf{Bound}(\mathbf{m}, i)$ (by definition of Bound on ω-markings).

Let S be a subset of $L_i \cup C_i$ and let us establish that $\gamma(\mathsf{Bound}(D, i)) \subseteq \gamma(S)$. Let $S' \subseteq S$ be the set of all the ω-markings of S that cover some ω-marking of D, i.e., $S' = \{\mathbf{m} \in S \mid \exists \mathbf{m}' \in D : \mathbf{m}' \preccurlyeq_e \mathbf{m}\}$. Since S and D are both subsets of $L_i \cup C_i$ and since $D \neq \emptyset$, we have $S' \neq \emptyset$. Since $S' \subseteq S$, we have $\gamma(S') \subseteq \gamma(S)$. Let $\mathbf{m}$ be a marking of S' and let $\mathbf{m}'$ be a marking of D s.t. $\mathbf{m}' \preccurlyeq_e \mathbf{m}$. By Lemma 5.4, we have $\mathbf{m}' \preccurlyeq_e \mathsf{Bound}(\mathbf{m}', i) \preccurlyeq_e \mathbf{m}$, since $\mathbf{m} \in S' \subseteq L_i \cup c_i$. Since this holds for any $\mathbf{m} \in S'$, we have $\subseteq \gamma(\cup_{\mathbf{m} \in D} \mathsf{Bound}(\mathbf{m}, i)) = \gamma(\mathsf{Bound}(D, i)) \subseteq \gamma(S')$. Hence, $\gamma(\mathsf{Bound}(D, i)) \subseteq \gamma(S)$. □

We can now prove that the pairs $\langle C_i, L_i \rangle$ constructed according to Definition 5.6 are *perfect pairs*.

Lemma 5.6 *Given a* SMPN $\mathcal{N} = \langle P, T, D^-, D^+ \rangle$ *with the adequate domain of limits $\langle \mathcal{L}, \preccurlyeq_e, \gamma \rangle$ and the sets $C_i \subseteq \mathbb{N}^{|P|}$ and $L_i \subseteq \mathcal{L}$ constructed following Definition 5.6, any pair $\langle C_i, L_i \rangle$ is a perfect pair.*

Proof. According to Definition 4.6, we have to show that, for any $\mathbf{m} \in L_i \cup C_i$, there exists one and only one canonical set $D \subseteq C_i \cup L_i$ s.t.

1. $\mathsf{Post}(\gamma(\mathbf{m})) \subseteq \gamma(D)$ and

2. there is no $S \subseteq L_i \cup C_i$ with $\mathsf{Post}(\gamma(\mathbf{m})) \subseteq \gamma(S) \subset \gamma(D)$.

First, remark that $\preccurlyeq_e$ is an antisymmetric WQO. This follows from the fact that the definition of $\preccurlyeq_e$ is based on $\leq$ (extended to $\mathbb{N} \cup \{\omega\}$), which is antisymmetric.

Let us first show that there exists at least one canonical set $D \subseteq L_i \cup C_i$ that respects the two points of the definition of perfect pair (the unicity will be established later). For that purpose, we choose $D = \mathsf{Max}^{\preccurlyeq_e}(\mathsf{Bound}(\mathsf{Post}(\mathbf{m}), i))$. Remark that D is canonical by Lemma 2.3 because $\preccurlyeq_e$ is antisymmetric, and we keep $\preccurlyeq_e$-maximal elements only in D.

Let us establish the first point of the definition of perfect pairs, i.e., $\mathsf{Post}(\gamma(\mathbf{m})) \subseteq \gamma(D)$. By definition of Bound and $\mathsf{Max}^{\preccurlyeq_e}$, for any $\mathbf{m}'$ in $\mathsf{Post}(\mathbf{m})$, there is $\mathbf{m}'' \in D$ s.t. $\mathbf{m}' \preccurlyeq_e \mathbf{m}''$. Let $\overline{\mathbf{m}}$ be a marking of $\mathsf{Post}(\gamma(\mathbf{m}))$ (remark that $\overline{\mathbf{m}}$ necessarily a marking, since $\gamma(\mathbf{m})$ is a set of markings only). By Proposition 3.2 (the extension of the transition relation to ω-marking is exact), this means that there is an ω-marking $\mathbf{m}' \in \mathsf{Post}(\mathbf{m})$ s.t. $\overline{\mathbf{m}} \preccurlyeq_e \mathbf{m}'$. Thus, there is $\mathbf{m}'' \in D$ s.t. $\overline{\mathbf{m}} \preccurlyeq_e \mathbf{m}''$. Since this holds for any marking $\overline{\mathbf{m}} \in \mathsf{Post}(\gamma(\mathbf{m}))$, we conclude that $\mathsf{Post}(\gamma(\mathbf{m})) \subseteq \gamma(D)$.

Then, in order to establish the second point of the definition of perfect pairs, we consider $S \subseteq L_i \cup C_i$ s.t. $\mathsf{Post}(\gamma(\mathbf{m})) \subseteq \gamma(S)$, and show that $\gamma(D) \subseteq \gamma(S)$. Remark that $\mathsf{Post}(\gamma(\mathbf{m})) \subseteq \mathbb{N}^{|P|}$, because $\gamma(\mathbf{m}) \subseteq \mathbb{N}^{|P|}$. Moreover, $\gamma(S)$ is a downward-closed set of $\mathbb{N}^{|P|}$ too, by definition of γ. Hence, $\mathsf{Post}(\gamma(\mathbf{m})) \subseteq \gamma(S)$ implies that $\gamma(\mathsf{Post}(\gamma(\mathbf{m}))) \subseteq \gamma(S)$. Since $S \subseteq L_i \cup C_i$ and $\gamma(\mathsf{Post}(\gamma(\mathbf{m}))) \subseteq \gamma(S)$, we have $\gamma(\mathsf{Bound}(\mathsf{Post}(\gamma(\mathbf{m})), i)) \subseteq \gamma(S)$, by Lemma 5.5. However, for any set S', we have: $\gamma\left(\mathsf{Max}^{\preccurlyeq_e}(S')\right) = \gamma(S')$, by definition of γ. Hence:

$$\gamma(\mathsf{Bound}(\mathsf{Post}(\gamma(\mathbf{m})), i)) = \gamma\left(\mathsf{Max}^{\preccurlyeq_e}(\mathsf{Bound}(\mathsf{Post}(\gamma(\mathbf{m})), i))\right) = \gamma(D)$$

and we conclude that $\gamma(D) \subseteq \gamma(S)$.

Thus, $\gamma(D)$ is a most precise downward-closed set of markings that is representable by elements of $L_i \cup C_i$ and that covers $\mathsf{Post}(\gamma(d))$. Let us conclude the proof by showing that D is the sole canonical representation (in $L_i \cup C_i$) of $\gamma(D)$. That part is *per absurdum* and relies on the fact that $\preccurlyeq_e$ is antisymmetric. Let us assume that there is $D' \subseteq L_i \cup C_i$ s.t. $D' \neq D$, D' is canonical and $\gamma(D) = \gamma(D')$. Since $D' \neq D$, there is, without loss of generality, $\mathbf{m} \in D \setminus D'$. Since $\gamma(D) = \gamma(D')$, there is $\mathbf{m}' \in D'$ s.t. $\mathbf{m} \preccurlyeq_e \mathbf{m}'$. Since $\mathbf{m} \notin D'$, we have $\mathbf{m} \neq \mathbf{m}'$. Hence, $\mathbf{m} \prec_e \mathbf{m}'$ because $\preccurlyeq_e$ is antisymmetric. Since $\gamma(D) = \gamma(D')$, again, there is $\mathbf{m}'' \in D$ s.t. $\mathbf{m} \preccurlyeq_e \mathbf{m}''$. We conclude that there are $\mathbf{m}$ and $\mathbf{m}''$ in D s.t. $\mathbf{m} \prec_e \mathbf{m}''$. Hence, D is not canonical. Contradiction.

□

From Lemma 5.6 and Lemma 4.1, the following corollary holds:

Corollary 5.3 *Given a $\preccurlyeq$-strongly monotonic* SMPN *net $\mathcal{N}$ with the adequate domain of limits $\langle \mathcal{L}, \preccurlyeq_e, \gamma \rangle$ and the sets $C_i \subseteq \mathbb{N}^{k_P}$ and $L_i \subseteq \mathcal{L}$ constructed following Definition 5.6,* Over $(\mathcal{N}, C_i, L_i)$ *is a degenerated And-Or graph.*

This result makes the task of building under– and over–approximation much easier, since the bound function provides a way to directly compute the unique and most precise over-approximation of any ω-marking.

5.4.3 Algorithm for the coverability problem

Let $C_0, C_1, \ldots, C_i, \ldots$ and $L_0, L_1, \ldots, L_i, \ldots$ be the sequences of sets of configurations and limit elements defined at Definition 5.6. By applying the schema presented in Section 4.3 to $\preccurlyeq$-strongly monotonic self-modifying Petri nets, we obtain Algorithm 5.2.

Algorithm 5.2: A forward algorithm to decide the coverability problem on SMPN.

```
Data: $\mathcal{N}$, a $\preccurlyeq$-strongly monotonic self-modifying Petri system
Data: $G_U$, the set of minimal element of the $\preccurlyeq$-upward-closed set $U$.
begin
    $i \leftarrow 1$;
    while (true) do
        if Reach (Under $(\mathcal{N}, C_i)$) $\cap U \neq \emptyset$ then
            return 'Reachable';
        else if $\gamma$ (Reach (OverWSTS $(\mathcal{N}, C_i, L_i)$)) $\cap U = \emptyset$ then
            return 'Unreachable';
        $i \leftarrow i + 1$ ;
end
```

In this algorithm we have confused the SMPN $\mathcal{N}$ with its corresponding WSTS $\mathcal{S}_{\mathcal{N}}$. The reachability of U during the 'Expand' phase can be efficiently decided by considering the lossy version of $\mathcal{N}$, and using the algorithm of Section 5.3. Since we have shown in Corollary 5.3 that we have perfect pairs in the case of SMPN, we can consider again OverWSTS $(\mathcal{S}, C_i, L_i)$ instead of Over $(\mathcal{S}, C_i, L_i)$ during the 'Enlarge' phase, and use the efficient algorithm of Section 5.3 in this case too.

Remark that this algorithm is *incremental*: Reach (Under $(\mathcal{N}, C_{i+1})$) can be computed by extending Reach (Under $(\mathcal{N}, C_i)$) for all $i \geq 0$. Similarly, one can construct Reach (OverWSTS $(\mathcal{N}, C_i, L_i)$) from Reach (Under $(\mathcal{N}, C_i)$).

As a consequence, we immediately obtain:

Theorem 5.2 *For any strongly monotonic* SMPN $\mathcal{N}$ *and any upward-closed set* U *of markings of* $\mathcal{N}$*, Algorithm 5.2 returns 'Reachable' if* $\mathsf{Reach}(\mathcal{N}) \cap U \neq \emptyset$*, 'Unreachable' otherwise.*

5.4.4 Handling EPN

We have purposely defined our algorithm to solve the coverability problem on the class of strongly monotonic SMPN. Let us show that this choice is not restrictive.

Handling PN We have seen in Section 2.3.3 that the class PN is a syntactic subclass of the class of strongly monotonic SMPN. Thus, we can directly apply our algorithm to PN.

Handling PN+T and PN+R We have seen in Section 2.3.3 that not every PN+T or PN+R is an SMPN. However, for any PN+T $\mathcal{N}$ and any upward-closed set U of markings of $\mathcal{N}$, it is not difficult to build, in polynomial time, an SMPN $\mathcal{N}'$ and an upward-closed set U' of markings of $\mathcal{N}'$ s.t. $\mathsf{Reach}(\mathcal{N}) \cap U = \emptyset$ iff $\mathsf{Reach}(\mathcal{N}') \cap U' = \emptyset$.

The set of places of $\mathcal{N}'$ is $P' = P \uplus \{p^{lock}\} \uplus \{p^t \mid t \in T_e\}$, where T_e is the set of extended transitions of $\mathcal{N}$. Initially, p^{lock} contains one token and all the places of the form p^t are empty. Let f be a function that associates to each place p of $\mathcal{N}'$ its ordinal in P'. That is, if $P' = \{p_1, \ldots p_k\}$, and p is a place of that set, then $p = p_{f(i)}$. Let g be a similar function for the transitions. Then, for any regular transition $t = \langle I, O, \perp, \perp, 0 \rangle \in T_r$ of $\mathcal{N}$, we create a transition t' in $\mathcal{N}'$ that satisfies the following constraints, for any $p \in P'$:

$$D^-_{g(t')f(p)}(\mathbf{m}) = \begin{cases} I(p) & \text{if } p \in P \\ 1 & \text{if } p = p^{lock} \\ 0 & \text{otherwise} \end{cases}$$
$$D^+_{g(t')f(p)}(\mathbf{m}) = \begin{cases} O(p) & \text{if } p \in P \\ 1 & \text{if } p = p^{lock} \\ 0 & \text{otherwise} \end{cases}$$

That is, t' has the same effect as t on the places of P. Moreover, a token has to be present in p^{lock} in order to enable t'. The places in $\{p^t \mid t \in T_e\}$ are not affected by t'.

Symmetrically, for each extended transition $t = \langle I, O, s, d, +\infty \rangle$ of $\mathcal{N}$, we create in $\mathcal{N}'$ two transitions t^1 and t^2 as follows. For any $p \in P'$ we have:

$$
\begin{aligned}
D^-_{g(t^1)f(p)}(\mathbf{m}) &= \begin{cases} I(p) & \text{if } p \in P \\ 1 & \text{if } p = p^{lock} \\ 0 & \text{otherwise} \end{cases} \\
D^+_{g(t^1)f(p)}(\mathbf{m}) &= \begin{cases} O(p) & \text{if } p \in P \setminus \{s, d\} \\ 1 & \text{if } p = p^{t} \\ 0 & \text{otherwise} \end{cases} \\
D^-_{g(t^2)f(p)}(\mathbf{m}) &= \begin{cases} \mathbf{m}(s) & \text{if } p = s \\ 1 & \text{if } p = p^{t} \\ 0 & \text{otherwise} \end{cases} \\
D^+_{g(t^2)f(p)}(\mathbf{m}) &= \begin{cases} O(s) & \text{if } p = s \\ \mathbf{m}(s) + O(d) & \text{if } p = d \\ 1 & \text{if } p = p^{lock} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

The construction is illustrated in Figure 5.3. The intuition of the construction is as follows. We split the effect of the transition $t \in T_e$ in two parts that are realised by the two transitions t^1 and t^2. This allows us to ensure that t^1 and t^2 respect the syntax of SMPN. Roughly speaking t^1 takes care of the constant effect of t on all the places $p \in P \setminus \{s, d\}$. As far as s and d are concerned, t^1 also *consumes* the constant amount of tokens that t consumes, but *does not create* any token. Then, t^2 transfers all the remaining tokens from s to d and creates in s and d the constant amount of tokens that t creates.

Remark that the firing of t^1 inhibits all the other transitions of the net, except t^2 (because t^1 consumes the token in p^{lock}). This allows us to guarantee that these two transitions will always be fired sequentially.

Moreover, it is not difficult to see that $\mathcal{N}'$ is strongly monotonic, according to Lemma 2.13.

Let us finish this sketch of the construction by showing how to adapt U. We let:

$$U' = \{\mathbf{m} \mid \mathbf{m}(p^{lock}) \geq 1 \wedge \exists \mathbf{m}' \in U : \forall p \in P : \mathbf{m}(p) = \mathbf{m}'(p)\}$$

Remark that we impose no restriction on the places outside $P \cup \{p^{lock}\}$. Hence, that set is upward-closed. It is not difficult to see that (proof omitted):

Lemma 5.7 *Let $\mathcal{N}$ be a* PN+T*, let U be an upward-closed set of markings of $\mathcal{N}$, and let $\mathcal{N}'$ and U' be respectively the strongly monotonic* SMPN *and the upward-closed set of markings obtained from $\mathcal{N}$ and U. Then,* $\mathsf{Reach}(\mathcal{N}) \cap U = \emptyset$ *iff* $\mathsf{Reach}(\mathcal{N}') \cap U = \emptyset$.

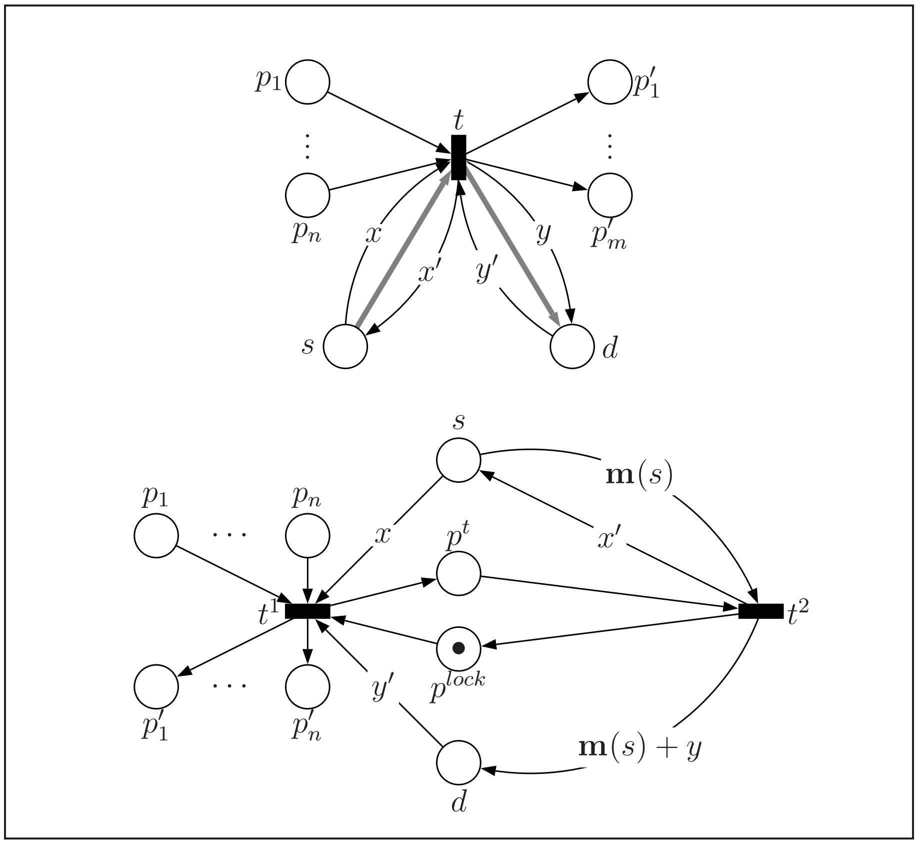


Figure 5.3: An example of translation of a PN+T extended transition t into two SMPN transitions t^1 and t^2.

Since every PN+R is a PN+T, the same holds for PN+R.

Remark that, according to Section 2.3.3, all the PN+T and all the PN+R whose extended transitions $t = \langle I, O, s, d, b\rangle$ respect $O(d) \geq I(s)$ are strongly monotonic SMPN too. This includes the 'Petri nets with transfer arcs' and the 'Petri nets with reset arcs', as defined, for instance, by Ciardo in [Cia94]. Hence, for these PN+T and PN+R, the above construction is not necessary.

Handling PN+NBA As stated in Section 2.3.3, the class PN+NBA is not a syntactic subclass of that of strongly monotonic SMPN. However, for any PN+NBA $\mathcal{N} = \langle P, T, \mathbf{m}_0\rangle$ and any upward–closed set U of markings of $\mathcal{N}$, we can again build, in polynomial time, a PN+R $\mathcal{N} = \langle P', T', \mathbf{m}'_0\rangle$ and an upward–closed set U' s.t. $\mathsf{Reach}\,(\mathcal{N}) \cap U = \emptyset$ iff $\mathsf{Reach}\,(\mathcal{N}') \cap U' = \emptyset$.

Let us consider the partition of T into T_e and T_r as defined in Definition 2.22, and a new place p_{Tr} (the trashcan place). We now show how to build $\mathcal{N}' = \langle P', T', \Sigma, \mathbf{m}'_0\rangle$. First, $P' = P \uplus \{p_{Tr}\}$. For each transition $t = \langle I, O, s, d, 1\rangle$ in T_e, we put in T' two new transitions $t^{=0} = \langle I, O, s, p_{Tr}, +\infty\rangle$ and $t^{\neq 0}\ \langle I_e, O_e, \perp, \perp, 0\rangle$, such that:

$$\forall p \in P : I_e(p) \;=\; \begin{cases} I(p) & \text{if } p \neq s \\ I(p)+1 & \text{otherwise} \end{cases}$$

$$\forall p \in P : O_e(p) \;=\; \begin{cases} O(p) & \text{if } p \neq s \\ O(p)+1 & \text{otherwise} \end{cases}$$

We also add into T' all the transitions $t = \langle I, O, \perp, \perp, 0\rangle$ of T_r (remark that, since $p_{Tr} \notin P$, this implies that these transitions have no effect on p_{Tr}). Finally, $\forall p \in P$: $\mathbf{m}'_0(p) = \mathbf{m}_0(p)$ and $\mathbf{m}'_0(p_{Tr}) = 0$. Figure 5.4 illustrates the construction.

The intuition behind this construction is as follows. Let $t = \langle I, O, s, d, 1\rangle$ be a transition of $\mathcal{N}$ bearing a non-blocking arc. Let $\mathbf{m}$ be a marking of $\mathcal{N}$. In the case where $\mathbf{m}(s) = I(s)$, the non-blocking arc t has no effect, because $I(s)$ tokens are removed from s, which empties s, before the non-blocking arc exerts its effect. Hence, in this case, firing transition $t^{=0}$ has exactly the same effect than firing t. On the other hand, if $\mathbf{m}(s) > I(s)$, the non-blocking arc will move one token from s to d. Thus, the transition actually removes $I(s)+1$ tokens from s and creates $O(d)+1$ tokens in d. This is exactly what $t^{\neq 0}$ does. Thus, in any case, we can always choose the *right transition* of $\mathcal{N}'$ to obtain the same effect as in $\mathcal{N}$.

The difference between t and the pair $t^{=0}$, $t^{\neq 0}$ is that $t^{=0}$ can fire *even when* $\mathbf{m}(s) > I(s)$, whereas $t^{\neq 0}$ should be fired in order to obtain the same effect. However, this is not problematic from the coverability point of view: in the case where $t^{=0}$ is fired with $\mathbf{m}(s) > I(s)$, the marking we obtain is *smaller* (on the places of P) than the marking we would have obtained by firing $t^{\neq 0}$.

Remark that $\mathcal{N}'$ is indeed a PN+R because none of its transitions consume tokens from p_{tR}, and each of its transitions is either a regular transition (from $\mathcal{N}$), or an

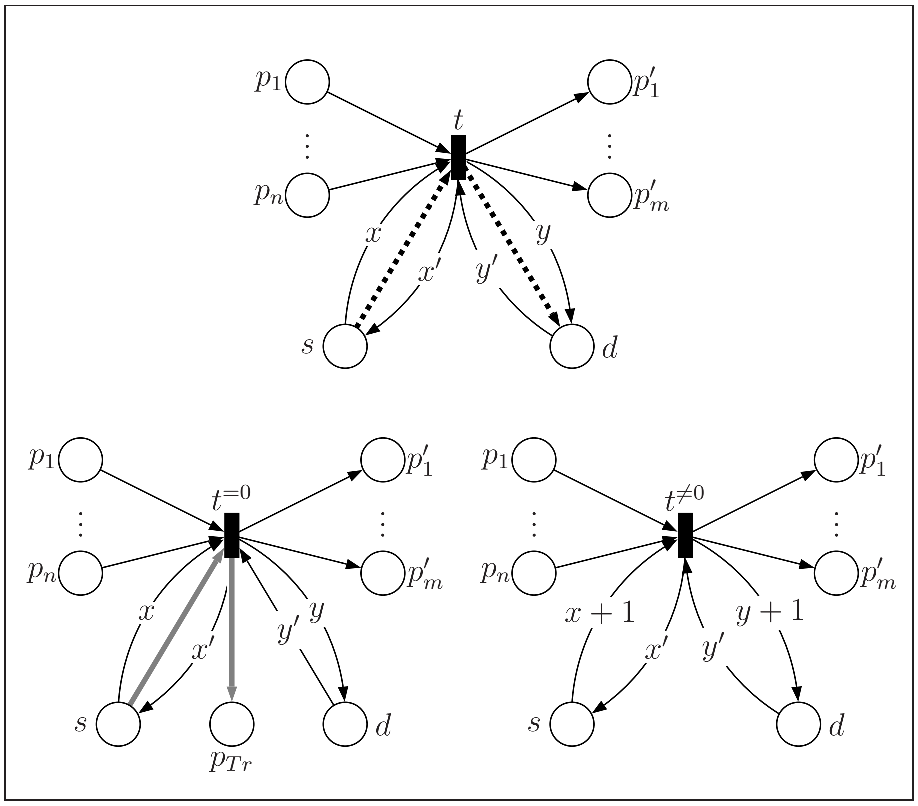


Figure 5.4: An example of translation of a PN+NBA extended transition t into a PN+R extended transition. Remark that the places $p_1, \ldots, p_n$, $p'_1, \ldots, p'_m$, s and d are the same for the two transitions at the bottom of the figure. They have been represented twice for the sake of clarity.

extended transition that transfers all the tokens to p_{Tr} (from which they can never escape).

To prove that this construction is correct, we need the following definitions and notations:

- Let $\mathbf{m}_1$ and $\mathbf{m}_2$ be two markings on some set of places P' and let $P \subseteq P$. Then, $\mathbf{m}_1 \preccurlyeq_P \mathbf{m}_2$ iff for any $p \in P$: $\mathbf{m}_1(p) \leq \mathbf{m}_2(p)$. We define accordingly $=_P$, $\prec_P$, $\succcurlyeq_P$ and $\succ_P$.
- In order to establish a correspondence between the sequences of transitions of $\mathcal{N}$ and those of $\mathcal{N}'$, we define two functions f and g as follows.
  1. $f : T \times \mathbb{N}^{|P|} \rightarrow T'$ is the function such that $\forall t \in T_r : f(t, \mathbf{m}) = t$ and $\forall t = \langle O, I, s, d, 1\rangle \in T_e : f(t, \mathbf{m}) = t^{\neq 0}$, if $\mathbf{m}(s) > I(s)$ and $f(t, \mathbf{m}) = t^{=0}$, otherwise. Remark that in the case where $f(t, \mathbf{m}) = t^{\neq 0}$, we are ensured that the non-blocking arc of t will actually have an effect, because there will be at least one token in s after $I(s)$ tokens have been removed from this place.
  2. $g : T' \rightarrow T$ is the function such that for all $t \in T_r$: $g(t) = t$ and for all $t \in T_e : g(t^{\neq 0}) = g(t^{=0}) = t$.

Then, we can establish the two following lemmata[2]

Lemma 5.8 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0\rangle$ be a* PN+NBA *and let $\mathcal{N}' = \langle P', T', \mathbf{m}'_0\rangle$ be the* PN+R *obtained from $\mathcal{N}$. Let $\sigma = \tau_1\tau_2 \ldots \tau_n$ be a firable sequence of transitions of $\mathcal{N}$. Let $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n$ be the markings s.t.*

$$\mathbf{m}_0 \xrightarrow{\tau_1} \mathbf{m}_1 \xrightarrow{\tau_2} \mathbf{m}_2 \ldots \mathbf{m}_{n-1} \xrightarrow{\tau_n} \mathbf{m}_n$$

Let $\sigma' = \tau'_1\tau'_2 \ldots \tau'_n$ be the sequence of transitions of $\mathcal{N}'$ s.t. for any $1 \leq i \leq n$: $\tau'_i = f(\tau_i, \mathbf{m}_i)$. Then, $\mathbf{m}'_0 \xrightarrow{\sigma'} \mathbf{m}'_n$ implies that $\mathbf{m}'_n =_P \mathbf{m}_n$.

Proof. By definition of σ' and function f, we always choose the *right* transition τ'_i that has exactly the same effect than the transition τ_i on the places of P (and does not modify p_{Tr}). □

Lemma 5.9 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0\rangle$ be a* PN+NBA *and let $\mathcal{N}' = \langle P', T', \mathbf{m}'_0\rangle$ be the* PN+R *obtained from $\mathcal{N}$. Let $\sigma' = \tau'_1\tau'_2 \ldots \tau'_n$ be a sequence of transitions of $\mathcal{N}'$ s.t. $\mathbf{m}'_0 \xrightarrow{\sigma'} \mathbf{m}'$. Let $\sigma = g(\tau'_1)g(\tau'_2) \ldots g(\tau'_n)$ be the corresponding sequence of transitions of $\mathcal{N}$. Then, $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}$ with $\mathbf{m}' \preccurlyeq_P \mathbf{m}$.*

[2]These results may seem too general to prove that the coverability set of $\mathcal{N}$ is equal to that of $\mathcal{N}'$. We have chosen to proceed this way because we will reuse these results later in Chapter 7 and Chapter 8.

Proof. Let $\sigma' = \tau'_1\tau'_2\ldots\tau'_n$ be a firable sequence of transitions of $\mathcal{N}'$. Let $\mathbf{m}'_1, \mathbf{m}'_2, \ldots, \mathbf{m}'_n$ be the markings of $\mathcal{N}'$ s.t. $\mathbf{m}'_0 \xrightarrow{\tau'_1} \mathbf{m}'_1 \xrightarrow{\tau'_2} \ldots \xrightarrow{\tau'_n} \mathbf{m}'_n$. Symmetrically, let $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n$ be the markings of $\mathcal{N}$ such that $\mathbf{m}_0 \xrightarrow{g(\tau'_1)} \mathbf{m}_1 \xrightarrow{g(\tau'_2)} \ldots \xrightarrow{g(\tau'_n)} \mathbf{m}_n$. We prove that σ is indeed a firable sequence of $\mathcal{N}$ and has the right property by showing, by induction on j, that $\mathbf{m}'_j \preccurlyeq_P \mathbf{m}_j$ for all $0 \leq j \leq n$.

- **Base case:** $j = 0$. Trivial
- **Inductive case:** $j = k + 1$. By induction hypothesis, $\mathbf{m}'_k \preccurlyeq_P \mathbf{m}_k$. Let us consider $g(\tau'_{k+1})$. If $g(\tau'_{k+1}) \in T_r$, then, $g(\tau'_{k+1})$ and τ'_{k+1} both have the same effect on the places of P. Hence, by monotonicity, $g(\tau'_{k+1})$ is firable from $\mathbf{m}_k$ and $\mathbf{m}'_{k+1} \preccurlyeq_P \mathbf{m}_{k+1}$. Otherwise, let us assume that $g(\tau'_{k+1}) = t = \langle I, O, s, d, 1\rangle \in T_e$. Then, τ'_{k+1} is either $t^{=0}$ or $t^{\neq 0}$. In both cases, the definition of $t^{=0}$ and $t^{\neq 0}$ ensures that $\mathbf{m}'_k(s) \geq I(s)$. Hence, $\mathbf{m}_k(s) \geq I(s)$. Since the number of tokens that are consumed by t, $t^{=0}$ and $t^{\neq 0}$ is the same on the other places, we are ensured that t is firable from $\mathbf{m}_k$.
  - In the case where $\tau'_{k+1} = t^{=0}$, we have $\mathbf{m}'_{k+1}(s) = O(s)$, and, for any place $p \in P$: $\mathbf{m}'_{k+1}(p) = \mathbf{m}'_k(p) - O(p) + I(p)$. By definition of t, we have $\mathbf{m}_{k+1}(s) \geq O(s)$ (whether the non-blocking arc has an effect or not). As far as $\mathbf{m}_{k+1}(d)$ is concerned, we have either $\mathbf{m}_{k+1}(d) = \mathbf{m}_k(d) + O(d) - I(d)$ or $\mathbf{m}_{k+1}(d) = \mathbf{m}_k(d) + O(d) - I(d) + 1$, depending on the effect of the non-blocking arc. Hence, $\mathbf{m}_{k+1}(d) \geq \mathbf{m}_k(d) + O(d) - I(d)$. Finally, for any place $p \in P \setminus \{s, d\}$: $\mathbf{m}_{k+1}(p) = \mathbf{m}_k(p) - O(p) + I(p)$. Hence, $\mathbf{m}'_{k+1} \preccurlyeq_P \mathbf{m}_{k+1}$.
  - In the case where $\tau'_{k+1} = t^{\neq 0}$, we have $\mathbf{m}'_k(s) \geq I(s) + 1$, by definition of τ'_{k+1}. Hence, $\mathbf{m}_k(s) \geq I(s) + 1$. Thus, the non-blocking arc of $g(\tau'_{k+1})$ has an effect this transition consumes $I(s) + 1$ tokens in s, and produces $O(s) + 1$ tokens in d. Hence, $g(\tau'_{k+1})$ and τ'_{k+1} both have the same effect on the places of P, and we have $\mathbf{m}'_{k+1} \preccurlyeq_P \mathbf{m}_{k+1}$ by monotonicity.

□

Thus, given an upward–closed set U of markings of $\mathcal{N}$, we can build the upward-closed set U' of markings of $\mathcal{N}'$ as follows:

$$U' = \{\mathbf{m}' \mid \exists \mathbf{m} \in U : \mathbf{m} =_P \mathbf{m}'\}$$

In this case, and thanks to Lemma 5.8 and Lemma 5.9, we obtain:

Corollary 5.4 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0\rangle$ and U be an* PN+NBA *and an upward–closed set of markings of $\mathcal{N}$. Let $\mathcal{N}' = \langle P \uplus \{p_{Tr}\}, T', \mathbf{m}'_0\rangle$ and U' be respectively the* PN+R *obtained from $\mathcal{N}$ and the upward–closed set of markings of $\mathcal{N}'$ obtained from U. Then,* $\mathsf{Cover}(\mathcal{N}) \cap U = \emptyset$ *iff* $\mathsf{Cover}(\mathcal{N}') \cap U' = \emptyset$.

Since any PN+R is also a PN+T, it can be associated to a strongly monotonic SMPN that has corresponding coverability properties. Hence, we obtain a procedure to turn any PN+NBA into a corresponding strongly monotonic SMPN on which EEC can be applied.

Finally, let us mention that for Lossy Petri nets, another monotonic extension of Petri nets [BM99], one can easily again build, in polynomial time, a corresponding SMPN that has the same coverability properties (by adding transitions that non-deterministically remove tokens from any place).

Thus, to the best of our knowledge, Algorithm 5.2 can handle *any monotonic extension of Petri nets in the literature.*

5.4.5 Experimental evaluation

We have implemented the techniques described so far in a prototype for the analysis of PN. Remark however, that theses techniques are not restricted to PN. They can also be applied to PN+T and PN+NBA. Our simple prototype, however, does not handle them. A more advanced prototype, that handles PN+T and PN+NBA too, has been described in [GRVB05] and is briefly discussed hereunder. We have run the prototype on about 12 examples from the literature. Table 5.1 reports on these results. The case studies retained here are mainly abstractions of mutual exclusion protocols (most of them taken from [Van03]).

When applied to these examples, the symbolic backward algorithm of [ACJT96] does not always produce a result within the time limit of 20 minutes we have fixed (column **Pre**). A heuristic presented in [DRVB01] uses place-invariants to guide the search and improves the performance of the prototype (which has been fined tuned during several years of research). Remark that the performances of the EEC prototype are close to those of the backward algorithm, although the EEC prototype is still in its infancy.

More experimental results can be found in [GRVB05]. In this paper, we describe an efficient algorithm to decide coverability of finite WSTS that is different from that of Section 5.3. Then, we present a prototype that implements the (accordingly improved) EEC procedure, and that is capable of analysing EPN. As in Table 5.1, we compare this prototype to a plain and an improved implementation of the backward algorithm. The efficiency of this prototype on PN is similar to the one we have presented here. On several examples of PN+T, it outperforms the backward approach: out of the 14 examples of PN+T that we consider, the plain backward approach can analyse only 9 example and the improved backward, 12 examples within the time limit. On the other hand, EEC successfully analyses all the 14 examples.

Other tools such as Fast [BFLP03, Fas] can analyse the same examples by using a forward procedure. Remark that these tools can handle a broader class of systems than EEC (like systems that are not WSTS, for instance). In practice, FAST does not always terminate on our set of examples [Fas].

5.5 Application to LCS

In this section, we show that our general algorithmic schema can be instantiated to decide the coverability problem on LCS too. For that purpose, we need an adequate domain of limits. Such a domain has already been introduced in Section 3.1.2. We prove in Section 5.5.1 that this domain is effective. Then we define in Section 5.5.2 the sequences of sets C_i and L_i that we shall use in our instantiation EEC to the LCS case. In Section 5.5.3, we explain how the sets of all the most precise subsets of $L_i \cup C_i$ that over-approximate the successors of a set of configurations can be computed in an efficient fashion. This result makes the task of constructing the And-Or graph more efficient in practice. Then, we present in Section 5.5.4 the instantiation of EEC to the case of LCS, and show in Section 5.5.5 that this algorithm performs well in practice. We close the chapter on a short discussion that shows, by means of a practical example, why And-Or graphs are necessary in the case of LCS.

5.5.1 Domain of limits

In order to apply the EEC algorithm to the case of LCS, we have to show that any LCS $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ with the adequate domain of limits $(\mathcal{L}(\Sigma, Q), \overline{\sqsubseteq}, \gamma)$ enforce the effectiveness criteria of Definition 4.1.

Theorem 5.3 *Any* LCS $\mathcal{C}$ *with the adequate domain of limits* $(\mathcal{L}(\Sigma, Q), \overline{\sqsubseteq}, \gamma)$ *are effective.*

Proof. We establish the theorem by proving that the four properties of Definition 4.1 hold:

(E_1) the sets $\mathsf{States}\,(\mathcal{C})$ and $\mathcal{L}(\Sigma, Q)$ are recursively enumerable. Indeed, as far as $\mathsf{States}\,(\mathcal{C})$ is concerned, each configuration $\langle q, W\rangle \in \mathsf{States}\,(\mathcal{C})$ can be encoded as word. Let us associate to every state $q \in Q$ a unique character $\mathsf{c}(q) \notin \Sigma$. Moreover, let $\#$ be a new character, which is not in Σ nor in $\{\mathsf{c}(q) \mid q \in Q\}$. Then, the encoding of $c = \langle q, W\rangle$ is $w_c = \mathsf{c}(q) \cdot W(f_1)\#W(f_2) \cdot \# \cdot W(f_2)\# \cdots \#W(f_n)$, where $F = \{f_1, f_2, \dots f_n\}$. Remark that $\{w_c \mid c \in \mathsf{States}\,(\mathcal{C})\}$ is a (possibly infinite) language on the finite alphabet $\Sigma \cup \{\mathsf{c}(q) \mid q \in Q\}$. Hence, it is recursively enumerable. A similar technique can be applied to encode each element of $\mathcal{L}(\Sigma, Q)$ into a word on a finite alphabet.

(E_2) it is shown in [ABJ98] that the transition relation of LCS is decidable;

Example					**EEC**		**Pre+Inv**		**Pre**	
cat.	**name**	**P**	**T**	**Safe**	**Time**	**Mem**	**Time**	**Mem**	**Time**	**Mem**
PN	basicME	5	4	$\surd$	0	2,040	0	1,308	0	2,076
PN	csm	14	13	$\surd$	0.03	2,856	0.09	2,516	0.11	2,512
PN	fms	22	20	$\surd$	13.61	20,456	0	1,356	28.49	8,048
PN	mesh2x2	32	32	$\surd$	0.96	5,152	0.89	2,740	0.81	2,740
PN	mesh3x2	52	54	$\surd$	3.25	11,396	10.92	4,568	8.57	4,604
PN	multipool	18	21	$\surd$	1.05	5,952	0	1,124	1.46	2,980
PN	pncsacover	31	36	$\times$	101.52	292,268	40.83	5,012	time out	
BPN	lamport	11	9	$\surd$	0.01	2,580	0.02	2,132	0.06	2,468
BPN	newdekker	16	14	$\surd$	0.06	2,880	0.05	2,260	0.54	2,536
BPN	newrtp	9	12	$\surd$	0	2,552	0	1,096	0.05	2,340
BPN	peterson	14	12	$\surd$	0.01	2,708	0.04	2,244	0.12	2,504
BPN	read-write	13	9	$\surd$	0.27	3,172	0.16	2,480	0.42	2,496

Table 5.1: Empirical evaluation of the EEC method on PN. Results obtained on Intel Xeon 3Ghz with 4Gb of memory. **cat.** : category of example (PN = (unbounded) Petri net, BPN = bounded Petri net), **P** : number of places, **T**: number of transitions. **EEC** : results obtained with the "Expand, Enlarge and Check" algorithm. **Pre + Inv**: results obtained with a backward approach, using invariant heuristics, **Pre**: same without invariants. All the memory consumptions are in KB and times in second.

(E_3) it is shown in [ABJ98] how to compute an operator that returns, given $c \in \mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$, $c' \in \mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$ such that $\gamma(c') = \mathsf{Post}(\gamma(c))$ (that is, how to extend the Post operator to sre, see Lemma 3.11). By using that operator and since $\overline{\sqsubseteq}$ is decidable following [ABJ98], we conclude that we can decide whether $\mathsf{Post}(\gamma(c)) \subseteq \gamma(c')$ for any $c, c' \in \mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$;

(E_4) We have discussed at the end of Section 3.1.2 how to decide whether $\gamma(C_1) \subseteq \gamma(C_2)$ for any finite sets $C_1, C_2 \subseteq \mathsf{States}(\mathcal{C}) \cup \mathcal{L}(\Sigma, Q)$.

□

5.5.2 Construction of the C_i's and the L_i's

The following definition explains how we construct the C_i's and L_i's in the case of LCS:

Definition 5.7 (Sequences of C_i's and L_i's for LCS) *The sequences of C_i's and L_i's are defined as follows:*

- $C_i = \{\langle q, W\rangle \in \mathcal{S}_{\mathcal{C}} \mid q \in Q, \forall f \in F : W(f) \in \Sigma^* \text{ and } |W(f)| \leq i\}$;
- $L_i = \{\langle q, E\rangle \in \mathcal{L}(\Sigma, Q) \mid q \in Q, \forall f \in F : E(f) \in L(\Sigma)^* \text{ and } |E(f)| \leq i\}$. ■

That is, C_i is the set of states where the contents of the channels are words of size at most i. Similarly, L_i is the set of limits that assign sre of size of most i to channels (plus the $\top$ element).

It is easy to see that

1. $C_i \subseteq C_{i+1}$ and $L_i \subseteq L_{i+1}$ for all $i \geq 0$;
2. for all $c \in \mathsf{States}(\mathcal{C})$ there exists $i \geq 0$ such that $c \in C_i$ and for all $\ell \in \mathcal{L}(\Sigma, Q)$ there exists $i \geq 0$ such that $\ell \in L_i$;
3. $\langle q_0, W_0\rangle \in C_0$ where $\forall c \in C : W_0(c) = \varepsilon$;
4. $\top \in L_0$.

5.5.3 Approximation of the successors

Unlike the SMPN case, And-Or graphs built during the 'Enlarge' phase are in general not degenerated when we consider LCS. Hence, the efficient algorithm of Section 5.3 cannot be applied during the 'Enlarge' phase in the present case. However, that phase can be made more efficient in practice by improving the construction of the And-Or graph.

According to Definition 4.5, the cornerstone in the construction of the And-Or graph is the computation, for any Or-node v, of the set of most precise subsets of $L_i \cup C_i$ that over-approximate $\mathsf{Post}\,(v)$. Since the sets L_i and C_i are always finite, this computation is clearly feasible by enumerating all their possible subsets and keeping the most precise ones. However, it is possible to compute these most precise approximations in a much more direct fashion, as we show now. Notice that, following the semantics of LCS, the Post operation can add at most one character to each channel. Hence, we only need to be able to approximate precisely any $c \in L_{i+1} \cup C_{i+1}$ by elements in $L_i \cup C_i$.

Over-approximation of an sre For that purpose, we first show, given an sre $s \neq \varepsilon$ and a natural number $i \geq 1$ such that $|s| \leq i+1$, how to compute the most complete and most precise set of sre (of size at most i) that over-approximate s. This is the purpose of the Approx function:

Definition 5.8 (The Approx function) *Let $s = d_1 \cdot d_2 \cdots d_n \neq \varepsilon$ be an sre and let $i \geq 1$ be a natural number s.t. $|s| \leq i+1$. Let $L(s)$ be the set defined as:*

$$L(s) = \bigcup_{j=1}^{n-1} \left\{ \begin{array}{c} d_1 \cdots d_{j-1} \cdot (c_1 + \ldots + c_m)^* \cdot d_{j+2} \cdots d_n \\ s.t. \\ \{c_1, \ldots, c_m\} = \alpha(d_j) \cup \alpha(d_{j+1}) \end{array} \right\}$$

Let $M \subseteq L(s)$ be a set s.t. for any $s' \in M$: there is no $s'' \in M$ with $s'' \sqsubseteq s'$. Then:

$$\mathsf{Approx}(s, i) = \begin{cases} \{s\} & \text{if } |s| \leq i \\ M & \text{otherwise} \end{cases}$$

■

Remark in particular that $\mathsf{Approx}(\varepsilon, i) = \{\varepsilon\}$ for any $i \geq 0$, since $|\varepsilon| = 0$. Remark that, since we have not assumed that we manipulate normal form sre only, there can be several sets that satisfy the definition of M. We simply assume that Approx returns one of these sets (when $|s| = i+1$). This is not problematic however, since all these sets represent the same set of words. Moreover, this can be avoided by always using normal form sre[3].

Example 5.3 *Let us consider the sre $s = (\mathtt{a} + \varepsilon) \cdot (\mathtt{b} + \mathtt{c})^* \cdot (\mathtt{a} + \varepsilon) \cdot (\mathtt{d} + \varepsilon)$. In that case, $L(s)$ is the set:*

$$\left\{ \begin{array}{c} (\mathtt{a} + \mathtt{b} + \mathtt{c})^* \cdot (\mathtt{a} + \varepsilon) \cdot (\mathtt{d} + \varepsilon) \\ (\mathtt{a} + \varepsilon) \cdot (\mathtt{a} + \mathtt{b} + \mathtt{c})^* \cdot (\mathtt{d} + \varepsilon) \\ (\mathtt{a} + \varepsilon) \cdot (\mathtt{b} + \mathtt{c})^* \cdot (\mathtt{a} + \mathtt{d})^* \end{array} \right\}$$

[3]As all the possible sets that satisfy the definition of M represent the same set of words, there is one and only one set of sre in normal form that represents it, by Lemma 3.4.

Remark that $(\mathtt{a}+\mathtt{b}+\mathtt{c})^* \cdot (\mathtt{a}+\varepsilon) \cdot (\mathtt{d}+\varepsilon)$ *and* $(\mathtt{a}+\varepsilon) \cdot (\mathtt{a}+\mathtt{b}+\mathtt{c})^* \cdot (\mathtt{d}+\varepsilon)$ *both represent the same set: they are both equivalent to the normal-form* sre $(\mathtt{a}+\mathtt{b}+\mathtt{c})^* \cdot (\mathtt{d}+\varepsilon)$*. On the other hand,* $(\mathtt{a}+\varepsilon) \cdot (\mathtt{b}+\mathtt{c})^* \cdot (\mathtt{a}+\mathtt{d})^*$ *represents a set that is incomparable to the set represented by the two other* sre*. Hence, the set* $\mathsf{Approx}(s, 3)$ *is either:*

$$\{(\mathtt{a}+\varepsilon) \cdot (\mathtt{a}+\mathtt{b}+\mathtt{c})^* \cdot (\mathtt{d}+\varepsilon), \quad (\mathtt{a}+\varepsilon) \cdot (\mathtt{b}+\mathtt{c})^* \cdot (\mathtt{a}+\mathtt{d})^*\}$$

or

$$\{(\mathtt{a}+\mathtt{b}+\mathtt{c})^* \cdot (\mathtt{a}+\varepsilon) \cdot (\mathtt{d}+\varepsilon), \quad (\mathtt{a}+\varepsilon) \cdot (\mathtt{b}+\mathtt{c})^* \cdot (\mathtt{a}+\mathtt{d})^*\}$$

$\Diamond$

The following proposition shows that $\mathsf{Approx}(s, i)$ returns a set containing the most precise sre of length at most i that over-approximate s:

Proposition 5.5 *For any* sre s*, for any* $i \in \mathbb{N}$ *s.t.* $|s| \leq i+1$*, for any* sre s'*:*

$$\begin{gathered} [\![s]\!] \subseteq [\![s']\!] \text{ and } |s'| \leq i \\ \text{implies} \\ \exists s'' \in \mathsf{Approx}(s, i) : [\![s'']\!] \subseteq [\![s']\!] \end{gathered}$$

Proof. First, remark that the proposition holds trivially when $|s| \leq i$, because in that case $\mathsf{Approx}(s, i) = \{s\}$.

Then remark that, by construction, any sre in $\mathsf{Approx}(s, i)$ is of size i. It is not difficult to see that any sre $s'' \in L(s)$ is s.t. $[\![s]\!] \subseteq [\![s'']\!]$. Hence, it is also the case for any sre $s'' \in \mathsf{Approx}(s, i)$. Let us show that $\mathsf{Approx}(s, i)$ contains the most precise sre.

Let $s = d_1 \cdots d_{i+1}$ and $s' = d'_1 \cdots d'_k$ be two sre s.t. $k \leq i$ and $[\![s]\!] \subseteq [\![s']\!]$. By Lemma 3.7, this implies that there exists a function $\rho : \{1 \ldots i+1\} \mapsto \{1 \ldots k\}$ s.t.

1. for any $1 \leq j \leq i+1$: $[\![d_j]\!] \subseteq [\![d'_{\rho(j)}]\!]$ and
2. for any $1 \leq j \leq i$: $\rho(j) = \rho(j+1)$ implies that $d'_{\rho(j)}$ is a $*$–dc–re.

Since $k < i+1$, there exists $1 \leq j \leq i$ s.t. $\rho(j) = \rho(j+1)$. Thus, $d'_{\rho(j)}$ is a $*$–dc–re with $[\![d_j \cdot d_{j+1}]\!] \subseteq [\![d'_{\rho(j)}]\!]$, by Lemma 3.7.

Let us show that there exists in $\mathsf{Approx}(s, i)$ an sre $\overline{s}$ s.t. $[\![\overline{s}]\!] \subseteq [\![s']\!]$. For that purpose, we first consider the sre $s'' = d''_1 \cdot d''_2 \cdots d''_i$ defined as follows:

1. for any $1 \leq \ell \leq j-1$: $d''_\ell = d_\ell$;
2. $d_j = (c_1 + \ldots + c_m)^*$ with $\alpha(d_j) \cup \alpha(d_{j+1}) = \{c_1, \ldots, c_m\}$;
3. for any $j+1 \leq \ell \leq i$: $d''_\ell = d_{\ell+1}$;

Remark that $s'' \in L(s)$ (where $L(s)$ corresponds to Definition 5.8). Let us show that $[\![s'']\!] \subseteq [\![s']\!]$. Let ρ' be the function $\{1 \dots i\} \mapsto \{1 \dots k\}$ s.t.:

$$\forall 1 \leq \ell \leq i : \rho'(\ell) = \begin{cases} \rho(\ell) & \text{if } 1 \leq \ell \leq j \\ \rho(\ell+1) & \text{if } j+1 \leq \ell \leq i \end{cases}$$

Since the dc–re $d''_1 \dots d''_{j-1}$ and d''_{j+1}, d''_i correspond to dc–re of s, we have: for any $1 \leq \ell \leq i$: $\ell \neq j$ implies that $[\![d''_\ell]\!] \subseteq [\![d'_{\rho'(\ell)}]\!]$. Moreover, $\alpha(d'_{\rho'(j)}) = \alpha(d'_{\rho(j)}) \supseteq \alpha(d_j) \cup \alpha(d_{j+1})$, and $\alpha(d''_j) = \alpha(d_j) \cup \alpha(d_{j+1})$, by construction. Hence, since d''_j and $d'_{\rho'(j)}$ are both $*$–dc–re, we have $[\![d''_j]\!] \subseteq [\![d'_{\rho'(j)}]\!]$. We conclude that for any $1 \leq \ell \leq i$: $[\![d''_\ell]\!] \subseteq [\![d'_{\rho'(\ell)}]\!]$.

Let us show that for any $1 \leq \ell < i$: $\rho'(\ell) = \rho'(\ell+1)$ implies that $d'_{\rho'(\ell)}$ is a $*$–dc–re. We consider three cases:

1. If $1 \leq \ell < j$, then

   $$\rho'(\ell) = \rho'(\ell+1) \text{ implies } d'_{\rho'(\ell)} \text{ is a } *\text{–dc–re.}$$

   is, by definition of ρ', equivalent to:

   $$\rho(\ell) = \rho(\ell+1) \text{ implies } d'_{\rho(\ell)} \text{ is a } *\text{–dc–re.}$$

   which is true by definition of ρ.

2. If $\ell = j$, then the implication is trivially true because $d'_{\rho(j)}$ has been identified as a $*$–dc–re.

3. If $j+1 \leq \ell < i$, then

   $$\rho'(\ell) = \rho'(\ell+1) \text{ implies } d'_{\rho'(\ell)} \text{ is a } *\text{–dc–re.}$$

   is, by definition of ρ', equivalent to:

   $$\rho(\ell+1) = \rho(\ell+2) \text{ implies } d'_{\rho(\ell+1)} \text{ is a } *\text{–dc–re.}$$

   which is true by definition of ρ.

Thus, for any $1 \leq \ell \leq i$: $[\![d''_\ell]\!] \subseteq [\![d'_{\rho'(\ell)}]\!]$ and for any $1 \leq \ell < i$: $\rho'(\ell) = \rho'(\ell+1)$ implies that $d'_{\rho'(\ell)}$ is a $*$–dc–re. By Lemma 3.7, we conclude that $[\![s'']\!] \subseteq [\![s']\!]$. Since $s'' \in L(s)$, and by Definition 5.8, we conclude that Approx(s, i) contains an sre $\overline{s}$ s.t. $|\overline{s}| \leq i$ and $[\![\overline{s}]\!] \subseteq [\![s'']\!]$. Hence the proposition. □

Approximation of the successors in $L_i \cup C_i$ Thanks to Proposition 5.5, it is easy to show how to compute, for any element[4] $d \in L_{i+1} \cup C_{i+1}$ the most precise subsets of $L_i \cup C_i$ that over-approximate d.

First, we extend the definition of Approx in order to handle limit elements, instead of sre. Let d be an element of $\mathcal{L}(\Sigma, Q)$. In the case where $d = \top$, we let $\mathsf{Approx}(d, i) = \{\top\}$. Otherwise, we assume that $d = \langle q, E\rangle$, and we define $\mathsf{Approx}(d, i)$ as:

$$\mathsf{Approx}(d, i) = \{\langle q, E'\rangle \mid \forall f \in F : E'(f) \in \mathsf{Approx}(E(f), i)\}$$

Let us show that, for any element $d \in L_{i+1}$, $\mathsf{Approx}(d, i)$ is the set of the most precise elements of L_i that over-approximate d:

Lemma 5.10 *Let d be an element of L_{i+1}, and let $D' \subseteq L_i$ be s.t. $\gamma(d) \subseteq \gamma(D')$. Then, there exists $\overline{d} \in \mathsf{Approx}(d, i)$ s.t. $\gamma(d) \subseteq \gamma\left(\overline{d}\right) \subseteq \gamma(D')$.*

Proof. The proof is trivial when $d = \top$. Otherwise, we assume that $d = \langle q, E\rangle$. Since $\gamma(d) \subseteq \gamma(D')$, there is $d' \in D'$ s.t. $\gamma(d) \subseteq \gamma(d')$, by Lemma 3.9. In the case where $d' = \top$, we have $\gamma(D') = \mathsf{States}(\mathcal{C})$, and we conclude immediately that there is $\overline{d} \in \mathsf{Approx}(d, i)$ s.t. $\gamma(d) \subseteq \gamma\left(\overline{d}\right) \subseteq \gamma(D') = \mathsf{States}(\mathcal{C})$. Otherwise, let us assume that $d' = \langle q, E'\rangle$ (remark that the state in d' has to be the same than the state in d in order to ensure that $\gamma(d) \subseteq \gamma(d')$). Since $\gamma(d) \subseteq \gamma(d')$, we have $[\![E(f)]\!] \subseteq [\![E'(f)]\!]$ for any channel f. Hence, by Proposition 5.5, there is, for any channel f, an sre $s_f \in \mathsf{Approx}(E(f), i)$ s.t. $[\![E(f)]\!] \subseteq [\![s_f]\!] \subseteq [\![E'(f)]\!]$. Let $\overline{d} = \left\langle q, \overline{E}\right\rangle$, where for any channel f: $\overline{E}(f) = s_f$. Remark that $\overline{d} \in L_i$, since for any f: $s_f \in \mathsf{Approx}(E(f), i)$. Moreover, $\overline{d} \in \mathsf{Approx}(d, i)$, by definition, and $\gamma(d) \subseteq \gamma\left(\overline{d}\right) \subseteq \gamma(d') \subseteq \gamma(D')$, by construction. □

Remark that all the elements of this set are pairwise $\overline{\sqsubseteq}$-incomparable, by definition of Approx (on the sre).

Thanks to that definition of Approx, one can define a PostApprox function, that, given a limit element $d \in L_i$ returns the most precise downward-closed sets representable in L_i and that over-approximate $\mathsf{Post}(d)$.

Definition 5.9 (The PostApprox function) *Given an LCS $\mathcal{C}$, its associated sequences $C_0, C_1, \ldots, C_i, \ldots$ and $L_0, L_1, \ldots, L_i, \ldots$, a natural number $i \geq 1$, and an element $d \in L_j$ for $1 \leq j \leq i$:*

$$\mathsf{PostApprox}(d, i) = \left\{ \{d'_1, d'_2, \ldots d'_k\} \;\middle|\; \begin{array}{c} \mathsf{Post}(d) = \{d_1, d_2, \ldots d_k\} \\ \textit{and} \\ \forall 1 \leq j \leq k : d'_j \in \mathsf{Approx}(d_j, i) \end{array} \right\}$$

■

[4]Remind that, by semantics of LCS, at most one character can be added by a transition to the channel at any time. Hence, we can restrict ourselves to $L_{i+1} \cup C_{i+1}$.

Let us show, thanks to Lemma 5.10, that $\mathsf{PostApprox}\,(d,i)$ is indeed the set we are looking for:

Lemma 5.11 *Let $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ be an* LCS, *and let $\left\langle \mathcal{L}(\Sigma,Q), \overline{\sqsubseteq}, \gamma\right\rangle$ be an adequate domain of limits for $\mathsf{States}\,(\mathcal{C})$. Let us consider the sequence of sets L_i built according to Definition 5.7. Let $i \geq 1$ be a natural number, and let let d be a limit element from L_j for some $1 \leq j \leq i$. Then, for every $D \subseteq L_i$ s.t. $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(D)$, there exists $D' \in \mathsf{PostApprox}\,(d,i)$ s.t. $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(D') \subseteq \gamma\,(D)$.*

Proof. $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(D)$ implies that, for every $d_j \in \mathsf{Post}\,(\gamma\,(d))$, there is $\overline{d} \in D$ s.t. $\gamma\,(d_j) \subseteq \gamma\left(\overline{d}\right)$, by Lemma 3.9. Thus, for any $d_j \in \mathsf{Post}\,(\gamma\,(d))$ there is $d'_j \in \mathsf{Approx}(d_j, i)$ s.t. $\gamma\,(d_j) \subseteq \gamma\left(d'_j\right) \subseteq \gamma\left(\overline{d}\right)$, by Lemma 5.10. By definition of $\mathsf{PostApprox}\,(d,i)$, the set $D' = \left\{d'_1, \ldots d'_k\right\}$ is in $\mathsf{PostApprox}\,(d,i)$. Moreover, $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(D') \subseteq \gamma\,(D)$, by construction. □

It remains to show how we can over-approximate an element of $c \in C_i$. For that purpose, we exploit the limit function (see Definition 3.5) that transforms such a configuration into a limit element $\mathsf{limit}\,(c)$ that has the same denotation. We can re-use the Approx function. For any $c \in C_{i+1}$, $\mathsf{Approx}(\mathsf{limit}\,(c)\,, i)$ is the set of the most precise over-approximations of c in $L_i \cup C_i$.

Lemma 3.10 has two important consequences. First of all, it implies that for any $i \geq 1$, for any $c \in C_j$ $(1 \leq j \leq i)$, the set $\mathsf{PostApprox}\,(\mathsf{limit}\,(c)\,, i)$ contains the most precise over-approximations of $\mathsf{Post}\,(c)$ representable in L_i (Lemma 5.11). Second, it also implies that one cannot find more precise representations of $\mathsf{Post}\,(d)$ by considering elements of C_i. That is, $\mathsf{PostApprox}\,(\mathsf{limit}\,(c)\,, i)$ contains the most precise over-approximations of $\mathsf{Post}\,(c)$ representable in $L_i \cup C_i$.

Proposition 5.6 *Let $\mathcal{C} = \langle Q, q_0, F, \Sigma, T\rangle$ be an* LCS, *with adequate domain of limits $\left\langle \mathcal{L}(\Sigma,Q), \overline{\sqsubseteq}, \gamma\right\rangle$. Let us consider the sequence of sets L_i and C_i built according to Definition 5.7. Let $i \geq 1$ be a natural number, and let let d be an element from $L_j \cup C_j$ for some $1 \leq j \leq i$. Then, for every $D \subseteq L_i \cup C_i$ s.t. $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(D)$, there exists $D' \in \mathsf{PostApprox}\,(\mathsf{limit}\,(d)\,, i)$ s.t. $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(D') \subseteq \gamma\,(D)$.*

Proof. Since $\gamma\,(\mathsf{limit}\,(d)) = \gamma\,(d)$ for every $d \in L_i \cup C_i$, by Lemma 3.10 and Definition 3.5 (extended to limit elements), we are ensured that for every $D \subseteq L_i$ s.t. $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(D)$, there exists $D' \in \mathsf{PostApprox}\,(\mathsf{limit}\,(d)\,, i)$ s.t. $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(D') \subseteq \gamma\,(D)$, by Lemma 5.11. Let us show, *per absurdum* that this holds for any $D \subseteq L_i \cup C_i$. We assume that there exists $E \subseteq L_i \cup C_i$ s.t. $\mathsf{Post}\,(\gamma\,(d)) \subseteq \gamma\,(E)$ and there is no $E' \in \mathsf{PostApprox}\,(\mathsf{limit}\,(d)\,, i)$ with $\gamma\,(E') \subseteq \gamma\,(E)$. Let $\overline{E} = (E \cap L_i) \cup \{\mathsf{limit}\,(c) \mid c \in E \cap C_i\}$, i.e., $\overline{E}$ is obtained from E by replacing in this set all the configurations by their corresponding limit. By Lemma 3.10, we have $\gamma\left(\overline{E}\right) = \gamma\,(E)$. Hence, by Lemma 5.11, there is $\overline{E}' \in \mathsf{Approx}(d,i)$ s.t. $\gamma\left(\overline{E}'\right) \subseteq \gamma\left(\overline{E}\right) = \gamma\,(E)$. Contradiction. □

Example 5.4 *Let us consider an* LCS $\mathcal{C}$ *with two channels* f_1 *and* f_2*. For the sake of clarity, we will represent configurations* $\langle q, W\rangle$ *of* $\mathcal{C}$ *as* $\langle q, W(f_1), W(f_2)\rangle$*, and limit elements* $\langle q, E\rangle$ *associated to* $\mathcal{C}$ *as* $\langle q, E(f_1), E(f_2)\rangle$*.*

Let us fix $i = 2$ *and let us consider*[5] $\ell = \langle q_1, (\mathtt{a}+\varepsilon)\cdot(\mathtt{b}+\varepsilon), (\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\varepsilon)\rangle \in L_i$*. Let us assume that two transitions are firable from* q_1*. The former moves the state to* q_2 *and adds a* $\mathtt{c}$ *in both channels. The latter moves the state to* q_3*, consumes a* $\mathtt{b}$ *in channel* f_1 *and adds an* $\mathtt{f}$ *to channel* f_2*. Then:*

$$\mathsf{Post}\,(\ell) = \{\ell_1, \ell_2\}$$

where:

$$\begin{aligned}
\ell_1 &= \langle q_2, (\mathtt{a}+\varepsilon)\cdot(\mathtt{b}+\varepsilon)\cdot(\mathtt{c}+\varepsilon), (\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\varepsilon)\cdot(\mathtt{c}+\varepsilon)\rangle\\
\ell_2 &= \langle q_3, (\mathtt{a}+\varepsilon), (\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\varepsilon)\cdot(\mathtt{f}+\varepsilon)\rangle
\end{aligned}$$

Let us now apply the Approx *function on the* sre *that make up* ℓ_1 *and* ℓ_2*. We obtain:*

$$\begin{aligned}
\mathsf{Approx}((\mathtt{a}+\varepsilon)\cdot(\mathtt{b}+\varepsilon)\cdot(\mathtt{c}+\varepsilon), i) &= \left\{(\mathtt{a}+\mathtt{b})^*\cdot(\mathtt{c}+\varepsilon), (\mathtt{a}+\varepsilon)\cdot(\mathtt{b}+\mathtt{c})^*\right\}\\
\mathsf{Approx}((\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\varepsilon)\cdot(\mathtt{c}+\varepsilon), i) &= \left\{(\mathtt{a}+\mathtt{d})^*\cdot(\mathtt{c}+\varepsilon), (\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\mathtt{c})^*\right\}\\
\mathsf{Approx}((\mathtt{a}+\varepsilon), i) &= \{(\mathtt{a}+\varepsilon)\}\\
\mathsf{Approx}((\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\varepsilon)\cdot(\mathtt{f}+\varepsilon), i) &= \left\{(\mathtt{a}+\mathtt{d})^*\cdot(\mathtt{f}+\varepsilon), (\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\mathtt{f})^*\right\}
\end{aligned}$$

As a consequence, we have $\mathsf{Approx}(\ell_1, i) = \{\ell_3, \ell_4, \ell_5, \ell_6\}$ *where:*

$$\begin{aligned}
\ell_3 &= \langle q_2, (\mathtt{a}+\mathtt{b})^*\cdot(\mathtt{c}+\varepsilon), (\mathtt{a}+\mathtt{d})^*\cdot(\mathtt{c}+\varepsilon)\rangle\\
\ell_4 &= \langle q_2, (\mathtt{a}+\mathtt{b})^*\cdot(\mathtt{c}+\varepsilon), (\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\mathtt{c})^*\rangle\\
\ell_5 &= \langle q_2, (\mathtt{a}+\varepsilon)\cdot(\mathtt{b}+\mathtt{c})^*, (\mathtt{a}+\mathtt{d})^*\cdot(\mathtt{c}+\varepsilon)\rangle\\
\ell_6 &= \langle q_2, (\mathtt{a}+\varepsilon)\cdot(\mathtt{b}+\mathtt{c})^*, (\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\mathtt{c})^*\rangle
\end{aligned}$$

and $\mathsf{Approx}(\ell_2, i) = \{\ell_7, \ell_8\}$ *where:*

$$\begin{aligned}
\ell_7 &= \langle q_3, (\mathtt{a}+\varepsilon), (\mathtt{a}+\mathtt{d})^*\cdot(\mathtt{f}+\varepsilon)\rangle\\
\ell_8 &= \langle q_3, (\mathtt{a}+\varepsilon), (\mathtt{a}+\varepsilon)\cdot(\mathtt{d}+\mathtt{f})^*\rangle
\end{aligned}$$

Thus,

$$\mathsf{PostApprox}\,(\ell, i) = \left\{\begin{array}{llll} \{\ell_3,\ell_7\}, & \{\ell_3,\ell_8\}, & \{\ell_4,\ell_7\}, & \{\ell_4,\ell_8\},\\ \{\ell_5,\ell_7\}, & \{\ell_5,\ell_8\}, & \{\ell_6,\ell_7\}, & \{\ell_6,\ell_8\} \end{array}\right\}$$

$\Diamond$

[5]We consider limit elements in the example since we have to transform any configuration into its corresponding limit anyway.

Construction of the And-Or graph The function PostApp provides us with a practical improvement of the construction of the And-Or graph, in the case of LCS. Indeed, when building the successors of an Or-node n, we can restrict ourselves to the And-nodes of $\mathsf{PostApprox}(\mathsf{limit}(n), i)$, because, by Proposition 5.6, this set contains representative of all the most precise downward-closed sets representable in $L_i \cup C_i$ that over-approximate $\mathsf{Post}(n)$. Since that set is typically smaller than $2^{C_i \cup L_i}$, this represents an improvement of the basic solution that consists in enumerating all the possible subsets of $C_i \cup L_i$.

In practice, we can completely ignore elements of $c \in C_i$ during the construction of the And-Or graph and always replace them by their corresponding $\mathsf{limit}(c) \in L_i$. This can be obtained by letting the initial node v_i of the And–Or graph be $\mathsf{limit}(c_0)$ (instead of c_0).

5.5.4 Algorithm for the coverability problem

By instantiating the EEC schema in the LCS case, we obtain Algorithm 5.3, that decides CPWSTS on LCS. As in the case of SMPN, the 'Expand' phase can be improved thanks to the algorithm of Section 5.3 (remark that in the present case, $\mathcal{C}$ is already lossy). The 'Enlarge' phase can use the techniques presented in the previous section to compute the And–Or graph in an efficient fashion.

Algorithm 5.3: A forward algorithm to decide the coverability problem on LCS.

```
Data: $\mathcal{C}$, a LCS
Data: $G_U$, the set of minimal element of the $\preccurlyeq$-upward-closed set $U$.
begin
    $i \leftarrow 1$;
    while (true) do
        if $\mathsf{Reach}(\mathsf{Under}(\mathcal{C}, C_i)) \cap U \neq \emptyset$ then
            return 'Reachable';
        else if $U$ is avoidable in $\mathsf{Reach}(\mathsf{Over}(\mathcal{N}, C_i, L_i))$ then
            return 'Unreachable';
        $i \leftarrow i + 1$ ;
end
```

5.5.5 Experimental evaluation

We have built a prototype to decide the coverability problem for LCS. It implements the improvements of the 'Expand' and 'Enlarge' phases presented above. Another improvement in the construction of the And-Or graph consists in computing only the

Case study	S	T	C	EEC
ABP	48	192	2	0.18
BRP$_1$	480	2,460	2	0.19
BRP$_2$	480	2,460	2	0.19
BRP$_3$	480	2,460	2	0.35
BRP$_4$	480	2,460	2	0.41
BRP$_5$	640	3,370	2	0.19

Table 5.2: Empirical evaluation of the EEC method on LCS. Results obtained on Intel Xeon 3Ghz with 4Gb of memory. **S** and **T**: number of states and transitions in the synchronised product; **C**: number of channels; **EEC**: execution time (in second) of the analysis of the synchronised product thanks to EEC.

states that are reachable from the initial state. Table 5.2 reports on the performance of the prototype when applied to various examples of the literature: the Alternating Bit Protocol (ABP), and the Bounded Retransmission Protocol (BRP), on which we verify five different properties [AABJ04]. Table 5.2 shows that our simple prototype performs well on these examples. All these examples have been provided to the tool under the form of several LCS, whose synchronised product has to be computed before the analysis (the time necessary to compute this synchronised product has not been taken into account in the figures of the table). Columns **S** and **T** report respectively on the number of states an number of transitions of the synchronised product.

5.5.6 Why we need And-Or graphs

Contrary to the SMPN, an And-Or graph is necessary in the case of LCS to ensure the termination of our algorithm. Let us illustrate this thanks to the LCS of Fig. 5.5. It is made up of one automaton and a single channel. Its set of reachable configurations is the $\precsim$-downward-closure of $\{\langle 1, \varepsilon\rangle\} \cup \{\langle 2, \mathtt{c}\cdot w_{ab}\rangle\}$, where w_{ab} can be any word made up of an arbitrary number of a and b's.

Let us suppose we want to prove that the number of c in the channel is always bounded, when the LCS reaches state 2. This property holds on the LCS of Fig 5.5, and it corresponds to showing that the $\precsim$-upward-closed set $\{c \mid \langle 2, \mathtt{cc}\rangle \precsim c\}$ is not reachable. Let us further suppose we are trying to compute an over-approximation of the LCS for some value $i \geq 2$ of the bound[6]. At some point, we will end up computing the set of successors of the configuration $\left\langle 2, \mathtt{c}\cdot \mathtt{a}^j \cdot \mathtt{b}^k\right\rangle$ with $|\mathtt{c}\cdot\mathtt{a}^j\cdot\mathtt{b}^k| = j+k+1 = i$, $j \geq 1$ and $k \geq 0$. Remark that this configuration is in C_i, but its successors are not. Hence we need to use limit elements to represent them.

[6]It is easy to see that the algorithm can't prove the safety of the system for $i = 1$. For this value, the only limit that contains a, b and c is $(\mathtt{a}+\mathtt{b}+\mathtt{c})^*$, which is clearly too coarse.

Actually, two *incomparable* sets of limit elements can be used for this purpose: $\ell_1 = \big\{\big\langle 2, (\mathtt{c}{+}\mathtt{a})^*{\cdot}(\mathtt{b}{+}\varepsilon)^{k+1}\big\rangle, \big\langle 2, (\mathtt{c}{+}\mathtt{a})^*{\cdot}(\mathtt{b}{+}\varepsilon)^{k}{\cdot}(\mathtt{a}{+}\varepsilon)\big\rangle\big\}$ and $\ell_2 = \big\{\big\langle 2, (\mathtt{c}{+}\varepsilon){\cdot}(\mathtt{a}{+}\mathtt{b})^*\big\rangle\big\}$. If we represent the over-approximation by an And-Or graph, this is not a problem, since we can choose between ℓ_1 and ℓ_2, and ℓ_2 allows us to prove the safety of the system (under the hypothesis that the other branches of the unfolding are also *safe*). On the other hand, if we use a *plain graph*, we have to *guess* which limit is the good one. By choosing ℓ_1, we are not able to prove the safety, which compels the algorithm to build another graph. As stated in Section 4.3.1, one could imagine two different ways to do this. The first solution would be to keep the same value of i, and build another graph (in which ℓ_2, for instance, will be chosen). It is not difficult to see that such a procedure could have to build an number of graphs that is exponential in the size of $L_i \cup C_i$. This solution is clearly less efficient than the PTime algorithm that explores And-Or graphs. The other solution could be to try to refine the bound, and build a new approximation for the value $i+1$. However, suppose now that the *bad guess* occurs repeatedly for any i. In this case, it is not difficult to see that the algorithm will fail to terminate and prove the safety of the system, although the system is safe !

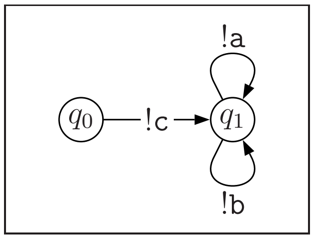


Figure 5.5: A LCS with one channel: locations are represented by circles (location q_0 is initial); transitions by arrows. The labels are the operations on the channel.

5.6 Discussion

In this chapter, we have shown how the (general) algorithmic schema introduced in Chapter 4 can be turned into efficient solutions to decide coverability on classes of WSTS that are interesting in practice: the strongly monotonic SMPN and the LCS. Some of the optimisations that we have introduced are targeted towards a special class of system. For instance, we have shown that the And-Or graph is degenerated in the case of strongly monotonic SMPN, and we have provided an efficient procedure to compute the most precise over-approximations of the successors of a LCS configuration.

The other optimisations are generic, in the sense that they can be applied to any class of WSTS that EEC can handle. It is the case of the efficient algorithm of Section 5.3, that can always be applied during the 'Expand' phase, whatever the class of WSTS we consider. The main optimisation of this algorithm consists in keeping *maximal elements* only during the exploration of the finite WSTS.

A natural question regarding the optimisation consisting in keeping maximal elements only is to know whether it could be applied to *infinite-state* WSTS. In the case of these systems, *acceleration techniques* have to be applied in order to compute, in a finite number of steps, representative of an infinite number of reachable configurations. The idea of keeping maximal elements combined with acceleration techniques has been applied by Finkel in [Fin91] to compute the coverability set of PN in an efficient fashion. We discuss this *Minimal Coverability Tree algorithm* in the next chapter, and show, unfortunately, that it is flawed, in the sense that it can compute an under-approximation of the coverability set.

This fact could be hint that keeping maximal elements only (in this case, maximal ω-markings) is not a correct optimisation when dealing with infinite state systems that require acceleration techniques. Nevertheless, one can still improve the algorithms in the case of infinite state systems, by using the new optimisation technique that we introduce in the next chapter.

Chapter 6

Efficient computation of the minimal coverability set for PN

In the two previous chapters, we have discussed the EEC algorithm that can be used to solve the coverability problem on a wide range of WSTS, including monotonic extensions of Petri nets. As we have seen, EEC decides the coverability problem without computing the coverability set: it is indeed not computable for most classes of WSTS that are used in practice.

The most notable exception is the class PN of (plain) Petri nets for which the coverability is computable. Having the coverability set at our disposal is interesting *per se*, because it allows us to decide other problems than the coverability problem such as (see section 2.6.1): PBEPN (place–boundedness), BOUNDEPN (boundedness), QLEPN (quasi–liveness),...

Hence, practical algorithms to compute the coverability set of PN are needed. A first algorithm to compute the coverability set of PN has been introduced by Karp and Miller in [KM69] (we have reviewed it in Section 3.2.4). However, as noted by Finkel in [Fin91], that algorithm is not efficient enough to handle Petri nets of large dimensions. This emphasises the need for an efficient algorithm to compute the coverability set of PN.

In the present chapter, we review two optimisations of the `Karp&Miller` algorithm, found in the literature. The first one is the minimal coverability tree (MCT for short) introduced by Finkel in [Fin91]. The latter is a variation of the former: it has been introduced in [Lut95] and implemented in the PEP tool [Gra97a].

It is important to mention that the paper [Fin91], that introduces the MCT algorithm, has been often cited in the literature since its publication. At the time we write this thesis, the site `scholar.google.com` counts 49 citations of [Fin91]. In some of these citations, such as [VBvdA01], the MCT algorithm is directly exploited as is. In other works such as [BCR01] or [LL00], applications of ideas borrowed from the MCT algorithm to other kinds of systems than PN are presented. The arguments developed by the authors of these papers explicitly refer to the proof of [Fin91].

Unfortunately, these algorithms are both *incorrect*: the MCT algorithm may compute an under-approximation of the coverability set, and the algorithm implemented in Pep, and designed to correct the flaw of MCT algorithm, might not terminate. We demonstrate these two errors by means of examples on which the algorithms fail, in Section 6.1 and Section 6.2 respectively.

Remark that the problem uncovered in the MCT algorithm does *not* imply that the results in [VBvdA01, LL00, BCR01] (the papers we were alluding to), or in other papers citing [Fin91] are flawed (they have to be reconsidered). But it is certainly not acceptable anymore to refer to the arguments developed in [Fin91] in the proof of another algorithm.

The idea applied in the MCT algorithm consists, roughly speaking, in keeping maximal markings only in the trees that are maintained during the computation. This is achieved by the means of reduction rules that cut (supposedly) irrelevant subtrees. The subtle mistakes we have uncovered in both algorithms tend to show that this optimisation cannot be applied in the case of Petri nets. Remember, however, that we have successfully drawn on the same idea in Section 5.3. Still, in this case, we were considering *finite* WSTS where no acceleration techniques are needed, which is not the case for PN in general.

As a consequence, we claim that any optimisation technique that has to be applied in the presence of accelerations, must, in some sense, take the acceleration into account. In the `Karp&Miller` procedure, the results of an acceleration is computed from a marking and its set of ancestors in the tree. The relation 'is an ancestor of' is thus the one that produces the accelerations. By considering markings isolatedly for pruning – as does the MCT algorithm – one looses the relationships that exists between them. Thus, accelerations can be missed because of *wrong cuts* in the tree.

According to these remarks, we introduce in Section 6.3.1 a new algorithm that computes the coverability set of Petri nets in an efficient fashion. Instead of keeping maximal sets of markings, this algorithm maintains maximal sets of *pairs* of markings (wrt to an order that we introduce). This allows to relate the individual markings, and cope with the accelerations. The compelling simplicity of the proofs should convince the reader that this new solution avoids the mistake of its predecessors.

Remark concerning the ω-markings Throughout this chapter, we will manipulate markings and ω-markings to represent downward-closed sets of markings. According to the discussions of Section 2.2 and Section 3.1.1, we will consider the ordered set $\left\langle \mathbb{N}^{|P|}, \preccurlyeq \right\rangle$ with its associated adequate domain of limits $\langle \mathcal{L}, \preccurlyeq, \gamma \rangle$. Remark that in the present case, we use the same symbol $\preccurlyeq$ to denote both the ordering ranging over $\mathbb{N}^{|P|}$

and the ordering ranging over $\left(\mathbb{N} \cup \{\omega\}\right)^{|P|}$ (which was denoted by $\preccurlyeq_e$ in the previous chapter[1]). This is not problematic however, since the definitions of these two orderings are the same when considering markings only (compare Definition 2.19 with that of $\preccurlyeq_e$ at the beginning of Section 3.1.1). As a consequence, notice that for any Petri net $\mathcal{N}$, the covering set of $\mathcal{N}$ is $\gamma\left(\mathsf{Reach}\left(\mathcal{N}\right)\right) = \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$.

We also rely on the extension of the Post operator to handle ω-markings, that has been defined in Section 3.1.1 (see in particular Proposition 3.2).

The content of this chapter is still unpublished material.

6.1 The minimal coverability tree algorithm

Let us first recall the algorithm introduced in [Fin91] to compute the coverability set of PN in an efficient fashion. This procedure can be regarded as an optimisation of the `Karp&Miller` procedure. The aim is to ensure that all the intermediate trees computed by the algorithm are as small as possible, in the sense that all the nodes in the trees have to be incomparable w.r.t to $\preccurlyeq$. This requirement stems from the observation that the transitions of a Petri net are monotonic w.r.t to $\preccurlyeq$. Hence, only the maximal ω-markings have to be kept during the construction of a coverability tree.

The principle of this optimisation consists in building a tree *à la* `Karp&Miller` and ensuring the minimality criterion through the application of *reduction rules.* These rules *remove* subtrees when comparable nodes are encountered. Unfortunately, as we will show in the present section, this algorithm is flawed, because it could, in some cases, compute a tree whose nodes do not cover the whole set of reachable markings.

This section is organised as follows: we first recall the algorithm proposed in [Fin91], then we expose our counter-example.

6.1.1 The algorithm

Algorithm 6.1 presents the minimal coverability tree algorithm of [Fin91]. Before discussing it, we introduce the two auxiliary functions removeSubtree and removeSubtreeExceptRoot that will be used in the reduction rules to suppress redundant subtrees. Given a labelled tree $\mathcal{T}$ and a node n of $\mathcal{T}$, removeSubtree$(n, \mathcal{T})$ removes the subtree rooted by n from $\mathcal{T}$. The function removeSubtreeExceptRoot$(n, \mathcal{T})$ is similar to removeSubtree$(n, \mathcal{T})$ except that the root node n is not removed.

We can now consider Algorithm 6.1. As one can see, this algorithm is very similar to the `Karp&Miller` procedure. The variable $\mathcal{T}$ holds the tree built by the algorithm. A

[1]We had chosen a different notation because we sometimes needed to explicitly make the difference between configurations and limit elements in the EEC algorithm. This is not the case anymore here.

set *to_treat* holds the nodes that are waiting to be processed (initially, this set contains the root node of the labelled tree, whose label is the initial marking of the PN). The main loop consists in picking up a node n from *to_treat*, and processing this node. First remark that, in the case where there is another node $\overline{n}$ of the tree that has the same label as n, this node does not need to be developed (see line (a)). Then, the Post of $\Lambda(n)$ is computed and all the markings $\mathbf{m} \in \mathsf{Post}(\Lambda(n))$ are considered successively. Two cases may occur:

1. Either, there exists an ancestor $\overline{n}$ of n that is labelled by an ω-marking strictly smaller than $\mathbf{m}$ (see line (b)). In this case, an acceleration occurs, and the result of the acceleration is assigned to the label of $\overline{n}$. As a consequence, the whole subtree of $\overline{n}$ is deleted and $\overline{n}$ is put back into *to_treat* because it has been modified. We also have to stop treating the successors of n, because this node has been deleted along with the subtree of $\overline{n}$. Hence the **break** statement of line (b.1). When this statement is executed, the algorithm immediately exits the **foreach** loop without considering the remaining markings in $\mathsf{Post}(\Lambda(n))$.

2. Or there is no such ancestor, and a new node with label $\mathbf{m}$ has to be added as a successor of n, *under the condition that* there is no node $\overline{n}$ whose label is strictly larger than $\mathbf{m}$ in the tree. If such a $\overline{n}$ exists, then, it is argued that n does not need to be present in the tree, since $\Lambda(n)$ is covered by $\Lambda(\overline{n})$, and since all the successors of $\Lambda(n)$ will be covered by some successors of $\Lambda(\overline{n})$, by monotonicity.

After that step of processing the successors of n, *to_treat* is updated by adding to it the newly computed successors of n. Remark that if n has been suppressed (in the case of an acceleration), *to_treat* will not be updated at that point (however, the node $\overline{n}$ that is labelled by the result of the acceleration has already been added to *to_treat*).

At this point, we are ensured that there are *no* two strictly comparable ω-markings along a branch of the tree. However, pairs of comparable nodes might remain in the tree[2], and, since the `MinimalCoverabilityTree` procedure aims at keeping $\preccurlyeq$-incomparable elements only, some nodes may need to be deleted. This is the purpose of the **while** loop of line (d).

6.1.2 Counter-example to the algorithm

In this section, we show, by means of an example, that Algorithm 6.1 may fail to compute a coverability set. More precisely, we present a Petri net and show a *possible* erroneous computation of the algorithm. Indeed, it is important to remark that Algorithm 6.1 is *non–deterministic*, since no constraints are given on the order in which

[2]This happens for instance when a newly added successor n has a label that is larger than another node n' that is not an ancestor of n and was present in the tree before the insertion of n.

Algorithm 6.1: The minimal coverability tree algorithm [Fin91].

```
Data: A PN $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$.
Result: The minimal coverability set of $\mathcal{N}$.
MinimalCoverabilityTree
begin
    $\mathcal{T} \leftarrow \langle N, B, n_0, \Lambda \rangle$ where $N = \{n_0\}$, $B = \emptyset$ and $\Lambda(n_0) = \mathbf{m}_0$ ;
    $to\_treat \leftarrow \{n_0\}$ ;
    while $to\_treat \neq \emptyset$ do
        Select some node $n$ in $to\_treat$;
        $to\_treat \leftarrow to\_treat \setminus \{n\}$ ;
(a)     if $\nexists \overline{n} \in N$ s.t. $\Lambda(\overline{n}) = \Lambda(n)$ then
            foreach $\mathbf{m} \in \mathsf{Post}(\Lambda(n))$ do
(b)             if $\exists \overline{n} : B^*(\overline{n}, n)$ and $\Lambda(\overline{n}) \prec \mathbf{m}$ then
                    Let $\overline{n}$ be the highest node s.t. $B^*(\overline{n}, n) \wedge \Lambda(\overline{n}) \prec \mathbf{m}$ ;
                    Let $\mathcal{A}$ be $\{n' \in N \mid B^*(n', n)\}$ ;
                    $\Lambda(\overline{n}) \leftarrow \mathsf{Accelerate}(\mathcal{A}, \mathbf{m})$ ;
                    $to\_treat \leftarrow \big(to\_treat \setminus \{n' \mid B^*(\overline{n}, n')\}\big) \cup \{\overline{n}\}$ ;
                    $\mathsf{removeSubtreeExceptRoot}(\overline{n}, \mathcal{T})$ ;
(b.1)               break ;
(c)             else if $\nexists \overline{n} \in N$ s.t. $\mathbf{m} \prec \Lambda(\overline{n})$ then
                    Let $n'$ be a new node s.t. $\Lambda(n') = \mathbf{m}$ ;
                    $N \leftarrow N \cup \{n'\}$;
                    $to\_treat \leftarrow to\_treat \cup \{n'\}$;
                    $B \leftarrow B \cup (n, n')$ ;
(d)         while $\exists n_1, n_2 \in N : \Lambda(n_1) \prec \Lambda(n_2)$ do
                $to\_treat \leftarrow to\_treat \setminus \{n \mid B^*(n_1, n)\}$ ;
                $\mathsf{removeSubtree}(n_1, \mathcal{T})$ ;
    return($\{\Lambda(n) \mid n \in T\}$) ;
end
```

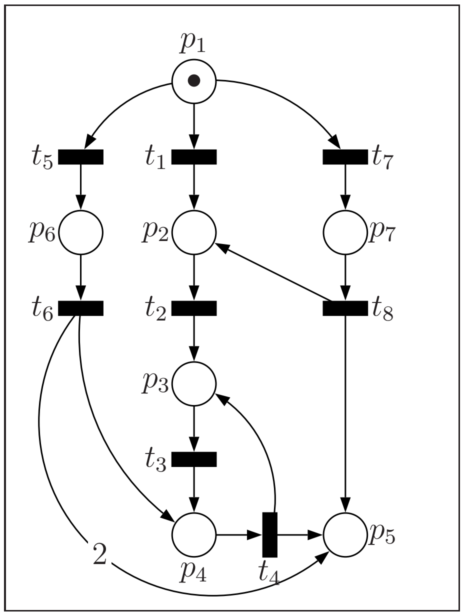


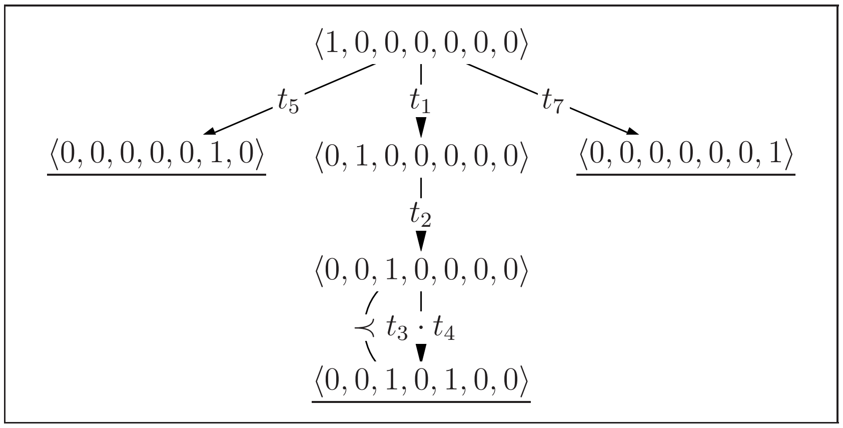
Figure 6.1: A PN on which the algorithm proposed in [Fin91] may not compute the whole coverability set. Notice that p_5 is unbounded.

the nodes have to be chosen in *to_treat*. Thus, the algorithm fails on our counter-example for the order we present, but might compute a correct result when another order is considered. However, we conjecture that the counter-example can be adapted to let the algorithm fail for any given order, which would imply that one cannot fix Algorithm 6.1 just by imposing a simple criterion on the order in which the nodes are selected. On the contrary, when discussing the counter-example in section 6.1.3 we will identify an important concept that is missing in the algorithm.

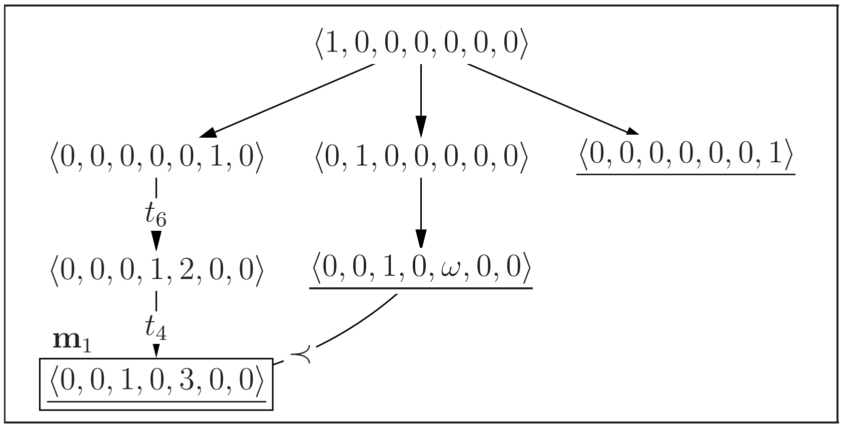
Let us now present the analysis of the net of Figure 6.1 by Algorithm 6.1. The main steps of the analysis are detailed at Figure 6.2 and Figure 6.3. When a subtree is deleted from $\mathcal{T}$, its root node is drawn shaded in the figure. The thick grey arrows on these figures are not part of the tree, and the reader should ignore them for the moment. Their precise meaning and utility will be discussed in section 6.1.3.

It is not difficult to see that not all the reachable markings in the net of Figure 6.1 are covered by the labels of the tree (Figure 6.3(b)) obtained at the end of the algorithm. Indeed, one cannot bound the number of tokens that any firable sequence of transitions produces in p_5. However, according to the result of the algorithm, no more than two tokens can ever be present in p_5. Let us provide more detailed comments on the main steps of the execution:

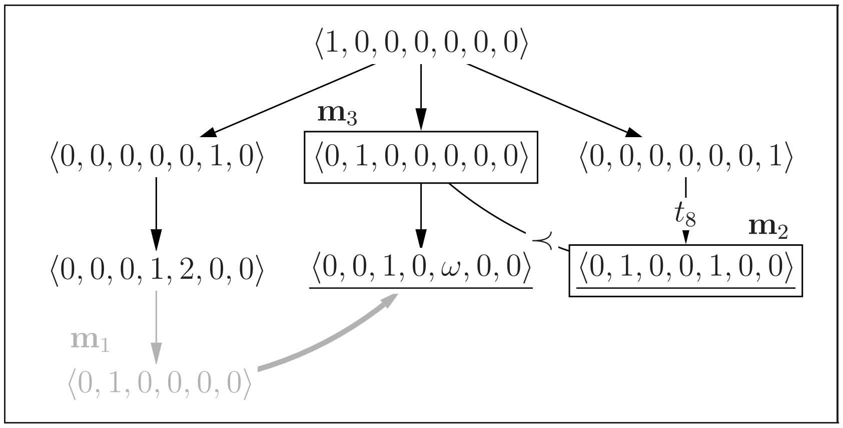
Step 1 – Figure 6.2(a) At the first step, the three successors of the initial marking are computed, and added to $\mathcal{T}$. Then, $n = \langle 0, 1, 0, 0, 0, 0, 0 \rangle$ is picked up from



(a) Step 1.

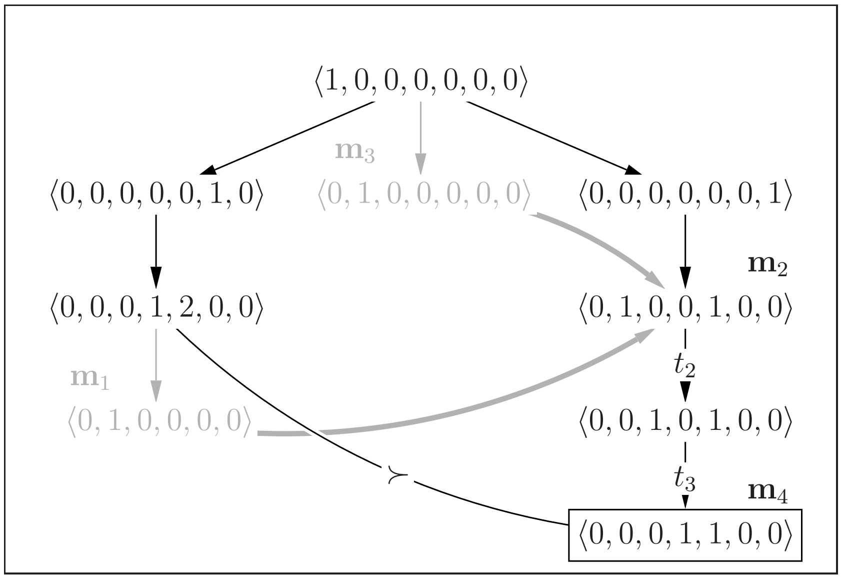


(b) Step 2.

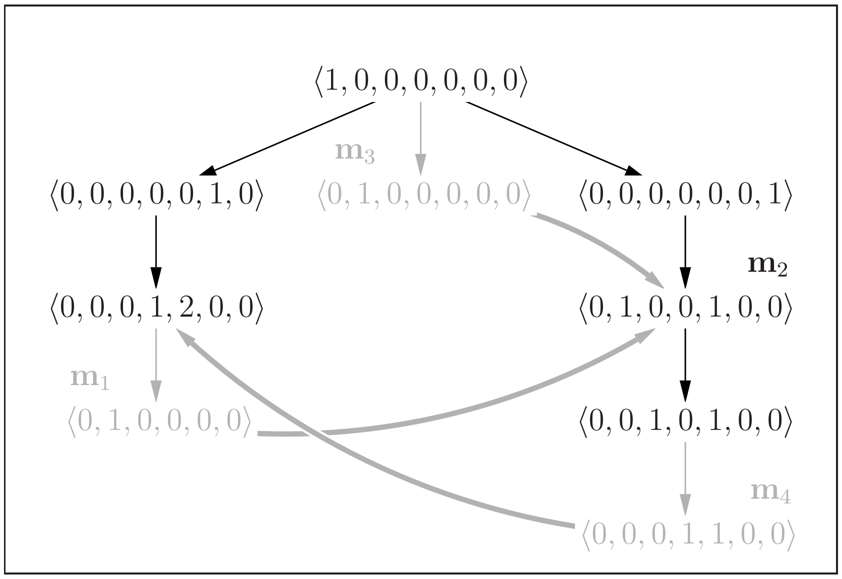


(c) Step 3.

Figure 6.2: A counter-example to the MCT algorithm. Nodes and edges in grey have been removed. An underlined markings means that the node is still in the frontier at the end of the step. Thick grey arrows represent the *proofs*.



(a) Step 4.



(b) The result of the algorithm.

Figure 6.3: A counter-example to the MCT algorithm (continued). Nodes and edges in grey have been removed (or their creation has been avoided). An underlined markings means that the node is still in the frontier at the end of the step. Thick grey arrows represent the *proofs*.

to_treat. Its successors are computed by firing t_2 then t_3 and the corresponding branches are added to $\mathcal{T}$. Remark that all the ω-markings obtained so far are incomparable. By the firing of t_4, we obtain $\langle 0,0,1,0,1,0,0\rangle$ which is strictly greater than its ancestor $\langle 0,0,1,0,0,0,0\rangle$. Thus, the algorithm enters the **if** at line (b). The Accelerate function returns the ω-marking $\langle 0,0,1,0,\omega,0,0\rangle$ which replaces its predecessor $\langle 0,0,1,0,0,0,0\rangle$ and is put again into *to_treat*.

Step 2 – Figure 6.2(b) Figure 6.2(b) shows the labelled tree obtained after the nodes $\langle 0,0,0,0,0,1,0\rangle$ and $\langle 0,0,0,1,2,0,0\rangle$ have been successively picked up from *to_treat* and their successors have been computed. One then obtains the marking $\mathbf{m}_1 = \langle 0,0,1,0,3,0,0\rangle$, which is strictly smaller than $\langle 0,0,1,0,\omega,0,0\rangle$. Hence, $\mathbf{m}_1$ is not added as a successor of $\langle 0,0,0,1,2,0,0\rangle$ (see line (c)).

Step 3 – Figure 6.2(c) At that point, the node $\langle 0,0,0,0,0,0,1\rangle$ is selected and its unique successor $\mathbf{m}_2 = \langle 0,1,0,0,1,0,0\rangle$ is computed. $\mathbf{m}_2$ is strictly larger than $\mathbf{m}_3 = \langle 0,1,0,0,0,0,0\rangle$, but $\mathbf{m}_3$ is not an ancestor of $\mathbf{m}_2$. Hence, $\mathbf{m}_2$ is put into *to_treat*, and the subtree rooted at $\mathbf{m}_3$ (including the ω-marking resulting from the acceleration) disappears from $\mathcal{T}$ (line (d)).

Step 4 – Figure 6.3(a) and 6.3(b) $\mathbf{m}_2$ is the next marking to be looked at. From this marking, we can compute two successive successors by unrolling the branch labelled $t_2 \cdot t_3$. This is shown at Figure 6.3(a). We obtain $\mathbf{m}_4 = \langle 0,0,0,1,1,0,0\rangle$ which is strictly smaller than $\langle 0,0,0,1,2,0,0\rangle$, and does not appear in the tree for that reason (line (c)). At that point, the set *to_treat* is empty and the algorithm terminates. The labelled tree $\mathcal{T}$ computed by the algorithm is shown at Figure 6.3(b). However, that labelled tree is not a coverability tree because some reachable markings are not covered by any node of this tree. Indeed, if it were the case, we would conclude that the place p_5 is bounded, which is obviously not the case: the sequence $t_1 \cdot t_2 \cdot (t_3 \cdot t_4)^n$ puts n tokens in p_5 and can be fired for any $n \geq 0$.

6.1.3 Discussion of the counter-example

Let us explain more precisely the problem that occurs when Algorithm 6.1 fails to compute a coverability set. The main idea exploited in Algorithm 6.1 is the following: when two nodes n and n' s.t. $\Lambda(n) \prec \Lambda(n')$ are present in the tree, then, by monotonicity, one can get rid of n (and its subtree). In that case, we say that n' *is a proof for* n, in the sense that n' carries enough information to allow us to cut the subtree rooted at n. This deletion seems acceptable because Petri nets are *monotonic* and because the algorithm makes the assumption that either the subtree rooted at n' will be fully developed, or n' will in turn be covered by another node n''.

Let us now show how this reasoning may fail. For that purpose, let us refer to Figure 6.2 and Figure 6.3, that present the flawed result of Algorithm 6.1 when it is applied on the Petri net of Figure 6.1. On Figure 6.2 and Figure 6.3, we have represented the notion of *proof* thanks to grey arrows: such an arrow going from node n to node n' means that *n' is a proof for n*. Remark that when the subtree rooted at $\mathbf{m}_3$ has been deleted (Figure 6.2(c) and 6.3(a)), we have *modified* the proof relation.

Now, if one considers the *proof* relation as well as the *edges* of the tree, the following cycle appears in the result of the algorithm, shown at Figure 6.3(b) (the names of the markings refer to those used along the steps of Figure 6.2 and Figure 6.3):

$$\mathbf{m}_1, \mathbf{m}_2, \langle 0,0,1,0,1,0,0\rangle, \mathbf{m}_4, \langle 0,0,0,1,2,0,0\rangle, \mathbf{m}_1$$

This cycle is problematic because, roughly speaking, the subtree rooted at $\mathbf{m}_1$ has not been developed with the assumption that the subtree rooted at $\langle 0,1,0,0,1,0,0\rangle$ would be completely developed. This implies that the subtree of $\mathbf{m}_4$ must be completely developed. However the development of that subtree has been stopped and discharged on the full construction of the subtree rooted at $\langle 0,0,0,1,2,0,0\rangle$. But this last subtree contains $\mathbf{m}_1$... There is thus a cycle in the assumptions that must be satisfied in order to ensure the construction of a proper coverability tree.

6.1.4 Remark concerning the proof of [Fin91]

Thanks to the counter-example we have just discussed, we are now able to identify the mistake in the proof of [Fin91]. It is to be found at page 232 of the proceedings [Roz93], in point 1.1 of the proof, that states that, for any $\mathbf{m}$ in $\mathsf{Reach}(\mathcal{N})$, there is a node n in the tree built by the MCT algorithm s.t. $\mathbf{m} \preccurlyeq \Lambda(n)$. That part of the proof is done by induction on the length of the sequence of transitions that allows to reach $\mathbf{m}$ from $\mathbf{m}_0$. The problem appears in the inductive part. The induction hypothesis states that for any marking $\mathbf{m}'$ with $\mathbf{m}_0 \xrightarrow{\sigma'} \mathbf{m}'$ and $|\sigma'| \leq x$, there is n' in the tree s.t. $\mathbf{m}' \preccurlyeq \Lambda(n')$. A marking $\mathbf{m}$, a sequence σ and a transition t s.t. $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}' \xrightarrow{t} \mathbf{m}$ with $|\sigma \cdot t| = x+1$ are considered. By induction hypothesis, there is n' in the tree s.t. $\mathbf{m}' \preccurlyeq \Lambda(n')$. By monotonicity, t is firable from $\mathbf{m}'$ and $\mathbf{m}' \xrightarrow{t} \mathbf{m}''$ with $\mathbf{m} \preccurlyeq \mathbf{m}''$. Two cases are considered: either there is a node n'' and an edge (n', n'') in the tree with $\mathbf{m}'' \preccurlyeq \Lambda(n'')$, or it is not the case. In the latter case, the proof states that '*by definition of the minimal coverability tree, there is $\overline{n}$ in the tree s.t. $\mathbf{m}'' \preccurlyeq \Lambda(\overline{n})$*'. However, the existence of this node $\overline{n}$ is not justified any further (and we have seen that it might not exist). Moreover, the induction hypothesis cannot be invoked at this point, because there is no guarantee that $\mathbf{m}''$ is reachable from $\mathbf{m}_0$ by a sequence of length $\leq x$.

6.2 The INA algorithm

As a matter of fact, the bug presented in the previous section has independently been discovered by the team of Prof. Starke, around 1995. To the best of our knowledge, however, this result has never been published, except under the form of the master's thesis (in German) of K. Lüttge [Lut95].

In the same work, a correction of Algorithm 6.1 is proposed. This new algorithm has been implemented into the INA tool [INA], which is part of the PEP toolbox [Gra97a]. This corrected algorithm is ensured to return a coverability set when it terminates (for the proof, see [Lut95]). Although the authors of [Lut95] provide a proof of termination of their algorithm, we show that the termination is *not guaranteed* in general (and so, the 'proof' of [Lut95] is flawed). Indeed, we provide, in this section, a counter-example on which the algorithm does not terminate. Let us begin by recalling the algorithm.

6.2.1 The algorithm

Crudely speaking, Algorithm 6.1 fails to compute a coverability tree because it sometimes *deletes too many nodes*. As we have seen in the previous section, it may be the case that a marking $\mathbf{m}$ is reachable, but is not kept in the tree, because it is covered by another marking labelling a node n. Hence, when a subtree containing n is deleted, the covering of $\mathbf{m}$ is not ensured anymore. The idea presented in [Lut95] tries to prevent this. It consists in ensuring that, whenever a subtree S gets deleted, the nodes that were covered by S remain covered by other nodes that are still in the tree. For this purpose, the two functions removeSubtree and removeSubtreeExceptRoot, have to be modified: after the actual deletion of the subtree, each remaining node in $\mathcal{T}$ but not in *to_treat* is considered. If one of these nodes has a successor that is not covered by $\mathcal{T}$, the successor is added to $\mathcal{T}$ and *to_treat*.

More precisely, the algorithm of [Lut95] is obtained, by replacing, in Algorithm 6.1, the calls to removeSubtree and removeSubtreeExceptRoot by two new functions removeSubtreeINA and removeSubtreeExceptRootINA. The first one is shown at Algorithm 6.2. It first removes from $\mathcal{T}$ the maximal subtree rooted at n. This is carried out through a call to removeSubtree, as defined in section 6.1.1. Then, the function scans all the nodes of $\mathcal{T}$ that are not waiting to be processed (i.e., not in *to_treat*), and tests whether their successors are all covered by nodes in $\mathcal{T}$. If it is not the case, a relevant node is added to $\mathcal{T}$. The function removeSubtreeExceptRootINA is obtained by replacing the call to RemoveSubtree by a call to RemoveSubtreeExceptRoot in Algorithm 6.2.

6.2.2 Counter-example to the algorithm

Let us now present a counter-example that shows that the algorithm proposed in [Lut95] may not terminate. Remark that, since this algorithm is based on Algorithm 6.1, it is non-deterministic too. Our remarks regarding the non-determinism of Algorithm 6.1 apply here too.

Algorithm 6.2: The function to remove subtrees in the algorithm of [Lut95].

```
removeSubtreeINA(node $n$, tree $\mathcal{T} = \langle N, root, B, \Lambda \rangle$)
begin
    removeSubtree($n$, $\mathcal{T}$);
    foreach $n' \in N \setminus to\_treat$ do
        foreach transition $t$ firable in $\Lambda(n')$ do
            let $\mathbf{m}$ be s.t.: $\Lambda(n') \xrightarrow{t} \mathbf{m}$;
            if there is no $n'' \in N$ s.t.: $\mathbf{m} \preccurlyeq \Lambda(n'')$ then
                /* A successor of $n'$ is not covered anymore */
                let $n_1$ be a new node with $\Lambda(n_1) = \mathbf{m}$;
                $N \leftarrow N \cup \{n_1\}$ ;
                $B \leftarrow B \cup \{(n', n_1)\}$ ;
                $to\_treat \leftarrow to\_treat \cup \{n_1\}$ ;
end
```

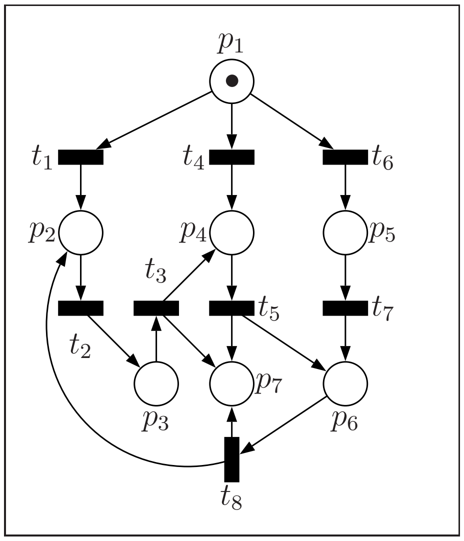
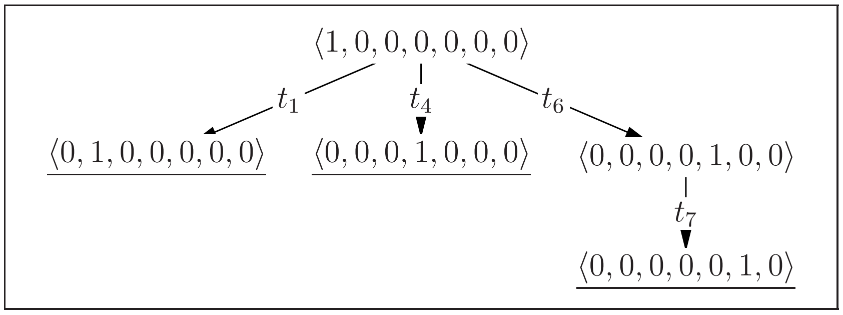
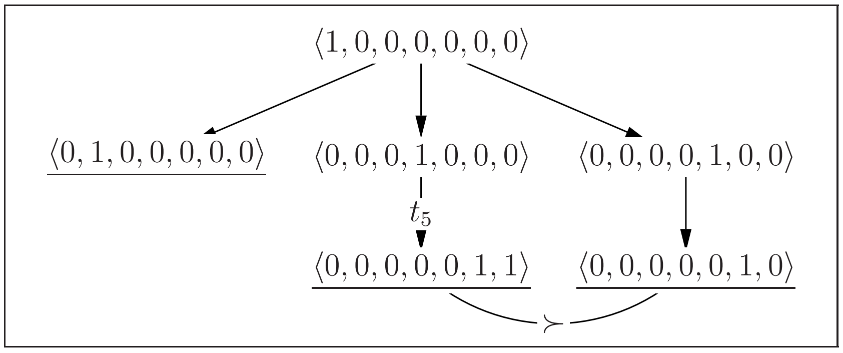


Figure 6.4: A PN on which the algorithm proposed in [Lut95] may not terminate.



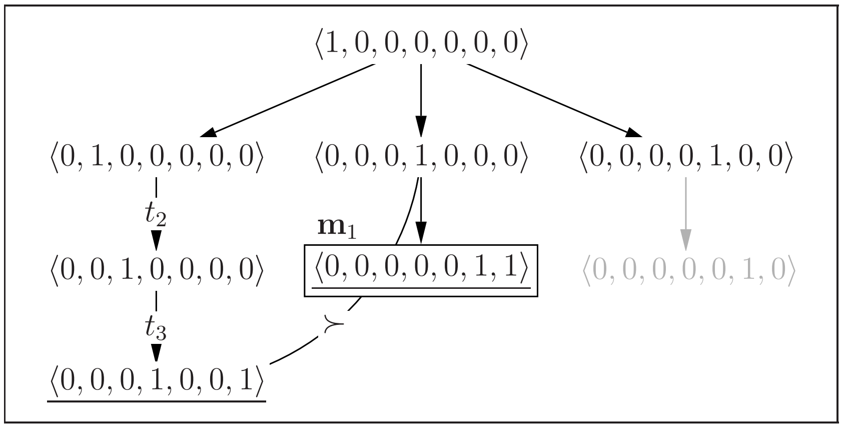
$$\begin{tikzcd}
 & \langle 1,0,0,0,0,0,0\rangle \arrow[dl, "t_1"'] \arrow[d, "t_4"] \arrow[dr, "t_6"] & \\
\underline{\langle 0,1,0,0,0,0,0\rangle} & \underline{\langle 0,0,0,1,0,0,0\rangle} & \langle 0,0,0,0,1,0,0\rangle \arrow[d, "t_7"] \\
 & & \underline{\langle 0,0,0,0,0,1,0\rangle}
\end{tikzcd}$$

(a) Step 1



$$\begin{tikzcd}
 & \langle 1,0,0,0,0,0,0\rangle \arrow[dl] \arrow[d] \arrow[dr] & \\
\underline{\langle 0,1,0,0,0,0,0\rangle} & \langle 0,0,0,1,0,0,0\rangle \arrow[d, "t_5"] & \langle 0,0,0,0,1,0,0\rangle \arrow[d] \\
 & \underline{\langle 0,0,0,0,0,1,1\rangle} \arrow[r, dash, bend right, "\succ"] & \underline{\langle 0,0,0,0,0,1,0\rangle}
\end{tikzcd}$$

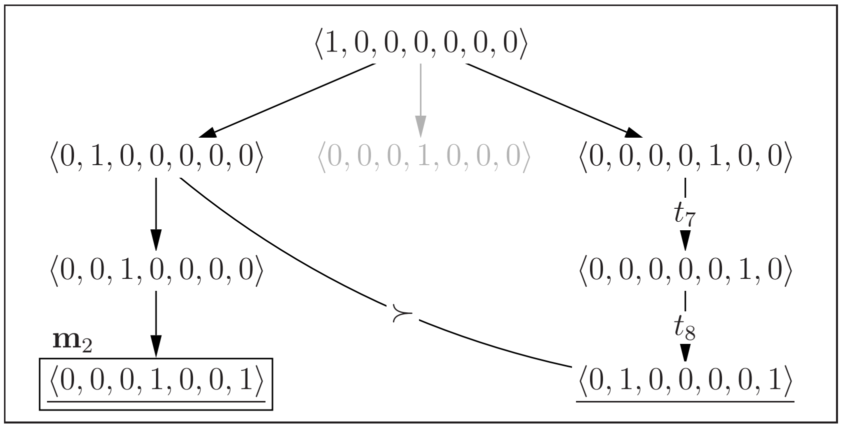
(b) Step 2



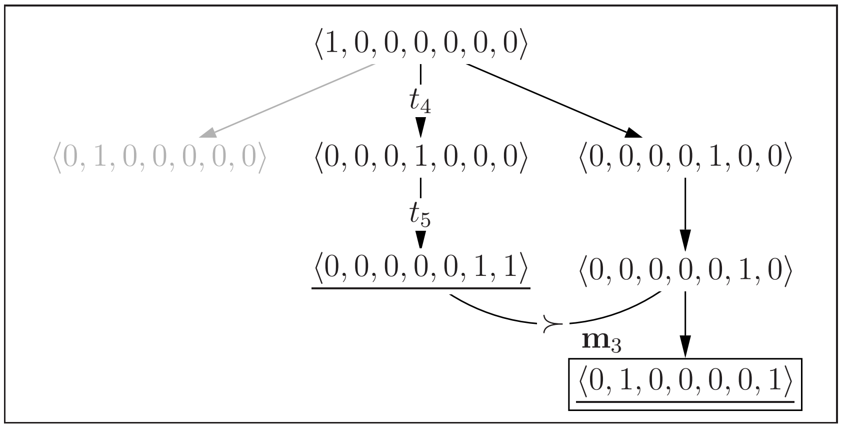
$$\begin{tikzcd}
 & \langle 1,0,0,0,0,0,0\rangle \arrow[dl] \arrow[d] \arrow[dr] & \\
\langle 0,1,0,0,0,0,0\rangle \arrow[d, "t_2"] & \langle 0,0,0,1,0,0,0\rangle \arrow[d] \arrow[d, dash, bend right] & \langle 0,0,0,0,1,0,0\rangle \arrow[d] \\
\langle 0,0,1,0,0,0,0\rangle \arrow[d, "t_3"] & \overset{\mathbf{m}_1}{\boxed{\underline{\underline{\langle 0,0,0,0,0,1,1\rangle}}}} \arrow[dl, dash, bend left, "\succ"] & \langle 0,0,0,0,0,1,0\rangle \\
\underline{\langle 0,0,0,1,0,0,1\rangle} & &
\end{tikzcd}$$

(c) Step 3 – After the deletion of m_1, $\langle 0,0,0,0,0,1,0\rangle$ is not covered anymore.

Figure 6.5: The counter-example to the algorithm of [Lut95]. Nodes in grey have been deleted from the tree. An underlined markings means that the node is in the frontier.



(a) Step 4 – After the deletion of m_2, $\langle 0,0,0,1,0,0,0\rangle$ is not covered anymore.



(b) Step 5 – After the deletion of m_3, $\langle 0,1,0,0,0,0,0\rangle$ is not covered anymore.

Figure 6.6: The counter-example to the algorithm of [Lut95] (continued). Nodes in grey have been deleted from the tree. An underlined markings means that the node is in the frontier.

The counter example consists to apply the algorithm on the Petri net shown in Figure 6.4. The main steps of the computation are shown at Figure 6.5 and Figure 6.6, and detailed below. When a subtree is deleted from $\mathcal{T}$, its root node is drawn shaded in the figure.

The counter-example exhibits a possibly cyclic behaviour of the algorithm: the tree one obtains at the end of step 5 (Figure 6.6(b)) is the same tree that is computed at the end of step 2 (Figure 6.5(b)), after the deletion of $\mathbf{m}_1$ (which is covered by a new marking). Thus, the algorithm cycles on that example. Let us now look into the main steps of the computation with more details:

Step 1 – Figure 6.5(a) The three successors of the initial marking are computed, by firing t_1, t_4 and t_6. Then, the successor by t_7 of $\langle 0,0,0,0,1,0,0\rangle$ is computed and added into the tree. Remark that all the markings computed so far are incomparable.

Step 2 – Figure 6.5(b) We fire t_5 from $\langle 0,0,0,1,0,0,0\rangle$ and obtain $\langle 0,0,0,0,0,1,1\rangle$, which is added to the tree. Since this marking is larger than the formerly computed marking $\langle 0,0,0,0,0,1,0\rangle$, the latter is removed from the tree (in the **while** loop at line (d)).

Step 3 – Figure 6.5(c) Then, $\langle 0,1,0,0,0,0,0\rangle$ is picked up from *to_treat* and its successors $\langle 0,0,1,0,0,0,0\rangle$ and $\langle 0,0,0,1,0,0,1\rangle$ are successively computed, added to the tree and picked up from *to_treat*. Hence, the subtree rooted at $\langle 0,0,0,1,0,0,0\rangle$ is deleted from $\mathcal{T}$, by the function removeSubtreeINA (in the **while** loop at line (d) of the algorithm). But this implies the deletion of $\mathbf{m}_1$ and the function detects that $\langle 0,0,0,0,0,1,0\rangle$ (shaded on the figure) is not covered anymore. It is put back into $\mathcal{T}$.

Step 4 – Figure 6.6(a) Since $\langle 0,0,0,0,0,1,0\rangle$ is in $\mathcal{T}$ and *to_treat* again, it can be selected for treatment. Its successor $\langle 0,1,0,0,0,0,1\rangle$ is computed and added to the tree. This marking is larger than $\langle 0,1,0,0,0,0,0\rangle$. Hence, we remove the subtree rooted at $\langle 0,1,0,0,0,0,0\rangle$, again by calling removeSubtreeINA (in the **while** loop at line (d)). The function detects that $\langle 0,0,0,1,0,0,0\rangle$, successor of the root node, is not covered anymore, because $\mathbf{m}_2$ has been deleted. Thus, $\langle 0,0,0,1,0,0,0\rangle$ is added to the tree.

Step 5 – Figure 6.6(b) The scenario we had observed at the previous step repeats: $\langle 0,0,0,1,0,0,0\rangle$ is selected from *to_treat*, and its successor $\langle 0,0,0,0,0,1,1\rangle$ is computed, added to $\mathcal{T}$ and selected for treatment. Since this successor is larger than $\langle 0,0,0,0,0,1,0\rangle$, the latter is deleted with its whole subtree, including $\mathbf{m}_3$. However, $\langle 0,0,0,1,0,0,0\rangle$ (successor of the root) is not covered anymore and is thus put back into $\mathcal{T}$ and *to_treat*.

At that point, we obtain the same tree that we had at the beginning of step 3: we can fire t_2 and t_3, suppress $\mathbf{m}_1$, and so forth. Thus, we can iterate steps 3–4–5 infinitely often.

6.3 An efficient algorithm to compute a coverability set of Petri nets

The two previous sections have demonstrated that it is a difficult task to devise an algorithm to compute the (minimal) coverability set of a Petri net by keeping $\preccurlyeq$–maximal elements only during the computation. Whether such an algorithm exists remains for us an open question.

Nevertheless, we present, in this section, a novel algorithm to compute the coverability set of PN. That algorithm is, in practice, more efficient than the `Karp&Miller` procedure.

We consider the problem from scratch, and define a simple procedure to compute the coverability set of a Petri net. Unlike the `Karp&Miller` procedure that builds a tree, this procedure builds sets of *pairs of markings*. This allows to keep the relationships between the markings that have been computed. These relationships are important in order to compute *accelerations* (similarly to the `Karp&Miller` procedure).

An acceleration can occur when we discover two markings $\mathbf{m}_1$ and $\mathbf{m}_2$ that enforce the two following conditions:

1. A sequence σ s.t. $\mathbf{m}_1 \xrightarrow{\sigma} \mathbf{m}_2$ has to exist. This will be guaranteed by our algorithm because, for every pair $(\mathbf{m}_1, \mathbf{m}_2)$ built by it, we have the guarantee that $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$.

2. We must have $\mathbf{m}_1 \prec \mathbf{m}_2$, which means that there is at least one place p s.t. the difference (in a precise sense that we make clear in the sequel) between $\mathbf{m}_1(p)$ and $\mathbf{m}_2(p)$ has to be strictly positive. That point has to be taken into account when pruning the sets of pairs during the computation, as we will see now.

The sets of pairs of markings are actually represented by means of their *maximal elements* wrt to an ordering *on pairs* $\sqsubseteq$ (and not an ordering on *individual markings*, as it is the case for the MCT algorithm). This point is the key optimisation of our algorithm, because sets of maximal elements are typically smaller that the sets they represent (see for instance Section 5.3).

The reason why we use an ordering relating pairs (rather than markings) is that we can take into account the difference between the two coordinates of the pairs, and ensure that *larger pairs* will produce *larger accelerations*. That is, $(\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2)$ will be regarded as *larger* than $(\mathbf{m}_1, \mathbf{m}_2)$ iff $\mathbf{m}_1 \preccurlyeq \overline{\mathbf{m}}_1$, $\mathbf{m}_2 \preccurlyeq \overline{\mathbf{m}}_2$ *and* for any place p, the *difference* between $\overline{\mathbf{m}}_2(p)$ and $\overline{\mathbf{m}}_1(p)$ is larger than or equal to the difference between $\mathbf{m}_1(p)$ and $\mathbf{m}_2(p)$.

If these conditions are enforced, it is safe to remove the pair $(\mathbf{m}_1, \mathbf{m}_2)$ from the set of pairs that have to be considered, because:

- by monotonicity, all the potential successors of $\mathbf{m}_1$ and $\mathbf{m}_2$ will be covered by some successors of $\overline{\mathbf{m}}_1$ and $\overline{\mathbf{m}}_2$ and
- any acceleration that can be created by the pair $(\mathbf{m}_1, \mathbf{m}_2)$ will be covered by an acceleration created by $(\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2)$.

These are, intuitively, the arguments and results that we introduce in this section. It is organised as follows. In section 6.3.1, we introduce a *difference operator* $\ominus$ on markings, and use it to define the $\sqsubseteq$ ordering on pairs of markings (its definition follows the intuition that we have sketched above). Then, we define for any PN $\mathcal{N}$ the *covering sequence* (of $\mathcal{N}$), a sequence of sets of pairs of markings, that are maximal wrt to $\sqsubseteq$. Then, we show that the covering sequence of $\mathcal{N}$ eventually converges to (a representation of) a coverability set of $\mathcal{N}$. For that purpose, we first provide the reader with several auxiliary lemmata in Section 6.3.2. We use them to prove that the covering sequence is both *sound* and *complete* wrt to the coverability set of $\mathcal{N}$, respectively in Section 6.3.3 and Section 6.3.4. This allows us to conclude, in Section 6.3.5, that the covering sequence eventually stabilises and allows to obtain a coverability set of $\mathcal{N}$. Finally, in section 6.3.6, we provide empirical results that prove the practical efficiency of this method, by comparing it to the `Karp&Miller` procedure and to the construction of the coverability graph.

6.3.1 The covering sequence

In this section we define a sequence of sets of pairs of markings that is computable and from which we can obtain a coverability set of any Petri net.

Auxiliary functions In order to define the covering sequence, we rely on the functions Flatten, AccelPair and TClosure defined as follows:

- For any $R \subseteq (\mathbb{N} \cup \{\omega\})^{|P|} \times (\mathbb{N} \cup \{\omega\})^{|P|}$:
$$\mathsf{Flatten}\,(R) = \{\mathbf{m} | \exists \mathbf{m}' : (\mathbf{m}, \mathbf{m}') \in R \text{ or } (\mathbf{m}', \mathbf{m}) \in R\}$$
- For any $\mathbf{m}_1, \mathbf{m}_2 \in (\mathbb{N} \cup \{\omega\})^{|P|}$ s.t. $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$, $\mathsf{AccelPair}\,(\mathbf{m}_1, \mathbf{m}_2) \in (\mathbb{N} \cup \{\omega\})^{|P|}$ is s.t.:
$$\forall p \in P : \mathsf{AccelPair}\,(\mathbf{m}_1, \mathbf{m}_2)\,(p) = \begin{cases} \mathbf{m}_1(p) & \text{if } \mathbf{m}_1(p) = \mathbf{m}_2(p) \\ \omega & \text{otherwise} \end{cases}$$
Remark that $\mathsf{AccelPair}\,(\mathbf{m}_1, \mathbf{m}_2) = \mathbf{m}_1$ iff $\mathbf{m}_1 = \mathbf{m}_2$;

- For any $R \subseteq (\mathbb{N} \cup \{\omega\})^{|P|} \times (\mathbb{N} \cup \{\omega\})^{|P|}$:

$$\begin{aligned}\mathsf{TClosure}(R) &= \big\{(\mathbf{m}, \mathbf{m}') \mid \exists \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k : \mathbf{m} = \mathbf{m}_1, \\ &\quad \mathbf{m}_k = \mathbf{m}' \wedge \forall 1 \le j < k : (\mathbf{m}_j, \mathbf{m}_{j+1}) \in R\big\}\end{aligned}$$

The ordering $\sqsubseteq$ In order to define $\sqsubseteq$, we first define $\ominus$, a difference operator on ω-markings.

Definition 6.1 (The $\ominus$ Operator) *Let $\mathbf{m}$ and $\mathbf{m}'$ be two ω-markings. Then, $\mathbf{m} \ominus \mathbf{m}'$ is a function $P \mapsto \mathbb{Z} \cup \{-\omega, \omega\}$ s.t.:*

$$\forall p \in P : (\mathbf{m} \ominus \mathbf{m}')(p) = \begin{cases} \omega & \text{if } \mathbf{m}(p) = \omega \\ -\omega & \text{if } \mathbf{m}'(p) = \omega \text{ and } \mathbf{m}(p) \neq \omega \\ \mathbf{m}(p) - \mathbf{m}'(p) & \text{otherwise} \end{cases}$$

■

Remark that, by this definition, for any pair of markings $\mathbf{m}$ and $\mathbf{m}'$, for any place p, we have $(\mathbf{m} \ominus \mathbf{m}')(p) = \omega$ if and only if $\mathbf{m}(p) = \omega$. Remark also that $m \ominus m'$ is in general not an ω-marking since $(m \ominus m')(p)$ can be negative for some $p \in P$. We can now define $\sqsubseteq$. In this definition, we extend the $\le$ partial ordering on $\mathbb{Z}$ to $\mathbb{Z} \cup \{-\omega, \omega\}$ as follows: for any $e \in \mathbb{Z}$: $-\omega < e < \omega$, $e \not\le -\omega$ and $\omega \not\le e$ (remark that the resulting ordering is still antisymmetric).

Definition 6.2 (The $\sqsubseteq$ Order) *Let $(\mathbf{m}_1, \mathbf{m}_2)$ and $(\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2)$ be two pairs of markings. Then $(\mathbf{m}_1, \mathbf{m}_2) \sqsubseteq (\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2)$ iff $\mathbf{m}_1 \preccurlyeq \overline{\mathbf{m}}_1$, $\mathbf{m}_2 \preccurlyeq \overline{\mathbf{m}}_2$ and for any place p: $(\mathbf{m}_2 \ominus \mathbf{m}_1)(p) \le (\overline{\mathbf{m}}_2 \ominus \overline{\mathbf{m}}_1)(p)$.* ■

It is easy to show that:

Proposition 6.1 *$\sqsubseteq$ is a partial order.*

Proof. The proposition stems from the fact that both $\preccurlyeq$ and $\le$ are partial orders. □

This implies in particular that $\sqsubseteq$ is reflexive and transitive. These properties will be important in some of the forthcoming proofs.

The covering sequence Let us now introduce the *covering sequence*, our new tool to compute the coverability set of PN:

Definition 6.3 (The Covering Sequence) *Given a PN $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, the covering sequence $\mathsf{CovSeq}(\mathcal{N})$ is the sequence of sets of pairs of markings $T_0, T_1, \ldots, T_i, \ldots$ defined inductively as follows:*

- $T_0 = \mathsf{Max}^{\sqsubseteq}\left(\{(\mathbf{m}_0, \mathbf{m}) \mid \mathbf{m} \in \mathsf{Post}\left(\mathbf{m}_0\right) \cup \{\mathbf{m}_0\}\}\right)$.
- *For any* $i \geq 1$*:* $A_i = \{(\mathbf{m}, \mathbf{m}') \mid \exists (\overline{\mathbf{m}}, \mathbf{m}) \in T_{i-1} : \overline{\mathbf{m}} \prec \mathbf{m} \wedge \mathbf{m}' = \mathsf{AccelPair}\left(\overline{\mathbf{m}}, \mathbf{m}\right)\}$.
- *For any* $i \geq 1$*:* $S_i = \{(\mathbf{m}, \mathbf{m}') \mid \mathbf{m}' \in \mathsf{Post}\left(\mathbf{m}\right) \wedge \mathbf{m} \in \mathsf{Flatten}\left(T_{i-1}\right)\}$.
- *For any* $i \geq 1$*:* $T_i = \mathsf{Max}^{\sqsubseteq}\left(\mathsf{TClosure}\left(S_i \cup A_i \cup T_{i-1}\right)\right)$. ■

The intuition behind this definition is as follows. The set T_0 contains the $\sqsubseteq$-maximal elements from the set of all the pairs of the form $(\mathbf{m}_0, \mathbf{m})$ s.t. $\mathbf{m}$ is a direct successor of $\mathbf{m}_0$ (or $\mathbf{m}_0$, in order to ensure that $T_0 \neq \emptyset$ even when no transitions are firable from $\mathbf{m}_0$). For any $i \geq 1$, A_i consists in computing all the possible accelerations based on the pairs that have been computed so far. S_i consists in computing all the pairs of the form $(\mathbf{m}_1, \mathbf{m}_2)$ s.t. $\mathbf{m}_1$ has been computed at a former step, and $\mathbf{m}_2$ is in $\mathsf{Post}\left(\mathbf{m}_1\right)$. The set T_i is obtained by taking the *transitive closure* of the union of A_i, S_i and T_{i-1}, which is the result of the previous steps of computations. The transitive closure allows to ensure that no acceleration is missed when computing A_{i+1}. Finally, we keep in T_i only the maximal elements with respect to $\sqsubseteq$.

Remark that, given the set T_i, one can compute in a finite amount of time the sets A_{i+1}, S_{i+1} and T_{i+1}. Since, T_0 is computable in a finite amount of time too, any finite prefix of any covering sequence $\mathsf{CovSeq}\left(\mathcal{N}\right)$ is computable. We will show in the sequel that for any Petri net $\mathcal{N}$ with $\mathsf{CovSeq}\left(\mathcal{N}\right) = T_0, T_1, \ldots, T_i, \ldots$, there exists a finite value k s.t.

$$\gamma\left(\mathsf{Flatten}\left(T_k\right)\right) = \gamma\left(\mathsf{Flatten}\left(T_{k-1}\right)\right) = \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right) = \mathsf{Cover}\left(\mathcal{N}\right)$$

6.3.2 Auxiliary lemmata

Let us introduce two properties of the covering sequence that will be helpful to prove its correctness. The first property relies on two auxiliary lemmata. The former lemma is a characterisation of the function $\mathsf{AccelPair}$:

Lemma 6.1 *Let* $\mathcal{N}$ *be a* PN *and let* $\mathbf{m}_1$ *and* $\mathbf{m}_2$ *be two* ω*-markings of* $\mathcal{N}$ *that respect* $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$ *and* $\gamma\left(\mathbf{m}_2\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_1\right)\right)$*. Then,*

$$\gamma\left(\mathsf{AccelPair}\left(\mathbf{m}_1, \mathbf{m}_2\right)\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_2\right)\right)$$

Proof. Let P' be the set of places $\{p \mid \mathbf{m}_1(p) < \mathbf{m}_2(p)\}$. Remark that, since $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$, $\mathbf{m}_1(p) = \mathbf{m}_2(p)$ for any $p \notin P'$. Since $\gamma\left(\mathbf{m}_2\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_1\right)\right)$, and $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$ there must exist a sequence of transitions σ that is firable from $\mathbf{m}_1$ and allows to increase the number of tokens in the places of P'. That is, there exists σ and a marking $\overline{\mathbf{m}}$ s.t.

(i) $\mathbf{m}_1 \xrightarrow{\sigma} \overline{\mathbf{m}}$, (ii) $\mathbf{m}_1 \preccurlyeq \overline{\mathbf{m}}$ and (iii) for any place $p \in P'$, $\overline{\mathbf{m}}(p) > \mathbf{m}_1(p)$. Indeed, let $\mathbf{m}'$ be defined as follows:

$$\forall p \in P : \mathbf{m}' = \begin{cases} 0 & \text{if } p \notin P' \text{ and } \mathbf{m}_1(p) = \omega \\ \mathbf{m}_1(p) & \text{if } p \notin P' \text{ and } \mathbf{m}_1(p) \neq \omega \\ \mathbf{m}_1(p) + 1 & \text{if } p \in P' \end{cases}$$

By Definition, we have $\mathbf{m}' \in \gamma(\mathbf{m}_2)$ but $\mathbf{m}' \notin \gamma(\mathbf{m}_1)$ (remark in particular that $p \in P'$ implies that $\mathbf{m}_1(p) \neq \omega$ and $\mathbf{m}_2(p) \geq \mathbf{m}_1(p) + 1$). Since $\mathbf{m}' \in \gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$, there exists a marking $\overline{\mathbf{m}}$ and a sequence of transitions σ s.t. $\mathbf{m}_1 \xrightarrow{\sigma} \overline{\mathbf{m}}$ and $\mathbf{m}' \preccurlyeq \overline{\mathbf{m}}$. Hence, $\mathbf{m}'(p) \leq \overline{\mathbf{m}}(p)$ for every place p. We consider three cases. (i) when $\mathbf{m}_1(p) = \omega$, we have necessarily $\overline{\mathbf{m}}(p) = \omega$. Hence, $\mathbf{m}_1(p) \leq \overline{\mathbf{m}}(p)$ for every place p s.t. $\mathbf{m}_1(p) = \omega$. (ii) when $\mathbf{m}_1(p) \neq \omega$ and $p \notin P'$, we have $\mathbf{m}'(p) = \mathbf{m}_1(p)$, by definition of $\mathbf{m}'$. Hence $\mathbf{m}_1(p) \leq \overline{\mathbf{m}}(p)$. (iii) when $p \in P'$ (hence $\mathbf{m}_1(p) \neq \omega$), we have $\mathbf{m}_1(p) < \mathbf{m}'(p)$, by definition of $\mathbf{m}'$ again. Hence $\mathbf{m}_1(p) < \overline{\mathbf{m}}(p)$. We conclude that $\mathbf{m}_1 \preccurlyeq \overline{\mathbf{m}}$ and that $\mathbf{m}_1(p) < \overline{\mathbf{m}}(p)$ for every $p \in P'$.

Let $\overline{\mathbf{m}}_i$ $(i \geq 1)$ be the marking s.t. $\mathbf{m}_1 \xrightarrow{\sigma^i} \overline{\mathbf{m}}_i$, i.e. the marking obtained after having fired i times σ from $\mathbf{m}_1$. Thus, since PN transitions have constant effect,

$$\forall i \geq 1 : \forall p : \overline{\mathbf{m}}_i(p) = \mathbf{m}_1(p) + i \cdot (\overline{\mathbf{m}}(p) - \mathbf{m}_1(p)) \tag{6.1}$$

Remark that, for any $i \geq 1 : \overline{\mathbf{m}}_i \in \mathsf{Post}^*(\mathbf{m}_1)$, and that, by monotonicity, $\forall i \geq 1 : \overline{\mathbf{m}}_i \preccurlyeq \overline{\mathbf{m}}_{i+1}$.

In the case where $p \in P'$, the value $\overline{\mathbf{m}}(p) - \mathbf{m}_1(p)$ is > 0. Hence, by (6.1), we have:

$$\forall p \in P' : \forall n \in \mathbb{N} : \exists k : \overline{\mathbf{m}}_k(p) > n \tag{6.2}$$

On the other hand, by definition of the acceleration function, and since $\overline{\mathbf{m}}_i \succcurlyeq \mathbf{m}_1$ for any $i \geq 1$:

$$\forall p \notin P' : \forall i \geq 1 : \overline{\mathbf{m}}_i(p) \geq \mathbf{m}_1(p) = \mathbf{m}_2(p) = \mathsf{AccelPair}(\mathbf{m}_1, \mathbf{m}_2)(p) \tag{6.3}$$

Let $\mathbf{m}$ be in $\gamma(\mathsf{AccelPair}(\mathbf{m}_1, \mathbf{m}_2))$. Thus, $\mathbf{m} \preccurlyeq \mathsf{AccelPair}(\mathbf{m}_1, \mathbf{m}_2)$ and for any place p: $\mathbf{m}(p) \neq \omega$. Hence, by (6.3), for any $p \notin P'$, for any $i \geq 1$, $\mathbf{m}(p) \leq \overline{\mathbf{m}}_i(p)$. Moreover, by (6.2), there exists, for any $p \in P'$, a value $k(p)$ s.t. $\overline{\mathbf{m}}_{k(p)}(p) > \mathbf{m}(p)$. Since the sequence $\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2, \ldots$ is $\preccurlyeq$-increasing, the marking $\overline{\mathbf{m}}_k$, with $k = \max\{k(p) \mid p \in P'\}$ is s.t. for any $p \in P' : \overline{\mathbf{m}}_k(p) \geq \mathbf{m}(p)$. We conclude that there exists $k \geq 1$ with $\overline{\mathbf{m}}_k \succcurlyeq \mathbf{m}$. Since $\overline{\mathbf{m}}_k \in \mathsf{Post}^*(\mathbf{m}_1)$, and since $\mathbf{m}_2 \succcurlyeq \mathbf{m}_1$, there exists, by monotonicity, a marking $\mathbf{m}'$ s.t. $\mathbf{m}' \in \mathsf{Post}^*(\mathbf{m}_2)$ and $\mathbf{m}' \succcurlyeq \overline{\mathbf{m}}_k \succcurlyeq \mathbf{m}$. Since this is true for any $\mathbf{m} \in \gamma(\mathsf{AccelPair}(\mathbf{m}_1, \mathbf{m}_2))$, we conclude that: $\gamma(\mathsf{AccelPair}(\mathbf{m}_1, \mathbf{m}_2)) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_2))$. □

The latter lemma is a direct consequence of the monotonicity of Petri nets:

Lemma 6.2 *Let $\mathcal{N}$ be a Petri net and let A and B be two sets of ω-markings of $\mathcal{N}$. Then, $\gamma(A) \subseteq \gamma(\mathsf{Post}^*(B))$ implies that $\gamma(\mathsf{Post}^*(A)) \subseteq \gamma(\mathsf{Post}^*(B))$.*

Proof. By monotonicity of Post, and since Post is well-defined on ω-markings (see Proposition 3.2) we have:

$$\forall X \subseteq \left(\mathbb{N} \cup \{\omega\}\right)^{|P|} : \gamma\left(\mathsf{Post}^*\left(X\right)\right) = \gamma\left(\mathsf{Post}^*\left(\gamma\left(X\right)\right)\right) \tag{6.4}$$

Thus:

$$\begin{array}{rrcll}
 & \gamma(A) & \subseteq & \gamma(\mathsf{Post}^*(B)) & \\
\Rightarrow & \gamma(\mathsf{Post}^*(\gamma(A))) & \subseteq & \gamma(\mathsf{Post}^*(\gamma(\mathsf{Post}^*(B)))) & \text{Monotonicity of } \mathsf{Post} \text{ and } \gamma \\
\Rightarrow & \gamma(\mathsf{Post}^*(A)) & \subseteq & \gamma(\mathsf{Post}^*(\mathsf{Post}^*(B))) & \text{By (6.4)} \\
\Rightarrow & \gamma(\mathsf{Post}^*(A)) & \subseteq & \gamma(\mathsf{Post}^*(B)) &
\end{array}$$

□

We can now prove an invariant of the covering sequence. It states that, whenever a pair $(\mathbf{m}_1, \mathbf{m}_2)$ is computed, the denotation of $\mathbf{m}_2$ is included into the denotation of $\mathsf{Post}^*(\mathbf{m}_1)$.

Lemma 6.3 *For any* PN $\mathcal{N}$ *with* $\mathsf{CovSeq}(\mathcal{N}) = T_0, T_1, \ldots, T_j, \ldots$, *for any* $i \geq 1$, *for any* $(\mathbf{m}_1, \mathbf{m}_2) \in T_i$*:* $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$.

Proof. The proof is by induction on i.

Base case $i = 1$. For any $(\mathbf{m}_1, \mathbf{m}_2) \in T_0$, $\{\mathbf{m}_2\} \subseteq \mathsf{Post}^*(\mathbf{m}_1)$. Hence, $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$, by $\subseteq$-monotony of γ.

Inductive case: $i = k + 1$. By induction hypothesis, for any $(\mathbf{m}_1, \mathbf{m}_2) \in T_k$: $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$. Let us show that the same holds for any $(\mathbf{m}_1, \mathbf{m}_2)$ in T_{k+1}. Let us first show that, for any pair $(\mathbf{m}_1, \mathbf{m}_2) \in S_{k+1} \cup A_{k+1}$, $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$:

1. If $(\mathbf{m}_1, \mathbf{m}_2) \in A_{k+1}$, then, there exists $\mathbf{m}_3$ s.t. $(\mathbf{m}_3, \mathbf{m}_1) \in T_k$, $\mathbf{m}_3 \prec \mathbf{m}_1$ and $\mathbf{m}_2 = \mathsf{AccelPair}(\mathbf{m}_3, \mathbf{m}_1)$. By induction hypothesis, $\gamma(\mathbf{m}_1) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_3))$. By Lemma 6.1, this implies that $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$.
2. If $(\mathbf{m}_1, \mathbf{m}_2) \in S_{k+1}$, then $\{\mathbf{m}_2\} \subseteq \mathsf{Post}^*(\mathbf{m}_1)$. Thus, $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$ by monotonicity of γ.

By combining these two points with the induction hypothesis, we conclude that any pair of markings in $T_k \cup S_{k+1} \cup A_{k+1}$ respects the lemma. Let $(\mathbf{m}_1, \mathbf{m}_2)$ be a pair of $\mathsf{TClosure}(T_k \cup S_{k+1} \cup A_{k+1})$. This implies that there are $\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2, \ldots, \overline{\mathbf{m}}_n$ s.t. the set

$$S = \{(\mathbf{m}_1, \overline{\mathbf{m}}_1), (\overline{\mathbf{m}}_n, \mathbf{m}_2)\} \cup \{(\overline{\mathbf{m}}_i, \overline{\mathbf{m}}_{i+1} \mid 1 \leq i \leq n-1\}$$

is included in $(T_k \cup S_{k+1} \cup A_{k+1})$, and any pair $(\mathbf{m}, \mathbf{m}') \in S$ respects $\gamma(\mathbf{m}') \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}))$.

Observe now that the following holds for any pairs $(\mathbf{m}, \mathbf{m}')$ and $(\mathbf{m}', \mathbf{m}'')$:

$$\begin{array}{c}\gamma(\mathbf{m}') \subseteq \gamma(\mathsf{Post}^*(\mathbf{m})) \text{ and } \gamma(\mathbf{m}'') \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}')) \\ \text{implies } \gamma(\mathbf{m}'') \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}))\end{array} \tag{6.5}$$

Indeed, by Lemma 6.2, $\gamma(\mathbf{m}') \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}))$ implies that $\gamma(\mathsf{Post}^*(\mathbf{m}')) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}))$. By transitivity of $\subseteq$, we obtain: $\gamma(\mathbf{m}'') \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}))$.

By applying (6.5) to $(\mathbf{m}_1, \overline{\mathbf{m}}_1)$ and $(\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2)$, we obtain $\gamma(\overline{\mathbf{m}}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$. We can iterate this reasoning and finally obtain $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$.

We conclude that, for any $(\mathbf{m}_1, \mathbf{m}_2)$ in $\mathsf{TClosure}(T_k \cup S_{k+1} \cup A_{k+1})$: $\gamma(\mathbf{m}_2) \subseteq \gamma(\mathsf{Post}^*(\mathbf{m}_1))$. Since $T_{k+1} \subseteq \mathsf{TClosure}(T_k \cup S_{k+1} \cup A_{k+1})$, the same holds for any pair $(\mathbf{m}_1, \mathbf{m}_2) \in T_{k+1}$. Hence the lemma.

□

We close the section by a second property of the covering sequence, stating that the sequence $\gamma(\mathsf{Flatten}(T_i))$ is increasing with respect to set inclusion.

Lemma 6.4 *Let $\mathcal{N}$ be a* PN *s.t.* $\mathsf{CovSeq}(\mathcal{N}) = T_0, T_1, \ldots, T_i, \ldots$ *Then, for any $i \geq 0$:*

$$\gamma(\mathsf{Flatten}(T_i)) \subseteq \gamma(\mathsf{Flatten}(T_{i+1}))$$

Proof. Clearly, we have $T_i \subseteq S_{i+1} \cup A_{i+1} \cup T_i \subseteq \mathsf{TClosure}(S_{i+1} \cup A_{i+1} \cup T_i)$. Hence, $\mathsf{Flatten}(T_i) \subseteq \mathsf{Flatten}(\mathsf{TClosure}(S_{i+1} \cup A_{i+1} \cup T_i))$. However, for any marking $\mathbf{m}$ in $\mathsf{Flatten}(\mathsf{TClosure}(S_{i+1} \cup A_{i+1} \cup T_i))$, there is

$$\overline{\mathbf{m}} \in \mathsf{Flatten}\left(\mathsf{Max}^{\sqsubseteq}(\mathsf{TClosure}(S_{i+1} \cup A_{i+1} \cup T_i))\right) = \mathsf{Flatten}(T_{i+1})$$

such that $\mathbf{m} \preccurlyeq \overline{\mathbf{m}}$, by definition of $\mathsf{Max}^{\sqsubseteq}$. We conclude that, for any $\mathbf{m} \in \mathsf{Flatten}(T_i)$, there is $\overline{\mathbf{m}} \in \mathsf{Flatten}(T_{i+1})$ s.t. $\mathbf{m} \preccurlyeq \overline{\mathbf{m}}$. Hence, $\gamma(\mathsf{Flatten}(T_i)) \subseteq \gamma(\mathsf{Flatten}(T_{i+1}))$. □

We have now at our disposal all the necessary results to prove that the covering sequence effectively computes the coverability set of any Petri net. The actual arguments of correctness are presented along the following two sections that are devoted respectively to proving that the covering sequence is both *sound* (any marking computed by the covering sequence is covered by some marking of the coverability set) and *complete* (any reachable marking of the Petri net will eventually be covered by a marking computed in the covering sequence).

6.3.3 Soundness of the covering sequence

The soundness of the covering sequence is established by showing that, for any $i \geq 1$, for any $\mathbf{m} \in \mathsf{Flatten}(T_i)$, the denotation of $\mathbf{m}$ is included in the denotation of $\mathsf{Post}^*(\mathbf{m}_0)$:

Lemma 6.5 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ be a* PN *s.t.* $\mathsf{CovSeq}\left(\mathcal{N}\right) = T_0, T_1, \ldots, T_i, \ldots$ *Then, for any $i \geq 1$, for any $\mathbf{m} \in \mathsf{Flatten}\left(T_i\right)$, $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$.*

Proof. The proof is by induction on i.

Base case $i = 0$. Trivial.

Inductive case $i = k + 1$. Let us first show that for any $\mathbf{m}$ in $\mathsf{Flatten}\left(S_i \cup A_i\right)$: $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$. We consider the two sets S_i and A_i separately:

1. In the case where $\mathbf{m} \in \mathsf{Flatten}\left(S_i\right)$, either there is a pair $(\mathbf{m}, \mathbf{m}') \in S_i$, or there is a pair $(\mathbf{m}', \mathbf{m}) \in S_i$. In the first case, $\mathbf{m} \in \mathsf{Flatten}\left(T_{i-1}\right)$, by construction. Hence, $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$ by induction hypothesis. In the latter case, $\mathbf{m}' \in \mathsf{Flatten}\left(T_{i-1}\right)$, thus, $\gamma\left(\mathbf{m}'\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$ by induction hypothesis. By Lemma 6.2, this implies that $\gamma\left(\mathsf{Post}^*\left(\mathbf{m}'\right)\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$. Moreover, $\mathbf{m} \in \mathsf{Post}\left(\mathbf{m}'\right)$, by construction. Hence, $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}'\right)\right)$. We conclude that $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$.

2. In the case where $\mathbf{m} \in \mathsf{Flatten}\left(A_i\right)$, there is either a pair $(\mathbf{m}, \mathbf{m}') \in A_i$, or a pair $(\mathbf{m}', \mathbf{m}) \in A_i$. In the first case, $\mathbf{m} \in \mathsf{Flatten}\left(T_{i-1}\right)$, by construction. Hence, $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$ by induction hypothesis. In the latter case, there exists a pair $(\overline{\mathbf{m}}, \mathbf{m}') \in T_{i-1}$ s.t. $\overline{\mathbf{m}} \prec \mathbf{m}'$ and $\mathbf{m} = \mathsf{AccelPair}\left(\overline{\mathbf{m}}, \mathbf{m}'\right)$, by construction. Thus, $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}'\right)\right)$, by Lemma 6.1. By induction hypothesis, and since $\mathbf{m}' \in \mathsf{Flatten}\left(T_{i-1}\right)$, $\gamma\left(\mathbf{m}'\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$. Thus, $\gamma\left(\mathsf{Post}^*\left(\mathbf{m}'\right)\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$, according to Lemma 6.2. We conclude that $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$.

Since T_{i-1} respects the lemma, for any $\mathbf{m} \in \mathsf{Flatten}\left(S_i \cup A_i \cup T_{i-1}\right)$, we have $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$. Moreover, for any set S of pairs, $\mathsf{Flatten}\left(\mathsf{TClosure}\left(S\right)\right) = \mathsf{Flatten}\left(S\right)$. Hence, for any $\mathbf{m} \in \mathsf{Flatten}\left(\mathsf{TClosure}\left(S_i \cup A_i \cup T_{i-1}\right)\right)$: $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$. Finally, since $T_i \subseteq \mathsf{TClosure}\left(S_i \cup A_i \cup T_{i-1}\right)$, we conclude that for any $\mathbf{m} \in \mathsf{Flatten}\left(T_i\right)$: $\gamma\left(\mathbf{m}\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$. □

As a consequence, we directly obtain our soundness result:

Corollary 6.1 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ be a* PN *and let its covering sequence* $\mathsf{CovSeq}\left(\mathcal{N}\right)$ *be $T_0, T_1, \ldots, T_i, \ldots$ Then, for any $i \geq 1$, $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$.*

6.3.4 Completeness of the covering sequence

In order to show that the covering sequence is *complete*, we show that it simulates, in some sense, the Karp&Miller algorithm. More precisely, for any node n of the Karp&Miller tree of a given PN $\mathcal{N}$, we show that there exists $k \geq 1$ and $\mathbf{m} \in \mathsf{Flatten}\left(T_k\right)$ s.t. $\Lambda\left(n\right) \preccurlyeq \mathbf{m}$. The actual value of k depends on the depth of n in the Karp&Miller tree, and can be bounded. Remark that since the Karp&Miller tree is guaranteed to be finite, and thanks to Lemma 6.5 this results also provides us with the key argument to prove that our covering sequence will eventually converge to the coverability set.

In order to draw the link between the covering sequence and the Karp&Miller algorithm, we first state the following property that characterises (with respect to $\mathsf{AccelPair}$) the sequence $\varsigma(n)$ and the marking $\mathsf{M}(n)$ that are associated to any node n of the Karp&Miller tree (see Section 3.2.4 and particularly Definition 3.10).

Lemma 6.6 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ be a Petri net and let $\mathcal{T} = \langle N, B, root, \Lambda \rangle$ be its Karp&Miller tree. Let $n \neq root$ be a node of $\mathcal{T}$. Let $\mathbf{m}'$ be s.t. $\mathsf{M}(n) \xrightarrow{\varsigma(n)} \mathbf{m}'$. Then, $\Lambda(n) \preccurlyeq \mathsf{AccelPair}(\mathsf{M}(n), \mathbf{m}')$.*

Proof. Let $P_a = \{p \in P \mid \Lambda(n)(p) = \omega \wedge \mathsf{M}(n)(p) \neq \omega\}$. By construction, we have:

$$\Lambda(n)(p) = \begin{cases} \omega & \text{if } p \in P_a \\ \mathsf{M}(n)(p) & \text{otherwise} \end{cases}$$

Moreover, by definition of $\mathsf{AccelPair}$, we have:

$$\mathsf{AccelPair}(\mathsf{M}(n), \mathbf{m}')(p) = \begin{cases} \omega & \text{if } \varsigma(n)(p) > 0 \\ \mathsf{M}(n)(p) & \text{otherwise} \end{cases}$$

However, by definition of $\varsigma(n)$, $p \in P_a$ implies that $\varsigma(n)(p) > 0$. Hence the lemma. □

We are now ready to show that our covering sequence simulates the Karp&Miller algorithm. In this lemma, we use the following notation, defined for any node n of any labelled tree $\mathcal{T} = \langle N, B, root, \Lambda \rangle$:

$$\forall n \in N : \mathsf{Ancestors}(\mathcal{T}, n) = \{n' \mid B^*(n', n)\}$$

Hence, for any node n, $\mathsf{Ancestors}(\mathcal{T}, n)$ is the set of 'ancestors' of n in $\mathcal{T}$ (n included). Thus, $\mathsf{Ancestors}(\mathcal{T}, n) \neq \emptyset$ for any n. The Lemma is as follows:

Lemma 6.7 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ be a Petri net and let $\mathcal{T} = \langle N, B, root, \Lambda \rangle$ be its Karp&Miller tree. Let n be a node of N. Then, there is $\widehat{\mathbf{m}}$ in $\mathsf{Flatten}(T_k)$ s.t. $\Lambda(n) \preccurlyeq \widehat{\mathbf{m}}$ for $k \leq \sum_{n' \in \mathsf{Ancestors}(\mathcal{T}, n)} (|\varsigma(n')| + 2)$.*

Proof. The proof is by induction on the length ℓ of the branch ending in n.

Base case $\ell = 0$. In that case, $n = root$ and $\Lambda(root) = \mathbf{m}_0$. By construction, there is in $\mathsf{Flatten}(T_0)$ a marking $\widehat{\mathbf{m}}$ s.t. $\mathbf{m}_0 \preccurlyeq \widehat{\mathbf{m}}$. Moreover $0 \leq \sum_{n' \in \mathsf{Ancestors}(\mathcal{T}, root)} (|\varsigma(n')| + 2) = |\varsigma(root)| + 2 = 2$.

Inductive case $\ell = i + 1$. Let $n_1, n_2, \ldots, n_i, n_{i+1}$ be a branch of $\mathcal{T}$ (hence, $n_1 = root$) of length ℓ. By induction hypothesis, there exists $k \leq \sum_{j=1}^{i} (|\varsigma(n_j)| + 2)$ s.t. $\gamma(\Lambda(n_i)) \subseteq \gamma(\mathsf{Flatten}(T_k))$. Let us us consider the sequence of transitions $\varsigma(n_{i+1})$. We consider two cases:

1. In the case where $\varsigma(n_{i+1})$ is the empty sequence, there exists a transition t s.t. $\Lambda(n_i) \xrightarrow{t} \Lambda(n_{i+1})$. By induction hypothesis, there exists $\overline{\mathbf{m}}_i \in \mathsf{Flatten}(T_k)$ s.t. $\overline{\mathbf{m}}_i \succcurlyeq \Lambda(n_i)$. Hence t is firable from $\overline{\mathbf{m}}_i$ and $\overline{\mathbf{m}}_i \xrightarrow{t} \overline{\mathbf{m}}$ implies that $\Lambda(n_{i+1}) \preccurlyeq \overline{\mathbf{m}}$. By construction, $(\overline{\mathbf{m}}_i, \overline{\mathbf{m}}) \in S_{k+1}$, and thus, there is $(\widehat{\mathbf{m}}_i, \widehat{\mathbf{m}}) \in T_{k+1}$ s.t. $(\overline{\mathbf{m}}_i, \overline{\mathbf{m}}) \sqsubseteq (\widehat{\mathbf{m}}_i, \widehat{\mathbf{m}})$. Hence,

$$\Lambda(n_{i+1}) \preccurlyeq \overline{\mathbf{m}} \preccurlyeq \widehat{\mathbf{m}} \in \mathsf{Flatten}(T_{k+1})$$

with $k+1 \leq k+2 \leq \sum_{j=1}^{i}(|\varsigma(n_j)|+2)+2 = \sum_{j=1}^{i+1}(|\varsigma(n_j)|+2)$.

2. In the case where $\varsigma(n_{i+1})$ is not empty, then let $\mathbf{m}'$ be s.t. $\mathsf{M}(n_{i+1}) \xrightarrow{\varsigma(n_{i+1})} \mathbf{m}'$. By Lemma 6.6, we have $\mathsf{AccelPair}(\mathsf{M}(n_{i+1}), \mathbf{m}') \succcurlyeq \Lambda(n_{i+1})$. Let us show, by induction on the length of $\varsigma(n_{i+1})$, that there exists in $T_{k+|\varsigma(n_{i+1})|+1}$ a pair $(\overline{\mathbf{m}}, \overline{\mathbf{m}}')$ s.t. $(\mathsf{M}(n_{i+1}), \mathbf{m}') \sqsubseteq (\overline{\mathbf{m}}, \overline{\mathbf{m}}')$.

   First remark that there is, in $\mathsf{Flatten}(T_{k+1})$, a marking $\widehat{\mathbf{m}}$ s.t. $\mathsf{M}(n_{i+1}) \preccurlyeq \widehat{\mathbf{m}}$, because, by definition of $\mathsf{M}(n_{i+1})$, there exists a transition t s.t. $\Lambda(n_i) \xrightarrow{t} \mathsf{M}(n_{i+1})$. Hence, we can invoke the arguments used in point 1 of the present proof. Thus, $\varsigma(n_{i+1})$ is firable from $\widehat{\mathbf{m}}$.

   Base case $|\varsigma(n_{i+1})| = 1$. In this case, $\varsigma(n_{i+1}) = t \in T$. By construction, and since t is firable from $\widehat{\mathbf{m}}$, the pair $(\widehat{\mathbf{m}}, \widehat{\mathbf{m}}')$ with $\widehat{\mathbf{m}} \xrightarrow{t} \widehat{\mathbf{m}}'$ is in S_{k+2}. Remark that $(\mathsf{M}(n_{i+1}), \mathbf{m}') \sqsubseteq (\widehat{\mathbf{m}}, \widehat{\mathbf{m}}')$ because PN transitions have constant effects. By construction, there is in T_{k+2} a pair $(\overline{\mathbf{m}}, \overline{\mathbf{m}}')$ s.t. $(\widehat{\mathbf{m}}, \widehat{\mathbf{m}}') \sqsubseteq (\overline{\mathbf{m}}, \overline{\mathbf{m}}')$, hence $(\mathsf{M}(n_{i+1}), \mathbf{m}') \sqsubseteq (\overline{\mathbf{m}}, \overline{\mathbf{m}}')$.

   Inductive case $|\varsigma(n_{i+1})| = m+1$. Let us assume that $\varsigma(n_{i+1}) = \sigma' \cdot t$, where $|\sigma'| = m$. Let $\mathbf{m}''$ be the marking s.t. $\mathsf{M}(n_{i+1}) \xrightarrow{\sigma'} \mathbf{m}'' \xrightarrow{t} \mathbf{m}'$. By induction hypothesis, there exists a pair $(\overline{\mathbf{m}}, \overline{\mathbf{m}}'') \in T_{k+m+1}$ s.t. $(\mathsf{M}(n_{i+1}), \mathbf{m}'') \sqsubseteq (\overline{\mathbf{m}}, \overline{\mathbf{m}}'')$. Hence, $\overline{\mathbf{m}}'' \succcurlyeq \mathbf{m}''$. Thus, t is firable from $\overline{\mathbf{m}}''$. Let $\overline{\mathbf{m}}'$ be the marking s.t. $\overline{\mathbf{m}}'' \xrightarrow{t} \overline{\mathbf{m}}'$. By monotonicity, $\mathbf{m}' \preccurlyeq \overline{\mathbf{m}}'$. Since $\overline{\mathbf{m}}'' \in \mathsf{Flatten}(T_{k+m+1})$, the pair $(\overline{\mathbf{m}}'', \overline{\mathbf{m}}')$ is in S_{k+m+2}. Hence, the pair $(\overline{\mathbf{m}}, \overline{\mathbf{m}}')$ is in $\mathsf{TClosure}(S_{k+m+2} \cup A_{k+m+2} \cup T_{k+m+1})$. Remark that, since $(\mathsf{M}(n_{i+1}), \mathbf{m}'') \sqsubseteq (\overline{\mathbf{m}}, \overline{\mathbf{m}}'')$, and since PN transitions have constant effects, $(\mathsf{M}(n_{i+1}), \mathbf{m}') \sqsubseteq (\overline{\mathbf{m}}, \overline{\mathbf{m}}')$. Finally, by construction, there is in T_{k+m+2} a pair $(\widehat{\mathbf{m}}, \widehat{\mathbf{m}}')$ s.t. $(\overline{\mathbf{m}}, \overline{\mathbf{m}}') \sqsubseteq (\widehat{\mathbf{m}}, \widehat{\mathbf{m}}')$, hence $(\mathsf{M}(n_{i+1}), \mathbf{m}') \sqsubseteq (\widehat{\mathbf{m}}, \widehat{\mathbf{m}}')$.

   Thus, there is, in $T_{k+|\sigma|+1}$ a pair $(\overline{\mathbf{m}}, \overline{\mathbf{m}}')$ s.t. $(\mathsf{M}(n_{i+1}), \mathbf{m}') \sqsubseteq (\overline{\mathbf{m}}, \overline{\mathbf{m}}')$. Hence, there is in $A_{k+|\sigma|+2}$ a pair $(\overline{\mathbf{m}}', \overline{\mathbf{m}}'')$ s.t. $\overline{\mathbf{m}}'' = \mathsf{AccelPair}(\overline{\mathbf{m}}, \overline{\mathbf{m}}')$. By definition of $\sqsubseteq$, $\mathsf{AccelPair}(\overline{\mathbf{m}}, \overline{\mathbf{m}}') \succcurlyeq \mathsf{AccelPair}(\mathsf{M}(n_{i+1}), \mathbf{m}')$. Moreover, by Lemma 6.6, $\mathsf{AccelPair}(\mathsf{M}(n_{i+1}), \mathbf{m}') \succcurlyeq \Lambda(n_{i+1})$. By construction, there is, in $T_{k+|\sigma|+2}$, a pair $(\widehat{\mathbf{m}}', \widehat{\mathbf{m}}'')$ s.t. $(\overline{\mathbf{m}}', \overline{\mathbf{m}}'') \sqsubseteq (\widehat{\mathbf{m}}', \widehat{\mathbf{m}}'')$. Hence, we have:

$$\Lambda(n_{i+1}) \preccurlyeq \mathsf{AccelPair}(\mathsf{M}(n_{i+1}), \mathbf{m}') \preccurlyeq \mathsf{AccelPair}(\overline{\mathbf{m}}, \overline{\mathbf{m}}') = \overline{\mathbf{m}}'' \preccurlyeq \widehat{\mathbf{m}}''$$

with $\widehat{\mathbf{m}}'' \in \mathsf{Flatten}\left(T_{k+|\sigma|+2}\right)$. Thus, there is in $\mathsf{Flatten}\left(T_{k+|\sigma|+2}\right)$ a marking $\widehat{\mathbf{m}}$ s.t. $\widehat{\mathbf{m}} \succcurlyeq \Lambda\left(n_{i+1}\right)$. Moreover, using induction hypothesis, we obtain: $k + |\varsigma\left(n_{i+1}\right)| + 2 \leq \sum_{j=1}^{i+1}(|\varsigma\left(n_j\right)| + 2)$. Hence the lemma.

□

6.3.5 Stabilisation of the covering sequence

Thanks to the results introduced in the previous section, we show that for any Petri net $\mathcal{N}$, the sequence $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right)$, eventually stabilises to the coverability set of $\mathcal{N}$:

Theorem 6.1 *For any* PN $\mathcal{N} = \langle P, T, \mathbf{m}_0\rangle$ *with* $\mathsf{CovSeq}\left(\mathcal{N}\right) = T_0, T_1, \ldots, T_i, \ldots$, *there exists* $k \in \mathbb{N}$ *s.t.:*

- *For any* $0 \leq i \leq k-1$*:* $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) \subset \gamma\left(\mathsf{Flatten}\left(T_{i+1}\right)\right)$*;*
- *For any* $i \geq k$*:* $\gamma\left(\mathsf{Flatten}\left(T_k\right)\right) = \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$*.*

Proof. The proof works as follows. We first establish the existence of a value $k \in \mathbb{N}$ s.t. for any $i \geq k$: $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) = \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$ and for any $0 \leq i < k$: $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) \subset \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$. Then we show that $\mathsf{Flatten}\left(T_0\right) \subset \mathsf{Flatten}\left(T_1\right) \subset \cdots \subset \mathsf{Flatten}\left(T_{k-1}\right)$.

Let $\mathcal{T} = \langle N, B, \mathit{root}, \Lambda\rangle$ be the `Karp&Miller` tree for $\mathcal{N}$. According to Theorem 3.5, $\gamma\left(\Lambda\left(N\right)\right) = \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$, and N is finite. By Lemma 6.7, $\Lambda\left(N\right) \subseteq \gamma\left(\mathsf{Flatten}\left(T_j\right)\right)$ for some j s.t.

$$j \leq \max_{n \in N} \sum_{n' \in \mathsf{Ancestors}(\mathcal{T}, n)} (|\varsigma\left(n'\right)| + 2)$$

Remark that j is thus a finite value. Thus, $\gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right) = \gamma\left(\Lambda\left(N\right)\right) \subseteq \gamma\left(\mathsf{Flatten}\left(T_j\right)\right)$ for some $j \leq \max_{n\in N} \sum_{n' \in \mathsf{Ancestors}(\mathcal{T},n)}(|\varsigma\left(n'\right)| + 2)$. In addition, we know that, for any $i \geq 0$, $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) \subseteq \gamma\left(\mathsf{Flatten}\left(T_{i+1}\right)\right)$, by Lemma 6.4. We conclude that:

$$\exists k \in \mathbb{N} : \left(\begin{array}{c} \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right) \subseteq \gamma\left(\mathsf{Flatten}\left(T_k\right)\right) \\ \text{and} \\ \forall 0 \leq i < k : \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right) \not\subseteq \gamma\left(\mathsf{Flatten}\left(T_i\right)\right) \end{array}\right) \tag{6.6}$$

By corollary 6.1, $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) \subseteq \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$, for any $i \geq 0$. Hence, by (6.6), there exists $k \in \mathbb{N}$ s.t. for any $i \geq k$: $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) = \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$ and for any $0 \leq i < k$: $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) \subset \gamma\left(\mathsf{Post}^*\left(\mathbf{m}_0\right)\right)$.

It remains to show that for any $0 \leq i \leq k-1$: $\gamma\left(\mathsf{Flatten}\left(T_i\right)\right) \subset \gamma\left(\mathsf{Flatten}\left(T_{i+1}\right)\right)$. We prove this by invoking Knaster–Tarski's theorem. Let F be the function: $F(X) =$

$\gamma(\mathsf{Post}(X)) \cup \gamma(\{\mathbf{m}_0\})$. Remark that, for any X, $F(X)$ is $\preccurlyeq$–downward–closed. Moreover F is monotonic w.r.t. to $\subseteq$ and by Tarski's theorem: $\mu X.F(X)$ exists, is unique and equal to $\gamma(\mathsf{Post}^*(\gamma(\mathbf{m}_0))) = \gamma(\mathsf{Post}^*(\mathbf{m}_0))$. Let us show that :

$$\forall 0 \le i \le k-1 \quad : \quad \gamma(F(\mathsf{Flatten}(T_i))) \subseteq \gamma(\mathsf{Flatten}(T_{i+1})) \tag{6.7}$$

Indeed, $\gamma(F(\mathsf{Flatten}(T_i))) = \gamma(\mathbf{m}_0) \cup \gamma(\mathsf{Post}(\mathsf{Flatten}(T_i)))$. By definition, $\gamma(\mathbf{m}_0) \subseteq \gamma(\mathsf{Flatten}(T_0))$. By Lemma 6.4, $\gamma(\mathsf{Flatten}(T_0)) \subseteq \gamma(\mathsf{Flatten}(T_i))$ for any $i \ge 0$. Moerover, $\mathsf{Post}(\mathsf{Flatten}(T_i)) \subseteq \mathsf{Flatten}(S_{i+1})$, by construction. As a consequence, $\gamma(\mathsf{Post}(\mathsf{Flatten}(T_i))) \subseteq \gamma(\mathsf{Flatten}(S_{i+1}))$. Finally, for any marking $\mathbf{m}$ in $\mathsf{Flatten}(S_{i+1})$, there is a marking $\overline{\mathbf{m}} \in \mathsf{Flatten}(T_{i+1})$ s.t. $\mathbf{m} \preccurlyeq \overline{\mathbf{m}}$, by definition of T_{i+1}. Hence, $\gamma(\mathsf{Post}(\mathsf{Flatten}(T_i))) \subseteq \gamma(\mathsf{Flatten}(S_{i+1})) \subseteq \gamma(\mathsf{Flatten}(T_{i+1}))$. We conclude that for any $i \ge 0$, $\gamma(F(\mathsf{Flatten}(T_i))) = \gamma(\mathbf{m}_0) \cup \gamma(\mathsf{Post}(\mathsf{Flatten}(T_i))) \subseteq \gamma(\mathsf{Flatten}(T_{i+1}))$, which implies (6.7).

Then, we know that for any $0 \le i \le k-1$, $\gamma(\mathsf{Flatten}(T_i)) \subset \gamma(\mu X.F(X))$. Hence, the function F is increasing on $\{\gamma(\mathsf{Flatten}(T_i)) \mid 1 \le i \le k-1\}$, that is, for any $1 \le i \le k-1$:

$$\begin{array}{rcll} \gamma(\mathsf{Flatten}(T_i)) & \subset & F\big(\gamma(\mathsf{Flatten}(T_i))\big) & \\ & = & \gamma\Big(F\big(\gamma(\mathsf{Flatten}(T_i))\big)\Big) & F\big(\gamma(\mathsf{Flatten}(T_i))\big) \\ & & & \text{is } \preccurlyeq \text{ - down.-cl.} \\ & = & \gamma\Big(F\big(\mathsf{Flatten}(T_i)\big)\Big) & F \text{ is monotonic} \end{array} \tag{6.8}$$

The last point holds, because, for any X, we have $\gamma(F(\gamma(X))) = \gamma(F(X))$. Indeed, $\gamma(F(\gamma(X))) \supseteq \gamma(F(X))$ by monotonicity of F and γ. Moreover, $\gamma(F(\gamma(X))) \subseteq \gamma(F(X))$ is equivalent to $\forall \mathbf{m} \in F(\gamma(X)) : \exists \mathbf{m}' \in F(X) : \mathbf{m} \preccurlyeq \mathbf{m}'$. This is true by monotonicity of F.

By combining (6.7) and (6.8), we have: $\forall 0 \le i \le k-1 : \gamma(\mathsf{Flatten}(T_i)) \subset \gamma(\mathsf{Flatten}(T_{i+1}))$. ☐

Thus, Theorem 6.1 provides us with an algorithm to compute the coverability set of any Petri net $\mathcal{N}$: compute the covering sequence $\mathsf{CovSeq}(\mathcal{N}) = T_0, T_1, \ldots, T_j, \ldots$, stop as soon as $\gamma(\mathsf{Flatten}(T_j)) = \gamma(\mathsf{Flatten}(T_{j+1}))$ and return $\mathsf{Flatten}(T_j)$.

6.3.6 Empirical results

This section demonstrates that the computation of the covering sequence is actually an efficient procedure to compute the coverability set of Petri nets, when compared to classical methods. We have implemented this approach and run it on several examples[3]

[3]These examples come from our test suite, available at `http://www.ulb.ac.be/di/ssd/eec`. Remark that this set of examples has already been used as a benchmark for several other tools such as FAST [BFLP03] or VAMPIRE [RV02].

of Petri nets of various dimensions. We have also implemented the Karp and Miller procedure and a standard version of the Coverability Graph as described in [Rei86].

Before we discuss the actual results, it is important to say a few words about the prototype itself. This prototype is still in a very early stage of development. Indeed, the definition of the covering sequence (Definition 6.3) is purely theoretical (in order to keep the proofs simple), and leaves many open question regarding its (efficient) implementation. For instance:

1. an algorithm implementing the computation of the covering sequence should maintain a notion of *frontier* in order to avoid redeveloping successors of markings that have already been computed.

2. the efficient computation of the transitive closure has to be investigated. For instance, a set $\mathcal{R}$ of pairs of markings can be represented by a graph, whose nodes are markings, and whose directed edges correspond to pairs in $\mathcal{R}$. In that case, the computation of $\mathsf{TClosure}\,(\mathcal{R})$ can be carried out on the graph either *explicitly*, i.e., by actually adding edges between every $\mathbf{m}$ and $\mathbf{m}'$ s.t. there is a path going from $\mathbf{m}$ to $\mathbf{m}'$; or *implicitly*, i.e., by considering that the pair $(\mathbf{m}, \mathbf{m}')$ is in $\mathcal{R}$ iff there is a path (and not necessarily an edge) between $\mathbf{m}$ and $\mathbf{m}'$ in the graph. In the former case, the insertion of a new pair is costly because it potentially involves the re-computation of the transitive closure. However, deletion of a pair is easy: it is sufficient to remove the associated edge. In the latter case, insertion of a pair amounts to inserting a single edge, but the deletion of a pair is more difficult. It is presently unclear which solution is the most efficient in practice.

3. the efficient computation of the $\sqsubseteq$-maximal elements has to be investigated too. An efficient datastructure to represent $\sqsubseteq$-downward-closed sets of pairs of markings by means of maximal elements only could be useful (based on the CST datastructure, for instance [DRVB04]).

The prototype we present here has been realised with the (interpreted) PYTHON language [VRD03, Pyt]. This choice has been made because it allows a rapid development of the prototype, and because many easy-to-use libraries are readily available to manipulate complex datastructures, such as graphs for instance. In the present case, we have relied on the PYTHON version of the BOOST graph library [SLL01, Boo], in order to represent sets of pairs of markings. The implementations of the other algorithms we compare to (`Karp&Miller`, Coverability graph) have been realised in PYTHON too.

Our aim in the development of this new algorithm was to devise an algorithm that computes the minimal coverability set of PN by maintaining intermediate structures that are as small as possible (this was also the aim pursued by A. Finkel in his solution). Thus, our first criterion to assess the efficiency of the covering sequence is the *size* of the structures that are computed. Our simple implementations of the three algorithms are sufficient for this comparison. However, these implementations have not been designed to be efficient from the running time point of view: we have already pointed out several issues regarding to covering sequence, that should be addressed for that purpose. Moreover, a compiled programming language, such as C++ should be considered. We have thus decided not to report the running times of the three algorithms.

Table 6.1 compares the practical efficiency of the *covering sequence* method against the Karp&Miller tree and the coverability graph. We report on the sizes of the structures that are built by these algorithms, as well as the size of the minimal coverability set. In the case of the *covering sequence*, we provide the maximal number of markings and pairs computed along the whole computation, as well as the size of the set of pairs when the fixed point is reached. Indeed, the size of the sets of pairs typically decreases during the computation of the last iterations (before convergence of the sequence), because we keep $\sqsubseteq$-maximal elements only. Remark that in the case of the other two algorithms, the size of the structure at the end of the computation is also the maximal size throughout the computation, because no cut are performed. It is interesting to remark that, on all these examples, the set $\mathsf{Flatten}\left(T_i\right)$ obtained at the end is exactly the minimal coverability set of the PN.

The performances of our algorithm are in worst cases comparable to the performances of the Coverability graph computation. Remark however that the examples that are less favourable to our method are rather trivial, and leave little room for optimisations. There is thus little hope, to compute the coverability set of these examples in a more efficient way than the `Karp&Miller` procedure. On more complex examples, our algorithm performs much better: it is able to compute the coverability set of Petri nets for which the other procedures suffer from the explosion of the number of computed elements.

6.4 Discussion

In this chapter, we have presented a counter-example on which the Minimal Coverability Tree algorithm of [Fin91] might compute an under-approximation of the coverability set. We have also shown by means of a counter-example, that the variation of this algorithm implemented in Pep, and supposed to correct the bug, might not terminate in some cases.

We have not tried to present here a fix of the MCT algorithm. In the case of *bounded Petri nets*, a quick fix of the MCT algorithm is quite easy to obtain. Indeed, suppose that instead of *deleting* nodes from the tree when applying the reduction rule, we *freeze* these node, that is, we remove them from the frontier (if necessary) and mark them in order not to consider them in the further steps of the algorithm (the algorithm will behave 'as if' these nodes were not present in the tree). At the end of the computation, one can easily detect that the algorithm has not computed a coverability set. It is the case iff $\gamma\left(\mathsf{Post}\left(\Lambda\left(N\right)\right)\right) \not\subseteq \gamma\left(\Lambda\left(N\right)\right)$, where N is the set of nodes of the computed tree

Example			**MCS**	CovSeq ($\mathcal{N}$)				**K&M**		**Cov. Graph**	
Name	P	T	#M	max #M	max #P	#M	#P	#V	#E	#V	#E
example1	3	3	1	4	6	1	1	6	5	5	5
basicME	5	4	3	3	9	3	9	5	4	3	4
manufacturing	13	6	1	3	2	1	1	32	32	11	22
csm	14	13	16	26	256	16	256	×	×	55	178
kanban	16	16	1	5	5	1	1	×	×	104	381
fms	22	20	24	216	1,450	24	576	×	×	×	×

Table 6.1: Empirical comparison of the covering sequence against other methods to compute the coverability set of PN. Column **Example** details the PN used in the experiment (P = number of places. T = number of transitions). Column **MCS** is the size of the minimal coverability set (number of markings). Column CovSeq ($\mathcal{N}$) reports on the efficiency of the covering sequence (#M (#P) = number of markings (pairs) at the end of the computation. max #M (max #P) = maximal number of markings (pairs) along the whole computation. Column **K&M** reports on the size of the Karp&Miller tree. Column **Cov. Graph** reports on the size of the coverability graph (#V = maximal number of vertices. #E maximal number of edges). A × in the column means that the computed structure has more than 2000 nodes, and that the corresponding algorithm has been stopped before termination.

(that is, without the frozen nodes). In this case, we can re-use the frozen nodes n s.t. $\gamma(n) \not\subseteq \gamma(\Lambda(N))$, insert them in the frontier and run the MCT algorithm once more, starting from this frontier. That approach has been followed in [GRVB05]. We refer the reader to this paper for more details.

Other ways to fix the MCT algorithm surely exist. For instance, one could maintain in the tree the *proofs* that are assumed by the MCT algorithm when cutting a subtree (see Section 6.1.3), and *avoid cycles*.

We have preferred to consider the problem from scratch, and devise a completely new approach to the computation of the coverability set. From our point of view, this approach offers several advantages. The proofs are neat and compact. The concepts at work are simpler. Moreover, besides the implementation issues that we have already mentioned in the previous section, these ideas pave the way for future research. It would be interesting to see whether such an approach could be applied to other monotonic systems (even those for which the coverability set is not computable: our approach could lead to a practical improvement). The main difficulty relies in the definition of a difference operator similar to $\ominus$ in the case of these systems. We leave these questions open for future works.

Part II

Expressiveness properties

Chapter 7

ω-languages defined by WSTS

As we have seen in the first part of the thesis, decidability properties of WSTS have been fairly well studied. Quite intriguingly, their *expressiveness properties* have seldom been addressed in the literature, if we except the notable exception of Petri nets (in [Pet81], for instance). This is rather remarkable, if we compare the study of WSTS to that of other models of computation such as Turing Machines, two counters machines [Min67], pushdown and finite automata [HMU01, Sal73], whose defined languages (in terms of finite or infinite words) have been fairly well characterised.

Thus, there are many open problems to be looked into, and the second part of this thesis is devoted to the study of properties of languages and ω-languages that are defined by (labelled) EWSTS and EPN [1]. We begin our study of expressiveness properties of WSTS by the languages of infinite words, or ω-languages. This might seem unusual to the reader who is acquainted with the classical works of the literature dealing, for instance, with the expressiveness of finite automata, and where languages of *finite* words are usually covered before ω-languages. We have chosen to begin with ω-languages because the results we are about to prove in the present chapter as well as the proof techniques used here are somewhat similar to some of the results that will appear in the next chapter (where we address finite words languages). However, the proofs are, in the case of ω-languages, much shorter and easier, and the present chapter will also serve as an introduction to the proof techniques that we exploit in both chapters.

Our interest in the study of ω-languages of WSTS comes from the fact that these systems can model *non-terminating reactive systems* that interact with an environment. To formally reason about the correctness of such systems, we need formal models of their behaviours. At some abstract level, the behaviour of a non-terminating reactive

[1]From these notions, it is easy to define the language of a strongly monotonic SMPN, which we do not address in this thesis. All the general results obtained on EWSTS can be directly applied to SMPN.

system within its environment can be seen as an *infinite sequence of events* (usually taken within a finite set of events). The semantics of those systems is thus a (usually infinite) set of those infinite behaviours, that is, an ω-language.

As we have seen already, several recent works [GS92, DRVB02, DEP99] have considered the modelling of *infinite* reactive systems (for instance, systems where the number of processes cannot be bounded) by means of *counting abstractions*, where the individual identities of the processes are hidden. Such systems are perfectly modelled by Petri nets, possibly extended with non-blocking or transfer arcs to model various communication procedures (see Section 2.3.3).

In the present chapter, we thus restrict our attention to subclasses of $L^\omega(\mathsf{WSTS})$, namely, the classes of ω-languages that are defined respectively by EPN (and their subclasses). In Section 7.1, we prove that PN+T are strictly more expressive than PN+NBA, and, by using similar techniques, we prove in Section 7.2 that PN+NBA are strictly more expressive than PN. We obtain thus a strict hierarchy of expressiveness of these three models. From our point of view, this result is interesting because, the strict separation of expressive power indicates the meaningfulness of the special arcs of PN+T and PN+NBA to model communication procedures found in distributed systems.

On the other hand, the proof techniques that we use in the present chapter seem interesting *per se*, because they are a direct exploitation of the monotonicity properties of these models, and of the properties of infinite sequences of markings.

The content of this chapter is essentially based on the two articles [FGRVB05] and [FGRVB06]. Section 7.4 contains unpublished material.

7.1 PN+T are more expressive than PN+NBA

In this section, we prove that PN+T are strictly more expressive, on ω-languages, than PN+NBA. Let us first give a rough sketch of the proof.

We prove this in two steps. First, we show that any ω-language accepted by a PN+NBA can be accepted by a PN+T (this is the purpose of Lemma 7.3 and Theorem 7.1). Then, we prove the strictness of the inclusion thanks to the PN+T $\mathcal{N}_1$ of Fig. 7.1. Namely, we show that $L^\omega(\mathcal{N}_1)$ contains at least the words $(\mathtt{a}^k\mathtt{b}^k)^\omega$, for any $k \geq 1$ (Lemma 7.4). On the other hand we show that $\mathcal{N}_1$ rejects the words whose prefixes belongs to $(\mathtt{a}^{n_3}\mathtt{b}^{n_3})^*\mathtt{a}^{n_3}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^+\mathtt{b}^{n_2}$ with $n_1 < n_2 < n_3$ (Lemma 7.5). We finally show that any PN+NBA accepting words of the form $(\mathtt{a}^k\mathtt{b}^k)^\omega$ also has to accept words whose prefixes belong to $(\mathtt{a}^{n_3}\mathtt{b}^{n_3})^*\mathtt{a}^{n_3}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^+\mathtt{b}^{n_2}$ with $n_1 < n_2 < n_3$. Since $\mathcal{N}_1$ rejects these words, we conclude that no PN+NBA can accept $L^\omega(\mathcal{N}_1)$.

7.1.1 PN+NBA are not more expressive than PN+T.

In order to establish this result, we re–use the construction, introduced in Section 5.4.4, that turns a PN+NBA $\mathcal{N}$ into a corresponding PN+T $\mathcal{N}'$ (actually a PN+R, but every PN+R is a PN+T, by definition). We handle the labels as follows: for any transition $t \in T_r$, the corresponding t' has the same label as t. For any $t \in T_e$, the corresponding $t^{=0}$ and $t^{\neq 0}$ have the same label as t. We also re–use the two functions f and g defined there to establish the correspondence between the transitions of $\mathcal{N}$ and $\mathcal{N}'$, as well as the notations $\preccurlyeq_P$, $=_P$, and so forth. This allows to establish the two following lemmata:

Lemma 7.1 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ be a* PN+NBA *and let $\mathcal{N}' = \langle P', T', \mathbf{m}'_0 \rangle$ be the* PN+T *obtained from $\mathcal{N}$. Let $\sigma = \tau_1 \tau_2 \ldots \tau_j \ldots$ be an infinite (firable) sequence of transitions of $\mathcal{N}$. Let $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_j, \ldots$ be the markings s.t.*

$$\mathbf{m}_0 \xrightarrow{\tau_1} \mathbf{m}_1 \xrightarrow{\tau_2} \mathbf{m}_2 \ldots \mathbf{m}_{j-1} \xrightarrow{\tau_j} \mathbf{m}_j \ldots$$

Let $\sigma' = \tau'_1 \tau'_2 \ldots \tau'_j \ldots$ be the corresponding sequence of transitions of $\mathcal{N}'$ s.t. for any $j \geq 1$: $\tau'_j = f(\tau_j, \mathbf{m}_{j-1})$.

Then σ' is firable from $\mathbf{m}'_0$ and for any $j \geq 1$: $\mathbf{m}'_j =_P \mathbf{m}_j$, where $\mathbf{m}'_1, \mathbf{m}'_2, \ldots, \mathbf{m}'_j, \ldots$ are the markings s.t.

$$\mathbf{m}'_0 \xrightarrow{\tau'_1} \mathbf{m}'_1 \xrightarrow{\tau'_2} \mathbf{m}'_2 \ldots \mathbf{m}'_{j-1} \xrightarrow{\tau'_j} \mathbf{m}'_j, \ldots$$

Proof. First remark that, by Lemma 5.8, σ' is indeed firable from $\mathbf{m}'_0$.

Then, for any $k \geq 1$, let $\sigma_k = \tau_1 \ldots \tau_k$ and $\sigma'_k = \tau'_1 \ldots \tau'_k$. Remark that σ_k is the unique prefix of length k of σ and that σ'_k is the unique prefix of length k of σ'. Thus, $\mathbf{m}_0 \xrightarrow{\sigma_k} \mathbf{m}_k$ and $\mathbf{m}'_0 \xrightarrow{\sigma'_k} \mathbf{m}'_k$. By Lemma 5.8, and by definition of σ', we have $\mathbf{m}'_k =_P \mathbf{m}_k$. □

Lemma 7.2 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ be a* PN+NBA *and let $\mathcal{N}' = \langle P', T', \mathbf{m}'_0 \rangle$ be the* PN+T *obtained from $\mathcal{N}$. Let $\sigma' = \tau'_1 \tau'_2 \ldots \tau'_j \ldots$ be a sequence of transitions of $\mathcal{N}'$. Let $\mathbf{m}'_1, \mathbf{m}'_2, \ldots, \mathbf{m}'_j, \ldots$ be the markings s.t.*

$$\mathbf{m}'_0 \xrightarrow{\tau'_1} \mathbf{m}'_1 \xrightarrow{\tau'_2} \mathbf{m}'_2 \ldots \mathbf{m}'_{j-1} \xrightarrow{\tau'_j} \mathbf{m}'_j \ldots$$

Let $\sigma = g(\tau'_1) g(\tau'_2) \ldots g(\tau'_j), \ldots$ be the corresponding sequence of transitions of $\mathcal{N}$.

Then, σ is firable from $\mathbf{m}_0$ and, for all $j \geq 1$: $\mathbf{m}_j \preccurlyeq_P \mathbf{m}'_j$, where $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_j, \ldots$ are the markings s.t.

$$\mathbf{m}_0 \xrightarrow{g(\tau'_1)} \mathbf{m}_1 \xrightarrow{g(\tau'_2)} \mathbf{m}_2 \ldots \mathbf{m}_{j-1} \xrightarrow{g(\tau'_j)} \mathbf{m}_j, \ldots$$

Proof. The result can be established by the same argument as for Lemma 7.1, by invoking Lemma 5.9. □

Thus, we have:

Lemma 7.3 *For any* PN+NBA $\mathcal{N}$*, it is possible to construct a* PN+T $\mathcal{N}'$ *such that* $L^\omega(\mathcal{N}) = L^\omega(\mathcal{N}')$.

Proof. Let σ be a firable sequence of transitions of $\mathcal{N}$, and let σ' be its corresponding sequence in $\mathcal{N}'$ (obtained thanks to function f). By construction, we have $\Lambda(\sigma') = \Lambda(\sigma)$. By Lemma 7.1, σ' is firable in $\mathcal{N}'$. Hence, $L^\omega(\mathcal{N}) \subseteq L^\omega(\mathcal{N}')$. By a similar reasoning and thanks to Lemma 7.2, we conclude that $L^\omega(\mathcal{N}) \supseteq L^\omega(\mathcal{N}')$. Hence, $L^\omega(\mathcal{N}) = L^\omega(\mathcal{N}')$. □

And thus, we immediately obtain:

Theorem 7.1 *For every ω-language L that is accepted by a* PN+NBA*, there exists a* PN+T *that accepts L.*

Proof. The Theorem stems directly from Lemma 7.3. □

7.1.2 PN+T are more expressive than PN+NBA

Let us now prove that $L^\omega(\mathsf{PN+NBA})$ is *strictly* included in $L^\omega(\mathsf{PN+T})$. We consider the PN+T $\mathcal{N}_1$ presented in Figure 7.1 with the initial marking $\mathbf{m}_0(p_1) = 1$ and $\mathbf{m}_0(p) = 0$ for $p \in \{p_2, p_3, p_4\}$. The two following Lemmata allow us to better understand the behaviour of $\mathcal{N}_1$.

Lemma 7.4 *For any $k \geq 1$, the word $\left(\mathsf{a}^k\mathsf{b}^k\right)^\omega$ is accepted by $\mathcal{N}_1$.*

Proof. The following holds for any $k \geq 1$. From the initial marking of $\mathcal{N}_1$, we can fire $t_1^k t_2 t_3^{k-1}$ (which accepts $\mathsf{a}^k\mathsf{b}^k$), and reach the marking $\mathbf{m}_1$ such that $\mathbf{m}_1(p_2) = 1$ and $\forall i \in \{1,3,4\} : \mathbf{m}_1(p_i) = 0$. Thus, t_4 is firable from $\mathbf{m}_1$ and does not transfer any token, but produces a token in p_3 and moves the token from p_2 to p_1. The marking that is obtained is $\mathbf{m}_2$ such that $\mathbf{m}_2(p_1) = \mathbf{m}_2(p_3) = 1$ and $\mathbf{m}_2(p_2) = \mathbf{m}_2(p_4) = 0$. It is not difficult to see now that $\mathbf{m}_2 \xrightarrow{t_1^{k-1}t_2t_3^{k-1}} \mathbf{m}_1 \xrightarrow{t_4} \mathbf{m}_2$. We can thus fire the sequence $t_1^{k-1}t_2t_3^{k-1}t_4$ arbitrarily often from $\mathbf{m}_2$. Hence $\left(\mathsf{a}^k\mathsf{b}^k\right)^\omega$ is accepted by $\mathcal{N}_1$. □

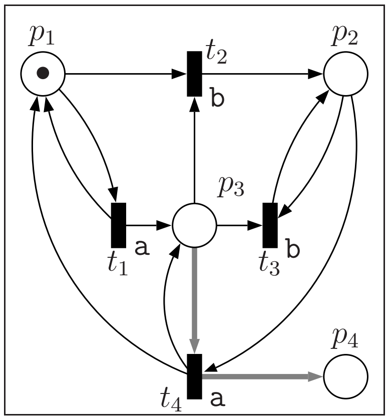


Figure 7.1: The PN+T $\mathcal{N}_1$.

Lemma 7.5 *Let n_1, n_2, n_3 and m be four natural numbers such that $0 < n_1 < n_2 < n_3$ and $m > 0$. Then, for any $k \geq 0$ the words*

$$(\mathtt{a}^{n_3}\mathtt{b}^{n_3})^k\mathtt{a}^{n_3}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^m(\mathtt{b}^{n_2}\mathtt{a}^{n_2})^\omega$$

are not accepted by $\mathcal{N}_1$.

Proof. The following holds for any n_1, n_2, n_3 with $0 < n_1 < n_2 < n_3$ and for any $m > 0$ and $k \in \mathbb{N}$. From the initial marking of $\mathcal{N}_1$, the only sequence of transitions labelled by $\mathtt{a}^{n_3}$ is $t_1^{n_3}$. Firing this sequence leads to the marking $\mathbf{m}_1$ such that $\mathbf{m}_1(p_1) = 1, \mathbf{m}_1(p_3) = n_3$ and $\mathbf{m}_1(p) = 0$ if $p \in \{p_2, p_4\}$. From $\mathbf{m}_1$ the only firable sequence of transitions labelled by $\mathtt{b}^{n_3}$ is $t_2t_3^{n_3-1}$. This leads to the marking $\mathbf{m}_2$ such that $\mathbf{m}_2(p_2) = 1$ and $\mathbf{m}_2(p) = 0$ if $p \neq p_2$. The only sequence of transitions firable from $\mathbf{m}_2$ and labelled by $\mathtt{a}^{n_3}$ is $t_4t_1^{n_3-1}$. Since $\mathbf{m}_2(p_3) = 0$, the transfer of t_4 has no effect when fired from $\mathbf{m}_2$. Hence, we reach $\mathbf{m}_1$ again after firing $t_4t_1^{n_3-1}$. By repeating the reasoning, we conclude that the only sequence of transitions firable from the initial marking and labelled by $(\mathtt{a}^{n_3}\mathtt{b}^{n_3})^k\mathtt{a}^{n_3}$ (when $k > 0$) is $t_1^{n_3}t_2t_3^{n_3-1}(t_4t_1^{n_3-1}t_2t_3^{n_3-1})^{k-1}t_4t_1^{n_3-1}$ and it leads to $\mathbf{m}_1$. In the case where $k = 0$, the sequence $t_1^{n_3}$ is firable and leads to $\mathbf{m}_1$ too. From $\mathbf{m}_1$, the only firable sequence of transitions labelled by $\mathtt{b}^{n_1}$ is $t_2t_3^{n_1-1}$. This leads to a marking similar to $\mathbf{m}_2$, noted $\mathbf{m}_2'$, except that p_3 contains $n_3 - n_1$ tokens. Then, the only firable sequence of transitions labelled by $\mathtt{a}^{n_1}$ is $t_4t_1^{n_1-1}$. In this case, the transfer of t_4 moves the $n_3 - n_1$ tokens from p_3 to p_4 and we reach a marking similar to $\mathbf{m}_1$, noted $\mathbf{m}_1'$, except that p_4 contains $n_3 - n_1$ tokens and p_3 contains n_1 tokens. From $\mathbf{m}_1'$, the only firable sequence of transitions labelled by $\mathtt{b}^{n_1}\mathtt{a}^{n_1}$ is $t_2t_3^{n_1-1}t_4t_1^{n_1-1}$. This sequence leads to $\mathbf{m}_1'$ and can thus be fired m times.

However, after firing $t_2t_3^{n_1-1}$ from $\mathbf{m}_1'$, we reach a marking $\mathbf{m}_2''$ similar to $\mathbf{m}_2$ except that p_4 contains $n_3 - n_1$ tokens and from which no transition labelled by $\mathtt{b}$ is firable. Since $n_2 > n_1$, we conclude that there is no sequence of transitions labelled by $\mathtt{b}^{n_2}$ that is firable from $\mathbf{m}_1'$, hence $(\mathtt{a}^{n_3}\mathtt{b}^{n_3})^k\mathtt{a}^{n_3}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^m(\mathtt{b}^{n_2}\mathtt{a}^{n_2})^\omega$ is not accepted by $\mathcal{N}_1$. □

Our proof that $L^\omega(\mathsf{PN+NBA})$ is *strictly* included in $L^\omega(\mathsf{PN+T})$ consists in showing that no $\mathsf{PN+NBA}$ can accept the language accepted by $\mathcal{N}_1$. For that purpose, we rely on the two previous lemmata, as well as the following result that characterises sequences of transitions of $\mathsf{PN+NBA}$:

Lemma 7.6 *Let $\mathcal{N} = \langle \mathcal{P}, \mathcal{T}, \Sigma, \mathbf{m}_0\rangle$ be a* $\mathsf{PN+NBA}$*, and let σ be a finite sequence of transitions of $\mathcal{N}$ that contains n occurrences of transitions in $\mathcal{T}_e$. Let $\mathbf{m}_1$, $\mathbf{m}'_1$, $\mathbf{m}_2$ and $\mathbf{m}'_2$ be four makings such that (i) $\mathbf{m}_1 \xrightarrow{\sigma} \mathbf{m}'_1$, (ii) $\mathbf{m}_2 \xrightarrow{\sigma} \mathbf{m}'_2$ and (iii) $\mathbf{m}_2 \succcurlyeq \mathbf{m}_1$. Then, for every place $p \in \mathcal{P}$: $\mathbf{m}'_2(p) - \mathbf{m}'_1(p) \geq \mathbf{m}_2(p) - \mathbf{m}_1(p) - n$.*

Proof. Let us consider a place $p \in \mathcal{P}$. First, we remark that when we fire σ from $\mathbf{m}_2$ instead of $\mathbf{m}_1$, its Petri net arcs will have the same effect on p. On the other hand, since we want to find a lower bound on $\mathbf{m}'_2(p) - \mathbf{m}'_1(p)$, we consider the situation where no non-blocking arcs affect p when σ is fired from $\mathbf{m}_1$, but they all remove one token from p when σ is fired from $\mathbf{m}_2$. In the latter case, the effect of σ on p is $\mathbf{m}'_1(p) - \mathbf{m}_1(p) - n$. We obtain thus: $\mathbf{m}'_2(p) \geq \max\{\mathbf{m}_2(p) + \mathbf{m}'_1(p) - \mathbf{m}_1(p) - n, 0\}$. Hence $\mathbf{m}'_2(p) \geq \mathbf{m}'_1(p) + \mathbf{m}_2(p) - \mathbf{m}_1(p) - n$, and thus: $\mathbf{m}'_2(p) - \mathbf{m}'_1(p) \geq \mathbf{m}_2(p) - \mathbf{m}_1(p) - n$. □

We can now show that no $\mathsf{PN+NBA}$ can accept $L^\omega(\mathcal{N}_1)$. Remark that the proof technique used hereafter relies on Lemmata 2.2 and 2.14, and is somewhat similar to a *pumping lemma* (such as Lemma 2.16, for instance).

Lemma 7.7 *No* $\mathsf{PN+NBA}$ *accepts $L^\omega(\mathcal{N}_1)$.*

Proof. Let $\mathcal{N}$ be a $\mathsf{PN+NBA}$ such that $L^\omega(\mathcal{N}_1) \subseteq L^\omega(\mathcal{N})$. We will show that this implies that $L^\omega(\mathcal{N}_1) \subsetneq L^\omega(\mathcal{N})$. As $L^\omega(\mathcal{N}_1) \subseteq L^\omega(\mathcal{N})$, by Lemma 7.4 we know that, for all $k \geq 1$, the word $(\mathtt{a}^k\mathtt{b}^k)^\omega$ belongs to $L^\omega(\mathcal{N})$. Suppose that $\mathbf{m}_{init}$ is the initial marking of $\mathcal{N}$. Thus, for all $k \geq 1$, there exists a marking $\widehat{\mathbf{m}}_k$, a finite sequence of transitions σ_k, a natural number ℓ_k and $n_k \geq 1$ such that:

$$\mathbf{m}_{init} \xrightarrow{(\mathtt{a}^k\mathtt{b}^k)^{\ell_k}\mathtt{a}^k} \widehat{\mathbf{m}}_k \xrightarrow{\Lambda(\sigma_k)} \widehat{\mathbf{m}}'_k, \widehat{\mathbf{m}}'_k \not\succcurlyeq \widehat{\mathbf{m}}_k \text{ and } \Lambda(\sigma_k) = (\mathtt{b}^k\mathtt{a}^k)^{n_k}$$

Indeed, if this were not the case, we would have $\mathbf{m}_{init} \xrightarrow{\mathtt{a}^k} \mathbf{m}_1 \xrightarrow{\mathtt{b}^k\mathtt{a}^k} \mathbf{m}_2 \ldots \xrightarrow{\mathtt{b}^k\mathtt{a}^k} \mathbf{m}_i \xrightarrow{\mathtt{b}^k\mathtt{a}^k} \ldots$ such that there does not exist $1 \leq i < j$ with $\mathbf{m}_i \preccurlyeq \mathbf{m}_j$. But from Lemma 2.2, this never occurs.

Let us consider the infinite sequence $\widehat{\mathbf{m}}_1, \widehat{\mathbf{m}}_2, \ldots, \widehat{\mathbf{m}}_i, \ldots$ Following Lemma 2.2 again, we may extract from it a sub-sequence $\widehat{\mathbf{m}}_{\rho(1)}, \widehat{\mathbf{m}}_{\rho(2)}, \ldots, \widehat{\mathbf{m}}_{\rho(n)}, \ldots$ such that: $\forall p \in P$: either $\forall i \geq 1 : \widehat{\mathbf{m}}_{\rho(i)}(p) = \widehat{\mathbf{m}}_{\rho(i+1)}(p)$ or $\forall i \geq 1 : \widehat{\mathbf{m}}_{\rho(i)}(p) < \widehat{\mathbf{m}}_{\rho(i+1)}(p)$. Let us denote by P' the set of places that strictly increase in that sequence.

Let n be the number of occurrences of transitions of T_e in $\sigma_{\rho(1)}$ and let us consider $\widehat{\mathbf{m}}_{\rho(1)}$, $\widehat{\mathbf{m}}_{\rho(2)}$, $\widehat{\mathbf{m}}_{\rho(n+3)}$, and $\mathbf{m}$ such that: $\widehat{\mathbf{m}}_{\rho(n+3)} \xrightarrow{\sigma_{\rho(1)}} \mathbf{m}$ (from Lemma 2.14, the sequence $\sigma_{\rho(1)}$ is firable from $\widehat{\mathbf{m}}_{\rho(n+3)}$ since $\widehat{\mathbf{m}}_{\rho(1)} \preccurlyeq \widehat{\mathbf{m}}_{\rho(n+3)}$ and $\sigma_{\rho(1)}$ is firable from $\widehat{\mathbf{m}}_{\rho(1)}$). We first prove that $\mathbf{m} \succcurlyeq \widehat{\mathbf{m}}_{\rho(2)}$.

We know that:

$$\widehat{\mathbf{m}}_{\rho(1)} \xrightarrow{\sigma_{\rho(1)}} \widehat{\mathbf{m}}'_{\rho(1)} \;\wedge\; \widehat{\mathbf{m}}'_{\rho(1)} \succcurlyeq \widehat{\mathbf{m}}_{\rho(1)} \tag{7.1}$$

$$\forall p \in \mathcal{P}' \;:\; \widehat{\mathbf{m}}_{\rho(n+3)}(p) \geq \widehat{\mathbf{m}}_{\rho(2)}(p) + n + 1 \tag{7.2}$$

$$\forall p \in P \setminus \mathcal{P}' \;:\; \widehat{\mathbf{m}}_{\rho(1)}(p) = \widehat{\mathbf{m}}_{\rho(2)}(p) = \widehat{\mathbf{m}}_{\rho(n+3)}(p) \tag{7.3}$$

Thus:

$$\begin{array}{lllll}
\textbf{(a)} & \forall p \in P' & : & \mathbf{m}(p) \geq \widehat{\mathbf{m}}'_{\rho(1)}(p) + \big(\widehat{\mathbf{m}}_{\rho(n+3)}(p) - \widehat{\mathbf{m}}_{\rho(1)}(p)\big) - n & \text{by Lemma 7.6} \\
\Rightarrow & \forall p \in P' & : & \mathbf{m}(p) \geq \widehat{\mathbf{m}}_{\rho(1)}(p) + \big(\widehat{\mathbf{m}}_{\rho(n+3)}(p) - \widehat{\mathbf{m}}_{\rho(1)}(p)\big) - n & \text{by (7.1)} \\
\Rightarrow & \forall p \in P' & : & \mathbf{m}(p) \geq \widehat{\mathbf{m}}_{\rho(n+3)}(p) - n & \\
\Rightarrow & \forall p \in P' & : & \mathbf{m}(p) \geq \widehat{\mathbf{m}}_{\rho(2)}(p) + 1 & \text{by (7.2)} \\
\Rightarrow & \forall p \in P' & : & \mathbf{m}(p) > \widehat{\mathbf{m}}_{\rho(2)}(p) &
\end{array}$$

(b) By monotonicity of PN+NBA (Lemma 2.14), we have that $\mathbf{m} \succcurlyeq \widehat{\mathbf{m}}'_{\rho(1)}$. Moreover, by (7.1), we have that $\widehat{\mathbf{m}}'_{\rho(1)} \succcurlyeq \widehat{\mathbf{m}}_{\rho(1)}$. Hence, $\forall p \in P : \mathbf{m}(p) \geq \widehat{\mathbf{m}}_{\rho(1)}(p)$. As a consequence, $\forall p \in P \setminus P' : \mathbf{m}(p) \geq \widehat{\mathbf{m}}_{\rho(2)}(p)$ from (7.3).

From (a) and (b), we obtain $\mathbf{m} \succcurlyeq \widehat{\mathbf{m}}_{\rho(2)}$, hence $\sigma_{\rho(2)}$ is firable from $\mathbf{m}$. And so:

$$\mathbf{m}_{init} \xrightarrow{\left(\mathsf{a}^{\rho(3+n)}\mathsf{b}^{\rho(3+n)}\right)^{\ell_{\rho(3+n)}}\mathsf{a}^{\rho(3+n)}} \widehat{\mathbf{m}}_{\rho(3+n)} \xrightarrow{\left(\mathsf{b}^{\rho(1)}\mathsf{a}^{\rho(1)}\right)^{n_{\rho(1)}}} \mathbf{m} \xrightarrow{\left(\mathsf{b}^{\rho(2)}\mathsf{a}^{\rho(2)}\right)^{n_{\rho(2)}}} \mathbf{m}'$$

Finally, let us prove that we can fire $\sigma_{\rho(2)}$ infinitely often from $\mathbf{m}'$. Since $\mathbf{m} \succcurlyeq \widehat{\mathbf{m}}_{\rho(2)}$ and $\widehat{\mathbf{m}}_{\rho(2)} \xrightarrow{\sigma_{\rho(2)}} \widehat{\mathbf{m}}'_{\rho(2)}$, we have by Lemma 2.14 that $\mathbf{m}' \succcurlyeq \widehat{\mathbf{m}}'_{\rho(2)} \succcurlyeq \widehat{\mathbf{m}}_{\rho(2)}$, hence $\mathbf{m}' \xrightarrow{\sigma_{\rho(2)}} \mathbf{m}''$ for some marking $\mathbf{m}'' \succcurlyeq \widehat{\mathbf{m}}'_{\rho(2)} \succcurlyeq \widehat{\mathbf{m}}_{\rho(2)}$. Since we can repeat the reasoning infinitely often from $\mathbf{m}''$, we conclude that $\sigma_{\rho(2)}$ can be fired infinitely often from $\mathbf{m}''$ and

$$\left(\mathsf{a}^{\rho(n+3)}\mathsf{b}^{\rho(n+3)}\right)^{\ell_{\rho(n+3)}}\mathsf{a}^{\rho(n+3)}\left(\mathsf{b}^{\rho(1)}\mathsf{a}^{\rho(1)}\right)^{n_{\rho(1)}}\left(\mathsf{b}^{\rho(2)}\mathsf{a}^{\rho(2)}\right)^{\omega}$$

is a word of $L^\omega(\mathcal{N})$ (with $\rho(n+3) > \rho(2) > \rho(1) > 0$ and $n_{\rho(1)} > 0$). But, following Lemma 7.5, this word is not in $L^\omega(\mathcal{N}_1)$. We conclude that $L^\omega(\mathcal{N}_1) \subsetneq L^\omega(\mathcal{N})$. □

We can now state the main Theorem of this section.

Theorem 7.2 PN+T *are more expressive, on infinite words, than* PN+NBA, *i.e.:* $L^\omega(\text{PN+NBA}) \subsetneq L^\omega(\text{PN+T})$.

Proof. From Theorem 7.1, we have that $L^\omega(\text{PN+NBA}) \subseteq L^\omega(\text{PN+T})$. However, there exist languages that can be recognised by a PN+T but not by PN+NBA, following Lemma 7.7. Hence, we conclude that $L^\omega(\text{PN+NBA}) \subsetneq L^\omega(\text{PN+T})$. □

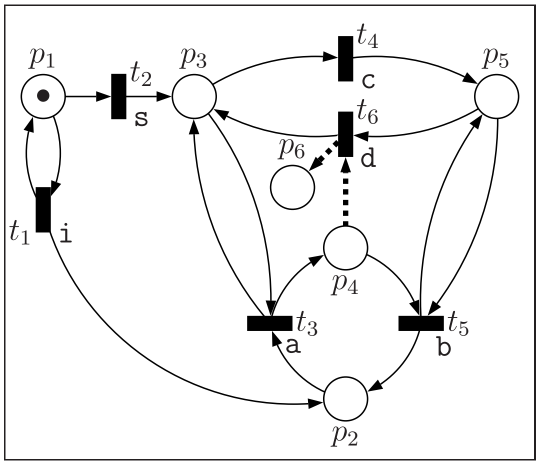


Figure 7.2: The PN+NBA $\mathcal{N}_2$.

Remark that Theorem 7.1 still holds in the case where we disallow ε-transitions, since the construction used in Lemma 7.3 does not require the use of ε-transitions. Moreover, since $\mathcal{N}_1$ contains no ε-transitions and since we have made no assumptions regarding the ε-transitions in the previous proofs, we obtain:

Corollary 7.1 PN+T *without ε-transitions are more expressive on infinite words than* PN+NBA, *i.e.: $L^{\omega}_{\not\varepsilon}(\mathsf{PN{+}NBA}) \subsetneq L^{\omega}_{\not\varepsilon}(\mathsf{PN{+}T})$.*

7.2 PN+NBA are more expressive than PN

In this section we prove that the class of ω-languages accepted by PN+NBA strictly contains the class of ω-languages accepted by PN.

The strategy adopted in the proof is similar to the one we have used in Section 7.1. We consider the PN+NBA $\mathcal{N}_2$ of Figure 7.2, and prove it accepts every word of the form $\mathtt{i}^k\mathtt{s}\big(\mathtt{a}^k\mathtt{c}\mathtt{b}^k\mathtt{d}\big)^{\omega}$, for $k \geq 1$ (Lemma 7.8), but rejects words of the form $\mathtt{i}^{n_3}\mathtt{s}\big(\mathtt{a}^{n_3}\mathtt{c}\mathtt{b}^{n_3}\mathtt{d}\big)^m\mathtt{a}^{n_3}\mathtt{c}\big(\mathtt{b}^{n_1}\mathtt{d}\mathtt{a}^{n_1}\mathtt{c}\big)^k\big(\mathtt{b}^{n_2}\mathtt{d}\mathtt{a}^{n_2}\mathtt{c}\big)^{\omega}$, for k big enough, and $0 < n_1 < n_2 < n_3$ (Lemma 7.9). Then, we prove Lemma 7.10, stating that any PN accepting at least the words of the first form must also accept the words of the latter form. We conclude that no PN can accept $L^{\omega}(\mathcal{N}_2)$. Since any PN is also a PN+NBA, the inclusion is immediate, and we obtain Theorem 7.3, that states the strictness of the inclusion $L^{\omega}(\mathsf{PN}) \subsetneq L^{\omega}(\mathsf{PN{+}NBA})$.

Let us consider the PN+NBA $\mathcal{N}_2$ in Figure 7.2, with the initial marking $\mathbf{m}_0$ such that $\mathbf{m}_0(p_1) = 1$ and $\mathbf{m}_0(p) = 0$ for $p \in \{p_2, p_3, p_4, p_5, p_6\}$.

Lemma 7.8 *For any $k \geq 0$, the word $\mathtt{i}^k\mathtt{s}\big(\mathtt{a}^k\mathtt{c}\mathtt{b}^k\mathtt{d}\big)^{\omega}$ is accepted by $\mathcal{N}_2$.*

Proof. The following holds for any $k \geq 0$. After firing the transitions $t_1^k t_2$ from the initial marking of $\mathcal{N}_2$, we reach the marking $\mathbf{m}_1$ such that $\mathbf{m}_1(p_2) = k$, $\mathbf{m}_1(p_3) = 1$, and $\mathbf{m}_1(p_j) = 0$ for $j \in \{1, 4, 5, 6\}$. Then, we can fire $t_3^k t_4$ from $\mathbf{m}_1$. This leads to the marking $\mathbf{m}_2$ such that $\mathbf{m}_2(p_4) = k$, $\mathbf{m}_2(p_5) = 1$, and $\mathbf{m}_2(p_j) = 0$ for $j \in \{1, 2, 3, 6\}$. From $\mathbf{m}_2$, t_5^k can be fired. This sequence of transitions moves the k tokens from p_4 to p_2. Then, from the resulting marking, t_6 can be fired. Since, p_4 is now empty, the effect of t_6 only consists in moving the token from p_5 to p_3 (its non-blocking arc has no effect) and we reach $\mathbf{m}_1$ again. Then, by applying the same reasoning, we fire infinitely often $t_3^k t_4 t_5^k t_6$. The word corresponding to such a sequence is $\mathtt{i}^k\mathtt{s}\big(\mathtt{a}^k\mathtt{c}\mathtt{b}^k\mathtt{d}\big)^\omega$. □

Lemma 7.9 *Let n_1, n_2 and n_3 be three natural numbers such that $0 < n_1 < n_2 < n_3$. Then, for all $m > 0$, for all $k \geq n_3 - n_1 - 1$: the words*

$$\mathtt{i}^{n_3}\mathtt{s}\big(\mathtt{a}^{n_3}\mathtt{c}\mathtt{b}^{n_3}\mathtt{d}\big)^m\mathtt{a}^{n_3}\mathtt{c}\big(\mathtt{b}^{n_1}\mathtt{d}\mathtt{a}^{n_1}\mathtt{c}\big)^k\big(\mathtt{b}^{n_2}\mathtt{d}\mathtt{a}^{n_2}\mathtt{c}\big)^\omega$$

are not accepted by $\mathcal{N}_2$.

Proof. In this proof, we will identify a sequence of transitions with the word it accepts (all the transitions have different labels). Clearly (see the proof of Lemma 7.8), the firing of $\mathtt{i}^{n_3}\mathtt{s}\big(\mathtt{a}^{n_3}\mathtt{c}\mathtt{b}^{n_3}\mathtt{d}\big)^m$ from $\mathbf{m}_0$ leads to a marking $\mathbf{m}_1$ such that $\mathbf{m}_1(p_2) = n_3$, $\mathbf{m}_1(p_3) = 1$, and $\forall i \in \{1, 4, 5, 6\} : \mathbf{m}_1(p_i) = 0$ (the non-blocking arc of t_6 hasn't consumed any token in p_4). By firing $\mathtt{a}^{n_3}\mathtt{c}\mathtt{b}^{n_1}\mathtt{d}$ from $\mathbf{m}_1$, we now have n_1 tokens in p_2, $n_3 - n_1 - 1$ tokens in p_4 and one token in p_6 (this time the non-blocking arc has moved one token since $n_1 < n_3$). Clearly, at each subsequent firing of $\mathtt{a}^{n_1}\mathtt{c}\mathtt{b}^{n_1}\mathtt{d}$, the non-blocking arc of t_6 will remove one token from p_4 and its marking will strictly decrease until it becomes empty. Let $\ell = n_3 - n_1 - 1$. It is easy to see that $\big(\mathtt{a}^{n_1}\mathtt{c}\mathtt{b}^{n_1}\mathtt{d}\big)^\ell$ leads to a marking $\mathbf{m}_2$ with $\mathbf{m}_2(p_2) = n_1$ $\mathbf{m}_2(p_3) = 1$ and $\forall j \in \{1, 4, 5\} : \mathbf{m}_2(p_j) = 0$. This characterisation also implies that we can fire $\mathtt{a}^{n_1}\mathtt{c}\mathtt{b}^{n_1}\mathtt{d}$ an arbitrary number of times from $\mathbf{m}_2$ because $\mathbf{m}_2 \xrightarrow{\mathtt{a}^{n_1}\mathtt{c}\mathtt{b}^{n_1}\mathtt{d}} \mathbf{m}_2$. On the other hand, it is not possible to fire $\mathtt{a}^{n_1}\mathtt{c}\mathtt{b}^{n_2}\mathtt{d}$, with $n_2 > n_1$, from $\mathbf{m}_2$. Indeed $\mathbf{m}_2 \xrightarrow{\mathtt{a}^{n_1}\mathtt{c}\mathtt{b}^{n_1}} \mathbf{m}_3$, with $\mathbf{m}_3(p_5) = 1$, $\mathbf{m}_3(p_2) = n_1$ and $\forall j \in \{1, 3, 4\} : \mathbf{m}_3(p_j) = 0$, which does not allow to fire the $\mathtt{b}$-labelled transition t_5 anymore. We conclude that, $\forall k \geq \ell = n_3 - n_1 - 1$, a sequence labelled by $\mathtt{i}^{n_3}\mathtt{s}\big(\mathtt{a}^{n_3}\mathtt{c}\mathtt{b}^{n_3}\mathtt{d}\big)^m\mathtt{a}^{n_3}\mathtt{c}\big(\mathtt{b}^{n_1}\mathtt{d}\mathtt{a}^{n_1}\mathtt{c}\big)^k\mathtt{b}^{n_2}\mathtt{d}\mathtt{a}^{n_2}\mathtt{c}$, is not firable in $\mathcal{N}_2$. Thus, we will not find in $L^\omega(\mathcal{N}_2)$ any word with this prefix, hence the Lemma. □

We are now ready to prove that no PN accepts exactly the ω-language of the PN+NBA $\mathcal{N}_2$.

Lemma 7.10 *No* PN *accepts $L^\omega(\mathcal{N}_2)$.*

Proof. Let $\mathcal{N}$ be a PN such that $L^\omega(\mathcal{N}_2) \subseteq L^\omega(\mathcal{N})$. We will show that this implies that $L^\omega(\mathcal{N}_2) \subsetneq L^\omega(\mathcal{N})$.

Suppose that $\mathbf{m}_{init}$ is the initial marking of $\mathcal{N}$. Following Lemma 7.8, since $L^\omega(\mathcal{N}_2) \subseteq L^\omega(\mathcal{N})$, we have $\forall k \geq 1 : \mathtt{i}^k\mathtt{s}\big(\mathtt{a}^k\mathtt{c}\mathtt{b}^k\mathtt{d}\big)^\omega \in L(\mathcal{N})$. Thus, for all $k \geq 1$, there exists a marking $\widehat{\mathbf{m}}_k$, a sequence of transitions σ_k and $\ell_k \in \mathbb{N}_0$ such that:

$$\mathbf{m}_{init} \xrightarrow{\mathtt{i}^k\mathtt{s}\big(\mathtt{a}^k\mathtt{c}\mathtt{b}^k\mathtt{d}\big)^{\ell_k}\mathtt{a}^k\mathtt{c}} \widehat{\mathbf{m}}_k \xrightarrow{\Lambda(\sigma_k)} \widehat{\mathbf{m}}'_k \text{ with } \widehat{\mathbf{m}}_k \preccurlyeq \widehat{\mathbf{m}}'_k \text{ and } \Lambda(\sigma_k) \in \big(\mathtt{b}^k\mathtt{d}\mathtt{a}^k\mathtt{c}\big)^+$$

Indeed, if it is not the case, we would have $\mathbf{m}_{init} \xrightarrow{\mathtt{i}^k\mathtt{s}\mathtt{a}^k\mathtt{c}} \mathbf{m}_1 \xrightarrow{\mathtt{b}^k\mathtt{d}\mathtt{a}^k\mathtt{c}} \dots \xrightarrow{\mathtt{b}^k\mathtt{d}\mathtt{a}^k\mathtt{c}} \mathbf{m}_i \xrightarrow{\mathtt{b}^k\mathtt{d}\mathtt{a}^k\mathtt{c}}$ $\dots$ such that there do not exist $1 \leq i < j$ with $\mathbf{m}_i \preccurlyeq \mathbf{m}_j$. But, from Lemma 2.2, this never occurs.

Let us consider the sequence $\widehat{\mathbf{m}}_1, \widehat{\mathbf{m}}_2, \widehat{\mathbf{m}}_3, \dots$ Following Lemma 2.2, we may extract an infinite sub-sequence $\widehat{\mathbf{m}}_{\rho(1)}, \widehat{\mathbf{m}}_{\rho(2)}, \widehat{\mathbf{m}}_{\rho(3)}, \dots$ such that $\forall p \in P$: either $\forall i \geq 1$: $\widehat{\mathbf{m}}_{\rho(i)}(p) = \widehat{\mathbf{m}}_{\rho(i+1)}(p)$ or $\forall i \geq 1 : \widehat{\mathbf{m}}_{\rho(i)}(p) < \widehat{\mathbf{m}}_{\rho(i+1)}(p)$.

Since $\widehat{\mathbf{m}}_{\rho(3)} \succcurlyeq \widehat{\mathbf{m}}_{\rho(1)}$ and $\sigma_{\rho(1)}$ has a non-negative and constant effect on each place (its effect is characterised by its Parikh vector, which is a tuple of naturals), we can fire $\sigma_{\rho(1)}$ any number of times from $\widehat{\mathbf{m}}_{\rho(3)}$: for all $k' \geq 0$ we have $\widehat{\mathbf{m}}_{\rho(3)} \xrightarrow{(\sigma_{\rho(1)})^{k'}} \mathbf{m}^{k'}$ with $\mathbf{m}^{k'} \succcurlyeq \widehat{\mathbf{m}}_{\rho(3)}$. Since $\widehat{\mathbf{m}}_{\rho(3)} \succcurlyeq \widehat{\mathbf{m}}_{\rho(2)}$ and $\sigma_{\rho(2)}$ has a constant non-negative effect on each place, $\sigma_{\rho(2)}$ can be fired infinitely often from $\mathbf{m}^{k'}$ for any $k' \geq 1$. Thus:

$$\mathbf{m}_{init} \xrightarrow{\mathtt{i}^{\rho(3)}\mathtt{s}\big(\mathtt{a}^{\rho(3)}\mathtt{c}\mathtt{b}^{\rho(3)}\mathtt{d}\big)^{\ell_{\rho(3)}}\mathtt{a}^{\rho(3)}\mathtt{c}} \widehat{\mathbf{m}}_{\rho(3)} \xrightarrow{(\sigma_{\rho(1)})^{k'}} \mathbf{m}^{k'} \xrightarrow{(\sigma_{\rho(2)})^\omega}$$

Following Lemma 7.9, if we choose k' large enough (that is, $k' \geq \rho(3) - \rho(1) - 1$), the word accepted by the previous sequence is not in $L^\omega(\mathcal{N}_2)$. Hence, $L^\omega(\mathcal{N}_2) \subsetneq L^\omega(\mathcal{N})$. □

Theorem 7.3 PN+NBA *are more expressive, on infinite words, than* PN. *That is,* $L^\omega(\mathsf{PN}) \subsetneq L^\omega(\mathsf{PN{+}NBA})$.

Proof. As the PN class is a syntactic subclass of the PN+NBA, each PN-language is also a PN+NBA-language. On the other hand, some PN+NBA-languages are not PN-languages, by Lemma 7.10. Hence the Theorem. □

Again, since PN is a syntactic subclass of PN+NBA and we have made no assumptions about the ε-transitions in the previous proofs, and since $\mathcal{N}_2$ contains no ε-transition, we obtain:

Corollary 7.2 PN+NBA *are more expressive than* PN, *on infinite words and without* ε*-transitions. That is,* $L^\omega_{\not\varepsilon}(\mathsf{PN}) \subsetneq L^\omega_{\not\varepsilon}(\mathsf{PN{+}NBA})$.

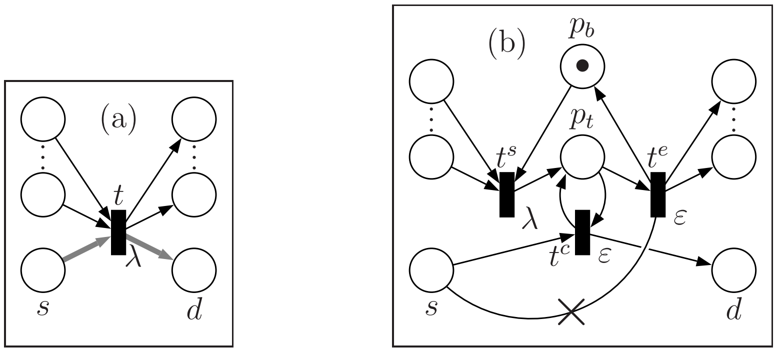


Figure 7.3: How to transform a PN+T (a) into a PN+R (b).

7.3 Reset nets

In this section we show how PN+R fit into our classification. We show that PN+R are as expressive, on ω-languages, as PN+T. It is important to remark here that our construction requires ε-transitions.

Remember that, by Definition 2.22, any PN+R is a PN+T. Thus, $L^\omega(\text{PN+R}) \subseteq L^\omega(\text{PN+T})$. Let us exhibit a construction to prove that any ω-language accepted by a PN+T can also be accepted by a PN+R. We consider the PN+T $\mathcal{N}_t = \langle P, T, \Sigma, \mathbf{m}_0 \rangle$, and build the PN+R $\mathcal{N}_r = \langle P', T', \Sigma, \mathbf{m}'_0 \rangle$ as follows. Let $P' = P \uplus \{p_b, p_{Tr}\} \uplus \{p_t | t \in T_e\}$. Then for each transition $t = \langle I, O, s, d, +\infty, \lambda \rangle \in T_e$, we put three transitions in T': $t^s = \langle I \uplus \{p_b\}, \{p_t\}, \bot, \bot, 0, \lambda \rangle$; $t^c = \langle \{p_t, s\}, \{p_t, d\}, \bot, \bot, 0, \varepsilon \rangle$; and $t^e = \langle \{p_t\}, O \uplus \{p_b\}, s, p_{Tr}, +\infty, \varepsilon \rangle$. For any $t = \langle I, O, \bot, \bot, 0, \lambda \rangle \in T_r$, we add a transition $t' = \langle I \uplus \{p_b\}, O \uplus \{p_b\}, \bot, \bot, 0, \lambda \rangle$ in T'. Finally, $\forall p \in P : \mathbf{m}'_0(p) = \mathbf{m}_0(p)$, $\mathbf{m}'_0(p_b) = 1$, $\mathbf{m}'_0(p_{Tr}) = 0$ and $\forall t \in T_e : \mathbf{m}'_0(p_t) = 0$. Figure 7.3 illustrates the construction.

Let us now prove that the PN+R obtained thanks to this construction has the same ω-language as the PN+T it corresponds to.

Lemma 7.11 *Let $\mathcal{N}_t$ be a* PN+T *and let $\mathcal{N}_r$ be the* PN+R *obtained from $\mathcal{N}_t$. Then, $L^\omega(\mathcal{N}_r) = L^\omega(\mathcal{N}_t)$.*

Proof. Let us first show that $L^\omega(\mathcal{N}_t) \subseteq L^\omega(\mathcal{N}_r)$. Let $\sigma = t_1 t_2 \ldots$ be an infinite sequence of transitions of $\mathcal{N}_t$. Then, $\mathcal{N}_r$ accepts $\Lambda(\sigma)$ thanks to σ' built as follows. Let us assume that $\mathbf{m}_0$ is the initial marking of $\mathcal{N}_t$. For any $i \geq 1$, let $\mathbf{m}_i$ be the marking of $\mathcal{N}_t$ s.t. $\mathbf{m}_0 \xrightarrow{t_1 \cdots t_i} \mathbf{m}_i$. Then, σ' is equal to $\sigma_1 \cdot \sigma_2 \cdots$, where, for any $i \geq 1$, σ_i is obtained as follows:

1. If $t_i = t \in T_r$, then $\sigma_i = t'$;
2. If $t_i = t \in T_e$, then $\sigma_i = t^s (t^c)^{(\mathbf{m}_{i-1}(s) - I(s))} t^e$.

Clearly $\Lambda(\sigma_i) = \Lambda(t_i)$ for any $i \geq 1$ and their respective effects are equal on the places in P (remark that the reset arc has no effect and that the markings of p_b and p_t respectively are the same before and after the firing of each σ_i).

Then, let us show that $L^\omega(\mathcal{N}_r) \subseteq L^\omega(\mathcal{N}_t)$. Let $\sigma' = t'_1 t'_2 \ldots$ be an infinite sequence of transitions of $\mathcal{N}_r$ We first extract from σ' the subsequences $t^s(t^c)^n t^e (n \in \mathbb{N})$ that correspond to a given extended transition t in $\mathcal{N}_t$. Thus, $\sigma' = \sigma'_1 \cdot \sigma'_2 \cdots$, where σ'_i is either a single regular transition t' corresponding to the single regular transition $t \in T_r$ or a sequence σ_t corresponding to the extended transition $t \in T_e$. This is possible since the firing of t^s removes the token from p_b and block the whole net (except t^c and t^e). Hence no transitions can interleave with $t^s(t^c)^* t^e$. Moreover, σ' cannot have a suffix of the form $t^s(t^c)^\omega$ since t^c decreases the marking of the source place of the transfer of the corresponding transition t.

Then, we build the sequence $\sigma = t_1 t_2 \ldots$ of $\mathcal{N}_t$ as follows. For any $i \geq 1$:

1. If $|\sigma'_i| = 1$, then, it is the single transition t' that corresponds to $t \in T_r$. In that case, we let $t_i = t$.

2. If $|\sigma'_i| > 1$, then, it is a sequence σ_t that corresponds to a single $t \in T_e$. We let $t_i = t$.

Clearly, $\Lambda(\sigma) = \Lambda(\sigma')$. Let us now prove that σ is firable, i.e. $\mathbf{m}_0 \xrightarrow{t_1} \mathbf{m}_1 \xrightarrow{t_2} \mathbf{m}_2 \xrightarrow{t_3} \ldots$, by showing that $\forall i \geq 0 : \mathbf{m}'_{k_i} \preccurlyeq_P \mathbf{m}_i$.

Base case: $i = 0$. The base case is trivially verified.

Induction Step: $i = \ell$. By induction hypothesis, we have that $\forall 0 \leq i \leq \ell - 1 : \mathbf{m}'_{k_i} \preccurlyeq_P \mathbf{m}_i$. In the case where t_ℓ is a regular transition, it has the same effect on the places in P as $\sigma_\ell = t'_{k_\ell}$ and it can occur since $\mathbf{m}'_{k_{\ell-1}} \preccurlyeq_P \mathbf{m}_{\ell-1}$. Hence $\mathbf{m}'_{k_\ell} \preccurlyeq_P \mathbf{m}_\ell$, by monotonicity. Otherwise t_ℓ is an extended transition and its effect on P corresponds to the effect of σ_ℓ. Let us observe the effect of σ_ℓ: some tokens will be taken from s (the source place of the transfer) and put into d (the destination) by t^c_ℓ. Finally, the tokens remaining in s will be removed by the reset arc of t^e_ℓ. Hence, σ_ℓ removes the same number of tokens from s than t_ℓ, and cannot put more tokens in d than t_ℓ does. Moreover, the effect of σ_ℓ on the other places is the same than t_ℓ. Thus $\mathbf{m}'_{k_\ell} \preccurlyeq_P \mathbf{m}_\ell$. □

Theorem 7.4 PN+R *are as expressive as* PN+T *on infinite words, i.e.* $L^\omega(\mathsf{PN{+}R}) = L^\omega(\mathsf{PN{+}T})$.

Proof. As any PN+R is a special case of PN+T, we have that $L^\omega(\mathsf{PN{+}R}) \subseteq L^\omega(\mathsf{PN{+}T})$. The other direction stems from Lemma 7.11. □

In the case where we disallow ε-transitions, the previous construction doesn't allow to prove whether $L^\omega_{\not\varepsilon}(\mathsf{PN{+}T}) \subseteq L^\omega_{\not\varepsilon}(\mathsf{PN{+}R})$ or not. Indeed, this construction requests the use of several ε-transitions (namely, the t_c and t_e transitions used in the widget that replaces any extended transition t. See Figure 7.3). However, we have that $L^{\omega}(\mathsf{PN{+}NBA}) \subsetneq L^{\omega}(\mathsf{PN{+}R})$ and $L^{\omega}_{\not\epsilon}(\mathsf{PN{+}NBA}) \subsetneq L^{\omega}_{\not\epsilon}(\mathsf{PN{+}R})$, since the PN+T $\mathcal{N}_1$ we have used in the proof of Lemma 7.7 satisfies our definition of PN+R (in this case, the place p_4 is the trashcan) and has no ε-transitions.

7.4 Decidability of LTLSatis on PN+T and PN+R

Another consequence of the results that we have obtained in the present chapter is that LTLSatis is undecidable on PN+T and PN+R. Indeed, for any PN+NBA $\mathcal{N}$, one can build a PN+R $\mathcal{N}'$ that has the same ω-language. Since LTLSatis is undecidable on PN+NBA, it has to be undecidable on PN+R. The undecidability on PN+T follows from the fact that every PN+R is a PN+T.

Theorem 7.5 LTLSatis *is undecidable on* PN+T *and* PN+R.

Proof. Let φ be a formula of action-based LTL, and let $\mathcal{N}$ be a PN+NBA. By Theorem 7.1, there exists an PN+R $\mathcal{N}'$ s.t. $L^{\omega}(\mathcal{N}) = L^{\omega}(\mathcal{N}')$. Remark that there exists an effective procedure to build such a $\mathcal{N}'$ (it is described in Section 5.4.4). Hence $L^{\omega}(\mathcal{N}) \models \varphi$ iff $L^{\omega}(\mathcal{N}') \models \varphi$. Thus, if there were an algorithm to decide LTLSatis on PN+R, this construction would provide us with an algorithm to decide LTLSatis on PN+NBA. By Theorem 3.20, this is not possible. Hence LTLSatis is undecidable on PN+R. The undecidability on PN+T follows from the fact that $\mathcal{N}'$ is also a PN+T, by Definition 2.22. $\square$

7.5 Discussion

In the introduction of this chapter, we have recalled how important EPN are to study the *non-terminating* behaviour of concurrent systems made up of an arbitrary number of communicating processes (once abstracted thanks to predicate- and counting-abstraction techniques [BCR01]). Our aim was thus to study and classify the expressive powers of these models, as far as ω-languages are concerned. This goal has been fulfilled, as shown in the summary of our results in Figure 7.4. Indeed, we have proved in Section 7.1 that any ω-language accepted by a PN+NBA can be accepted by a PN+T, but that there exist ω-languages that are recognised by a PN+T but not by a PN+NBA. A similar result has been demonstrated for PN+NBA and PN in Section 7.2. These results hold with or without ε-transitions. Finally, in Section 7.3 we have drawn a link between these results and the class PN+R. Remark that the exact relationship between $L^{\omega}_{\not\epsilon}(\mathsf{PN{+}T})$ and $L^{\omega}_{\not\epsilon}(\mathsf{PN{+}R})$ still has to be investigated: it is clear that $L^{\omega}_{\not\epsilon}(\mathsf{PN{+}R}) \subseteq L^{\omega}_{\not\epsilon}(\mathsf{PN{+}T})$, by definition. However, we are not aware of

With ε-transitions:

$$L^\omega(\mathsf{PN}) \subsetneq L^\omega(\mathsf{PN{+}NBA}) \subsetneq \left\{ \begin{array}{c} L^\omega(\mathsf{PN{+}T}) \\ \| \\ L^\omega(\mathsf{PN{+}R}) \end{array} \right.$$

Without ε-transitions:

$$L^\omega_{\not\varepsilon}(\mathsf{PN}) \subsetneq L^\omega_{\not\varepsilon}(\mathsf{PN{+}NBA}) \subsetneq \left\{ \begin{array}{c} L^\omega_{\not\varepsilon}(\mathsf{PN{+}T}) \\ \rotatebox{90}{$\subseteq$} \\ L^\omega_{\not\varepsilon}(\mathsf{PN{+}R}) \end{array} \right.$$

Figure 7.4: Summary of the results on ω-languages.

a proof showing that $L^\omega_{\not\varepsilon}(\mathsf{PN{+}R}) \supseteq L^\omega_{\not\varepsilon}(\mathsf{PN{+}T})$, nor of an example of PN+T without ε-labelled transition whose language cannot be accepted by a PN+R without ε-labelled transition.

A stated in the introduction, these results find a natural continuation in the next chapter, where we study the expressiveness of PN, PN+NBA and PN+T, as well as WSTS in general, in terms of *finite words* languages.

Chapter 8

Well-structured languages

The present chapter is dedicated to the study of finite words languages of WSTS. As we have seen in Chapter 2, four main classes of languages of WSTS can be defined, depending on the form of the accepting sets of configurations. As a matter of fact, it is the class $L^G(\mathsf{WSTS})$ of *well-structured languages* (and its subclasses $L^G(\mathsf{PN})$, $L^G(\mathsf{PN{+}NBA})$ and $L^G(\mathsf{PN{+}T})$) that will retain our attention here. Section 8.1 motivates this choice, by showing that the class $L^G(\mathsf{WSTS})$ enjoys nice properties that do not necessarily hold on the classes $L^L(\mathsf{WSTS})$ and $L^T(\mathsf{WSTS})$. For instance, $L^G(\mathsf{WSTS})$ is closed under several operations (union, intersection, concatenation, and so forth), and one can decide whether the language of a given EWSTS is empty, for a given upward-closed set of accepting configurations. On the other hand, unfortunately, we prove that universality is undecidable on this class. Nevertheless, the positive results are strong enough to justify the study of $L^G(\mathsf{WSTS})$ in the first place.

This study begins in Section 8.2 with the statement of three *pumping lemmata* for WSTS, PN and PN+NBA. These lemmata say, roughly speaking, that, when a given language L is in $L^G(\mathsf{WSTS})$ (resp. $L^G(\mathsf{PN})$, $L^G(\mathsf{PN{+}NBA})$ and if this language contains an infinite set of words of a given form $\{w_1, w_2, \ldots, w_j, \ldots\}$, we can deduce that infinitely many other words – built upon $w_1, w_2, \ldots$ – also belong to L. The proofs of these lemmata seem to us interesting *per se*, because the arguments at work are direct exploitations of the monotonicity property of WSTS, and of the peculiar properties of WQO.

Then, Section 8.3 exploits these pumping lemmata to show several results about well-structured languages, and languages of PN, PN+NBA and PN+T. For instance, we prove that several well-studied languages, such as the language of all palindromes on a given alphabet, is not a WSL. Thanks to our pumping lemma on WSL, that proof is deceptively simple (particularly if we compare it to a similar proof in [Pet81]). Moreover, as in the case of infinite words, we strictly separate, thanks to our pumping lemmata, the expressive power of PN, PN+NBA and PN+T, when upward-closed accepting conditions are considered. All these results show the interest of the three pumping lemmata.

Finally, we close the chapter by proving several closure properties of WSL (and, more particularly, of $L^G(\mathsf{PN+T})$).

Remark that the results that concern the separation in the expressiveness of PN+T, PN+NBA and PN in the present chapter are very similar to those we have presented about ω-languages in the previous one. As a matter of fact, the proof techniques will be similar too, and we will sometimes re-use results of Chapter 7. For instance, the two EPN that we have used in the proofs of Chapter 7 to separate the expressive powers, will be used in this chapter for the same purpose (as far as finite words are concerned).

The content of this chapter is mainly based on the (submitted) article [GRVB06c]. Section 8.3.6 constitutes unpublished material.

8.1 Well-structured languages

This section defines the notion of *well-structured language* or WSL for short, and motivates this choice. As stated before, we will concentrate on the study of $L^G(\mathsf{WSTS})$ (sometimes more specifically on $L^G(\mathsf{EWSTS})$), because this class enjoys interesting properties that do not hold on $L^L(\mathsf{WSTS})$ nor on $L^T(\mathsf{WSTS})$. Moreover, the class $L^P(\mathsf{WSTS})$ is a (strict) subclass of $L^G(\mathsf{WSTS})$.

8.1.1 Positive results on $L^G(\mathsf{WSTS})$

Let us begin our argumentation in favour of the class $L^G(\mathsf{WSTS})$ with three positive results.

$\mathbf{L^G}(\mathsf{EWSTS}) \neq \mathbf{R.E.} = \mathbf{L^L}(\mathsf{EWSTS}) = \mathbf{L^T}(\mathsf{EWSTS})$ The first motivation comes from Proposition 3.6 that states that $L^L(\mathsf{EWSTS})$ and $L^T(\mathsf{EWSTS})$ are both equal to the set of recursively enumerable languages. Since this class of languages corresponds to the languages of Turing Machines, and since Rice's theorem states that only trivial properties of these languages can be decided, there is no hope that interesting properties are decidable on $L^L(\mathsf{EWSTS})$ and $L^T(\mathsf{EWSTS})$. On the other hand, we will prove in section 8.3.1 (see Proposition 8.4) that some CFL (context-free languages) are not in $L^G(\mathsf{EWSTS})$. This implies that $L^G(\mathsf{EWSTS}) \neq$ R.E., which is not surprising since the emptiness problem is decidable (see hereunder).

Emptiness is decidable. The second motivation is to be found in the fact that the emptiness problem is decidable on EWSTS. The fact that EmptyWsts is decidable for the class EWSTS, when C' is $\overline{\leq}$-upward-closed, can easily be deduced from the fact that CPWSTS is decidable on EWSTS:

Theorem 8.1 *The emptiness problem* EMPTYWSTS *is decidable for the class of* EWSTS, *when we consider* $\overline{\leq}$*–upward–closed accepting sets.*

Proof. From the definition of CPWSTS, it is not difficult to see that, given an EWSTS $\mathcal{S}$ and an upward-closed set U of configurations of $\mathcal{S}$, the language $L(\mathcal{S}, U) = \emptyset$ iff the answer to the coverability problem is *negative* on $\mathcal{S}$ and U. Since CPWSTS is decidable on the class EWSTS (see Theorem 3.4), this provides us with an effective procedure to test the emptiness of the language of an EWSTS when an upward–closed set of accepting configurations is considered. □

L^G(WSTS) **is a full** AFL **closed under intersection** The third motivation is that L^G(WSTS) is a full AFL closed under intersection, which is a strong indication that it is a class worth of attention. The proof consists in showing that, given two languages L_1 and L_2 in L^G(WSTS), there are WSTS that accept respectively $L_1 \cap L_2$, $L_1 \cup L_2$, $L_1 \cdot L_2$, L_1^+, $L_1 \cap L_R$ (where L_R is any regular language), $h(L_1)$ and $h^{-1}(L_1)$ (where h is any arbitrary homomorphism). Remark that we only prove here the *existence* of these WSTS, and these constructions are thus *not effective* in general, since we have not fixed any formalism to describe WSTS. However, we will present in section 8.3.4 effective constructions for these operations when the WSTS considered are PN+T.

In order to show that L^G(WSTS) is a full AFL closed under intersection, we first introduce a construction that turns any labelled WSTS $\mathcal{S}$ into another labelled WSTS $\mathcal{S}_s$ that accepts the same language as $\mathcal{S}$ does (for any set of accepting configurations) and that is *simply monotonic*:

Definition 8.1 (SIMPLY MONOTONIC LABELLED WSTS) *A labelled* WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq}, \Sigma \rangle$ *is* simply monotonic *iff for any* $c_1, c_2, c_3 \in C$, *for any* $a \in \Sigma \cup \{\varepsilon\}$*:* $c_1 \overset{a}{\Rightarrow} c_2$ *and* $c_1 \overline{\leq} c_3$ *implies that there exists* $c_4 \in C$ *s.t.t* $c_3 \overset{a}{\Rightarrow} c_4$ *and* $c_2 \overline{\leq} c_4$. ■

The construction works as follows. First, given a WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq}, \Sigma \rangle$, and a configuration $c \in C$, we let $\varepsilon\text{–closure}^{\Rightarrow}(c) = \{c' \mid c \overset{\varepsilon}{\Rightarrow}^* c'\}$. Remark that, for any $c \in C$: $c \in \varepsilon\text{–closure}^{\Rightarrow}(c)$. Then, for any WSTS $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq}, \Sigma \rangle$, we build the WSTS $\mathcal{S}_S = \langle C, c_0, \Rightarrow_s, \overline{\leq}, \Sigma \rangle$ s.t.:

$$\Rightarrow_s = \left\{ (c, a, c') \;\middle|\; \exists c_1, c_2 \in C : \begin{array}{c} c_1 \in \varepsilon\text{–closure}^{\Rightarrow}(c) \ \wedge \\ c_1 \overset{a}{\Rightarrow} c_2 \ \wedge \\ c' \in \varepsilon\text{–closure}^{\Rightarrow}(c_2) \end{array} \right\} \cup \{(c, \varepsilon, c) \mid c \in C\}$$

We can now show that this new transition relation enjoys the desired monotonicity property:

Lemma 8.1 *Let $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq}, \Sigma \rangle$ be a* WSTS *and let $\mathcal{S}_s = \langle C, c_0, \Rightarrow_s, \overline{\leq}, \Sigma \rangle$ be obtained from $\mathcal{S}$ by the above construction. Then, for any $c_1, c_2, c_3 \in C$, for any $a \in \Sigma \cup \{\varepsilon\}$: $c_1 \overline{\leq} c_3$ and $c_1 \overset{a}{\Rightarrow}_s c_2$ implies that there exists c_4 s.t. $c_2 \overline{\leq} c_4$ and $c_3 \overset{a}{\Rightarrow}_s c_4$.*

Proof. Let c_1, c_2, c_3 be three configurations of C and let $a \in \Sigma \cup \{\varepsilon\}$ be a letter s.t. $c_1 \overset{a}{\Rightarrow}_s c_2$ and $c_1 \overline{\leq} c_3$. Remark that, by definition, $c_1 \overset{a}{\Rightarrow}_s c_2$ implies that $c_1 \overset{a}{\Rightarrow}^* c_2$. Hence, by monotonicity of $\Rightarrow$, there exists c_4 s.t. $c_3 \overset{a}{\Rightarrow}^* c_4$ and $c_2 \overline{\leq} c_4$. By definition of $\Rightarrow^*$, and since a is a single character, this means either that $c_3 \overset{a}{\Rightarrow}_s c_4$, or that there are two configurations c and c' s.t. $c_3 \overset{\varepsilon}{\Rightarrow}^* c \overset{a}{\Rightarrow} c' \overset{\varepsilon}{\Rightarrow}^* c_4$. Hence, $c \in \varepsilon\text{−}\mathsf{closure}^{\Rightarrow}(c_3)$, $c_4 \in \varepsilon\text{−}\mathsf{closure}^{\Rightarrow}(c')$, and we conclude that $c_3 \overset{a}{\Rightarrow}_s c_4$. □

Thus, $\mathcal{S}_s$ is indeed a simply monotonic WSTS. Let us show that for any set of accepting configurations C', both $\mathcal{S}$ and $\mathcal{S}_s$ accept the same language.

Proposition 8.1 *Let $\mathcal{S} = \langle C, c_0, \Rightarrow, \overline{\leq}, \Sigma \rangle$ be a* WSTS *and let $\mathcal{S}_s = \langle C, c_0, \Rightarrow_s, \overline{\leq}, \Sigma \rangle$ be the simply monotonic* WSTS *obtained from $\mathcal{S}$. Then, for any $C' \subseteq C$: $L(\mathcal{S}, C') = L(\mathcal{S}_s, C')$.*

Proof. First remark that $\Rightarrow_s \subseteq \Rightarrow^*$. Hence, $L(\mathcal{S}, C') \subseteq L(\mathcal{S}_s, C')$. Let us show that $L(\mathcal{S}_s, C') \subseteq L(\mathcal{S}, C')$. Let us consider $w \in L(\mathcal{S}_s, C')$ and let us show that $w \in L(\mathcal{S}, C')$.

By definition of L, there is $c \in C'$ s.t. $c_0 \overset{w}{\Rightarrow}_s^* c$. Hence, by definition of $\Rightarrow_s^*$ there is $k \geq 1$ s.t. there are $c_1, c_2, \ldots, c_k \in C$ and $b_1, b_2, \ldots b_{k-1} \in \Sigma \cup \{\varepsilon\}$ with $c_1 = c_0$, $c_k = c$, $b_1 \cdot b_2 \cdots b_{k-1} = w$ and $c_1 \overset{b_1}{\Rightarrow}_s c_2 \overset{b_2}{\Rightarrow}_s \cdots \overset{b_{k-1}}{\Rightarrow}_s c_k$. Without loss of generality, we assume that there is no $1 \leq i \leq k-1$ s.t. $c_i = c_{i+1}$ and $b_i = \varepsilon$. Indeed, if such transitions appear in the sequence, they can be removed because they do not add any character to the words, and are not necessary to reach C'.

By definition of $\Rightarrow_s$, there are $\overline{c}_1, \overline{c}_2, \ldots \overline{c}_{k-1}$ and $\widehat{c}_1, \widehat{c}_2, \ldots, \widehat{c}_{k-1}$ in C s.t.:

$$c_1 \overset{\varepsilon}{\Rightarrow}^* \overline{c}_1 \overset{b_1}{\Rightarrow} \widehat{c}_1 \overset{\varepsilon}{\Rightarrow}^* c_2 \overset{\varepsilon}{\Rightarrow}^* \overline{c}_2 \overset{b_2}{\Rightarrow} \widehat{c}_2 \overset{\varepsilon}{\Rightarrow}^* \cdots \overset{\varepsilon}{\Rightarrow}^* \overline{c}_{k-1} \overset{b_{k-1}}{\Rightarrow} \widehat{c}_{k-1} \overset{\varepsilon}{\Rightarrow}^* c_k$$

Since $c_k \in C'$, this implies that

$$\varepsilon \cdot b_1 \cdot \varepsilon \cdot \varepsilon \cdot b_2 \cdot \varepsilon \cdots \varepsilon \cdot b_{k-1} \cdot \varepsilon = b_1 \cdot b_2 \cdots b_{k-1} = w \in L(\mathcal{S}, C')$$

□

Theorem 8.2 *L^G(*WSTS*) is a full* AFL*, closed under intersection.*

Proof. According to Definition 2.36, one has to show seven closure properties (the six properties that define an AFL, plus the closure under intersection) in order to establish this result. In the sequel, we assume that $\mathcal{S}_1 = \langle C_1, i_1, \Sigma_1, \Rightarrow_1, \overline{\leq}_1 \rangle$ and $\mathcal{S}_2 = \langle C_2, i_2, \Sigma_2, \Rightarrow_2, \overline{\leq}_2 \rangle$ are two WSTS (with $C_1 \cap C_2 = \emptyset$), and that U_1 and U_2 are their associated upward-closed sets of accepting states. In order to make the proofs easier, we further assume that both $\mathcal{S}_1$ and $\mathcal{S}_2$ are simply monotonic. According to Proposition 8.1, this is not restrictive since, for any labelled WSTS $\mathcal{S}$, there exists a simply monotonic WSTS $\mathcal{S}_s$ that accepts the same language. We finally assume that $h : \Sigma_1 \mapsto \Sigma_1^*$ is a homomorphism s.t. $h(\varepsilon) = \varepsilon$, according to the definition from [Gin75, Sal73]. We prove the closure of the seven operations by showing the existence of a WSTS $\mathcal{S} = \left\langle C, i, \Sigma, \Rightarrow, \overline{\leq} \right\rangle$ and a set of accepting states U, s.t. $L(\mathcal{S}, U)$ is the result of the operation in question. We ensure that $L(\mathcal{S}, U)$ is a WSL by proving that $\overline{\leq}$ is a WQO, $\Rightarrow$ is $\overline{\leq}$-monotonic and U is upward-closed.

Intersection Let us show that there are $\mathcal{S}$ and U s.t. $L(\mathcal{S}, U) = L(\mathcal{S}_1, U_1) \cap L(\mathcal{S}_2, U_2)$. $\mathcal{S}$ is built as follows: $C = C_1 \times C_2$; $i = (i_1, i_2)$; $\Sigma = \Sigma_1 \cap \Sigma_2$; $\overline{\leq} = \{\big((c_1, c_2), (c_1', c_2')\big) \mid c_1 \overline{\leq}_1 c_1' \wedge c_2 \overline{\leq}_2 c_2'\}$. The transition relation $\Rightarrow$ is defined as:

$$\begin{aligned} \Rightarrow \; = \; & \{\big((c_1, c_2), a, (c_1', c_2')\big) \mid c_1 \overset{a}{\Rightarrow}_1 c_1' \wedge \; c_2 \overset{a}{\Rightarrow}_2 c_2' \wedge a \in \Sigma\} \cup \\ & \{\big((c_1, c_2), \varepsilon, (c_1', c_2')\big) \mid \big(c_1 \overset{\varepsilon}{\Rightarrow}_1 c_1' \wedge c_2 = c_2'\big) \text{ or } \big(c_1 = c_1' \wedge c_2 \overset{\varepsilon}{\Rightarrow}_2 c_2'\big)\} \end{aligned}$$

Finally, $U = \{(c_1, c_2) \mid c_1 \in U_1 \wedge c_2 \in U_2\}$.

Clearly, $L(\mathcal{S}, U) = L(\mathcal{S}_1, U_1) \cap L(\mathcal{S}_2, U_2)$. Let us prove that $\overline{\leq}$, $\Rightarrow$ and U have the desired properties:

- **$\overline{\leq}$ is a WQO** Let $\varsigma = (c_1^1, c_1^2), (c_2^1, c_2^2), \ldots, (c_n^1, c_n^2), \ldots$ be an infinite sequence of elements of C. Since $\overline{\leq}_1$ is a WQO on C_1, following Lemma 2.2, one can extract from ς an infinite subsequence

$$\varsigma' = (c_{\rho(1)}^1, c_{\rho(1)}^2), (c_{\rho(2)}^1, c_{\rho(2)}^2), \ldots, (c_{\rho(n)}^1, c_{\rho(n)}^2), \ldots$$

  such that for any $j \geq 1$: $c_{\rho(j)}^1 \overline{\leq}_1 c_{\rho(j+1)}^1$. Since $\overline{\leq}_2$ is a WQO on the elements of C_2, there are, in ς', two positions k and ℓ s.t. $k < \ell$ and $c_{\rho(k)}^2 \overline{\leq}_2 c_{\rho(\ell)}^2$. Hence, $(c_{\rho(k)}^1, c_{\rho(k)}^2) \overline{\leq} (c_{\rho(\ell)}^1, c_{\rho(\ell)}^2)$, which proves that $\overline{\leq}$ is a WQO, according to Definition 2.4.

- **$\Rightarrow$ is $\overline{\leq}$-monotonic** Let (c_1^1, c_1^2), (c_2^1, c_2^2), and (c_3^1, c_3^2) be three configurations of C. We consider two cases. Either there is $a \in \Sigma$ s.t. $(c_1^1, c_1^2) \overset{a}{\Rightarrow} (c_2^1, c_2^2)$ and $(c_1^1, c_1^2) \overline{\leq} (c_3^1, c_3^2)$. By definition of $\Rightarrow$ and $\overline{\leq}$, this implies that $c_1^1 \overset{a}{\Rightarrow}_1 c_2^1$, $c_1^2 \overset{a}{\Rightarrow}_2 c_2^2$, $c_1^1 \overline{\leq}_1 c_3^1$ and $c_1^2 \overline{\leq}_2 c_3^2$. Since $\Rightarrow_1$ and $\Rightarrow_2$ are resp. $\overline{\leq}_1$- and $\overline{\leq}_2$- simply monotonic, there are $c \in C_1$ and $c' \in C_2$ s.t.: $c_3^1 \overset{a}{\Rightarrow}_1 c$, $c_3^2 \overset{a}{\Rightarrow}_2 c'$, $c_2^1 \overline{\leq}_1 c$ and $c_2^2 \overline{\leq}_2 c'$. The first two point imply that $(c_3^1, c_3^2) \overset{a}{\Rightarrow} (c, c')$. The last two points imply that $(c_2^1, c_2^2) \overline{\leq} (c, c')$.

  On the other hand, if $(c_1^1, c_1^2) \overset{\varepsilon}{\Rightarrow} (c_2^1, c_2^2)$ then either (i) $c_1^1 \overset{\varepsilon}{\Rightarrow}_1 c_2^1$ and $c_1^2 = c_2^2$ or (ii) $c_1^2 \overset{\varepsilon}{\Rightarrow}_1 c_2^2$ and $c_1^1 = c_2^1$. In the first case, since $\Rightarrow_1$ is simply monotonic, and since $c_1^1 \overline{\leq}_1 c_3^1$, there exists c_4^1 s.t. $c_3^1 \overset{\varepsilon}{\Rightarrow}_1 c_4^1$ and $c_2^1 \overline{\leq} c_4^1$. Thus, $(c_4^1, c_1^2) \overline{\leq} (c_3^1 c_1^2)$ by definition of $\overline{\leq}$ and $(c_3^1, c_1^2) \overset{\varepsilon}{\Rightarrow} (c_4^1, c_1^2)$ by definition of $\Rightarrow$. The second case is similar.

- *U* **is $\overline{\leq}$-upward-closed** Let (c_1^1, c_1^2) and (c_2^1, c_2^2), both in C, be s.t. $(c_1^1, c_1^2)\overline{\leq}(c_2^1, c_2^2)$ and $(c_1^1, c_1^2) \in U$. Let us show that $(c_2^1, c_2^2) \in U$ too. Since $(c_1^1, c_1^2) \in U$, we have $c_1^1 \in U_1$ and $c_1^2 \in U_2$, by definition of U. Since $(c_1^1, c_1^2)\overline{\leq}(c_2^1, c_2^2)$, $c_1^1\overline{\leq}_1 c_2^1$ and $c_1^2\overline{\leq}_2 c_2^2$, by definition of $\overline{\leq}$. But U_1 and U_2 are resp. $\overline{\leq}_1$- and $\overline{\leq}_2$-upward-closed, which implies that $c_2^1 \in U_1$ and $c_2^2 \in U_2$. Hence $(c_2^1, c_2^2) \in U$.

Union Let us show that there are $\mathcal{S}$ and U such that $L(\mathcal{S}, U) = L(\mathcal{S}_1, U_1) \cup L(\mathcal{S}_2, U_2)$. We let $C = \{i\} \uplus C_1 \uplus C_2$; $\Sigma = \Sigma_1 \cup \Sigma_2$; $\overline{\leq} = \overline{\leq}_1 \cup \overline{\leq}_2 \cup \{(i, i)\}$; $U = U_1 \cup U_2$ and $\Rightarrow = \{(i, \varepsilon, i_1), (i, \varepsilon, i_2)\} \cup \Rightarrow_1 \cup \Rightarrow_2$.

Clearly, $L(\mathcal{S}, U) = L(\mathcal{S}_1, U_1) \cup L(\mathcal{S}_2, U_2)$. Let us show that $\mathcal{S}$ has the desired properties. By definition, $\Rightarrow$ is $\overline{\leq}$-monotonic (remark that i is $\overline{\leq}$-incomparable to any other element of C). Thus, it remains to prove that:

- **$\overline{\leq}$ is a WQO** Let $\varsigma = c_0, c_2, \ldots, c_n, \ldots$ be an infinite sequence of elements of C. Because it is infinite, one can extract, from that sequence, an infinite subsequence $\varsigma' = c_{j_1}, c_{j_2}, c_{j_3}, \ldots$, s.t. either $\forall k \geq 1 : c_{j_k} \in C_1$ or $\forall k \geq 1 : c_{j_k} \in C_2$ or $\forall k \geq 1 : c_{j_k} = i$. In the case where $\forall k \geq 1 : c_{j_k} = i$, there are clearly two positions $k < \ell$ s.t. $c_{j_k}\overline{\leq}c_{j_\ell}$, since $i\overline{\leq}i$. Otherwise, since $\overline{\leq}_1$ and $\overline{\leq}_2$ are both WQO, there exist two positions k and ℓ s.t. $k < \ell$ and either $c_{j_k}\overline{\leq}_1 c_{j_\ell}$ or $c_{j_k}\overline{\leq}_2 c_{j_\ell}$. In either cases, this implies that $c_{j_k}\overline{\leq}c_{j_\ell}$, which proves that $\overline{\leq}$ is a WQO following Definition 2.4.

- *U* **is $\overline{\leq}$-upward-closed** Let c_1, c_2 be two configurations in C s.t. $c_1 \in U$ and $c_1\overline{\leq}c_2$. Let us show that $c_2 \in U$. We consider two cases: either $c_1 \in U_1$ or $c_1 \in U_2$. In the former case, since c_1 and c_2 are $\overline{\leq}$-comparable, we deduce that $c_2 \in C_1$ and thus, $c_1\overline{\leq}_1 c_2$, by definition of $\overline{\leq}$. Hence, $c_2 \in U_1$, since U_1 is $\overline{\leq}_1$-upward-closed. This implies that $c_2 \in U$. The same reasoning can be applied to the latter case.

Concatenation Let us show that there are $\mathcal{S}$ and U such that $L(\mathcal{S}, U) = L(\mathcal{S}_1, U_1) \cdot L(\mathcal{S}_2, U_2)$. We let $C = C_1 \cup C_2$; $i = i_1$; $\Sigma = \Sigma_1 \cup \Sigma_2$; $\Rightarrow = \{(c, \varepsilon, i_2) \mid c \in U_1\} \cup \Rightarrow_2 \cup \Rightarrow_1$; $\overline{\leq} = \overline{\leq}_1 \cup \overline{\leq}_2$ and $U = U_2$.

Clearly, $L(\mathcal{S}, U)$ is the concatenation of $L(\mathcal{S}_1, U_1)$ and $L(\mathcal{S}_2, U_2)$. The transition relation $\Rightarrow$ is $\overline{\leq}$-monotonic from its definition. Indeed, let c_1, c_2 and c_3 be three configurations from C and $a \in \Sigma \cup \{\varepsilon\}$ be a character s.t. $c_1 \overset{a}{\Rightarrow} c_2$ and $c_1\overline{\leq}c_3$. In the case where $\{c_1, c_2, c_3\} \subseteq C_1$ or $\{c_1, c_2, c_3\} \subseteq C_2$, there exists $c_4\overline{\geq}c_2$ in $C = C_1 \cup C_2$ s.t. $c_3 \overset{a}{\Rightarrow} c_4$, by monotonicity of $\Rightarrow_1$ and $\Rightarrow_2$. In the case where $c_1 \in C_1$ and $c_2 \in C_2$, we have $c_1 \in U_1$, $c_2 = i_2$ and $a = \varepsilon$, by construction. Hence, $c_3 \in U$ and $c_3 \overset{\varepsilon}{\Rightarrow} i_2$ by construction again. Remark that it is not possible that $c_1 \in C_2$ and $c_2 \in C_1$. Since $U_2 = U$ is $\overline{\leq}_2$-upward-closed, $\overline{\leq} = \overline{\leq}_1 \cup \overline{\leq}_2$ and $C_1 \cap C_2 = \emptyset$, we conclude that U is $\overline{\leq}$-upward-closed. Finally, one can show that $\overline{\leq}$ is a WQO by reusing the same reasoning as for the union.

Iteration Let us show that there are $\mathcal{S}$ such that $L(\mathcal{S}, U) = L(\mathcal{S}_1, U_1)^+$. We consider a new configuration $i_0 \notin C_1$ and let $C = C_1 \cup \{i_0\}$; $i = i_0$; $\overline{\leq} = \overline{\leq}_1 \cup \{(i_0, i_0)\}$; $\Rightarrow = \{(i_0, \varepsilon, i_1)\} \cup \{(c, \varepsilon, i_0) \mid c \in U_1\} \cup \Rightarrow_1$ and $U = U_1$.

From these definitions, it is trivial to see that $L(\mathcal{S}, U) = L(\mathcal{S}_1, U_1)^+$, $\overline{\leq}$ is a WQO, $\Rightarrow$ is $\overline{\leq}$-monotonic, and U is $\overline{\leq}$-upward-closed.

Intersection with regular languages It is not difficult to see that any deterministic finite-state automaton is a WSTS, when we choose the equality between states as WQO. Hence, any regular language is a WSL. Since WSL are closed under intersection (see above), the closure with regular languages holds too.

Arbitrary homomorphism Let us show that there are S and U such that $L(\mathcal{S}, U) = h\big(L(\mathcal{S}_1, U_1)\big)$. We extend the set of states C_1 with elements from $C_1 \times \Sigma \times \mathbb{N}$ in the following way: $C = C_1 \uplus \{(c, a, j) \mid c \in C_1 \wedge a \in \Sigma \cup \{\varepsilon\} \wedge 0 \leq j \leq |h(a)| \wedge \exists c' : c \overset{a}{\Rightarrow}_1 c'\}$. Intuitively, these extra states are the intermediate states that have to appear along the path from c to c' when reading $h(a)$. More precisely, (c, a, j) is the state reached after having read the j first characters of $h(a)$ from c. We also let $i = i_1$ and $\overline{\leq} = \overline{\leq}_1 \cup \{\big((c_1, a, j), (c_2, a, j)\big) \mid (c_1, a, j), (c_2, a, j) \in C \wedge c_1 \overline{\leq}_1 c_2\}$. The transition relation is built according to the intuition we have sketched when introducing C:

$$\Rightarrow = \left\{ \begin{array}{l|c} \big(c, \varepsilon, (c, a, 0)\big), & \\ \big((c, a, 0), w_1, (c, a, 1)\big), & a \in \Sigma \cup \{\varepsilon\} : \\ \quad\vdots & c \overset{a}{\Rightarrow}_1 c' \text{ and} \\ \big((c, a, |h(a)| - 1), w_{|h(a)|}, (c, a, |h(a)|)\big) & h(a) = w_1 w_2 \dots w_{|h(a)|} \\ \big((c, a, |h(a)|), \varepsilon, c'\big) & \end{array} \right\}$$

Finally, $U = U_1$.

By construction, $L(\mathcal{S}, U) = h\big(L(\mathcal{S}_1, U_1)\big)$, and U is a $\overline{\leq}$-upward-closed set. It remains to show that:

- $\overline{\leq}$ **is a** WQO Let us suppose it is not the case. Then, there exists a sequence of elements of C: $\varsigma = c_1, c_2, \dots, c_n, \dots$ s.t. for any $k \geq 1$, for any $1 \leq n < k$: $c_n \overline{\not\leq} c_k$ (each configuration is $\overline{\leq}$-incomparable to all the previous ones). Remark that, since $\overline{\leq}_1$ is a WQO on the elements of C_1 and since $c \overline{\leq}_1 c'$ implies $c \overline{\leq} c'$ (by definition of $\overline{\leq}$), one cannot find in ς, infinitely many elements from C_1. Otherwise, the infinite subsequence of ς made of all the elements $c_i \in C_1$ would be an infinite sequence of $\overline{\leq}_1$-incomparable elements from C_1. But this cannot exist since $\overline{\leq}_1$ is a WQO. Thus, there is, in ς, an infinite subsequence $\varsigma' = c_{j_1}, c_{j_2}, \dots c_{j_n}, \dots$ s.t. for any $k \geq 1$: *(i)* $c_{j_k} \notin C_1$ and *(ii)* for any $1 \overline{\leq} n < k$: $c_{j_n} \overline{\not\leq} c_{j_k}$.

  By definition of a homomorphism, the value $\ell = \max_{a \in \Sigma \cup \{\varepsilon\}}\{|h(a)|\}$ is a finite value. Hence, there exists $0 \leq \ell' \leq \ell$ and a character a of $\Sigma \cup \{\varepsilon\}$ s.t. the sequence $(c_{j_1}, a, \ell'), (c_{j_2}, a, \ell'), \dots, (c_{j_n}, a, \ell'), \dots$ is an infinite subsequence of ς' and for any

$n < k$: $(c_{j_n}, a, \ell')\overline{\not\preceq}(c_{j_k}, a, \ell')$. However, this implies that for any $n < k$: $c_{j_n}\overline{\not\preceq}_1 c_{j_k}$, which contradicts the fact that $\overline{\preceq}_1$ is a WQO.

- **$\Rightarrow$ is $\overline{\preceq}$-monotonic** Let us show that, for any $c_1, c_2, c_3 \in C$, and for any $a \in \Sigma$ s.t. $c_1 \overset{a}{\Rightarrow} c_2$ and $c_1\overline{\preceq}c_3$, there exists c_4 s.t. $c_3 \overset{a}{\Rightarrow}^* c_4$ and $c_2\overline{\preceq}c_4$. We consider two cases.

  1. Either $c_1 \in C_1$. In that case, by definition of $\Rightarrow$, we have $a = \varepsilon$ and $c_2 = (c_1, b, 0)$ for some b. By construction, there is thus $c_1' \in C_1$ s.t. $c_1 \overset{b}{\Rightarrow}_1 c_1'$. Moreover, $c_1\overline{\preceq}c_3$ implies that $c_3 \in C_1$, and thus that, $c_1\overline{\preceq}_1 c_3$. Since $\Rightarrow_1$ is $\overline{\preceq}_1$-simply monotonic, there is c_3' s.t. $c_3 \overset{b}{\Rightarrow}_1 c_3'$. Hence, by construction, the configuration $c_4 = (c_3, b, 0)\overline{\succeq}(c_1, b, 0)$ is in C, and satisfies $c_3 \overset{\varepsilon}{\Rightarrow} c_4$.
  2. Or, $c_1 \notin C_1$. In that case $c_1 = (c', b, i)$ and $c_3 = (c'', b, i)$ with $c'\overline{\preceq}_1 c''$, for some b. Again, we have to consider two subcases.
     (a) In the case where $i < |h(b)|$, $c_2 = (c', b, i + 1)$, by construction. We can choose $c_4 = (c'', b, i + 1)$, which satisfies the conditions.
     (b) In the case where $i = |h(b)|$, c_2 is a configuration of C_1 s.t. $c' \overset{b}{\Rightarrow}_1 c_2$. By definition of $\Rightarrow$, we have: $c_1 = (c', b, |h(b)|) \overset{\varepsilon}{\Rightarrow} c_2$. By $\overline{\preceq}_1$-simple monotonicity of $\Rightarrow_1$, there exists a configuration c_4 s.t. $c'' \overset{b}{\Rightarrow}_1 c_4$ and $c_2\overline{\preceq}_1 c_4$. Thus, $c_2\overline{\preceq}c_4$, and, by definition of $\Rightarrow$, we have $c_3 = (c'', b, |h(b)|) \overset{\varepsilon}{\Rightarrow} c_4$. Hence, c_4 satisfies the conditions.

  In any case, we conclude that $\Rightarrow$ is $\overline{\preceq}$-monotonic.

Inverse homomorphism Let us build $\mathcal{S}$ and U s.t. $L(\mathcal{S}, U) = h^{-1}\big(L(\mathcal{S}_1, U_1)\big)$. We let $C = C_1$; $i = i_1$; $\overline{\preceq} = \overline{\preceq}_1$; $\Rightarrow = \{(c_1, a, c_2) \mid a \in \Sigma \cup \{\varepsilon\} \wedge \exists w \in \Sigma^* : h(a) = w \wedge c_1 \overset{w}{\Rightarrow}_1^* c_2\}$ and $U = U_1$.

Clearly, $L(S, U) = h^{-1}\big(L(S_1, U_1)\big)$. By definition, U is $\overline{\preceq}$-upward-closed and $\overline{\preceq}$ is a WQO. It remains to show that $\Rightarrow$ is $\overline{\preceq}$-monotonic. Let c_1, c_2, c_3 be three configurations in C s.t. $c_1 \overset{a}{\Rightarrow} c_2$ for some a, and $c_1\overline{\preceq}c_3$. By definition of $\Rightarrow$, there exists $w \in \Sigma^*$ s.t. $h(a) = w$ and $c_1 \overset{w}{\Rightarrow}_1^* c_2$. Moreover, $c_3 \in C_1$ and $c_1\overline{\preceq}_1 c_3$, by definition. By using an inductive reasoning on the length of w, one can show that there exists $c_4 \in C_1$ s.t. $c_3 \overset{w}{\Rightarrow}_1^* c_4$ and $c_2\overline{\preceq}_1 c_4$. Hence, $c_4 \in C$ and $c_3 \overset{a}{\Rightarrow} c_4$, by definition of $\Rightarrow$. □

Remark 8.1 L^P(WSTS) *is not a full* AFL. *Indeed, let us consider the alphabet* $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. *Clearly, the language* $\mathcal{L}_\mathtt{a} = \{\mathtt{a}, \varepsilon\}$ *is in* L^P(WSTS). *Let* $h : \Sigma \mapsto \Sigma^*$ *be an homomorphism s.t.* $h(\mathtt{a}) = \mathtt{bb}$. *Then,* $h(\mathcal{L}_\mathtt{a}) = \{(\mathtt{bb}), \varepsilon\}$ *is not in* L^P(WSTS) *because it is not prefix-closed (the word* b *is missing).*

8.1.2 Negative result: undecidability of universality

Unfortunately, the universality problem is undecidable on EWSTS, even when we consider upward-closed sets of accepting configurations. The proof consists in showing that the universality problem is undecidable on PN+NBA. In order to prove the undecidability of universality for PN+NBA, we reduce PBEPN in the case where the EPN is a PN+NBA, to this problem. Since PBEPN is undecidable on the class PN+NBA (see Theorem 3.13), and since any PN+NBA defines an EWSTS, we obtain the result.

Given a PN+NBA $\mathcal{N} = \langle P, T, \Sigma, \mathbf{m}_0 \rangle$ and a place $p \in P$, the reduction consists in building a new PN+NBA $\mathcal{N}_p = \langle P', T', \{\mathsf{a}\}, \mathbf{m}'_0 \rangle$ s.t. $\mathcal{N}_p$ accepts (with $\mathbb{N}^{|P'|}$ as accepting set) the universal language (i.e., a^*) if and only if the place p is unbounded in $\mathcal{N}$. The construction works as follows:

- $P' = P \uplus \{run, stop\}$;
- $T' = \{\langle I \cup \{run\}, O \cup \{run\}, s, d, b, \varepsilon \rangle \mid \exists \lambda : \langle I, O, s, d, b, \lambda \rangle \in T\} \cup \{t_a, t_f\}$, where: $t_a = \langle \{stop, p\}, \{stop\}, \perp, \perp, 0, \mathsf{a} \rangle$ and $t_f = \langle \{run\}, \{stop\}, \perp, \perp, 0, \varepsilon \rangle$.
- $\mathbf{m}'_0(run) = 1$, $\mathbf{m}'_0(stop) = 0$ and $\forall p' \in P : \mathbf{m}'_0(p') = \mathbf{m}_0(p')$.

In other words, $\mathcal{N}_p$ is similar to $\mathcal{N}$ except that its transitions (but t_a) are labelled by ε and may fire only if the place run is marked. Besides this, the transitions of $\mathcal{N}_p$ that have been adapted from transitions of $\mathcal{N}$ have the same effect in $\mathcal{N}_p$ than in $\mathcal{N}$. The transition t_f moves the unique token from run to $stop$. This has the effect to prevent the transitions in $T' \setminus \{t_a\}$ from firing further. Hence, t_a only (labelled by a) can be fired after t_f has been fired. Since t_a consumes one token from place p, that transition can be fired at most k times where k is the number of tokens in p when firing t_f.

The following lemma states that the the construction we have just introduced is correct:

Lemma 8.2 *Let $\mathcal{N} = \langle P, T, \Sigma, \mathbf{m}_0 \rangle$ bet a* PN+NBA *and let $p \in P$ be a place of $\mathcal{N}$. Let us assume that $\mathcal{N}_p = \langle P', T', \{\mathsf{a}\}, \mathbf{m}'_0 \rangle$ is the* PN+NBA *obtained from $\mathcal{N}$ and p as described above. Then, the place boundedness problem for $\mathcal{N}$ and p has a negative answer iff $L^G(\mathcal{N}_p, \mathbb{N}^{|P'|}) = \mathsf{a}^*$.*

Proof. If the place-boundedness problem has a negative answer for $\mathcal{N}$ and p, then, for any $k \in \mathbb{N}$, there is a sequence of transitions σ s.t. $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}$ with $\mathbf{m}(p) \geq k$. Let σ' be the sequence of transitions of $\mathcal{N}_p$ obtained by replacing in σ each transition by its corresponding transition in $\mathcal{N}_p$. Let $\mathbf{m}'$ be the marking s.t. $\mathbf{m}'_0 \xrightarrow{\sigma'} \mathbf{m}'$ in $\mathcal{N}_p$. By construction, we have: $\mathbf{m}'(run) = 1$ and for any $p \in P$: $\mathbf{m}'(p) = \mathbf{m}(p)$. In particular, this implies that $\mathbf{m}'(p) \geq k$. Hence, the sequence $t_f t_a^k$ is firable from $\mathbf{m}'$. Since the accepting upward-closed set is $\mathbb{N}^{|P'|}$, the sequence $\sigma' t_f t_a^k$ is accepting, with

$\Lambda(\sigma' t_f t_a^k) = \mathtt{a}^k$. This holds for any $k \in \mathbb{N}$, and we conclude that $\mathcal{N}_p$ accepts $\mathtt{a}^*$, and is thus universal because the alphabet of $\mathcal{N}_p$ is $\{\mathtt{a}\}$.

On the other hand, if $\mathcal{N}_p$ accepts $\mathtt{a}^*$, then, for any $k \in \mathbb{N}$, there exists a sequence of transitions σ' s.t. $\sigma' t_f t_a^k$ is firable from $\mathbf{m}'_0$. This holds because t_a is the only transition of $\mathcal{N}_p$ that is labelled by $\mathtt{a}$, and because t_f has to be fired before t_a can fire. Moreover, no ε-labelled transition can be fired once t_f has fired because it removes the token from place *run*. Let $\mathbf{m}'$ be the marking s.t. $\mathbf{m}'_0 \xrightarrow{\sigma'} \mathbf{m}'$. Clearly, $\mathbf{m}'(run) = 1$ and $\mathbf{m}'(p) \geq k$. Hence, the sequence σ' contains transitions from $T' \setminus \{t_f, t_a\}$ only. Thus, by construction of $\mathcal{N}_p$, there exists a sequence of transitions σ of $\mathcal{N}$ s.t. $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}$ with $\mathbf{m}(p) \geq k$ (σ is obtained from σ' by replacing each transition t' of $\mathcal{N}_p$ by its corresponding transitions t in $\mathcal{N}$). Since this is true for any $k \in \mathbb{N}$, it implies that p is *unbounded* in $\mathcal{N}$. □

This allows us to obtain the following proposition:

Proposition 8.2 *The universality problem for* PN+NBA *is undecidable, when we consider upward-closed sets of accepting configurations.*

Proof. From Theorem 3.13, we know that the place boundedness problem is undecidable for PN+NBA. Lemma 8.2 reduces this problem to the universality problem. Hence, the latter is undecidable. □

Since any PN+NBA defines an EWSTS, we immediately obtain:

Theorem 8.3 UnivEWsts *is undecidable when we consider upward-closed sets of accepting configurations.*

8.1.3 Well-structured languages

It should now be clear that the class $L^G(\mathsf{WSTS})$ enjoys interesting properties: the emptiness is decidable on this class, under reasonable effectiveness assumptions (Theorem 8.1), and it forms a full AFL closed under intersection (Theorem 8.2). Moreover, the transition relation of WSTS is, by definition, $\leq$-monotonic. Thus, $\leq$-upward-closed sets are perfectly suited accepting conditions for these systems. The only drawback we could identify on this class of languages is that universality is undecidable. Nevertheless, we believe that the positive results are sufficient to restrict ourselves to the study of $L^G(\mathsf{WSTS})$. We call these languages the *well-structured languages*:

Definition 8.2 (Well-Structured Language) *L is a* well-structured language *(*WSL *for short) iff* $L \in L^G(\mathsf{WSTS})$. ■

In the rest of this chapter, we will show that WSL are indeed a class of languages worth of interest by proving several non-trivial properties for them. In particular, we will prove in Section 8.2 three pumping lemmata, that we will apply in Section 8.3 to obtain many properties and results about WSL and their subclasses.

8.2 Pumping lemmata

This section presents three lemmata that show the limitations in the expressiveness of WSTS (for the first one), PN (for the second one), and PN+NBA (for the third one). All these lemmata have a similar statement: if a given WSTS (resp. PN, PN+NBA) accepts an infinite set of words $\{w_1, w_2, \ldots\}$ with a given structure, then it must also accept other words that are built upon the words $w_1, w_2, \ldots$ In some sense, these lemmata allow to “inflate” the set of accepted words. For that reason, we have chosen to call them *pumping lemmata*, owing to their similarities to the classical pumping lemmata for regular and context-free languages (see for instance [HMU01]).

The proof techniques rely on properties of infinite sequences of configurations (equipped with a WQO), and monotonicity properties. The usefulness of these pumping lemmata will be demonstrated in Section 8.3, where we apply them to obtain several results about WSL.

8.2.1 A pumping lemma for WSL

Our first pumping lemma deals with WSL, and is very easy to prove:

Lemma 8.3 *Let L be a* WSL*, and let $w_1, w_2, \ldots$ be an infinite sequence of words s.t. $\forall k \geq 1$: $w_k = B_k \cdot E_k \in L$. Then, there exist $i < j$ s.t. $B_j \cdot E_i \in L$.*

Proof. Let $S = \left\langle C, c_0, \Sigma, \Rightarrow, \overline{\leq} \right\rangle$ be a WSTS s.t. $L(S, \mathcal{U}) = L$ for some $\overline{\leq}$-upward-closed set $\mathcal{U}$. For any $k \geq 1$, let $c_k \in C$ be a configuration s.t. $c_0 \overset{B_k}{\Rightarrow}^* c_k \overset{E_k}{\Rightarrow}^* c'_k$, with $c'_k \in \mathcal{U}$. Since $\overline{\leq}$ is a WQO, there is $i < j$ s.t. $c_i \overline{\leq} c_j$. Hence, $c_0 \overset{B_j}{\Rightarrow}^* c_j \overset{E_i}{\Rightarrow}^* c'$, with $c'_i \overline{\leq} c'$ by monotonicity. Thus, $c' \in U$ and $B_j \cdot E_i \in L$. □

8.2.2 A pumping lemma for PN

Our second pumping lemma states properties of languages of Petri nets (more precisely, languages in the class L^G(PN)). This lemma will be exploited mainly in section 8.3.2, to strictly separate the expressive power of PN and PN+NBA. Other results of interest that one can obtain thanks to this lemma are mentioned in section 8.3.5.

The proof of the pumping lemma on WSL (see Lemma 8.3 above) exploited the properties of WQO and the monotonicity property in a rather straightforward fashion: from a well-chosen infinite sequence of configurations, we had extracted two comparable elements (property of $\preccurlyeq$). Thanks to these two comparable elements, and by the monotonicity property, we have devised a new execution of the WSTS that allows to prove the lemma.

We follow the same pattern in the proof of the present pumping lemma for PN. Thus, starting from some well-chosen executions of the PN, we build *infinite sequences* of comparable markings that are reached along these sequences. This construction exploits the properties of $\preccurlyeq$. However, it is much more intricate in the present case than in the case of Lemma 8.3 and deserves some attention. This is the purpose of lemma 8.4, that we introduce now.

Intuitively, Lemma 8.4 shows that, given a matrix $\mathcal{M}$ with infinitely many lines and columns that contains tuples of natural numbers, and given a natural number n, it is possible to build n infinite increasing sequences of elements of $\mathcal{M}$ that enjoy some properties which are necessary to prove the pumping lemma. These n sequences are obtained by the means of n functions $f_1, f_2, \ldots, f_n$ which take their values in $\mathbb{N}_0 \times \mathbb{N}_0$, and are thus meant to *select* elements from $\mathcal{M}$. Thus the first infinite sequence to consider will be $\mathcal{M}(f_1(1)), \mathcal{M}(f_1(2)), \ldots$; the second $\mathcal{M}(f_2(1)), \mathcal{M}(f_2(2)), \ldots$ and so forth. The lemma is as follows:

Lemma 8.4 *Let $\mathcal{M}$ be a matrix with an infinite number of lines and columns, and whose elements are numbered by pairs in $\mathbb{N}_0 \times \mathbb{N}_0$ and take their values in $\mathbb{N}^k$ (for $k \geq 1$).*

For any $n \geq 1$, there are n functions $\mathbb{N}_0 \mapsto \mathbb{N}_0 \times \mathbb{N}_0$, denoted by $f_1, f_2, \ldots, f_n$ such that the following holds (where $f_i^l(x)$ and $f_i^c(x)$ denote respectively the first and second coordinate of $f_i(x)$):

1. *For any $1 \leq i \leq n$, for any $x \geq 1$: $f_i^c(x) \leq i \cdot f_i^l(x)$;*
2. *For any $1 \leq i \leq n$ and $1 \leq j \leq n$, for any $x \geq 1$: $f_i^l(x) = f_j^l(x)$;*
3. *For any $1 \leq i \leq n$, for any $1 \leq p \leq k$: either for any $x \geq 1$, $\mathcal{M}(f_i(x))(p) < \mathcal{M}(f_i(x+1))(p)$ or, for any $x \geq 1$, $\mathcal{M}(f_i(x))(p) = \mathcal{M}(f_i(x+1))(p)$;*
4. *For any $1 \leq i < j \leq n$, for any $x \geq 1$: $0 < f_j^c(x) - f_i^c(x) < f_j^c(x+1) - f_i^c(x+1)$;*
5. *For any $1 \leq i \leq n$, for any $x \geq 1$: $f_i^l(x) < f_i^l(x+1)$.*

Proof. The proof is constructive and by induction on n.

Base case: $n = 1$. Let us consider the sequence:

$$S = \mathcal{M}(1,1), \mathcal{M}(2,1), \mathcal{M}(3,1), \ldots$$

By lemma 2.2, there exists a strictly increasing function $\rho : \mathbb{N}_0 \mapsto \mathbb{N}_0$ s.t. the following is a subsequence of S:

$$\mathcal{M}(\rho(1), 1), \mathcal{M}(\rho(2), 1), \mathcal{M}(\rho(3), 1), \ldots$$

with the following property: for any $1 \leq p \leq k$: either for any $i \geq 1$: $\mathcal{M}(\rho(i), 1)(p) < \mathcal{M}(\rho(i+1), 1)(p)$ or, for any $i \geq 1$: $\mathcal{M}(\rho(i), 1)(p) = \mathcal{M}(\rho(i+1), 1)(p)$. We define f_1 as follows:

$$\text{for any } x \geq 1 : f_1(x) = (\rho(x), 1) \tag{8.1}$$

Let us check that the lemma holds on this function:

1. We have to show that for any $x \geq 1$: $f_1^c(x) \leq 1 \cdot f_1^l(x)$. By (8.1), this is equivalent to $\forall x \geq 1 : 1 \leq \rho(x)$, which is true by definition of ρ.
2. Trivial for $n = 1$.
3. This holds by (8.1) and definition of ρ.
4. Trivial for $n = 1$.
5. We have to show that for any $x \geq 1$: $f_1^l(x) < f_1^l(x+1)$. By (8.1), this is equivalent to $\forall x \geq 1 : \rho(x) < \rho(x+1)$, which is true by definition of ρ.

Inductive case: $n > 1$**.** Let us assume there are $n-1$ functions $g_1, g_2, \ldots, g_{n-1}$ that respect the lemma and let us show how to build n functions $f_1, f_2, \ldots, f_n$ that respect the lemma.

We first define a function g_n as follows:

$$\text{for any } x \geq 1 : g_n(x) = (g_{n-1}^l(x), g_{n-1}^c(x) + x) \tag{8.2}$$

Let us now consider the sequence:

$$\mathcal{M}(g_n(1)), \mathcal{M}(g_n(2)), \mathcal{M}(g_n(3)), \ldots$$

By Lemma 2.2, there exists a strictly increasing function $\rho : \mathbb{N}_0 \mapsto \mathbb{N}_0$ s.t.:

$$\mathcal{M}(g_n(\rho(1))), \mathcal{M}(g_n(\rho(2))), \mathcal{M}(g_n(\rho(3))), \ldots$$

has the following property:

$$\forall 1 \leq p \leq k : \left\{ \begin{array}{rl} \text{either} & \forall i \geq 1 : \mathcal{M}(g_n(\rho(i)))(p) < \mathcal{M}(g_n(\rho(i+1)))(p) \\ \text{or} & \forall i \geq 1 : \mathcal{M}(g_n(\rho(i)))(p) = \mathcal{M}(g_n(\rho(i+1)))(p) \end{array} \right. \tag{8.3}$$

We can now define $f_1, f_2, \ldots, f_n$ as follows:

$$\text{For any } 1 \leq i \leq n : \text{for any } x \geq 1 : f_i(x) = g_i(\rho(x)) \tag{8.4}$$

Let us show that they satisfy the lemma. We prove each point of the lemma by considering several subcases:

1. (a) **In the case where $1 \leq i \leq n-1$:**

$$\begin{aligned} & \forall x \geq 1 : f_i^c(x) \leq i \cdot f_i^l(x) \\ \iff & \forall x \geq 1 : g_i^c(\rho(x)) \leq i \cdot g_i^l(\rho(x)) \quad \text{by (8.4)} \end{aligned}$$

and the latter is true by induction hypothesis (point 1).

   (b) **In the case where $i = n$:**

$$\begin{aligned} & \forall x \geq 1 : f_n^c(x) \leq n \cdot f_n^l(x) \\ \iff & \forall x \geq 1 : g_n^c(\rho(x)) \leq n \cdot g_n^l(\rho(x)) && \text{by (8.4)} \\ \iff & \forall x \geq 1 : g_{n-1}^c(\rho(x)) + \rho(x) \leq n \cdot g_{n-1}^l(\rho(x)) && \text{by (8.2)} \\ \iff & \forall x \geq 1 : g_{n-1}^c(\rho(x)) - (n-1) \cdot g_{n-1}^l(\rho(x)) \\ & \qquad \leq g_{n-1}^l(\rho(x)) - \rho(x) \end{aligned}$$

We show that the last point is valid by establishing that, for any $x \geq 1$, (i) the left-hand side of the inequation $g_{n-1}^c(\rho(x)) - (n-1) \cdot g_{n-1}^l(\rho(x)) \leq 0$ and (ii) the right-hand side of the inequation $g_{n-1}^l(\rho(x)) - \rho(x) \geq 0$. The first point stems from the induction hypothesis, point 1. The latter, holds since, by induction hypothesis (point 5): $0 < g_{n-1}^l(1) < g_{n-1}^l(2) < g_{n-1}^l(3), \ldots$ Hence, for any $x \geq 1 : g_{n-1}^l(x) \geq x$, and thus for any $x \geq 1 : g_{n-1}^l(x) - x \geq 0$.

2. Without loss of generality, we assume that $j \leq i$.

   (a) **In the case where $1 \leq j < i \leq n-1$:**

$$\begin{aligned} & \forall x \geq 1 : f_i^l(x) = f_j^l(x) \\ \iff & \forall x \geq 1 : g_i^l(\rho(x)) = g_j^l(\rho(x)) \quad \text{by (8.4)} \end{aligned}$$

The last point is true by induction hypothesis (point 2).

   (b) **In the case where $i = n$ and $1 \leq j \leq n-1$:**

$$\begin{aligned} & \forall x \geq 1 : f_n^l(x) = f_j^l(x) \\ \iff & \forall x \geq 1 : g_n^l(\rho(x)) = g_j^l(\rho(x)) && \text{by (8.4)} \\ \iff & \forall x \geq 1 : g_{n-1}^l(\rho(x)) = g_j^l(\rho(x)) && \text{by (8.2)} \end{aligned}$$

The last point is true by induction hypothesis (point 2).

   (c) **In the case where $i = j$:** the point is trivially true.

3. First remark that:

$$\begin{aligned} & \forall 1 \leq i \leq n : \forall 1 \leq p \leq k : \\ & \qquad \begin{cases} \text{either} & \forall x \geq 1 : \mathcal{M}(f_i(x))(p) < \mathcal{M}(f_i(x+1))(p) \\ \text{or} & \forall x \geq 1 : \mathcal{M}(f_i(x))(p) = \mathcal{M}(f_i(x+1))(p) \end{cases} \\ \text{iff} \quad & \forall 1 \leq i \leq n : \forall 1 \leq p \leq k : \qquad\qquad \text{by (8.4)} \\ & \qquad \begin{cases} \text{either} & \forall x \geq 1 : \mathcal{M}(g_i(\rho(x)))(p) < \mathcal{M}(g_i(\rho(x+1)))(p) \\ \text{or} & \forall x \geq 1 : \mathcal{M}(g_i(\rho(x)))(p) = \mathcal{M}(g_i(\rho(x+1)))(p) \end{cases} \end{aligned}$$

   (a) **In the case where $1 \leq i \leq n-1$**, this last point is true by induction hypothesis (point 3) and the fact that $\rho(x) < \rho(x+1)$.

   (b) **In the case where $i = n$**, this last point is true by (8.3).

4. (a) **In the case where $1 \leq i < j \leq n-1$:**

$$\begin{aligned} & \forall x \geq 1 : 0 < f_j^c(x) - f_i^c(x) < f_j^c(x+1) - f_i^c(x+1) \\ \iff & \forall x \geq 1 : 0 < g_j^c(\rho(x)) - g_i^c(\rho(x)) \\ & \qquad < g_j^c(\rho(x+1)) - g_i^c(\rho(x+1)) \qquad \text{by (8.4)} \end{aligned}$$

This last point is true by induction hypothesis (point 4) and the fact that $\rho(x) < \rho(x+1)$.

(b) **In the case where $1 \leq i \leq n-1$ and $j = n$:**

$$\begin{aligned} & \forall x \geq 1 : 0 < f_n^c(x) - f_i^c(x) < f_n^c(x+1) - f_i^c(x+1) \\ \iff & \forall x \geq 1 : 0 < g_n^c(\rho(x)) - g_i^c(\rho(x)) \\ & \qquad < g_n^c(\rho(x+1)) - g_i^c(\rho(x+1)) \qquad \text{by (8.4)} \\ \iff & \forall x \geq 1 : 0 < g_{n-1}^c(\rho(x)) + \rho(x) - g_i^c(\rho(x)) \\ & \qquad < g_{n-1}^c(\rho(x+1)) + \rho(x+1) - g_i^c(\rho(x+1)) \quad \text{by (8.2)} \end{aligned}$$

This can be proved by showing two points.

First: $\forall x \geq 1 : 0 < g_{n-1}^c(\rho(x)) + \rho(x) - g_i^c(\rho(x))$. This holds because (*i*) $\forall x \geq 1 : \rho(x) > 0$ (by definition of ρ) and (*ii*) $\forall x \geq 1 : g_{n-1}^c(\rho(x)) - g_i^c(\rho(x)) \geq 0$ (in the case where $i \neq n-1$, we have $g_{n-1}^c(\rho(x)) - g_i^c(\rho(x)) > 0$ by induction hypothesis, point 4. In the case where $i = n-1$, we have $g_{n-1}^c(\rho(x)) - g_i^c(\rho(x)) = 0$).

Second:

$$\begin{aligned} & \forall x \geq 1 : g_{n-1}^c(\rho(x)) + \rho(x) - g_i^c(\rho(x)) \\ & \qquad < g_{n-1}^c(\rho(x+1)) + \rho(x+1) - g_i^c(\rho(x+1)) \\ \iff & \forall x \geq 1 : g_{n-1}^c(\rho(x)) - g_i^c(\rho(x)) - \big(g_{n-1}^c(\rho(x+1)) - g_i^c(\rho(x+1))\big) \\ & \qquad < \rho(x+1) - \rho(x) \end{aligned}$$

This last point is valid, because (*i*), the left-hand side $g_{n-1}^c(\rho(x)) - g_i^c(\rho(x)) - (g_{n-1}^c(\rho(x+1)) - g_i^c(\rho(x+1))$ of the inequation is ≤ 0 (when $i \neq n-1$, it is < 0 by induction hypothesis (point 4), and when $i = n-1$, it is $= 0$) and (*ii*), the right-hand side $\rho(x+1) - \rho(x)$ is > 0, by definition of ρ.

5. (a) **In the case where $1 \leq i \leq n-1$:**

$$\begin{aligned} & \forall x \geq 1 : f_i^l(x) < f_i^l(x+1) \\ \iff & \forall x \geq 1 : g_i^l(\rho(x)) < g_i^l(\rho(x+1)) \quad \text{by (8.4)} \end{aligned}$$

This last point is true by induction hypothesis (point 5) and the fact that $\rho(x) < \rho(x+1)$.

(b) **In the case where $i = n$:**

$$\begin{aligned} & \forall x \geq 1 : f_n^l(x) < f_n^l(x+1) \\ \iff & \forall x \geq 1 : g_n^l(\rho(x)) < g_n^l(\rho(x+1)) \qquad \text{by (8.4)} \\ \iff & \forall x \geq 1 : g_{n-1}^l(\rho(x)) < g_{n-1}^l(\rho(x+1)) \quad \text{by (8.2)} \end{aligned}$$

This last point is true by induction hypothesis (point 5) and the fact that $\rho(x) < \rho(x+1)$.

□

Equipped with this lemma, we can state and prove our pumping lemma for PN.

Lemma 8.5 *Let $\mathcal{N}$ be a* PN *and U be an $\preccurlyeq$-upward-closed set of markings of $\mathcal{N}$.* **If** *there exists an infinite sequence of words $w_1, w_2, \ldots$ such that for any $i \geq 1$, there exist two words B_i, E_i with $B_i w_i^* E_i \subseteq L(\mathcal{N}, U)$,* **then** *there exist $0 < n_1 < n_2 < n_3$ such that for any $K \geq 0$, there exists $K' \geq K$ and $i_1 \geq 0, i_2 \geq 0$ such that the word $B_{n_3} w_{n_3}^{i_1} w_{n_1}^{K'} w_{n_2}^{i_2} E_{n_2}$ is in $L(\mathcal{N}, U)$.*

The proof of the lemma relies greatly on the fact that PN transitions have a constant effect. Before giving the proof, we provide the reader with a sketch that presents the main arguments in order to make the task of reading the proof easier. Throughout this explanation, we refer to peculiar markings using the same notations as in the proof. The reader is advised to refer to Figure 8.1, 8.2 and 8.3 to get the intuition of the meaning of these notations.

The proof is constructive. From the fact that the PN accepts the words $B_i w_i^* E_i$ for any $i \geq 1$, we build, by applying Lemma 8.4, infinite sequences of markings that are ordered (this is the purpose of the first two steps of the proof). Then, at the third step, we exploit these ordering properties, as well as the monotonicity of the PN and the fact that their transitions have constant effect, to show that a sequence of transitions with the desirable form is firable, and leads to the accepting $\preccurlyeq$-upward-closed set of markings.

Step 1 Let N denote the value $2^{|P|}+1$. For all $i \geq 1$, we consider the infinite sequence of words $B_i w_i^N E_i$, $B_i w_i^{2N} E_i$, $B_i w_i^{3N} E_i, \ldots$, $B_i w_i^{jN} E_i, \ldots$ For each of these words, we select an accepting sequence of transitions and consider the markings that are reached along this sequence. For instance, when considering the sequence that accepts $B_i w_i^{jN} E_i$, we select $jN+1$ markings $\mathbf{m}_j^k$ $(1 \leq k \leq jN+1)$ s.t.:

$$\mathbf{m}_{init} \xrightarrow{B_i} \mathbf{m}_j^1 \xrightarrow{w_i} \mathbf{m}_j^2 \xrightarrow{w_i} \cdots \xrightarrow{w_i} \mathbf{m}_j^{jN+1} \xrightarrow{E_i}$$

For any $i \geq 1$, we build an infinite matrix $\mathcal{M}_i$. The jth line of $\mathcal{M}_i$ contains all the markings that have been selected along the run accepting $B_i w_i^{jN} E_i$ (in the same order as in the run). Hence, we obtain a matrix with infinitely many lines. In order to obtain infinitely many elements on each line, we pad the matrix with $0^{|P|} = \langle 0, 0, \ldots, 0 \rangle$ markings. Figure 8.1 presents an example of such a matrix.

Then, we apply lemma 8.4 on $\mathcal{M}_i$ and build N functions $f_{(i,1)}, f_{(i,2)}, \ldots, f_{(i,N)}$. These functions allow us to select elements in the matrix $\mathcal{M}_i$. The selected elements are arranged into a new matrix $\mathcal{M}_i^{\preccurlyeq}$ with N columns and infinitely many lines (see Figure 8.2 for an informal illustration of the construction). $\mathcal{M}_i^{\preccurlyeq}$ is built column by column: the j-th column contains the elements selected by $f_{(i,j)}$, i.e., the first element of the j-th columns is the element of $\mathcal{M}_i$ whose coordinates are given by $f_{(i,j)}(1)$, the second element is the element $f_{(i,j)}(2)$ in $\mathcal{M}_i$, and so on.

$$\begin{pmatrix}
\mathbf{m}_1^1 & \mathbf{m}_1^2 & \cdots & \mathbf{m}_1^{N+1} & 0^{|P|} & \cdots & 0^{|P|} & 0^{|P|} & \cdots & 0^{|P|} & 0^{|P|} & \cdots \\
\mathbf{m}_2^1 & \mathbf{m}_2^2 & \cdots & \mathbf{m}_2^{N+1} & \mathbf{m}_2^{N+2} & \cdots & \mathbf{m}_2^{2N+1} & 0^{|P|} & \cdots & 0^{|P|} & 0^{|P|} & \cdots \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & & & & & \\
\mathbf{m}_j^1 & \mathbf{m}_j^2 & \cdots & \mathbf{m}_j^{N+1} & \mathbf{m}_j^{N+2} & \cdots & \mathbf{m}_j^{2N+1} & \mathbf{m}_j^{2N+2} & \cdots & \mathbf{m}_j^{jN+1} & 0^{|P|} & \cdots \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & &
\end{pmatrix}$$

Figure 8.1: An illustration of the construction of $\mathcal{M}_i$.

The new matrix $\mathcal{M}_i^{\preccurlyeq}$ has interesting properties upon which we rely in the rest of the proof. All these properties are direct consequences of Lemma 8.4. The most important ones are:

1. Each column of $\mathcal{M}_i^{\preccurlyeq}$ forms an infinitely increasing sequence of markings (according to point 3 of Lemma 8.4).
2. Each line of $\mathcal{M}_i^{\preccurlyeq}$ is actually a subsequence of one of the lines of $\mathcal{M}_i$ (by point 2 of Lemma 8.4). Thus, if $\mathbf{m}$ and $\mathbf{m}'$ are two markings taken from the same line of $\mathcal{M}_i^{\preccurlyeq}$ (with $\mathbf{m}$ appearing before $\mathbf{m}'$), we are sure that there exists a sequence of transitions that is firable from $\mathbf{m}$ and produces $\mathbf{m}'$. For that reason, we will sometimes refer to lines of $\mathcal{M}_i^{\preccurlyeq}$ as *runs*.
3. Let us consider two lines ℓ_1 and ℓ_2 of $\mathcal{M}_i^{\preccurlyeq}$ s.t. $\ell_1 < \ell_2$. Let $\mathbf{m}_1$, $\mathbf{m}_2$ be two markings of line ℓ_1 that appear respectively in columns number k_1 and k_2 with $k_1 < k_2$. Let $\mathbf{m}'_1$ and $\mathbf{m}'_2$ be two markings of line ℓ_2 that appear respectively in columns k_1 and k_2. Let σ and σ' be the sequences s.t. $\mathbf{m}_1 \xrightarrow{\sigma} \mathbf{m}_2$ and $\mathbf{m}'_1 \xrightarrow{\sigma'} \mathbf{m}'_2$. Then, the number of w_i's that labels σ' is strictly larger that the number of w_i's that labels σ (this stems from point 4 of Lemma 8.4). This property is important since we want to be able to construct sequences of the form $B_{n_3} w_{n_3}^{i_1} w_{n_1}^{K'} w_{n_2}^{i_2} E_{n_2}$ with K' arbitrarily large.

Step 2 The second step consists in selecting an infinite subset S of $\{\mathcal{M}_i^{\preccurlyeq} \mid i \geq 1\}$. We do this by building a sequence of runs such that the jth run in the sequence is the first run appearing in $\mathcal{M}_j^{\preccurlyeq}$. Again, we extract the sub-sequence S where markings appearing in different runs are $\preccurlyeq$-ordered by applying successively Lemma 2.2. In this case, only markings appearing along the $2^{|P|}+1$ first "columns" are $\preccurlyeq$-ordered.

Step 3 Finally, we show how to split and combine parts of runs appearing in the $\mathcal{M}_i$'s and S to obtain a run that allows the PN to accept a word of the desired form. This is shown in Figure 8.3.

In order to build this sequence, we rely on several variables, namely: c_1, c_2, n and x. At the present step of the proof, we introduce some constraints that relate

$$\mathcal{M}_i = \left(\begin{array}{ccccccccccccccccc}
\boxed{\mathcal{M}_i(f_{(i,1)}(1))} & \boxed{\boxed{\mathcal{M}_i(f_{(i,2)}(1))}} & \boxed{\boxed{\boxed{\mathcal{M}_i(f_{(i,3)}(1))}}} & 0^{|P|} & \cdots & & & & & & & & & & & & \\
\times & \times & \times & \times & \times & \times & 0^{|P|} & \cdots & & & & & & & & & \\
\boxed{\mathcal{M}_i(f_{(i,1)}(2))} & \times & \boxed{\boxed{\mathcal{M}_i(f_{(i,2)}(2))}} & \times & \boxed{\boxed{\boxed{\mathcal{M}_i(f_{(i,3)}(2))}}} & \times & \times & \times & \times & 0^{|P|} & \cdots & & & & & & \\
\times & \times & \boxed{\mathcal{M}_i(f_{(i,1)}(3))} & \times & \times & \times & \boxed{\boxed{\mathcal{M}_i(f_{(i,2)}(3))}} & \times & \times & \boxed{\boxed{\boxed{\mathcal{M}_i(f_{(i,3)}(3))}}} & \times & \times & 0^{|P|} & \cdots & & & \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & 0^{|P|} & \cdots \\
\vdots & & & \vdots & & & \vdots & & & \vdots & & & \vdots & & & \vdots &
\end{array}\right)$$

$$\mathcal{M}_i^{\preccurlyeq} = \left(\begin{array}{ccc}
\boxed{\mathcal{M}_i(f_{(i,1)}(1))} & \boxed{\boxed{\mathcal{M}_i(f_{(i,2)}(1))}} & \boxed{\boxed{\boxed{\mathcal{M}_i(f_{(i,3)}(1))}}} \\
\boxed{\mathcal{M}_i(f_{(i,1)}(2))} & \boxed{\boxed{\mathcal{M}_i(f_{(i,2)}(2))}} & \boxed{\boxed{\boxed{\mathcal{M}_i(f_{(i,3)}(2))}}} \\
\boxed{\mathcal{M}_i(f_{(i,1)}(3))} & \boxed{\boxed{\mathcal{M}_i(f_{(i,2)}(3))}} & \boxed{\boxed{\boxed{\mathcal{M}_i(f_{(i,3)}(3))}}} \\
\vdots & \vdots & \vdots
\end{array}\right)$$

Figure 8.2: An illustration of the construction of $\mathcal{M}_i^{\preccurlyeq}$ for $N = 3$, i.e., $|P| = 1$. Each $\times$ represents a marking reached in the PN. Lemma 8.4 has been applied in order to obtain three functions $f_{(i,1)}$, $f_{(i,2)}$ and $f_{(i,3)}$.

$$\begin{tikzcd}
\mathcal{M}^{\preccurlyeq}_{\rho(1)}(K) = \mathbf{m}_{init} \arrow[r, "B_{\rho(1)}"] & \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K,1) \arrow[r, dotted, no head] & {} \arrow[r, "w_{\rho(1)}"] & \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K,c_1) \arrow[r, Rightarrow, "w_{\rho(1)}"] & \cdots \arrow[r, Rightarrow, "w_{\rho(1)}"] & \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K,c_2) \arrow[r, "w_{\rho(1)}"] \arrow[dd, Rightarrow] & \cdots \arrow[r, "E_{\rho(1)}"] & \mathbf{n}_{\rho(1),K} \\
\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x) = \mathbf{m}_{init} \arrow[r, Rightarrow, "B_{\rho(n)}"] & \mathcal{M}^{\preccurlyeq}_{\rho(n)}(x,1) \arrow[r, Rightarrow, "w_{\rho(n)}"] & \cdots \arrow[r, Rightarrow, "w_{\rho(n)}"] & \mathcal{M}^{\preccurlyeq}_{\rho(n)}(x,c_1) \arrow[u, Rightarrow] \arrow[rrr, dotted, no head] & & & {} \arrow[r, "E_{\rho(n)}"] & \mathbf{n}_{\rho(n),x} \\
\mathcal{M}^{\preccurlyeq}_{\rho(2)}(1) = \mathbf{m}_{init} \arrow[r, "B_{\rho(2)}"] & \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1,1) \arrow[rrr, dotted, no head] & & & {} \arrow[r, "w_{\rho(2)}"] & \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1,c_2) \arrow[r, Rightarrow, "w_{\rho(2)}"] & \cdots \arrow[r, Rightarrow, "E_{\rho(2)}"] & \mathbf{n}_{\rho(2),1}
\end{tikzcd}$$

Figure 8.3: The firable sequence (along the $\Rightarrow$'s) that accepts a word of the form $B_{n_3} w_{n_3}^{i_1} w_{n_1}^{K'} w_{n_2}^{i_2} E_{n_2}$.

x and n to c_1, c_2 and K. These constraints are meant to produce a sequence of transitions that accepts a word of the desired form. The main (and most technical) part of step 3 consists in showing that these constraints are satisfiable. The first part of the sequence is the prefix of $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x)$, up to the "column" c_1 (see Figure 8.3). At that point, we are guaranteed that the marking we obtain is larger than $\mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1)$. This allows us to continue the sequence with a part of $\mathcal{M}^{\preccurlyeq}_{\rho(1)}(K)$, starting at "column" c_1 and ending at "column" c_2. Again, by exploiting the properties of the sequences built at steps 1 and 2, as well as the constant effect of PN transitions, we are ensured that the marking we have reached is larger than $\mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)$. This allows us to finish the sequence with the suffix of $\mathcal{M}^{\preccurlyeq}_{\rho(2)}(1)$. The word accepted by this sequence is of the desired form, since we have correctly chosen the values of x and n (in particular, the central part of the word is longer than K times $|w_{n_1}|$).

We are now ready to present the proof of Lemma 8.5.

Proof. Let $\mathcal{N}$ be a PN with set of places P and initial marking $\mathbf{m}_{init}$, such that $\{B_i w_i^* E_i\} \subseteq L(\mathcal{N}, U)$ for all $i \geq 1$. Let N denote the value $2^{|P|} + 1$.

$\boxed{\textbf{Step 1}}$ For any $i \geq 1$, let S_i be an infinite sequence of runs accepting the words of the form $B_i w_i^{j \cdot N} E_i$, with $j \geq 1$. That is, S_i is a sequence of runs:

$$\begin{array}{l}
\mathbf{m}_{init} \xrightarrow{\upsilon_1} \mathbf{m}_1^1 \xrightarrow{\varsigma_1^1} \mathbf{m}_1^2 \xrightarrow{\varsigma_1^2} \cdots \xrightarrow{\varsigma_1^N} \mathbf{m}_1^{N+1} \xrightarrow{\upsilon_1'} \mathbf{n}_{i,1}\\
\mathbf{m}_{init} \xrightarrow{\upsilon_2} \mathbf{m}_2^1 \xrightarrow{\varsigma_2^1} \mathbf{m}_2^2 \xrightarrow{\varsigma_2^2} \cdots \quad \cdots \xrightarrow{\varsigma_2^{2\cdot N}} \mathbf{m}_2^{2\cdot N+1} \xrightarrow{\upsilon_2'} \mathbf{n}_{i,2}\\
\vdots\\
\mathbf{m}_{init} \xrightarrow{\upsilon_j} \mathbf{m}_j^1 \xrightarrow{\varsigma_j^1} \mathbf{m}_j^2 \xrightarrow{\varsigma_j^2} \cdots \quad \cdots \quad \cdots \xrightarrow{\varsigma_j^{j\cdot N}} \mathbf{m}_j^{j\cdot N+1} \xrightarrow{\upsilon_j'} \mathbf{n}_{i,j}
\end{array}$$

where for any $\ell \geq 1$: $\mathbf{n}_{i,\ell} \in U$, $\Lambda(\upsilon_\ell) = B_i$ and $\Lambda(\upsilon'_\ell) = E_i$. Moreover, $\forall \ell \geq 1 : \forall 1 \leq k \leq \ell \cdot N : \Lambda(\varsigma_\ell^k) = w_i$.

Let $0^{|P|}$ denote the marking that ranges over $|P|$ places and assigns 0 token to each place. For any $i \geq 1$, we build, an infinite matrix $\mathcal{M}_i$, whose values are either markings met along the runs of S_i or $0^{|P|}$. More precisely, for any $j \geq 1, k \geq 1$ we have:

$$\mathcal{M}_i(j,k) = \begin{cases} \mathbf{m}_j^k & \text{if } 1 \leq k \leq j \cdot N + 1 \\ 0^{|P|} & \text{otherwise} \end{cases} \tag{8.5}$$

For any $i \geq 1$, we can apply Lemma 8.4 to $\mathcal{M}_i$, and obtain N functions that respect the five points of the lemma. We denote these functions by $f_{(i,1)}, f_{(i,2)}, \ldots f_{(i,N)}$. Thanks to these functions, we build infinitely many sequences of N markings. We represent these sequences under the form of a new matrix $\mathcal{M}_i^{\preccurlyeq}$, with N columns and infinitely many lines (each line corresponds to a sequence). $\mathcal{M}_i^{\preccurlyeq}$ is defined as follows (where $f^l_{(i,k)}(j)$ and $f^c_{(i,k)}(j)$ denote respectively the first and second coordinate of $f_{(i,k)}(j)$):

$$\forall i \geq 1 : \forall j \geq 1 : \forall 1 \leq k \leq N : \mathcal{M}_i^{\preccurlyeq}(j,k) = \mathcal{M}_i(f^l_{(i,k)}(j), f^c_{(i,k)}(j)) \tag{8.6}$$

For any $i, j \geq 1$, let $\mathcal{M}_i^{\preccurlyeq}(j)$ denote $\mathcal{M}_i^{\preccurlyeq}(j,1), \mathcal{M}_i^{\preccurlyeq}(j,2), \dots \mathcal{M}_i^{\preccurlyeq}(j,N)$, i.e. the sequence of markings that appears on the j-th line of $\mathcal{M}_i^{\preccurlyeq}$. Let us expose several properties of these sequences that will be useful in the sequel of the proof:

1. For any $i \geq 1$, $j \geq 1$, the sequence $\mathcal{M}_i^{\preccurlyeq}(j)$ corresponds to a run of S_i. More precisely, $\mathcal{M}_i^{\preccurlyeq}(j)$ is a subsequence of the markings in the $f^l_{(i,1)}(j)$−th run of S_i. According to the definitions of $\mathcal{M}_i$ and $\mathcal{M}_i^{\preccurlyeq}$ (see (8.5) and (8.6)), this can be proved by establishing three points:
   (a) The markings of $\mathcal{M}_i^{\preccurlyeq}(j)$ have all been taken in the same run of S_i since $f^l_{(i,1)}(j) = f^l_{(i,2)}(j) = \cdots = f^l_{(i,N)}(j)$, by point 2 of Lemma 8.4.
   (b) The ordering of the markings along the run has been preserved since the sequence $f^c_{(i,1)}(j), f^c_{(i,2)}(j), \dots, f^c_{(i,N)}(j)$ is strictly increasing, from point 4 of Lemma 8.4.
   (c) All the selected markings in $\mathcal{M}_i^{\preccurlyeq}(j)$ exist in the $f^l_{(i,1)}(j)$-th run of S_i, i.e., they are all different from the $0^{|P|}$ markings we have added when building $\mathcal{M}_i$. Since the ordering of the markings has been preserved, it is sufficient to show that the last marking of $\mathcal{M}_i^{\preccurlyeq}(j)$ corresponds to a marking of the $f^l_{(i,1)}(j)$-th run of S_i, i.e., that $f^c_{(i,N)}(j) \leq N \cdot f^l_{(i,1)}(j) + 1$. By point (a) above, this is equivalent to $f^c_{(i,N)}(j) \leq N \cdot f^l_{(i,N)}(j) + 1$, which is true by point 1 of Lemma 8.4.

2. Since $\mathcal{M}_i^{\preccurlyeq}(j)$ is a subsequence of markings that appear in a run of S_i, there exists, for any $i \geq 1$, $j \geq 1$ and $1 \leq k_1 < k_2 \leq N$ a sequence of transitions $\sigma_i^j(k_1,k_2)$ s.t.:
$$\mathcal{M}_i^{\preccurlyeq}(j,k_1) \xrightarrow{\sigma_i^j(k_1,k_2)} \mathcal{M}_i^{\preccurlyeq}(j,k_2)$$
Moreover, for any $i \geq 1$, $j \geq 1$ and $1 \leq k \leq N$, there are two sequences of transitions $\sigma_i^j(\cdot,k)$ and $\sigma_i^j(k,\cdot)$ s.t.:
$$\mathbf{m}_{init} \xrightarrow{\sigma_i^j(\cdot,k)} \mathcal{M}_i^{\preccurlyeq}(j,k) \xrightarrow{\sigma_i^j(k,\cdot)} \mathbf{n} \quad (\mathbf{n} \in U)$$
By (8.5) and (8.6), these sequences are labelled as follows (for any $i, j \geq 1$):
$$\forall 1 \leq k_1 < k_2 \leq N : \Lambda(\sigma_i^j(k_1,k_2)) = w_i^{(f^c_{(i,k_2)}(j) - f^c_{(i,k_1)}(j))} \tag{8.7}$$
$$\forall 1 \leq k \leq N : \Lambda(\sigma_i^j(\cdot,k)) = B_i w_i^{(f^c_{(i,k)}(j)-1)} \tag{8.8}$$
$$\forall 1 \leq k \leq N : \Lambda(\sigma_i^j(k,\cdot)) = w_i^{(j\cdot N+1-f^c_{(i,k)}(j))} E_i \tag{8.9}$$

Finally, let us introduce the following notation. Let w be a (possibly empty) word and v be a non-empty word. Then, we let $\|w\|_v = i$ iff $w = v^i$. By (8.7) and point 4 of Lemma 8.4, the following holds:

$$\begin{aligned}&\forall i, j \geq 1 : \forall 1 \leq k_1 < k_2 \leq N : \\ &\qquad w_i \neq \varepsilon \text{ implies } \|\Lambda(\sigma_i^j(k_1, k_2))\|_{w_i} < \|\Lambda(\sigma_i^{j+1}(k_1, k_2))\|_{w_i}\end{aligned}$$

That is, when $w_i \neq \varepsilon$, the word that labels the sequence leading from the k_1-th marking of the j-th run of $\mathcal{M}_i^{\preccurlyeq}$ to its k_2-th marking is strictly shorter than the word labelling the corresponding sequence in the $(j+1)$-th run of $\mathcal{M}_i^{\preccurlyeq}$. In particular, since $w_i \neq \varepsilon$ implies that $\|\Lambda(\sigma_i^1(k_1, k_2))\|_{w_i} \geq 1$, we have:

$$\begin{gathered}\forall i \geq 1 : \forall 1 \leq k_1 < k_2 \leq N : \\ w_i \neq \varepsilon \text{ implies } \|\Lambda(\sigma_i^j(k_1, k_2))\|_{w_i} \geq j\end{gathered} \tag{8.10}$$

3. Let us first introduce the following notation. Let $\mathcal{S}$ be an infinite sequence of runs $\mathcal{S}(1), \mathcal{S}(2), \ldots$, s.t. each run $\mathcal{S}(i)$ is made up of N markings $\mathcal{S}(i,1)$, $\mathcal{S}(i,2)$, $\ldots$, $\mathcal{S}(i,N)$. Then, for any $1 \leq k \leq N$, we denote by $\mathit{Places}(\mathcal{S}, k)$ the set of places s.t.

   $$\mathit{Places}(\mathcal{S}, k) = \big\{p \in P \mid \forall i \geq 1 : \mathcal{S}(i,k)(p) < \mathcal{S}(i+1,k)(p)\big\}$$

   By (8.5) and (8.6), and by Lemma 8.4, point 3, for any $1 \leq k \leq N$ and $i \geq 1$, the set $\mathit{Places}(\mathcal{M}_i^{\preccurlyeq}, k)$ is s.t.:

   $$\begin{aligned}&\forall i \geq 1 : \forall 1 \leq k \leq N : \forall p \in P : \\ &\quad p \in \mathit{Places}(\mathcal{M}_i^{\preccurlyeq}, k) \text{ iff } \forall j \geq 1 : \mathcal{M}_i^{\preccurlyeq}(j,k)(p) < \mathcal{M}_i^{\preccurlyeq}(j+1,k)(p) \\ &\quad p \notin \mathit{Places}(\mathcal{M}_i^{\preccurlyeq}, k) \text{ iff } \forall j \geq 1 : \mathcal{M}_i^{\preccurlyeq}(j,k)(p) = \mathcal{M}_i^{\preccurlyeq}(j+1,k)(p)\end{aligned} \tag{8.11}$$

   In particular, this implies that $\mathcal{M}_i^{\preccurlyeq}(1,k), \mathcal{M}_i^{\preccurlyeq}(2,k), \ldots, \mathcal{M}_i^{\preccurlyeq}(j,k), \ldots$ is an increasing sequence (w.r.t. $\preccurlyeq$):

   $$\forall i \geq 1 : \forall 1 \leq k \leq N : \forall j \geq 1 : \mathcal{M}_i^{\preccurlyeq}(j,k) \preccurlyeq \mathcal{M}_i^{\preccurlyeq}(j+1,k) \tag{8.12}$$

$\boxed{\textbf{Step 2}}$ To finish with the construction, we consider the infinite sequence of runs $\mathcal{M}_1^{\preccurlyeq}(1), \mathcal{M}_2^{\preccurlyeq}(1), \ldots$ made up of the first runs (lines) of all $\mathcal{M}_i^{\preccurlyeq}$. From this sequence, we extract the infinite subsequence $S = \mathcal{M}_{\rho(1)}^{\preccurlyeq}(1), \mathcal{M}_{\rho(2)}^{\preccurlyeq}(1), \ldots$ by successively applying Lemma 2.2. We construct S such that:

1. For any $1 \leq j \leq N$ the sequence $\mathcal{M}_{\rho(1)}^{\preccurlyeq}(1,j), \mathcal{M}_{\rho(2)}^{\preccurlyeq}(1,j), \ldots$ is increasing:

   $$\forall k \geq 1 : \mathcal{M}_{\rho(k)}^{\preccurlyeq}(1,j) \preccurlyeq \mathcal{M}_{\rho(k+1)}^{\preccurlyeq}(1,j) \tag{8.13}$$

2. For any $1 \leq j \leq N$, the places in the set $Places(S, j) \subseteq P$ strictly increase along the sequence $\mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, j), \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, j), \ldots$ and all the other places stay constant along the sequence. Thus, we let:

$$Places(S, j) = \\ \left\{p \in P \mid \forall k \geq 1 : \mathcal{M}^{\preccurlyeq}_{\rho(k)}(1, j)(p) < \mathcal{M}^{\preccurlyeq}_{\rho(k+1)}(1, j)(p)\right\} \tag{8.14}$$

   Let c_1 and c_2 be such that $1 \leq c_1 < c_2 \leq N$ and $Places(S, c_1) = Places(S, c_2)$. Remark that c_1 and c_2 always exist because there are $2^{|P|} = N - 1$ subsets of P.

3. The sets of strictly increasing places of the selected $\mathcal{M}^{\preccurlyeq}_i$ are equal:

$$\forall 1 \leq j \leq N : \forall k \geq 1 : Places(\mathcal{M}^{\preccurlyeq}_{\rho(k)}, j) = Places(\mathcal{M}^{\preccurlyeq}_{\rho(k+1)}, j) \tag{8.15}$$

   This is possible because there are $2^{|P|}$ subsets of P.

$\boxed{\textbf{Step 3}}$ The rest of the proof consists in showing that there are $0 < n_1 < n_2 < n_3$ s.t. for any $K \in \mathbb{N}$ there are $i_1 \geq 0$, $i_2 \geq 0$ and $K' \geq K$ s.t. the word $w = B_{n_3} w_{n_3}^{i_1} w_{n_1}^{K'} w_{n_2}^{i_2} E_{n_2}$ is accepted by $\mathcal{N}$.

We first choose the values of n_1 and n_2 as follows: $n_1 = \rho(1)$ and $n_2 = \rho(2)$ (where ρ is the function defined at the beginning of step 2). Then, we show how to compute n_3. Actually, we let $n_3 = \rho(n)$ for a well-chosen value of n. We provide a constraint (see equation (8.16) in the sequel) on n that we prove satisfiable and that we exploit at the end of the proof. Equipped with the values n_1, n_2 and n_3, we show that, for any $K \in \mathbb{N}$, it is possible to compute a value x s.t. the sequence of transitions $\sigma = \sigma^x_{\rho(n)}(\cdot, c_1) \cdot \sigma^K_{\rho(1)}(c_1, c_2) \cdot \sigma^1_{\rho(2)}(c_2, \cdot)$ accepts a word of the desired form.

Choice of n Let $\mathbf{m}_n$ be the marking such that $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) \xrightarrow{\sigma^1_{\rho(1)}(c_1, c_2)} \mathbf{m}_n$ (with c_1 and c_2 as defined at then of point 2, step 2). Remark that, since we are dealing with Petri nets, the sequence $\sigma^1_{\rho(1)}(c_1, c_2)$ has a constant effect (i.e., characterised by a vector of natural constants) equal to $\mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_2) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)$. Thus $\mathbf{m}_n = \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_2) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)$. We choose $n > 2$ such that:

$$\mathbf{m}_n = \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_2) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1) \succcurlyeq \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2) \tag{8.16}$$

Let us show that such a n exists. First notice that $\sigma^1_{\rho(1)}(c_1, c_2)$ is firable from $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)$ for all $n > 2$, because $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) \succcurlyeq \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)$ following (8.13), and the fact that $\rho(n) > \rho(1)$. Then, recall that $Places(S, c_1) = Places(S, c_2)$, i.e. the places that strictly increase along S are the same in columns c_1 and c_2. Let us show that, for any place p, $\mathbf{m}_n(p) \geq \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)(p)$. For that purpose, we consider two cases:

1. If $p \in \mathit{Places}(S, c_1)$, then, the sequence $\mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p), \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_1)(p), \ldots$ is strictly growing by (8.14) and, for any $n \geq 1$, $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p) \geq n - 1$. Thus there exists $n \geq 1$ s.t. $\forall p \in \mathit{Places}(S, c_1) : \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p) \geq \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)(p) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_2)(p) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p)$. This is equivalent to $\forall p \in \mathit{Places}(S, c_1) : \mathbf{m}_n(p) \geq \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)(p)$, by definition of $\mathbf{m}_n$.

2. On the other hand, for any $p \in P \setminus \mathit{Places}(S, c_1)$, we have: $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p) = \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p)$ and $\mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)(p) = \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_2)(p)$, by (8.14) again. Hence, $\forall p \in P \setminus \mathit{Places}(S, c_1)$: $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p) - \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)(p) = \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_2)(p)$, and thus, we have that, for any place $p \in P \setminus \mathit{Places}(S, c_1)$: $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_2)(p) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p) = \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)(p)$. Hence, for any place $p \in P \setminus \mathit{Places}(S, c_1)$, $\mathbf{m}_n(p) = \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)(p)$.

From these two points, we conclude that there exists n s.t. $\mathbf{m}_n \succcurlyeq \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)$.

Choice of x We choose $x > K$ such that:

$$\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1) \succcurlyeq \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1) \tag{8.17}$$

One can prove that such an x always exists by the same reasoning as in the choice of n, and by the fact that $\mathit{Places}(\mathcal{M}^{\preccurlyeq}_{\rho(n)}, c_1) = \mathit{Places}(\mathcal{M}^{\preccurlyeq}_{\rho(1)}, c_1)$ (see (8.15) above). Indeed, $\forall p \in \mathit{Places}(\mathcal{M}^{\preccurlyeq}_{\rho(1)}, c_1)$, the sequence $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p), \mathcal{M}^{\preccurlyeq}_{\rho(n)}(2, c_1)(p), \ldots$ is strictly increasing by (8.11) and (8.15), and we can thus choose x large enough to have $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1)(p) \geq \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1)(p) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p)$, for any place p in the set $\mathit{Places}(\mathcal{M}^{\preccurlyeq}_{\rho(1)}, c_1)$. On the other hand, for any $p \in P \setminus \mathit{Places}(\mathcal{M}^{\preccurlyeq}_{\rho(1)}, c_1)$: $\mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1)(p) = \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p)$, by (8.11) and (8.15). Thus, $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1)(p) \geq \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1)(p) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p)$ if and only if $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1)(p) \geq \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p)$. This last point is true by (8.11). We conclude that for any $p \in P$: $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1)(p) \geq \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1)(p) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1)(p) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)(p)$.

The next step amounts to showing that the sequence σ is firable. From $\mathbf{m}_{init}$, we fire $\sigma^x_{\rho(n)}(\cdot, c_1)$ and reach $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1)$. From that marking, we can fire the sequence $\sigma^K_{\rho(1)}(c_1, c_2)$. This is possible because $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1) \succcurlyeq \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1)$. Indeed, by (8.17): $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1) \succcurlyeq \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1) + \big(\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)\big)$. However, we know that $\big(\mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1)\big) \succcurlyeq 0^{|P|}$, by (8.13). This implies that $\mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1) \succcurlyeq \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1)$ and we have:

$$\mathbf{m}_{init} \xrightarrow{\sigma^x_{\rho(n)}(0, c_1)} \mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1) \xrightarrow{\sigma^K_{\rho(1)}(c_1, c_2)} \mathbf{m}$$

To finish the sequence, we have to show that $\mathbf{m} \succcurlyeq \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2)$. Since the effect of $\sigma^K_{\rho(1)}(c_1, c_2)$ is constant and equal to $\mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_2) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1)$, we have:

$$\begin{array}{rcll}
\mathbf{m} & = & \mathcal{M}^{\preccurlyeq}_{\rho(n)}(x, c_1) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_2) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1) & \\
\Rightarrow \mathbf{m} & \succcurlyeq & \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1) & \\
 & & +\mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_2) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_1) & \text{by (8.17)} \\
\Rightarrow \mathbf{m} & \succcurlyeq & \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(K, c_2) & \\
\Rightarrow \mathbf{m} & \succcurlyeq & \mathcal{M}^{\preccurlyeq}_{\rho(n)}(1, c_1) - \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_1) + \mathcal{M}^{\preccurlyeq}_{\rho(1)}(1, c_2) & \text{by (8.12)} \\
\Rightarrow \mathbf{m} & \succcurlyeq & \mathcal{M}^{\preccurlyeq}_{\rho(2)}(1, c_2) & \text{by (8.16)}
\end{array}$$

We can thus fire $\sigma^1_{\rho(2)}(c_2, \cdot)$ from $\mathbf{m}$ and obtain $\mathbf{m}'$ such that $\mathbf{m}' \succcurlyeq \mathbf{n}_{\rho(2),1}$ (by monotonicity), which implies that $\mathbf{m}' \in U$. Thus, $\mathcal{N}$ accepts $\Lambda(\sigma)$, which is of the form $B_{n_3} w^{i_1}_{n_3} w^{K'}_{n_1} w^{i_2}_{n_2} E_{n_2}$ with $n_1 = \rho(1)$, $n_2 = \rho(2)$ and $n_3 = \rho(n)$, $i_1 \geq 0$, and $i_2 \geq 0$. The former implies that $0 < n_1 < n_2 < n_3$, by definition of ρ. We finish the proof by considering two cases:

1. If $w_{n_1} = \varepsilon$, then clearly, for any $j \geq 0$: $w^j_{n_1} = \varepsilon$. In particular $w^K_{n_1} = \varepsilon = w^{K'}_{n_1}$. Thus, for any $j \geq 0$, the word $B_{n_3} w^{i_1}_{n_3} w^j_{n_1} w^{i_2}_{n_2} E_{n_2}$ satisfies the lemma.
2. If $w_{n_1} \neq \varepsilon$, it remains to show that the central part of the accepted word is long enough, i.e., that $K' \geq K$. This stems from the fact that, by construction, $K' = \|\Lambda(\sigma^K_{\rho(1)}(c_1, c_2))\|_{w_i}$ and that $\|\Lambda(\sigma^K_{\rho(1)}(c_1, c_2))\|_{w_i} \geq K$ by (8.10).

In both cases, we conclude that the word we have built, and that is accepted by the PN, satisfies the lemma. □

8.2.3 A pumping lemma for PN+NBA

Let us now turn our attention to our third pumping lemma (for PN+NBA):

Lemma 8.6 *Let $\mathcal{N}$ be a* PN+NBA *and U be an $\preccurlyeq$-upward-closed set of markings of $\mathcal{N}$.* **If** *there exists an infinite sequence of words $w_1, w_2, \ldots$ such that for any $i \geq 1$, there exist two words B_i, E_i with $B_i w^*_i E_i \subseteq L(\mathcal{N}, U)$,* **then** *there exist $i_1 \geq 0$, $i_2 > 0$, $i_3 \geq 0$ and $0 < n_1 < n_2 < n_3$ such that the word $B_{n_3} w^{i_1}_{n_3} w^{i_2}_{n_1} w^{i_3}_{n_2} E_{n_2}$ is in $L(\mathcal{N}, U)$.*

Once again, since the proof of Lemma 8.6 is rather technical, we first sketch it informally. The proof may be decomposed into two steps:

Step 1 We build an infinite sequence of runs whose $i-th$ element is a run that accepts the word $B_i w^{2^{|P|}}_i E_i$ (where P is the set of places of the PN+NBA considered).

Then, we build a sub-sequence of these runs by applying successively Lemma 2.2. Those sub-sequences have the property that markings appearing in different runs are $\preccurlyeq$-ordered. The increasing sequences appear along the $2^{|P|}+1$ first "columns".

Step 2 Finally, we show how to split and combine parts of runs appearing in the runs in order to obtain a new run that allows the PN+NBA to accept a word of the desired form.

In order to build this sequence, we rely on several variables, namely: c_1, c_2 and n. At the present step of the proof, we present several constraints on c_1, c_2 and n. These constraints are meant to produce a sequence of transitions that accepts a word of the desired form. The main (and most technical) part of step 2 consists to show that these constraints are satisfiable.

Proof. Let $\mathcal{N}$ be a PN+NBA with set of places $\mathcal{P}$ and initial marking $\mathbf{m}_{init}$ such that $B_i w_i^* E_i \subseteq L(\mathcal{N}, U)$.

$\boxed{\textbf{Step 1}}$ Since $B_i w_i^* E_i \subseteq L(\mathcal{N}, U)$ for all $i \geq 1$, the word $B_i w_i^{2^{|\mathcal{P}|}} E_i$ is accepted by $\mathcal{N}$. Let us consider an infinite sequence of runs that accept the words: $B_1 w_1^{2^{|P|}} E_1$, $B_2 w_2^{2^{|P|}} E_2, \ldots$, $B_j w_j^{2^{|P|}} E_j, \ldots$, i.e.,

$$\begin{array}{l}
\mathbf{m}_{init} \xrightarrow{\upsilon_1} \mathbf{m}_1^1 \xrightarrow{\varsigma_1^1} \mathbf{m}_1^2 \xrightarrow{\varsigma_1^2} \cdots \xrightarrow{\varsigma_1^{2^{|\mathcal{P}|}}} \mathbf{m}_1^{2^{|\mathcal{P}|}+1} \xrightarrow{\upsilon_1'} \mathbf{n}_1 \\
\mathbf{m}_{init} \xrightarrow{\upsilon_2} \mathbf{m}_2^1 \xrightarrow{\varsigma_2^1} \mathbf{m}_2^2 \xrightarrow{\varsigma_2^2} \cdots \xrightarrow{\varsigma_2^{2^{|\mathcal{P}|}}} \mathbf{m}_2^{2^{|\mathcal{P}|}+1} \xrightarrow{\upsilon_2'} \mathbf{n}_2 \\
\vdots \qquad\quad \vdots \qquad\qquad\qquad\qquad \vdots \\
\mathbf{m}_{init} \xrightarrow{\upsilon_i} \mathbf{m}_i^1 \xrightarrow{\varsigma_i^1} \mathbf{m}_i^2 \xrightarrow{\varsigma_i^2} \cdots \xrightarrow{\varsigma_i^{2^{|\mathcal{P}|}}} \mathbf{m}_i^{2^{|\mathcal{P}|}+1} \xrightarrow{\upsilon_i'} \mathbf{n}_i \\
\vdots \qquad\quad \vdots \qquad\qquad\qquad\qquad \vdots
\end{array}$$

where for any $i \geq 1$: $\Lambda(\upsilon_i) = B_i$, $\Lambda(\upsilon_i') = E_i$, $\mathbf{n}_i \in U$ and for any $1 \leq j \leq 2^{|\mathcal{P}|}$: $\Lambda(\varsigma_i^j) = w_i$.

By applying Lemma 2.2 successively, we can construct an infinite subsequence of that sequence:

$$\begin{array}{l}
\mathbf{m}_{init} \xrightarrow{\upsilon_{\rho(1)}} \mathbf{m}_{\rho(1)}^1 \xrightarrow{\varsigma_{\rho(1)}^1} \mathbf{m}_{\rho(1)}^2 \xrightarrow{\varsigma_{\rho(1)}^2} \cdots \xrightarrow{\varsigma_{\rho(1)}^{2^{|\mathcal{P}|}}} \mathbf{m}_{\rho(1)}^{2^{|\mathcal{P}|}+1} \xrightarrow{\upsilon_{\rho(1)}'} \mathbf{n}_{\rho(1)} \\
\mathbf{m}_{init} \xrightarrow{\upsilon_{\rho(2)}} \mathbf{m}_{\rho(2)}^1 \xrightarrow{\varsigma_{\rho(2)}^1} \mathbf{m}_{\rho(2)}^2 \xrightarrow{\varsigma_{\rho(2)}^2} \cdots \xrightarrow{\varsigma_{\rho(2)}^{2^{|\mathcal{P}|}}} \mathbf{m}_{\rho(2)}^{2^{|\mathcal{P}|}+1} \xrightarrow{\upsilon_{\rho(2)}'} \mathbf{n}_{\rho(2)} \\
\ldots
\end{array}$$

such that, for any $1 \leq j \leq 2^{|\mathcal{P}|}+1$, the sequence $\mathbf{m}_{\rho(1)}^j \mathbf{m}_{\rho(2)}^j \ldots$ is increasing:

$$\forall 1 \leq j \leq 2^{|\mathcal{P}|}+1 : \forall k \geq 1 : \mathbf{m}_{\rho(k)}^j \preccurlyeq \mathbf{m}_{\rho(k+1)}^j \tag{8.18}$$

and, for any $1 \leq j \leq 2^{|\mathcal{P}|}+1$, there exists a set of places, noted $Places(j)$, whose marking strictly increases along the sequence $\mathbf{m}^j_{\rho(1)}\mathbf{m}^j_{\rho(2)}\ldots$ while the other places stay constant:

$$\forall 1 \leq j \leq 2^{|\mathcal{P}|}+1 : \forall k \geq 1 : \mathbf{m}^j_{\rho(k)}(p) < \mathbf{m}^j_{\rho(k+1)}(p) \text{ iff } p \in Places(j) \tag{8.19}$$

Since, there are $2^{|P|}$ subsets of P, there exist $1 \leq c_1 < c_2 \leq 2^{|P|}+1$ such that $Places(c_1) = Places(c_2)$.

In the following, we denote by $\sigma_{\rho(j)}(k_1,k_2)$ with $k_1 < k_2$ the sequence $\varsigma^{k_1}_{\rho(j)} \cdot \ldots \cdot \varsigma^{k_2-1}_{\rho(j)}$. We also denote by $\sigma_{\rho(j)}(\cdot,k)$, the sequence $\upsilon_{\rho(j)} \cdot \varsigma^1_{\rho(j)} \cdot \ldots \cdot \varsigma^{k-1}_{\rho(j)}$; and by $\sigma_{\rho(j)}(k,\cdot)$ the sequence $\varsigma^k_{\rho(j)} \cdot \ldots \cdot \varsigma^{2^{|P|}}_{\rho(j)} \cdot \upsilon'_{\rho(j)}$.

$\boxed{\textbf{Step 2}}$ The rest of the proof consists in devising a word of $L(\mathcal{N},U)$ that is of the form $B_{n_3}w^{i_1}_{n_3}w^{i_2}_{n_1}w^{i_3}_{n_2}E_{n_2}$, with $i_1 \geq 0$, $i_2 > 0$, $i_3 \geq 0$ and $0 < n_1 < n_2 < n_3$. The sequence of transitions that accepts this word (called σ) is built as follows:

$$\sigma = \sigma_{\rho(n)}(\cdot,c_1) \cdot \sigma_{\rho(1)}(c_1,c_2) \cdot \sigma_{\rho(2)}(c_2,\cdot)$$

for a well-chosen value of n. We next explain how to compute this value.

We choose $n > 2$ such that, when firing $\sigma_{\rho(1)}(c_1,c_2)$ from $\mathbf{m}^{c_1}_{\rho(n)}$, we reach a marking $\mathbf{m} \not\succcurlyeq \mathbf{m}^{c_2}_{\rho(2)}$. Let us show that such a n always exists. First, remark that for any $n > 2$: $\sigma_{\rho(1)}(c_1,c_2)$ is firable from $\mathbf{m}^{c_1}_{\rho(n)}$ since, by (8.18), $\mathbf{m}^{c_1}_{\rho(n)} \not\succcurlyeq \mathbf{m}^{c_1}_{\rho(1)}$. Let k be the number of non-blocking arcs in $\sigma_{\rho(1)}(c_1,c_2)$. By Lemma 7.6, we have that

$$\forall p \in P : \mathbf{m}(p) \geq \mathbf{m}^{c_1}_{\rho(n)}(p) + \mathbf{m}^{c_2}_{\rho(1)}(p) - \mathbf{m}^{c_1}_{\rho(1)}(p) - k \tag{8.20}$$

But, since $Places(c_1) = Places(c_2)$, we can state the following. For any place $p \in Places(c_1)$ and for any $n \geq 1$: $\mathbf{m}^{c_1}_{\rho(n)}(p) \geq n-1$, since by (8.19) the sequence $\mathbf{m}^{c_2}_{\rho(1)}(p), \mathbf{m}^{c_2}_{\rho(2)}(p), \ldots$ is strictly increasing. In particular, if we choose n such that

$$n > \max_{p \in Places(c_1)} \left(\mathbf{m}^{c_2}_{\rho(2)}(p) - \mathbf{m}^{c_2}_{\rho(1)}(p) + \mathbf{m}^{c_1}_{\rho(1)}(p)\right) + k$$

we have $\forall p \in Places(c_1) : \mathbf{m}^{c_1}_{\rho(n)}(p) \geq \mathbf{m}^{c_2}_{\rho(2)}(p) - \mathbf{m}^{c_2}_{\rho(1)}(p) + \mathbf{m}^{c_1}_{\rho(1)}(p) + k$ and thus:

$$\forall p \in Places(c_1) : \mathbf{m}^{c_1}_{\rho(n)}(p) + \mathbf{m}^{c_2}_{\rho(1)}(p) - \mathbf{m}^{c_1}_{\rho(1)}(p) - k \geq \mathbf{m}^{c_2}_{\rho(2)}(p) \tag{8.21}$$

By (8.20) and (8.21), we obtain:

$$\forall p \in Places(c_1) : \mathbf{m}(p) \geq \mathbf{m}^{c_2}_{\rho(2)}(p) \tag{8.22}$$

On the other hand, for any place p, the monotonicity property of $\mathsf{PN{+}NBA}$ implies that $\mathbf{m}(p) \geq \mathbf{m}^{c_2}_{\rho(1)}(p)$. And since, by (8.19): $\forall p \in P \setminus Places(c_1) : \mathbf{m}^{c_2}_{\rho(1)}(p) = \mathbf{m}^{c_2}_{\rho(2)}(p)$, we obtain:

$$\forall p \in P \setminus Places(c_1) : \mathbf{m}(p) \geq \mathbf{m}^{c_2}_{\rho(2)}(p) \tag{8.23}$$

By (8.22) and (8.23), we conclude that $\mathbf{m} \succcurlyeq \mathbf{m}^{c_2}_{\rho(2)}$.

Thus, the sequence of transitions $\sigma = \sigma_{\rho(n)}(\cdot, c_1) \cdot \sigma_{\rho(1)}(c_1, c_2) \cdot \sigma_{\rho(2)}(c_2, \cdot)$ is firable from $\mathbf{m}_{init}$ (with n computed as explained above) and leads to a marking $\mathbf{m}'$, i.e $\mathbf{m}_{init} \xrightarrow{\sigma} \mathbf{m}'$. Since $\mathbf{m} \succcurlyeq \mathbf{m}^{c_2}_{\rho(2)}$, we also have that $\mathbf{m}' \succcurlyeq \mathbf{n}_{\rho(2)}$, by monotonicity. Hence $\mathbf{m}' \in U$, and the word $\Lambda(\sigma) \in L(\mathcal{N}, U)$. It is not difficult to see that by the previous construction this word is of the form: $B_{n_3} w^{i_1}_{n_3} w^{i_2}_{n_1} w^{i_3}_{n_2} E_{n_2}$ with *(i)* $n_3 = \rho(n)$, $n_1 = \rho(1)$ and $n_2 = \rho(2)$, hence $0 < n_1 < n_2 < n_3$, and *(ii)* $i_1 \geq 0$, $i_2 = c_2 - c_1 > 0$, $i_3 \geq 0$. □

8.3 Properties of WSL

In this section, we apply the pumping lemmata of the previous section to obtain several results about WSL and languages of EPN. Section 8.3.1 presents properties of WSL that can be proved thanks to Lemma 8.3. Then, the pumping lemmata on PN and PN+NBA are exploited in sections 8.3.2 and 8.3.3 to prove a strict hierarchy among the languages of PN, PN+NBA and PN+T; as well as in section 8.3.4, to obtain closure properties of languages of EPN. Finally, section 8.3.5 shows that some of the results that have been obtained thanks to the pumping lemma on WSL can also be obtained thanks to the pumping lemmata on PN and PN+NBA.

8.3.1 Consequences of Lemma 8.3

We first study several classical languages and show that they are not well-structured. These languages are: the set of all words of the form $\mathtt{a}^n\mathtt{b}^n$, the set of all words of the form $\mathtt{a}^n\mathtt{b}^m$ with $m \geq n$, and the set of all palindromes (of even length).

- $\mathcal{L} = \{\mathtt{a}^n\mathtt{b}^n | n \geq 1\} \notin L^G(\mathsf{WSTS})$. Suppose that $\mathcal{L} \in L^G(\mathsf{WSTS})$. Since, $\forall k \geq 1 : \mathtt{a}^k\mathtt{b}^k \in \mathcal{L}$, we can apply Lemma 8.3 (letting $B_k = \mathtt{a}^k$ and $E_k = \mathtt{b}^k$, for any $k \geq 1$). We conclude that there is $i < j$ s.t. $\mathtt{a}^j\mathtt{b}^i \in \mathcal{L}$, which is a contradiction. Notice that this results is also a consequence of Theorem 8.2 and Theorem 8.1, following the reasoning given in [Pet81, pages 175–176].

- $\mathcal{L}^{\geq} = \{\mathtt{a}^n\mathtt{b}^m | m \geq n\} \notin L^G(\mathsf{WSTS})$. The proof is similar to the previous one.

- $\mathcal{L}^R = \{w \cdot w^R\} \notin L^G(\mathsf{WSTS})$. Let Σ be an alphabet and $w = \mathtt{a}_1 \cdot \ldots \cdot \mathtt{a}_n \in \Sigma^*$, we define the mirror of w, as the word $w^R = \mathtt{a}_n \cdot \ldots \cdot \mathtt{a}_1$. Let us suppose $\mathcal{L}^R \in L^G(\mathsf{WSTS})$. Since $\{\mathtt{a}^n\mathtt{bba}^n \mid n \geq 0\} \subseteq \mathcal{L}^R$, we can apply Lemma 8.3 (letting $B_k = \mathtt{a}^k\mathtt{b}$ and $E_k = \mathtt{ba}^k$, for any $k \geq 1$). We conclude that there exist $i < j$ such that $\mathtt{a}^j\mathtt{bba}^i \in \mathcal{L}^R$, which is a contradiction. Hence $\mathcal{L}^R \notin L^G(\mathsf{WSTS})$.

These results allow us to show that neither the class of WSL, nor $L^G(\mathsf{PN})$, nor $L^G(\mathsf{PN{+}NBA})$, nor $L^G(\mathsf{PN{+}T})$ are closed under complement.

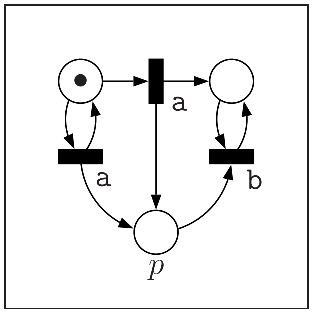


Figure 8.4: The PN $\mathcal{N}_3$ that accepts (i) $\{\mathtt{a}^n\mathtt{b}^m \mid m < n\}$ when $\{\mathbf{m} \mid \mathbf{m}(p) \geq 1\}$ is the set of accepting markings and (ii) $\{\mathtt{a}^n\mathtt{b}^m \mid m \leq n\}$ when $\mathbb{N}^3$ is the set of accepting markings.

Proposition 8.3 *L^G(WSTS), L^G(PN), L^G(PN+NBA), L^G(PN+R) and L^G(PN+T) are not closed under complement.*

Proof. Let us consider the PN $\mathcal{N}_3$ of Figure 8.4. It should be clear that $L(\mathcal{N}_3, U) = \{\mathtt{a}^n\mathtt{b}^m \mid m < n\}$ for $U = \{\mathbf{m} \mid \mathbf{m}(p) \geq 1\}$. It is well-known [Pet81] that L^G(PN) is closed under union and that the regular languages are all in L^G(PN). Hence, $\{\mathtt{a}^n\mathtt{b}^m \mid m < n\} \cup \big((\mathtt{a}+\mathtt{b})^* \setminus \mathtt{a}^*\mathtt{b}^*\big)$ is in L^G(PN), but also in L^G(PN+NBA), L^G(PN+R) and in L^G(PN+T), since PN is a syntactic subclass of theirs. However, its complement is $\mathcal{L}^{\geq} = \{\mathtt{a}^n\mathtt{b}^m \mid m \geq n\}$, which is not a WSL (see above). ☐

Finally, we can also exploit the previous results to show that the class of WSL is incomparable to the class of context-free languages.

Proposition 8.4 *The class L^G(WSTS) is incomparable to the class of context-free languages.*

Proof. CFL $\not\subseteq L^G$(WSTS) stems from the fact that $\mathcal{L}$, which is well-known to be a CFL, is not in L^G(WSTS). We prove that L^G(WSTS) $\not\subseteq$ CFL thanks to $\mathcal{L}_1 = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i \geq j \geq k \geq 0\}$. First, consider the PN $\mathcal{N}_4$ of Figure 8.5. It should be clear that $L(\mathcal{N}_4, \mathbb{N}^5) = \mathcal{L}_1$. Hence, $\mathcal{L}_1 \in L^G$(WSTS). On the other hand, we prove that $\mathcal{L}_1$ is not a CFL thanks to Lemma 2.16 (the pumping lemma for CFL)

For that purpose, we have to devise, for any constant $n \in \mathbb{N}$, a word $\omega_n \in \mathcal{L}_1$ such that $|\omega_n| \geq n$ and, for any words u, v, w, x and y respecting (i) $\omega_n = u \cdot v \cdot w \cdot x \cdot y$, (ii) $|v \cdot w \cdot x| \leq n$ and (iii) $|v \cdot x| > 0$, we can find $i \geq 0$ s.t. $u \cdot v^i \cdot w \cdot x^i \cdot y \notin \mathcal{L}_1$.

For any $n \geq 0$, we let $\omega_n = \mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$. Clearly $\omega_n \in \mathcal{L}_1$ and $|\omega_n| \geq n$, for any n. Let us consider all the possible values of u, v,…, y that respect the three conditions above, and let us show that, for all these values, there exists an $i \geq 0$ such that $u \cdot v^i \cdot w \cdot x^i \cdot y \notin \mathcal{L}_1$.

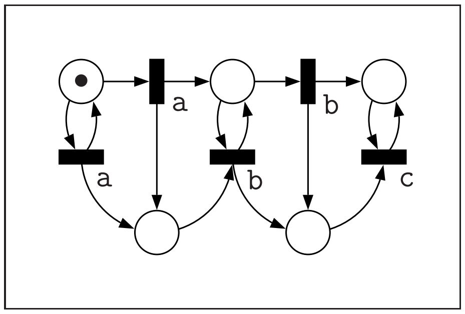


Figure 8.5: The PN $\mathcal{N}_4$ that accepts $\mathcal{L}_1 = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i \geq j \geq k \geq 0\}$ when $U = \mathbb{N}^5$ is the set of accepting markings.

- *If either v or x contain at least two different characters*, the word $u \cdot v^2 \cdot w \cdot x^2 \cdot y$ is clearly not a word of $\mathcal{L}_1$.

- *Otherwise, if $v \in \mathtt{a}^*$*, then, since $|v \cdot w \cdot x| \leq n$, it is not possible that x contains a $\mathtt{c}$. Hence, since x does not contain two different characters, there are two possibilities. Either $x \in \mathtt{a}^*$. In that case, we choose $i = 0$ and the word $u \cdot v^0 \cdot w \cdot x^0 \cdot y$ is of the form $\mathtt{a}^{n-|v \cdot x|}\mathtt{b}^n\mathtt{c}^n$, and is clearly not in $\mathcal{L}_1$, since $|v \cdot x| > 0$. Otherwise, $x \in \mathtt{b}^*$. In that case, we choose $i = 0$ again and we obtain a word of the form $\mathtt{a}^{n-|v|}\mathtt{b}^{n-|x|}\mathtt{c}^n$, which is not in $\mathcal{L}_1$ because $|v \cdot x| > 0$.

- *Otherwise, i.e., $v \in \mathtt{b}^*$ or $v \in \mathtt{c}^*$*, we choose $i = 2$, and the word $u \cdot v^2 \cdot w \cdot x^2 \cdot y$ contains either more $\mathtt{b}$'s or more $\mathtt{c}$'s than $\mathtt{a}$'s. Hence, it does not belong to $\mathcal{L}_1$.

□

8.3.2 PN+NBA are more expressive than PN

In this section we prove that the class of languages accepted by PN+NBA strictly contains the class of languages accepted by PN (when the acceptance condition is an $\preccurlyeq$-upward-closed set). Since the class PN forms a syntactic subclass of PN+NBA, we obtain this result by showing that there is a language accepted by a PN+NBA that cannot be accepted by any PN.

Separation of PN+NBA and PN The strategy adopted in the proof is as follows. We look into the PN+NBA $\mathcal{N}_2$ of Figure 7.2, with initial marking $\mathbf{m}_0$ such that $\mathbf{m}_0(p_1) = 1$ and $\mathbf{m}_0(p) = 0$ for $p \in \{p_2, p_3, p_4, p_5, p_6\}$ and set of accepting markings $U = \mathbb{N}^6$. We prove that it accepts every word of the form $\mathtt{i}^k\mathtt{s}\big(\mathtt{a}^k\mathtt{c}\mathtt{b}^k\mathtt{d}\big)^j$, for $k \geq 0$ and $j \geq 0$ (Lemma 8.7), but not those of the form $\mathtt{i}^{n_3}\mathtt{s}\mathtt{a}^{n_3}\mathtt{c}(\mathtt{b}^{n_3}\mathtt{d}\mathtt{a}^{n_3}\mathtt{c})^{i_1}\big(\mathtt{b}^{n_1}\mathtt{d}\mathtt{a}^{n_1}\mathtt{c}\big)^k\big(\mathtt{b}^{n_2}\mathtt{d}\mathtt{a}^{n_2}\mathtt{c}\big)^{i_2}\mathtt{b}^{n_2}\mathtt{d}$, for k big enough, and $0 < n_1 < n_2 < n_3$ (Lemma 8.8). Then we invoke Lemma 8.5 (pumping lemma on PN) to prove that every PN accepting the words of the first form also accepts words of the latter, which implies that no PN accepts $L(\mathcal{N}_2, \mathbb{N}^6)$.

Let us first state Lemma 8.7 and Lemma 8.8. Remark that the first one is a direct consequence of Lemma 7.8. Hence, we omit its proof.

Lemma 8.7 *For any $k \geq 0$, for any $j \geq 0$, the word $\mathtt{i}^k\mathtt{s}\big(\mathtt{a}^k\mathtt{cb}^k\mathtt{d}\big)^j$ is in $L(\mathcal{N}_2, \mathbb{N}^6)$.*

Lemma 8.8 *Let n_1, n_2 and n_3 be three natural numbers such that $0 < n_1 < n_2 < n_3$. The words*

$$\mathtt{i}^{n_3}\mathtt{sa}^{n_3}\mathtt{c}(\mathtt{b}^{n_3}\mathtt{da}^{n_3}\mathtt{c})^{i_1}\big(\mathtt{b}^{n_1}\mathtt{da}^{n_1}\mathtt{c}\big)^k\big(\mathtt{b}^{n_2}\mathtt{da}^{n_2}\mathtt{c}\big)^{i_2}\mathtt{b}^{n_2}\mathtt{d}$$

are not in $L(\mathcal{N}_2, \mathbb{N}^6)$, for all $i_1 \geq 0$, $k \geq n_3 - n_1$ and $i_2 \geq 0$.

Proof. In this proof, we will identify a sequence of transitions with the word it accepts (all the transitions have different labels and there are no silent transitions). Clearly (see the proof of Lemma 7.8), for any $n_3 \geq 0$, $m \geq 0$, the firing of $\mathtt{i}^{n_3}\mathtt{s}\big(\mathtt{a}^{n_3}\mathtt{cb}^{n_3}\mathtt{d}\big)^m$ from $\mathbf{m}_0$ leads to a marking $\mathbf{m}_1$ such that $\mathbf{m}_1(p_2) = n_3$, $\mathbf{m}_1(p_3) = 1$, and $\forall i \in \{1, 4, 5, 6\} : \mathbf{m}_1(p_i) = 0$ (the non-blocking arc of t_6 hasn't consumed any token in p_4). By firing $\mathtt{a}^{n_3}\mathtt{cb}^{n_1}\mathtt{d}$ from $\mathbf{m}_1$, we now have n_1 tokens in p_2, $n_3 - n_1 - 1$ tokens in p_4 and one token in p_6 (this time the non-blocking arc has moved one token since $n_1 < n_3$). Clearly, at each subsequent firing of $\mathtt{a}^{n_1}\mathtt{cb}^{n_1}\mathtt{d}$, the non-blocking arc of t_6 will remove one token from p_4 and the marking of this place will strictly decrease until p_4 becomes empty. Let $\ell = n_3 - n_1 - 1$. It is easy to see that that firing $\mathtt{a}^{n_3}\mathtt{cb}^{n_1}\mathtt{d}\big(\mathtt{a}^{n_1}\mathtt{cb}^{n_1}\mathtt{d}\big)^\ell$ from $\mathbf{m}_1$ leads to a marking $\mathbf{m}_2$ with $\mathbf{m}_2(p_2) = n_1$, $\mathbf{m}_2(p_3) = 1$, $\mathbf{m}_2(p_6) = n_3 - n_1$ and $\forall j \in \{1, 4, 5\} : \mathbf{m}_2(p_j) = 0$. This characterisation also implies that we can fire $\mathtt{a}^{n_1}\mathtt{cb}^{n_1}\mathtt{d}$ an arbitrary number of times from $\mathbf{m}_2$ because $\mathbf{m}_2 \xrightarrow{\mathtt{a}^{n_1}\mathtt{cb}^{n_1}\mathtt{d}} \mathbf{m}_2$. On the other hand, it is not possible to fire $\mathtt{a}^{n_1}\mathtt{cb}^{n_2}\mathtt{d}$, with $n_2 > n_1$, from $\mathbf{m}_2$. Indeed $\mathbf{m}_2 \xrightarrow{\mathtt{a}^{n_1}\mathtt{cb}^{n_1}} \mathbf{m}_3$, with $\mathbf{m}_3(p_5) = 1$, $\mathbf{m}_3(p_2) = n_1$, $\mathbf{m}_3(p_6) = n_3 - n_1$ and $\forall j \in \{1, 3, 4\} : \mathbf{m}_3(p_j) = 0$, which does not allow to fire the b-labelled transition t_5 anymore. We conclude that, $\forall k \geq n_3 - n_1$, a sequence labelled by $\mathtt{i}^{n_3}\mathtt{s}\big(\mathtt{a}^{n_3}\mathtt{cb}^{n_3}\mathtt{d}\big)^m\mathtt{a}^{n_3}\mathtt{c}\big(\mathtt{b}^{n_1}\mathtt{da}^{n_1}\mathtt{c}\big)^k\mathtt{b}^{n_2}\mathtt{da}^{n_2}\mathtt{c}$, is not firable in $\mathcal{N}_2$. Thus, we will not find in $L(\mathcal{N}_2, \mathbb{N}^6)$ any word with this prefix, hence the Lemma. □

Thanks to these lemmata, we can prove Proposition 8.5.

Proposition 8.5 *There is no PN $\mathcal{N}$ and no $\preccurlyeq$-upward-closed set U s.t. $L(\mathcal{N}, U) = L(\mathcal{N}_2, \mathbb{N}^6)$.*

Proof. By Lemma 8.7, any PN $\mathcal{N}$ s.t. $L(\mathcal{N}, \mathcal{U}) = L(\mathcal{N}_2, \mathbb{N}^6)$ for some $\preccurlyeq$-upward-closed set of accepting markings $\mathcal{U}$, must accept $\mathtt{i}^k\mathtt{s}\big(\mathtt{a}^k\mathtt{cb}^k\mathtt{d}\big)^j$, for any $k \geq 1$ and $j \geq 0$. Hence, we can apply Lemma 8.5, by letting $B_k = \mathtt{i}^k\mathtt{sa}^k\mathtt{c}$, $E_k = \mathtt{b}^k\mathtt{d}$ and $w_k = \mathtt{b}^k\mathtt{da}^k\mathtt{c}$, for any $k \geq 1$. We conclude that $\mathcal{N}$ also accepts a word of the form:

$\mathtt{i}^{n_3}\mathtt{s}\mathtt{a}^{n_3}\mathtt{c}\big(\mathtt{b}^{n_3}\mathtt{d}\mathtt{a}^{n_3}\mathtt{c}\big)^{i_1}\big(\mathtt{b}^{n_1}\mathtt{d}\mathtt{a}^{n_1}\mathtt{c}\big)^{L'}\big(\mathtt{b}^{n_2}\mathtt{d}\mathtt{a}^{n_2}\mathtt{c}\big)^{i_2}\mathtt{b}^{n_2}\mathtt{d}$ such that $0 < n_1 < n_2 < n_3$ and $L' \geq n_3 - n_1$. Since it is not in $L(\mathcal{N}_2, U)$, by Lemma 8.8, there is no PN $\mathcal{N}$ with an $\preccurlyeq$-upward-closed set $\mathcal{U}$ s.t. $L(\mathcal{N}, \mathcal{U}) = L(\mathcal{N}_2, \mathbb{N}^6)$. □

Thus, we conclude that:

Theorem 8.4 $L^G(\mathsf{PN}) \subset L^G(\mathsf{PN{+}NBA})$.

Proof. $L^G(\mathsf{PN}) \subseteq L^G(\mathsf{PN{+}NBA})$ is trivial since PN is a syntactic subclass of PN+NBA. The strictness of the inclusion is given by Proposition 8.5. □

8.3.3 PN+T are more expressive than PN+NBA

Let us now prove a similar result about the classes PN+NBA and PN+T: the class of languages that can be accepted by some PN+T strictly contains the class of languages accepted by any given PN+NBA. For this purpose, we first show that a PN+T can always *simulate* a PN+NBA, hence $L^G(\mathsf{PN{+}NBA}) \subseteq L^G(\mathsf{PN{+}T})$. Then, we prove, thanks to Lemma 8.6, that there is a language that can be recognised by a PN+T, but not by a PN+NBA, which implies the strictness of the inclusion.

Simulation of a PN+NBA by a PN+T As we did in the case of ω-languages, we re–use the construction of Section 5.4.4 to associate to any PN+NBA $\mathcal{N}$ with upward–closed set of accepting markings U, a PN+T $\mathcal{N}'$ and a set U' that define the same language as $\mathcal{N}$ and U.

More precisely, given a PN+NBA $\mathcal{N}$, we consider the PN+T $\mathcal{N}'$ as defined in Section 5.4.4 (it is actually a PN+R. However, remember that every PN+R is a PN+T). We handle the labels as follows: for any transition $t \in T_r$, the corresponding t' has the same label as t. For any $t \in T_e$, the corresponding $t^{=0}$ and $t^{\neq 0}$ have the same label as t. Given an upward–closed set U of markings of $\mathcal{N}$, we consider the upward–closed set U' of $\mathcal{N}'$ as defined in the same section. We also re–use the notations $\preccurlyeq_P$, $=_P$, and so forth, as well as the two functions f and g defined there to establish a correspondence between the sequences of $\mathcal{N}$ and those of $\mathcal{N}'$.

This allows us to show that:

Lemma 8.9 *Let $\mathcal{N} = \langle P, T, \mathbf{m}_0\rangle$ be an* PN+NBA *and let U be a $\preccurlyeq$–upward–closed set of markings of $\mathcal{N}$. Let $\mathcal{N}' = \langle P', T', \mathbf{m}_0'\rangle$ and U' be the* PN+T *and the upward–closed set of markings of $\mathcal{N}'$ obtained from $\mathcal{N}$ and U (see Section 5.4.4). Then, $L(\mathcal{N}, U) = L(\mathcal{N}', U')$.*

Proof. Let $\sigma = \tau_1\tau_2\ldots\tau_n$ be a sequence of transitions of $\mathcal{N}$, and, for any $1 \leq i \leq n$, let $\mathbf{m}_i$ be the marking s.t. $\mathbf{m}_{i-1} \xrightarrow{\tau_i} \mathbf{m}_i$. Let us assume that $\mathbf{m}_n \in U$, hence

$\Lambda(\sigma) \in L(\mathcal{N}, U)$. Let $\sigma' = \tau'_1\tau'_2 \cdots \tau'_n$ be the sequence of $\mathcal{N}'$ s.t. for any $1 \leq i \leq n$: $\tau'_i = f(\tau_i, \mathbf{m}_i)$. By construction, $\Lambda(\sigma') = \Lambda(\sigma)$. By Lemma 5.8, σ' is firable in $\mathcal{N}'$ and $\mathbf{m}_n =_P \mathbf{m}'_n$, hence, $\mathbf{m}'_n \in U'$, by definition of U', and since $\mathbf{m}_n \in U$. Hence, $\Lambda(\sigma') \in L(\mathcal{N}', U')$. We conclude that $L(\mathcal{N}, U) \subseteq L(\mathcal{N}', U')$.

By a similar reasoning on a sequence of transitions σ' of $\mathcal{N}'$ that reaches U', and thanks to Lemma 5.9, we conclude that there exists a firable sequence of transitions σ of $\mathcal{N}$ that reaches U and s.t. $\Lambda(\sigma) = \Lambda(\sigma')$. Hence, $L(\mathcal{N}, U) \supseteq L(\mathcal{N}', U')$.

We conclude that $L(\mathcal{N}, U) = L(\mathcal{N}', U')$. □

Separation of PN+T and PN+NBA Let us now prove that L^G(PN+NBA) is *strictly* included in L^G(PN+T). We consider the PN+T $\mathcal{N}_1$ presented in Figure 7.1 with the initial marking $\mathbf{m}_0(p_1) = 1$ and $\mathbf{m}_0(p) = 0$ for $p \in \{p_2, p_3, p_4\}$. The two following Lemmata allow us to better understand the behaviour of $\mathcal{N}_1$. Remark that the first one is a direct consequence of Lemma 7.4. Hence we omit its proof too.

Lemma 8.10 *For any $k \geq 1$, for any $j \geq 0$, the word $\left(\mathtt{a}^k\mathtt{b}^k\right)^j$ is in $L(\mathcal{N}_1, \mathbb{N}^4)$.*

Lemma 8.11 *Let n_1, n_2, n_3 be three natural numbers such that $0 < n_1 < n_2 < n_3$. For any $i_1 \geq 0$, $i_2 > 0$ and $i_3 \geq 0$, the words of the form:*

$$\mathtt{a}^{n_3}(\mathtt{b}^{n_3}\mathtt{a}^{n_3})^{i_1}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^{i_2}(\mathtt{b}^{n_2}\mathtt{a}^{n_2})^{i_3}\mathtt{b}^{n_2}$$

are not in $L(\mathcal{N}_1, \mathbb{N}^4)$.

Proof. The following holds for any n_1, n_2, n_3 with $0 < n_1 < n_2 < n_3$. From the initial marking of $\mathcal{N}_1$, the only sequence of transitions labelled by $\mathtt{a}^{n_3}$ is $t_1^{n_3}$. Firing this sequence leads to the marking $\mathbf{m}_1$ such that $\mathbf{m}_1(p_1) = 1, \mathbf{m}_1(p_3) = n_3$ and $\mathbf{m}_1(p) = 0$ if $p \in \{p_2, p_4\}$. From $\mathbf{m}_1$ the only firable sequence of transitions labelled by $\mathtt{b}^{n_3}$ is $t_2t_3^{n_3-1}$. This leads to the marking $\mathbf{m}_2$ such that $\mathbf{m}_2(p_2) = 1$ and $\mathbf{m}_2(p) = 0$ if $p \neq p_2$. The only sequence of transitions firable from $\mathbf{m}_2$ and labelled by $\mathtt{a}^{n_3}$ is $t_4t_1^{n_3-1}$. Since $\mathbf{m}_2(p_3) = 0$, the transfer of t_4 has no effect when fired from $\mathbf{m}_2$. Hence, we reach $\mathbf{m}_1$ again after firing $t_4t_1^{n_3-1}$. By repeating the reasoning, we conclude that the only sequence of transitions firable from the initial marking and labelled by $(\mathtt{a}^{n_3}\mathtt{b}^{n_3})^{i_1}\mathtt{a}^{n_3}$ (when $i_1 > 0$) is $t_1^{n_3}t_2t_3^{n_3-1}(t_4t_1^{n_3-1}t_2t_3^{n_3-1})^{i_1-1}t_4t_1^{n_3-1}$ and leads to $\mathbf{m}_1$. In the case where $i_1 = 0$, the sequence $t_1^{n_3}$ is firable and leads to $\mathbf{m}_1$ too. From $\mathbf{m}_1$, the only firable sequence of transitions labelled by $\mathtt{b}^{n_1}$ is $t_2t_3^{n_1-1}$. This leads to a marking similar to $\mathbf{m}_2$, noted $\mathbf{m}'_2$, except that p_3 contains $n_3 - n_1$ tokens. Then, the only firable sequence of transitions labelled by $\mathtt{a}^{n_1}$ is $t_4t_1^{n_1-1}$. In this case, the transfer of t_4 moves the $n_3 - n_1$ tokens from p_3 to p_4 and we reach a marking similar to $\mathbf{m}_1$, noted $\mathbf{m}'_1$, except that p_4 contains $n_3 - n_1$ tokens and p_3 contains n_1 tokens. From $\mathbf{m}'_1$, the only firable sequence of transitions labelled by $\mathtt{b}^{n_1}\mathtt{a}^{n_1}$ is $t_2 t_3^{n_1-1} t_4 t_1^{n_1-1}$ and leads to $\mathbf{m}_1'$. Hence, the sequence $(t_2 t_3^{n_1-1} t_4 t_1^{n_1-1})^{i_2}$ is firable from $\mathbf{m}_1'$.

However, after firing $t_2 t_3^{n_1-1}$ from $\mathbf{m}_1'$, we reach a marking $\mathbf{m}_2''$ similar to $\mathbf{m}_2$ except that p_4 contains $n_3 - n_1$ tokens and from which no transition labelled by $\mathtt{b}$ is firable. Since $n_2 > n_1$, we conclude that there is no sequence of transitions labelled by $\mathtt{b}^{n_2}$ that is firable from $\mathbf{m}_1'$, hence $\mathtt{a}^{n_3}(\mathtt{b}^{n_3}\mathtt{a}^{n_3})^{i_1}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^{i_2}(\mathtt{b}^{n_2}\mathtt{a}^{n_2})^{i_3}\mathtt{a}^{n_2}$ with $i_1 \geq 0, i_2 > 0, i_3 \geq 0$ is not in $L(\mathcal{N}_1, \mathbb{N}^4)$. □

Thanks to these two lemmata, and thanks to Lemma 8.6, we can now prove Proposition 8.6, that states that no PN+NBA can accept the language of $\mathcal{N}_1$.

Proposition 8.6 *There is no* PN+NBA *and no $\preccurlyeq$-upward-closed set U s.t. $L(\mathcal{N}, U) = L(\mathcal{N}_1, \mathbb{N}^4)$.*

Proof. By Lemma 8.10, any PN+NBA $\mathcal{N}$ s.t. $L(\mathcal{N}, U) = L(\mathcal{N}_1, \mathbb{N}^4)$ for some $\preccurlyeq$-upward-closed set U, accepts $\left(\mathtt{a}^j\mathtt{b}^j\right)^k$, for any $j \geq 1, k \geq 1$. Thus, we can apply Lemma 8.6, by letting $B_i = \mathtt{a}^i$, $E_i = \mathtt{b}^i$ and $w_i = \mathtt{b}^i\mathtt{a}^i$, for all $i \geq 1$, and obtain that $\mathcal{N}$ accepts a word of the form: $\mathtt{a}^{n_3}(\mathtt{b}^{n_3}\mathtt{a}^{n_3})^{i_1}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^{i_2}(\mathtt{b}^{n_2}\mathtt{a}^{n_2})^{i_3}\mathtt{b}^{n_2}$ with $0 < n_1 < n_2 < n_3$ and $i_2 > 0$. Since, by Lemma 8.11, this word is not in $L(\mathcal{N}_1, \mathbb{N}^4)$, there can be no PN+NBA $\mathcal{N}$ with an $\preccurlyeq$-upward-closed-set U s.t.: $L(\mathcal{N}, \mathcal{U}) = L(\mathcal{N}_1, \mathbb{N}^4)$. □

The last two propositions allow us to conclude that:

Theorem 8.5 $L^G(\mathsf{PN{+}NBA}) \subset L^G(\mathsf{PN{+}T})$

Proof. $L^G(\mathsf{PN{+}NBA}) \subseteq L^G(\mathsf{PN{+}T})$ is given by Lemma 8.9. The strictness of the inclusion is given by Proposition 8.6. □

8.3.4 Closure Properties of EPN

The pumping lemmata on PN and PN+NBA can also be used to show that neither $L^G(\mathsf{PN})$ nor $L^G(\mathsf{PN{+}NBA})$ are closed under iteration.

Theorem 8.6 $L^G(\mathsf{PN})$ *and* $L^G(\mathsf{PN{+}NBA})$ *are not closed under iteration.*

Proof. Let us consider the PN $\mathcal{N}_3$ of Figure 8.4. Clearly, $L = \{\mathtt{a}^n\mathtt{b}^m | n \geq m\} = L(\mathcal{N}_3, \mathbb{N}^3)$. Hence, $L \in L^G(\mathsf{PN})$ and $L \in L^G(\mathsf{PN{+}NBA})$).

Let us show, *per absurdum*, that $L^+ \notin L^G(\mathsf{PN})$. Suppose that there is a PN $\mathcal{N}$ and an upward-closed set U s.t. $L(\mathcal{N}, U) = L^+$. Let $B_i = \mathtt{a}^i$, $w_i = \mathtt{b}^i\mathtt{a}^i$ and $E_i = \mathtt{b}_i$ for all $i \geq 1$. Thanks to Lemma 8.5, we obtain that $L(\mathcal{N}, U)$ contains a word of the form:

$$\mathtt{a}^{n_3}(\mathtt{b}^{n_3}\mathtt{a}^{n_3})^{i_1}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^{K}(\mathtt{b}^{n_2}\mathtt{a}^{n_2})^{i_2}\mathtt{b}^{n_2}$$

with $n_1 < n_2 < n_3$, $K \geq 1$, which is not in L^+. Contradiction.

Let us show, *per absurdum* that $L^+ \not\in L^G(\mathsf{PN{+}NBA})$. Suppose that there is a $\mathsf{PN{+}NBA}$ $\mathcal{N}'$ and an upward-closed set U' s.t. $L(\mathcal{N}', U') = L^+$. Again, let $B_i = \mathtt{a}^i$, $w_i = \mathtt{b}^i\mathtt{a}^i$ and $E_i = \mathtt{b}_i$ for all $i \geq 1$. Thanks to Lemma 8.6, we obtain that $L(\mathcal{N}', U')$ contains a word of the form:

$$\mathtt{a}^{n_3}(\mathtt{b}^{n_3}\mathtt{a}^{n_3})^{i_1}(\mathtt{b}^{n_1}\mathtt{a}^{n_1})^{i_2}(\mathtt{b}^{n_2}\mathtt{a}^{n_2})^{i_3}\mathtt{b}^{n_2}$$

with $i_1 \geq 0$, $i_2 > 0$, $i_3 \geq 0$ and $0 < n_1 < n_2 < n_3$, which is not in L^+. Contradiction. □

In [Pet81], Peterson proves that $L^L(\mathsf{PN})$ is not closed under iteration, but does not address the case of $L^G(\mathsf{PN})$ (which we have solved here) and mentions the case of $L^P(\mathsf{PN})$ as an open problem (see page 186 of [Pet81]). We can now solve it easily:

Theorem 8.7 *$L^P(\mathsf{PN})$ and $L^P(\mathsf{PN{+}NBA})$ are not closed under iteration.*

Proof. We consider again the PN $\mathcal{N}_3$ of Figure 8.4 and the language $L = \{\mathtt{a}^n\mathtt{b}^m | n \geq m\}$. By definition of prefix languages, $L = L(\mathcal{N}_3, \mathbb{N}^3) \in L^P(\mathsf{PN}) \subseteq L^P(\mathsf{PN{+}NBA})$. By Theorem 8.6, and the facts that $L^P(\mathsf{PN}) \subseteq L^G(\mathsf{PN})$ $L^P(\mathsf{PN{+}NBA}) \subseteq L^G(\mathsf{PN{+}NBA})$, we conclude that $L^+ \not\in L^P(\mathsf{PN})$ and $L^+ \not\in L^P(\mathsf{PN{+}NBA})$. Hence the Theorem. □

Following Definition 2.36, Theorem 8.6 allows us to deduce that:

Corollary 8.1 *$L^G(\mathsf{PN})$ and $L^G(\mathsf{PN{+}NBA})$ are not full* AFL*.*

On the other hand, it is easy to show that:

Theorem 8.8 *$L^G(\mathsf{PN{+}T})$ is a full* AFL*, closed under intersection.*

Proof. We consider two $\mathsf{PN{+}T}$ $\mathcal{N}_1 = \langle P_1, T_1, \Sigma_1, \mathbf{m}_0^1 \rangle$ and $\mathcal{N}_2 = \langle P_2, T_2, \Sigma_2, \mathbf{m}_0^2 \rangle$ and two upward-closed sets U_1 and U_2, and we assume that the set of places and transitions of this two nets are disjoint. For each property to prove we show how to build an upward-closed set U and a $\mathsf{PN{+}T}$ $\mathcal{N} = \langle P, T, \Sigma, \mathbf{m}_0 \rangle$ s.t. $L(\mathcal{N}, U)$ is the desired language. Since the proofs that $\mathcal{N}$ accepts the right language are quite immediate, we do not provide them here. We rather report the main ideas of the construction which should be clear enough to convince the reader.

Union: $L(\mathcal{N}_1, U_1) \cup L(\mathcal{N}_2, U_2) \in L^G(\mathsf{PN{+}T})$. We build $\mathcal{N}$ as follows. $P = P_1 \uplus P_2 \uplus \{p_{init}, p_1, p_2\}$. For each transition $t = \langle I, O, s, d, b, \lambda \rangle \in T_1$, we put in T a transition $t' = \langle I \cup \{p_1\}, O \cup \{p_1\}, s, d, b, \lambda \rangle$. Symmetrically, for each transition $t = \langle I, O, s, d, b, \lambda \rangle \in T_2$, we put in T a transition $t' = \langle I \cup \{p_2\}, O \cup \{p_2\}, s, d, b, \lambda \rangle$. We also add to T two transitions $t_1 = \langle \{p_{init}\}, O_1, \perp, \perp, 0, \varepsilon \rangle$ and $t_2 = \langle \{p_{init}\}, O_2, \perp, \perp, 0, \varepsilon \rangle$ where $O_1(p) = \mathbf{m}_0^1(p)$ for all $p \in P_1$, $O_1(p_1) = 1$ and $O_1(p) = 0$ for all $p \in P_2 \cup \{p_2\}$; and $O_2(p) = \mathbf{m}_0^2(p)$

for all $p \in P_2$, $O_2(p_2) = 1$ and $O_2(p) = 0$ for all $p \in P_1 \cup \{p_1\}$. We let $\Sigma = \Sigma_1 \cup \Sigma_2$. The accepting upward-closed set is:

$$U = \big\{\mathbf{m} \mid \mathbf{m} \in_{P_1} U_1\big\} \cup \big\{\mathbf{m} \mid \mathbf{m} \in_{P_2} U_2\big\}$$

where $\mathbf{m} \in_P U$ means that the projection of the marking $\mathbf{m}$ on the set of places P is in U. More precisely, let $\mathbf{m}' : P \mapsto \mathbb{N}$ be the marking s.t. for any $p \in P$: $\mathbf{m}'(p) = \mathbf{m}(p)$. Then, $\mathbf{m} \in_P U$ iff $\mathbf{m}'$ is in U. Remark that $\{\mathbf{m} \mid \mathbf{m} \in_{P_1} U_1\}$ is upward-closed because U_1 is upward-closed. Similarly, $\{\mathbf{m} \mid \mathbf{m} \in_{P_2} U_2\}$ is upward-closed too. We conclude that U is upward-closed because the union of two upward-closed sets is an upward-closed set. Finally, we let $\mathbf{m}_0$ be s.t. $\mathbf{m}_0(p_{init}) = 1$ and $\mathbf{m}_0(p) = 0$ for any $p \neq p_{init}$.

It is not difficult to see that $\mathcal{N}$ accepts exactly $L(\mathcal{N}_1, U_1) \cup L(\mathcal{N}_2, \mathcal{U}_2)$. Indeed, any transition of $\mathcal{N}$ that corresponds to a transition of $\mathcal{N}_1$ (resp. $\mathcal{N}_2$) can be fired only if there is a token in p_1 (p_2). In the initial marking, only t_1 and t_2 are enabled. Firing t_1 puts a token in p_1 which enables the sub-net that corresponds to $\mathcal{N}_1$ (and accepts words from $L(\mathcal{N}_1, U_1)$ only). Symmetrically, t_2 enables the subnet that corresponds to $\mathcal{N}_2$.

Concatenation: $L(\mathcal{N}_1, U_1) \cdot L(\mathcal{N}_2, U_2) \in L^G(\mathsf{PN{+}T})$. We build $\mathcal{N}$ as follows. $P = P_1 \uplus P_2 \uplus \{p_1, p_2\}$. For any transition $t = \langle I, O, s, d, b, \lambda\rangle$ in T_1, we put in T a transition $t' = \langle I \cup \{p_1\}, O \cup \{p_1\}, s, d, b, \lambda\rangle$. For each transition $t = \langle I, O, s, d, b, \lambda\rangle$ in T_2, we put in T a transition $t' = \langle I \cup \{p_2\}, O \cup \{p_2\}, s, d, b, \lambda\rangle$. We also add to T a transition $t_{\mathbf{m}}$ for any $\mathbf{m} \in \mathsf{UGen}\,(U_1)$, where $t_{\mathbf{m}} = \langle I, O, \bot, \bot, 0, \varepsilon\rangle$ s.t.:

$$\forall p \in P : I(p) = \begin{cases} 1 & \text{if } p = p_1 \\ \mathbf{m}(p) & \text{if } p \in P_1 \\ 0 & \text{otherwise} \end{cases} \qquad O(p) = \begin{cases} 1 & \text{if } p = p_2 \\ \mathbf{m}_0^2(p) & \text{if } p \in P_2 \\ 0 & \text{otherwise} \end{cases}$$

Notice that following Lemma 2.8 and since $\preccurlyeq$ is a WQO, $\mathsf{UGen}\,(U_1)$ is finite. Hence, we only add a finite number of transitions $t_{\mathbf{m}}$.

We also let $\Sigma = \Sigma_1 \cup \Sigma_2$. The initial marking $\mathbf{m}_0$ is s.t.

$$\forall p \in P : \mathbf{m}_0(p) = \begin{cases} 1 & \text{if } p = p_1 \\ \mathbf{m}_0^1(p) & \text{if } p \in P_1 \\ 0 & \text{otherwise} \end{cases}$$

Finally, the accepting upward-closed set U is: $U = \big\{\mathbf{m} \mid \mathbf{m} \in_{P_2} U_2\big\}$

It is rather straightforward to see that $L(\mathcal{N}, U) = L(\mathcal{N}_1, U_1) \cdot L(\mathcal{N}_2, U_2)$. Indeed, in the initial marking, a token is present in p_1, which enables the transitions that corresponds to those of $\mathcal{N}_1$ but no token is present in p_2, which inhibits all the transitions that correspond to transitions of $\mathcal{N}_2$. Moreover, $\mathbf{m}_0$ corresponds to $\mathbf{m}_0^1$ as far as the places of $\mathcal{N}_1$ are concerned. Hence, a sequence of transitions that accepts a word from $L(\mathcal{N}_1, U_1)$ can be fired from $\mathbf{m}_0$. When a marking that corresponds to an accepting marking of $\mathcal{N}_1$ is reached, one of the $t_{\mathbf{m}}$ transitions can fire (and they can fire in this case only). This is due to Lemma 2.8 and $\preccurlyeq$ is a WQO that ensure that all the accepting markings of $\mathcal{N}_1$ are greater to at least one $\mathbf{m} \in \mathsf{UGen}(U_1)$ (and only those markings are). This firing moves the token from p_1 to p_2 and creates a marking that corresponds to $\mathbf{m}_0^2$ on the places of $\mathcal{N}_2$. This inhibits the subnet that corresponds to $\mathcal{N}_1$ and enables the subnet that corresponds to $\mathcal{N}_2$. That subnet is then ready to accept a word from $L(\mathcal{N}_2, U_2)$.

Intersection: $L(\mathcal{N}_1, U_1) \cap L(\mathcal{N}_2, U_2) \in L^G(\mathsf{PN{+}T})$. We build $\mathcal{N}$ as follows. For any transition t, let λ_t be the label of t. We let

$$P = P_1 \uplus P_2 \uplus \{p_{lock}\} \uplus \{p_{t_1,t_2} \mid t_1 \in T_1 \wedge t_2 \in T_2 \wedge \lambda_{t_1} = \lambda_{t_2} \neq \varepsilon\}$$

That is, P contains all the places of $\mathcal{N}_1$ and $\mathcal{N}_2$, a special place p_{lock} that we will use to inhibit transitions of $\mathcal{N}$, and a place p_{t_1,t_2} per pair of transitions from $\mathcal{N}_1$ and $\mathcal{N}_2$ that have the same label (different from ε).

For each ε-labelled transition $t = \langle I, O, s, d, b, \varepsilon\rangle$ of $T_1 \cup T_2$, we add to T the transition $t' = \langle I \cup \{p_{lock}\}, O \cup \{p_{lock}\}, s, d, b, \varepsilon\rangle$. Thus, t' can fire if and only if a token is present in place p_{lock}. Beside this, its effect is the same as in $\mathcal{N}_1$ or $\mathcal{N}_2$.

For every transition $t_1 = \langle I_1, O_1, s_1, d_1, b_1, \lambda_1\rangle$ of T_1 and every transition $t_2 = \langle I_2, O_2, s_2, d_2, b_2, \lambda_2\rangle$ of T_2 s.t. $\lambda_1 = \lambda_2 \neq \varepsilon$, we add to T two transitions $t = \langle I_1 \cup \{p_{lock}\}, O_1 \cup \{p_{t_1,t_2}\}, s_1, d_1, b_1, \lambda_1\rangle$ and $t' = \langle I_2 \cup \{p_{t_1,t_2}\}, O_2 \cup \{p_{lock}\}, s_2, d_2, b_2, \varepsilon\rangle$. Remark that these two transitions t and t' are meant to fire sequentially, and that, once t has fired, no other transition can fire before the corresponding t' fires (because t consumes the token in p_{lock}).

The initial markings is $\mathbf{m}_0$ defined as follows:

$$\forall p \in P : \mathbf{m}_0(p) = \begin{cases} \mathbf{m}_0^1(p) & \text{if } p \in P_1 \\ \mathbf{m}_0^2(p) & \text{if } p \in P_2 \\ 1 & \text{if } p = p_{lock} \\ 0 & \text{otherwise} \end{cases}$$

The accepting upward-closed set is defined as:

$$U = \{\mathbf{m} \mid \mathbf{m} \in_{P_1} U_1 \text{ and } \mathbf{m} \in_{P_2} U_2 \text{ and } \mathbf{m}(p_{lock}) \geq 1\}$$

U is indeed upward-closed. Let $\mathbf{m}_1$ and $\mathbf{m}_2$ be two markings s.t. $\mathbf{m}_1 \in U$ and $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$, and let us show that $\mathbf{m}_2 \in U$. Since $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$, we have (i) for any $p \in P_1$: $\mathbf{m}_1(p) \leq \mathbf{m}_2(p)$; (ii) for any $p \in P_2$: $\mathbf{m}_1(p) \leq \mathbf{m}_2(p)$; and (iii) $\mathbf{m}_1(p_{lock}) \leq \mathbf{m}_2(p_{lock})$. Since U_1 is upward-closed and since $\mathbf{m}_1 \in_{P_1} U_1$, point (i) implies that $\mathbf{m}_2 \in_{P_1} U_1$. Similarly, we deduce that $\mathbf{m}_2 \in_{P_2} U_2$ from point (ii). Finally, since $1 \leq \mathbf{m}_1(p_{lock})$, we have $1 \leq \mathbf{m}_2(p_{lock})$. Hence $\mathbf{m}_2 \in U$.

It is not difficult to see that $L(\mathcal{N}, U) = L(\mathcal{N}_1, U_1) \cap L(\mathcal{N}_2, U_2)$. Indeed, for any pair of transitions t_1 and t_2 respectively from $\mathcal{N}_1$ and $\mathcal{N}_2$ that have the same label, there are two transitions in $\mathcal{N}$ that, when fired sequentially, have the same effect than t_1 and

t_2 on their respective input and output places. The place p_{lock} ensures that the two transitions of $\mathcal{N}$ that correspond to t_1 and t_2 will fire sequentially. The transitions of $\mathcal{N}_1$ and $\mathcal{N}_2$ that are labelled by ε do not require any synchronisation and can thus fire independently. Hence, any pair of executions of $\mathcal{N}_1$ and $\mathcal{N}_2$ that have the same label can be simulated by an execution of $\mathcal{N}$, and any execution of $\mathcal{N}$ (ending in a marking $\mathbf{m}$ s.t. $\mathbf{m}(p_{lock}) = 1$) corresponds to a pair of executions of $\mathcal{N}_1$ and $\mathcal{N}_2$ with the same label.

Iteration: $L^+(\mathcal{N}_1, U_1) \in L^G(\mathsf{PN{+}T})$. The idea is similar to the construction for the concatenation. Let us assume that $P_1 = \{p_1, p_2, \ldots p_n\}$. We build $\mathcal{N}$ as follows. The set of places $P = P_1 \uplus \{p_{lock}, p_{Tr}, p'_1, p'_2, \ldots p'_n\}$. The set of transitions is:

$$\begin{array}{rcl} T & = & \{\langle I \cup \{p_{lock}\}, O \cup \{p_{lock}\}, s, d, b, \lambda\rangle \mid \langle I, O, s, d, b, \lambda\rangle \in T_1\} \\ & \uplus & \{t_{\mathbf{m}} \mid \mathbf{m} \in \mathsf{Min}^{\preccurlyeq}(U_1)\} \\ & \uplus & \{t'_1, t'_2, \ldots, t'_n\} \end{array}$$

where the transitions t'_i and $t_{\mathbf{m}}$ are defined as follows. For any $1 \leq i < n$, $t'_i = \langle \{p'_i\}, O_i, p_i, p_{Tr}, +\infty, \varepsilon\rangle$ with:

$$\forall p \in P : O_i(p) = \begin{cases} \mathbf{m}_0^1(p) & \text{if } p = p_i \\ 1 & \text{if } p = p'_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

The transition t'_n is $\langle \{p'_n\}, O_n, p_n, p_{Tr}, +\infty, \varepsilon\rangle$ with:

$$\forall p \in P : O_n(p) = \begin{cases} \mathbf{m}_0^1(p) & \text{if } p = p_n \\ 1 & \text{if } p = p_{lock} \\ 0 & \text{otherwise} \end{cases}$$

Finally, for every $\mathbf{m} \in \mathsf{Min}^{\preccurlyeq}(U)$, $t_{\mathbf{m}} = \langle I_{\mathbf{m}}, \{p'_1\}, \bot, \bot, 0, \varepsilon\rangle$, with:

$$\forall p \in P : I_{\mathbf{m}}(p) = \begin{cases} \mathbf{m}(p) & \text{if } p \in P_1 \\ 1 & \text{if } p = p_{lock} \\ 0 & \text{otherwise} \end{cases}$$

The initial marking $\mathbf{m}_0$ is s.t. $\mathbf{m}_0(p_{lock}) = 1$, for every $p \in P_1$, $\mathbf{m}_0(p) = \mathbf{m}_0^1(p)$ and for every $p \in P \setminus (P_1 \cup \{p_{lock}\})$, $\mathbf{m}_0(p) = 0$. Finally, the accepting upward-closed set is $U = \{\mathbf{m} \mid \exists \mathbf{m}' \in U_1 : \forall p \in P_1 : \mathbf{m}'(p) \leq \mathbf{m}(p)\}$.

Let us show why the construction is correct. $\mathcal{N}$ contains all the transitions of $\mathcal{N}_1$ (with the same labels), that have been adapted in order to fire only if there is at least one token in p_{lock}, which is true initially. Hence, $\mathcal{N}$ can start its execution by firing a sequence of transitions that is labelled by a word in $L(\mathcal{N}_1, U_1)$ and put into the places of P_1 a marking that is in U_1. At that point, the global marking of $\mathcal{N}$ is thus in U. Thus, the word read so far (which is indeed in $L(\mathcal{N}_1, U_1)^*$) is accepted. Nevertheless, the net can continue its execution, because, once a marking of U has been reached, one of the $t_{\mathbf{m}}$ transitions can fire, by monotonicity. This removes the token from p_{lock}, which inhibits all the (adapted) transitions from $\mathcal{N}_1$. At that point, the only firable sequence of transitions is $t'_1 t'_2 t'_3 \dots t'_n$ (labelled by ε). Each t'_i transition has the effect to restore the initial marking of p_i, by first transferring all the tokens from p_i to p_{Tr} (a trash can place), and then, produce into p_i exactly $\mathbf{m}_0^1(p_i)$ tokens. The last transition t'_n of the sequence also produces a token into p_{lock}, which allows the (adapted) transitions from $\mathcal{N}_1$ to fire anew. Since the initial marking has been restored, a new word from $L(\mathcal{N}_1, U_1)$ can be read. This allows to reach again a marking in U, and so on. Thus, every word in $L(\mathcal{N}_1, U_1)^+$ is in $L(\mathcal{N}, U)$.

On the other hand, in the case where the sequence of (adapted) transitions from $\mathcal{N}_1$ does not produce a marking $\mathbf{m}$ that corresponds to a marking of U_1, then, (i) the marking $\mathbf{m}$ is not in U and is thus not accepting, and (ii) no transition of the form $t_{\mathbf{m}}$ can fire. Hence, the net is blocked until a marking corresponding to an accepting marking of $\mathcal{N}_1$ is reached. We conclude that $L(\mathcal{N}, U) \subseteq L(\mathcal{N}_1, U_1)^+$. Hence, $L(\mathcal{N}, U) = L(\mathcal{N}_1, U_1)^+$.

Arbitrary homomorphism: $h(L(\mathcal{N}_1, \mathcal{U}_1)) \in L^G(\mathsf{PN{+}T})$. Let h be a homomorphism that maps each character a of Σ_1 to a sequence of characters $h(a)$ of an alphabet Σ' (and ε to itself). Again, we denote the label of any transition t by λ_t. We build $\mathcal{N}$ as follows. We let $\Sigma = \Sigma'$. We define the set of places P as:

$$P = P_1 \uplus \{p_{lock}\} \uplus \bigcup_{t \in T_1} \{p_{t,i} | 1 \le i < |h(\lambda_t)|\}$$

As usual, the place p_{lock} is meant to lock the net, i.e., prevent undesired transitions to fire, when necessary. The places $p_{t,i}$ act as intermediary states when reading the word $h(\lambda_t)$ for any $t \in T_1$ with $|h(\lambda_t)| \ge 1$. More precisely, a token in $p_{t,i}$ means that the net has accepted the prefix of length i of $h(\lambda_t)$ so far.

T is built according to these ideas. For any transition $t = \langle I, O, s, d, b, \lambda \rangle$ of T_1, we consider two cases. If $h(\lambda) = \varepsilon$ or $h(\lambda) \in \Sigma_1$, we add to T a single transition $t' = \langle I \cup \{p_{lock}\}, O \cup \{p_{lock}\}, s, d, b, h(\lambda) \rangle$. Otherwise $|h(\lambda)| > 1$, and we assume that $h(\lambda) = w_1 w_2 \cdots w_n$. We add to T the n transitions $t_1, t_2, \dots t_n$ defined as follows. $t_1 = \{I \cup \{p_{lock}\}, O \cup \{p_{t,1}\}, s, d, b, w_1\}$. For any $1 < i < n$, $t_i = \langle \{p_{t,i-1}\}, \{p_{t,i}\}, \perp, \perp, 0, w_i \rangle$. Finally, $t_n = \langle \{p_{t,n-1}\}, \{p_{lock}\}, \perp, \perp, 0, w_n\} \rangle$.

The initial marking $\mathbf{m}_0$ is s.t.:

$$\forall p \in P : \mathbf{m}_0(p) = \begin{cases} \mathbf{m}_0^1(p) & \text{if } p \in P_1 \\ 1 & \text{if } p = p_{lock} \\ 0 & \text{otherwise} \end{cases}$$

The accepting upward-closed set U is $\{\mathbf{m} | \mathbf{m} \in_{P_1} U_1 \text{ and } \mathbf{m}(p_{lock}) \ge 1\}$. For the justification that U is upward-closed, we refer the reader to the arguments used in the case of the intersection.

Clearly, $L(\mathcal{N}, U) = h\big(L(\mathcal{N}_1, U_1)\big)$. Indeed, each transition t of $\mathcal{N}_1$ with label λ and s.t. $h(\lambda) \leq 1$, is replaced by a transition t' with label $h(\lambda)$, that has the same effect on the places of P_1 but which can be fired only if the token is present in p_{lock}. Moreover, each transition t of $\mathcal{N}_1$ with label λ and s.t. $h(\lambda) > 1$ is replaced by a set of transitions that, when fired sequentially, accept $h(\lambda)$ and have the same effect as t on the places of P_1. Thanks to the places of the form $p_{t,i}$ and thanks to p_{lock}, we ensure that these transitions are indeed fired sequentially.

Inverse homomorphism: $h^{-1}(L(\mathcal{N}_1, U_1)) \in L^G(\mathsf{PN{+}T})$. Let Σ' be an alphabet and let h be a homomorphism that maps any word on Σ' to a word on Σ_1. The PN+T $\mathcal{N}$ is built as follows. First of all, we build a PN $\mathcal{N}_o = \langle P_o, T_o, \Sigma_1 \uplus \{\alpha_a \mid a \in \Sigma'\}, \mathbf{m}_0^o\rangle$ that will act as an observer and repeatedly accepts all the words of the form $h(a)$ for any $a \in \Sigma'$.

More precisely, $\mathcal{N}_o$ is defined as follows. Its set of places is:

$$P_o = \{p_{init}\} \uplus \{p_{a,i} \mid a \in \Sigma' \wedge 1 \leq i \leq |h(a)|\}$$

The set of transitions is:

$$T_o = \{t_{a,i} \mid a \in \Sigma' \wedge 1 \leq i \leq |h(a)|\} \cup \{t_a^h \mid a \in \Sigma'\}$$

where, for any $a \in \Sigma'$ s.t. $h(a) = w_1 w_2 \cdots w_n$: (i) $t_a^h = \langle\{p_{a,n}\}, \{p_{init}\}, \perp, \perp, 0, \alpha_a\rangle$; (ii) $t_{a,1} = \langle\{p_{init}\}, \{p_{a,1}\}, \perp, \perp, 0, w_1\rangle$; and (iii) for any $1 < i \leq n$, we let: $t_{a,i} = \langle\{p_{a,i-1}\}, \{p_{a,i}\}, \perp, \perp, 0, w_i\rangle$. Moreover, for any $a \in \Sigma'$ s.t. $h(a) = \varepsilon$, we have $t_a^h = \langle\{p_{init}\}, \{p_{init}\}, \perp, \perp, 0, \alpha_a\rangle$. The initial marking $\mathbf{m}_0^o$ puts a token in p_{init} only. The accepting set is $U_o = \{\mathbf{m} \mid \mathbf{m}(p_{init}) \geq 1\}$. Thus, any accepting sequence of transitions of $\mathcal{N}_o$ is labelled by a word of the form $h(a_1) \cdot \alpha_{a_1} \cdot h(a_2) \cdot \alpha_{a_2} \cdots h(a_n) \cdot \alpha_{a_n}$, where all the a_i's belong to Σ' (remark that it holds when $h(a) = \varepsilon$ too).

The next step amounts to computing a new PN+T $\mathcal{N}'$ and a new upward-closed set U' from $\mathcal{N}_1$, $\mathcal{N}_o$, U_1 and U_o by applying the same procedure as in the case of the intersection, *except that* we treat all the transitions labelled by α_a for some $a \in \Sigma'$ as if they were labelled by ε (in other words, we replace all the α_a labels in $\mathcal{N}_o$ by ε, compute the intersection, then restore the labels. Remember that the ε-labelled transitions are unaffected by the construction we have presented for the intersection. Thus, all the transitions of the form t_a^h appear *as is* in the resulting net). What we obtain is a net that accepts all the words of the form $h(a_1) \cdot \alpha_{a_1} \cdot h(a_2) \cdot \alpha_{a_2} \cdots h(a_n) \cdot \alpha_{a_n}$ such that $h(a_1) \cdot h(a_2) \cdots h(a_n) = h(a_1 \cdot a_2 \cdots a_n)$ is in $L(\mathcal{N}_1, U_1)$. We obtain $\mathcal{N}$ by replacing the label λ_t of any transition t in $\mathcal{N}'$ as follows: if $\lambda_t = \alpha_a$ for $a \in \Sigma'$, we let $\lambda_t = a$, otherwise, we let $\lambda_t = \varepsilon$. We also let $U = U'$. Hence, $L(\mathcal{N}, U)$ is the set of all the words of the form $a_1 \cdot a_2 \cdots a_n$ s.t. $h(a_1 \cdot a_2 \cdots a_n) \in L(\mathcal{N}_1, U_1)$. This is exactly $h^{-1}\big(L(\mathcal{N}_1, U_1)\big)$. □

Remark that the constructions at work in the previous proof involve regular Petri transitions only, except for the case of the iteration. Thus, the above constructions can

	Operation							**Full**
Model	Compl.	$\cup$	Concat.	$\cap$	Iter.	Hom.	Inv. Hom.	AFL
L^G(PN)	$\times$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\surd$	$\times$
L^G(PN+NBA)	$\times$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\surd$	$\times$
L^G(PN+T)	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
L^G(PN+R)	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Table 8.1: Closure properties of L^G(PN), L^G(PN+NBA), L^G(PN+R) and L^G(PN+T).

be re-used to show that L^G(PN), L^G(PN+NBA) and L^G(PN+R) are all closed under union, concatenation, intersection, homomorphism and inverse homomorphism:

Theorem 8.9 L^G(PN), L^G(PN+NBA) *and* L^G(PN+R) *are closed under union, concatenation, intersection, homomorphism and inverse homomorphism.*

Proof. The proof can be established thanks to the constructions used in the proof of Theorem 8.8. □

In the case of the iteration, one needs to add the t'_i transitions, that bear a transfer arc. However, these transitions behave as resets, since they transfer the tokens to the p_{Tr} place, from which the tokens can never escape. Thus, in the case where the original net $\mathcal{N}_1$ is a PN+R, the $\mathcal{N}$ obtained through the construction for the iteration will be a PN+R too. As a consequence, we obtain:

Theorem 8.10 L^G(PN+R) *is a full* AFL*, closed under intersection.*

Proof. The proof is the same as for Theorem 8.8 (simply replace PN+T by PN+R). Remark in particular that the construction used in the case of the iteration involves PN+R transitions only. □

Table 8.1 reports on the closure properties obtained along the present section.

Remark 8.2 L^P(PN+T) *is not a full* AFL*. The justification is the same as in the case of* L^P(WSTS)*. That is, let us consider the language* $L = \{\varepsilon, \mathsf{a}\}$ *on the alphabet* $\Sigma = \{\mathsf{a}, \mathsf{b}\}$*, and the homomorphism* h *s.t.* $h(\mathsf{a}) = \mathsf{bb}$*. Then,* $L \in L^P$(PN+T)*, but* $h(L) = \{\varepsilon, \mathsf{bb}\} \notin L^P$(PN+T) *because it is not prefix-closed (it does not contain the prefix* b *of* bb*).*

8.3.5 Some remarks about the pumping lemmata

It is interesting to compare, on the one hand, Lemma 8.3, and, on the other hand, Lemma 8.5 and Lemma 8.6. Indeed, Lemma 8.3 provides us with a property that holds on any WSL, where Lemma 8.5 and Lemma 8.6 deal with restricted subclasses of WSTS (namely, PN and PN+NBA). Because they focus on these two peculiar classes, these two lemmata allow us to state more precise properties than the one that is given by Lemma 8.3.

In particular, when we restrict ourselves to the class PN, Lemma 8.5 is more general than Lemma 8.3: by letting $w_i = \varepsilon$ for any $i \geq 1$ in Lemma 8.5, we re-obtain Lemma 8.3. Hence, we can obtain thanks to Lemma 8.5 several results[1] that we had previously proved with Lemma 8.3 in section 8.3.1. From our point of view, this is another argument in favour of the interest of Lemma 8.5. A similar conclusion can be drawn when comparing Lemma 8.3 to Lemma 8.6 for the class PN+NBA.

8.3.6 PN+R and Ciardo's conjecture

As in the case of ω-languages of EPN, we have no result to strictly separate $L^G_{\notin}(\mathsf{PN{+}T})$ and $L^G_{\notin}(\mathsf{PN{+}R})$. The only result that stems from the definition is that $L^G_{\notin}(\mathsf{PN{+}R}) \subseteq L^G_{\notin}(\mathsf{PN{+}T})$.

A tentative proof that $L^G_{\notin}(\mathsf{PN{+}T}) \supset L^G_{\notin}(\mathsf{PN{+}R})$ could work as follows: devise a PN+T $\mathcal{N}_t$ and an upward–closed set U of markings, and show that there is no PN+R $\mathcal{N}_r$ and no upward–closed set U' s.t. $L(\mathcal{N}_t, U) = L(\mathcal{N}_r, U')$.

For that purpose, it seems interesting to consider a conjecture stated by Ciardo in [Cia94], that we recall now. In this paper, Ciardo discusses *Petri nets with reset arcs*[2], instead of PN+R.

As stated before, these two formalisms are equivalent from the point of view of coverability. They are also equivalent from the point of view of G–type language, i.e., for any Petri net with reset arc $\mathcal{N}$ and any upward–closed set of accepting markings U, there is a PN+R $\mathcal{N}'$ and an upward–closed set U' s.t. $L(\mathcal{N}, U) = L(\mathcal{N}', U')$, and vice–versa. It is sufficient to transform the reset arc of the Petri net with reset arcs into a reset arc of the PN+R, actually a transfer arc from the place that must be reset, to p_{Tr}. The upward–closed set U is then adapted to take into account p_{Tr}, which can contain any number of tokens.

In [Cia94], however, Ciardo does not consider upward–closed sets of accepting markings, but finite sets of accepting markings, and states the following conjecture regarding the PN+T of Figure 8.6 (whose accepting marking is $\mathbf{m}_a = \langle 0, 0, 0, 1, 0 \rangle$):

Conjecture 8.1 *[Cia94] There is no Petri net with reset arcs $\mathcal{N}$ without ε-transitions and no finite set of markings S s.t. $L(\mathcal{N}, S) = L(\mathcal{N}_5, \{\mathbf{m}_a\})$.*

[1]The results we allude to are: $\mathcal{L} \notin L^G(\mathsf{PN})$, $\mathcal{L}^{\geq} \notin L^G(\mathsf{PN})$, $\mathcal{L}^R \notin L^G(\mathsf{PN})$ and $L^G(\mathsf{PN})$ is not closed under complement.

[2]see Remark 3.3, at the end of Section 3.3.1 for a reminder of the definition of that class.

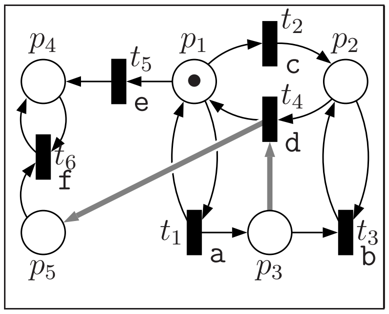


Figure 8.6: The PN+T $\mathcal{N}_5$ used in Ciardo's conjecture stating that there is no Petri net with reset arc $\mathcal{N}$ (without ε-transitions) and no finite accepting set S s.t. $L(\mathcal{N}, S) = L(\mathcal{N}_5, \{\mathbf{m}_a\})$.

Remark that the proof of this conjecture would imply that Petri nets with reset arcs are strictly less expressive that PN+T, for L–type languages, when ε–labelled transitions are disallowed. By definition of Petri nets with reset arcs, this conjecture would also imply that there is no PN+R $\mathcal{N}_r$ (without ε-transitions) and no finite set of accepting markings S_r s.t. $L(\mathcal{N}_r, S_r) = L(\mathcal{N}_5, \{\mathbf{m}_a\})$.

Two natural questions about this conjecture are: 'Is it correct ?' and 'Can we adapt it to prove or conjecture that $L^G_{\not\varepsilon}(\mathsf{PN+R}) \subset L^G_{\not\varepsilon}(\mathsf{PN+T})$?'

The language of $\mathcal{N}_5$ To answer either of these questions, we consider $\mathcal{N}_5$ more carefully. Remark that the language $L(\mathcal{N}_5, \{\mathbf{m}_a\})$ is actually [Cia94]:

$$\left\{\begin{array}{c} \mathtt{a}^{m_1}\mathtt{c}\mathtt{b}^{n_1}\mathtt{d}\mathtt{a}^{m_2}\mathtt{c}\mathtt{b}^{n_2}\mathtt{d}\cdots\mathtt{a}^{m_k}\mathtt{c}\mathtt{b}^{n_k}\mathtt{d}\mathtt{e}\mathtt{f}^{\ell} \\ \text{s.t.} \\ k \in \mathbb{N} \wedge \forall 1 \le i \le k : m_i \ge n_i \wedge \ell = \sum_{i=1}^{k}(m_i - n_i) \end{array}\right\}$$

Indeed, any accepting execution of $\mathcal{N}_5$ can be split into two parts. In a *first part*, a token is initially present in p_1, p_3 is empty, and subsequences of the form $\sigma_i = t_1^{m_i} t_2 t_3^{n_i} t_4$ (labelled by $\mathtt{a}^{m_i}\mathtt{c}\mathtt{b}^{n_i}\mathtt{d}$) may be fired successively (with $m_i \ge n_i$). Each time t_1 fires, a token is added to p_3. Hence, p_3 counts the number of a's in the current σ_i. That is, when t_2 fires, the markings of p_3 is exactly m_i. The firing of t_2 moves the token to p_2 and t_3 (labelled by b) can fire at most m_i times, which removes one token at a time from p_3. Thus, when t_4 fires, p_3 contains $m_i - n_i$. That amount of tokens is *transferred* to p_5. At that point, the token is back in p_1, p_3 is empty, and the next subsequence σ_{i+1} can be fired. After the firing of $\sigma_1 \cdot \sigma_2 \cdots \sigma_k$, the token is in p_1, places p_2, p_3 and p_4 are empty, and p_5 contains exactly $\sum_{i=1}^{k}(m_i - n_i)$ tokens.

The net can then enter the *second part* of its execution by firing t_5 (labelled by e) which moves the token to p_4. At that point, the only possible execution to reach $\mathbf{m}_a$ is to fire t_6 (labelled by f) exactly $\ell = \sum_{i=1}^{k}(m_i - n_i)$ times, because ℓ is the amount of tokens in p_5, and t_6 removes exactly one token at a time from p_5.

Counter–example to the conjecture Unfortunately, the conjecture is wrong: the Petri net with reset arcs (and without ε-transitions) $\mathcal{N}_6$ of Figure 8.7 is such that $L\big(\mathcal{N}_6, \{\mathbf{m}_a\}\big) = L\big(\mathcal{N}_5, \{\mathbf{m}_a\}\big)$, as shown by Proposition 8.7 hereunder. Remark that, in this figure, we have adopted the same graphical convention as in the case of $\mathsf{PN+R}$. That is, we represent reset arcs by an edge bearing a cross. It is also important to bear in mind that there is no trashcan place in this net.

Proposition 8.7 $L(\mathcal{N}_6, \{\mathbf{m}_a\}) = L(\mathcal{N}_5, \{\mathbf{m}_a\})$.

Proof. First, remark that $\mathcal{N}_5$ and $\mathcal{N}_6$ have a very similar structure. Let p_i be a place of $\mathcal{N}_5$. We say that the place $\overline{p}_i$ is its *corresponding* place in $\mathcal{N}_6$ (and vice-versa: there is thus a one-to-one correspondence between the places of $\mathcal{N}_5$ and $\mathcal{N}_6$). Similarly, to each transition t_i of $\mathcal{N}_5$, corresponds one and only one transition $\overline{t}_i$ of $\mathcal{N}_6$. Thus, to any sequence of transitions σ of $\mathcal{N}_5$ corresponds one and only one sequence of transitions $\overline{\sigma}$ of $\mathcal{N}_6$, with $\Lambda(\sigma) = \Lambda(\overline{\sigma})$. We prove that $L(\mathcal{N}_6, \{\mathbf{m}_a\}) \supseteq L(\mathcal{N}_5, \{\mathbf{m}_a\})$ and $L(\mathcal{N}_6, \{\mathbf{m}_a\}) \subseteq L(\mathcal{N}_5, \{\mathbf{m}_a\})$ independently:

$\boxed{L(\mathcal{N}_6, \{\mathbf{m}_a\}) \supseteq L(\mathcal{N}_5, \{\mathbf{m}_a\})}$ Let us consider the generic accepting sequence of transitions $\sigma = t_1^{m_1} t_2 t_3^{n_1} t_4 t_1^{m_2} t_2 t_3^{n_2} t_4 \cdots t_1^{m_k} t_2 t_3^{n_k} t_4 t_5 t_6^{\ell}$ of $\mathcal{N}_5$, with $\ell = \sum_{i=1}^{k}(m_i - n_i)$, and let $\overline{\sigma}$ be its corresponding sequence in $\mathcal{N}_6$. Let us show that $\overline{\sigma}$ is firable and leads to $\mathbf{m}_a$. For any $1 \leq i \leq k$, let $\sigma_i = t_1^{m_i} t_2 t_3^{n_i} t_4$, and let $\overline{\sigma}_i$ be its corresponding sequence. Moreover, for any $1 \leq i \leq k$, let $\mathbf{m}_i$ be s.t. $\mathbf{m}_0 \xrightarrow{\sigma_1 \cdots \sigma_i} \mathbf{m}_i$ (where $\mathbf{m}_0$ is the initial marking of $\mathcal{N}_5$), and let $\overline{\mathbf{m}}_i$ be s.t. $\overline{\mathbf{m}}_0 \xrightarrow{\overline{\sigma}_1 \cdots \overline{\sigma}_i} \overline{\mathbf{m}}_i$ (where $\overline{\mathbf{m}}_i$ is the initial marking of $\mathcal{N}_6$). According to [Cia94], we know that for any $1 \leq i \leq k$: $\mathbf{m}_i\big(\{p_2, p_3, p_4\}\big) = 0$, $\mathbf{m}_i(p_1) = 1$ and $\mathbf{m}_i(p_5) = \sum_{j=1}^{i}(m_j - n_j)$. It is easy to show by induction on i that for any $1 \leq i \leq k$: $\overline{\mathbf{m}}_i = \mathbf{m}_i$. Hence $\overline{\sigma}_1 \cdots \overline{\sigma}_k$ is firable in $\mathcal{N}_6$ and leads to $\overline{\mathbf{m}}_k = \mathbf{m}_k = \langle 1, 0, 0, 0, \ell \rangle$ with $\ell = \sum_{i=1}^{k}(m_i - n_i)$. From that point, it is easy to see that the sequence $t_5 t_6^{\ell}$ is firable and leads to $\mathbf{m}_a$. Hence, for any sequence of σ of $\mathcal{N}_5$ that lead to $\mathbf{m}_a$, the sequence $\overline{\sigma}$ of $\mathcal{N}_6$ leads to $\mathbf{m}_a$ with $\Lambda(\sigma) = \Lambda(\overline{\sigma})$. Thus, $L(\mathcal{N}_6, \{\mathbf{m}_a\}) \supseteq L(\mathcal{N}_5, \{\mathbf{m}_a\})$.

$\boxed{L(\mathcal{N}_6, \{\mathbf{m}_a\}) \subseteq L(\mathcal{N}_5, \{\mathbf{m}_a\})}$ The other direction can be proved by similar arguments and is thus omitted.

We conclude that $L(\mathcal{N}_6, \{\mathbf{m}_a\}) = L(\mathcal{N}_5, \{\mathbf{m}_a\})$. □

Thus, Ciardo's proposal is not suitable to strictly separate Petri nets with reset arcs (nor $\mathsf{PN+R}$) from $\mathsf{PN+T}$, when considering finite sets of accepting markings and disallowing ε–labelled transitions. Still, can we use it to prove that $L^G_{\not\varepsilon}(\mathsf{PN+R}) \subset L^G_{\not\varepsilon}(\mathsf{PN+T})$? For that purpose, one should devise an upward–closed set U s.t. there is no $\mathsf{PN+R}$ $\mathcal{N}'$ and no upward–closed set U' with $L(\mathcal{N}', U') = L(\mathcal{N}_5, U)$. At that point, it is not clear to us whether such an upward–closed set exists (for instance, neither $\mathbb{N}^5$ nor $\uparrow(\{\mathbf{m}_a\})$ are suitable).

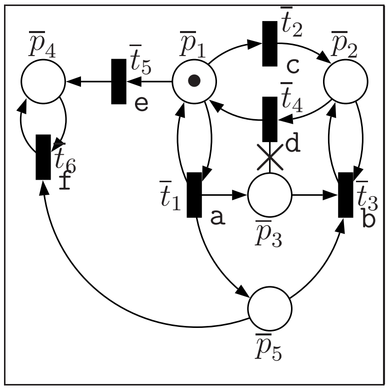


Figure 8.7: The Petri net with reset arcs $\mathcal{N}_6$ used in the proof that Ciardo's conjecture does not hold.

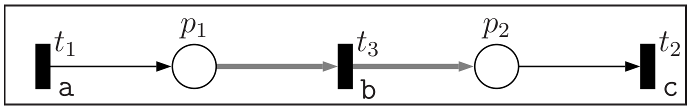


Figure 8.8: The PN+T $\mathcal{N}_7$: a new conjecture to separate PN+T and PN+R.

A new conjecture Nevertheless, we suggest $\mathcal{N}_7$ (see Figure 8.8) with accepting set of markings $\{\langle 0, 0\rangle\}$ as a new candidate of PN+T whose language cannot be accepted by a PN+R without ε-transitions:

Conjecture 8.2 *There is no* PN+R $\mathcal{N}$ *without ε-transitions and no finite set of markings S s.t. $L(\mathcal{N}, S) = L(\mathcal{N}_7, \{\langle 0, 0\rangle\})$.*

From our point of view, this net can also be useful in order to strictly separate the expressive powers of PN+R and PN+T, when we consider $\preccurlyeq$–upward–closed sets of accepting markings and disallow ε–labelled transitions:

Conjecture 8.3 *There is no* PN+R $\mathcal{N}$ *without ε-transitions and no upward–closed set of markings U s.t. $L(\mathcal{N}, U) = L(\mathcal{N}_7, \uparrow(\{\langle 0, 0\rangle\}))$.*

8.4 Discussion

In this chapter, we have addressed the expressiveness of WSTS (and various subclasses thereof, such as the different kinds of EPN), in terms of *finite words languages*, when the *set of accepting configurations is upward-closed*. These languages have been coined as *Well-structured languages*. We believe that the choice of upward-closed sets of accepting markings is consistent with the *monotonicity* property of WSTS, one of their main characteristics.

Indeed, that approach has turned to be fruitful. As a remainder, these are the most important results that we have obtained in this chapter:

- As far as the class $L^G(\mathsf{WSTS})$ of well-structured languages is concerned, we shown that it forms a full AFL closed under intersection. We have also proved a pumping lemma (Lemma 8.3), that allows to prove classical results of the literature, and to obtain new ones, very easily in both cases (see Section 8.3.1).
- As far as the class $L^G(\mathsf{PN})$ is concerned, we have proved another pumping lemma (Lemma 8.5), that allows to obtain interesting results about this class. The most notable is surely the strict separation of $L^G(\mathsf{PN})$ and $L^G(\mathsf{PN{+}NBA})$ (see Section 8.3.2).
- A third pumping lemma (Lemma 8.6), specific to $\mathsf{PN{+}NBA}$, has been proved. Here again, we have been able to obtain several results regarding $L^G(\mathsf{PN{+}NBA})$ thanks to this pumping lemma. In particular, we have been able to strictly separate the classes $L^G(\mathsf{PN{+}NBA})$ and $L^G(\mathsf{PN{+}T})$ (Section 8.3.3).

In the proofs of all our pumping lemmata, we have heavily relied on properties that are specific to WSTS: the monotonicity, and the possibility to extract an increasing sequence of configurations from any infinite sequence of configurations (Lemma 2.2). To the best of our knowledge, theses approach had never been exploited when dealing with expressiveness properties of these models.

Several problems remain open, however, such as the separation of $\mathsf{PN{+}T}$ and $\mathsf{PN{+}R}$ (see Section 8.3.6), whether we consider upward-closed or finite sets of accepting markings.

Chapter 9

Conclusion

In the introduction of this thesis, we have thoroughly explained why the development of formal methods to verify computer systems is nowadays a critical issue. To achieve this goal, it is extremely important to identify classes of models that are suitable to formally describe the (often complex) semantics of computer systems such as distributed, communicating and embedded systems. A thorough command of the properties of these models, as well as efficient algorithms to analyse them are necessary.

In this thesis, we have considered a broad class of models, known as the *Well-Structured Transition Systems*. That choice has been motivated along Chapter 2 and Chapter 3:

1. WSTS subsume several well-known classes of models that have been often studied in the literature, and that are known to be useful in practice for modelling various kinds of computer systems. These models are mainly the Lossy channel systems and the Petri nets (and their monotonic extensions, such as EPN, or monotonic SMPN).

2. Several non-trivial properties are decidable on WSTS. The most notable one is the *coverability*, which amounts to decide the reachability of a set of states that has a special form (it is upward-closed). Hence, the verification of many realistic safety properties can be reduced to deciding the coverability problem.

Moreover, as we have seen in Chapter 8, it is quite natural, owing to the monotonicity property of WSTS, to study the class of finite-words languages that are defined by WSTS, and upward-closed sets of accepting states.

At the end of Section 3.5, after our short survey of the literature regarding WSTS, we had identified several open problems regarding that class of systems. Let us briefly review them and recall which new results we have obtained about them. These new results often raise new questions. We thus seize the opportunity of this concluding chapter to suggest open problems and possible directions for future research.

The lack of a forward algorithm for the coverability problem

As we have seen in Chapter 3, the only general algorithm to answer coverability on WSTS was hitherto the *backward algorithm* of [ACJT96]. The main drawback of a backward algorithm, compared to a forward traversal, is its poor efficiency.

In Chapter 4 and Chapter 5 we have described, from a theoretical and a practical point of view, *Expand, Enlarge and Check*, an efficient forward algorithm to decide coverability on any WSTS (under some reasonable effectiveness requirements).

The practical efficiency of this solution has been made possible partly because of the efficient algorithm to decide coverability on finite WSTS, introduced in Section 5.3. That algorithm has allowed us to speed up the *Expand* phase of EEC. In some cases, when the And-Or graph is degenerated, the *Enlarge* phase can be improved too by this method. In the general case however, the whole portion of the And-Or graph that is reachable from its initial node has to be built in order to check for the avoidability of the upward-closed set. That graph can in some cases be huge and intractable in practice. It would thus be interesting to devise a more efficient method to decide whether a given upward-closed set is avoidable in an And-Or graph. Such a method should work *without building the whole graph*. For that purpose, it could exploit the monotonicity property of WSTS in order to prune branches of the And-Or graph that do not have to be explored (and hence, do not have to be built). Remark that it is the same monotonicity property that has allowed us to obtain an efficient algorithm for finite WSTS in Section 5.3. The algorithms presented in [LS98] (that have already been exploited in a similar context, see [CDF$^+$05]) could be relevant for that problem.

The efficient computation of the minimal coverability set for Petri nets

Although this is not true for the general case of WSTS, the covering set is computable in the case of Petri nets. Algorithms to compute the covering set actually compute a *coverability set*, which is a representation thereof. The most famous algorithm to compute a coverability set of a Petri net is the Karp&Miller algorithm [KM69]. The main drawback of this solution is that it turns out to be rather inefficient in practice.

In [Fin91], the Minimal Coverabiliy Tree algorithm has been introduced as an optimisation of the Karp&Miller algorithm. As we have seen in Chapter 6, that algorithm is not correct. This is unfortunately also the case for a variation of the MCT, introduced in [Lut95], implemented in the Pep tool, and supposed to correct the bug. In the case of the MCT, an under-approximation might be computed. In the case of the algorithm implemented in Pep, the termination is not guaranteed.

In Chapter 6, we have introduced a new algorithm to compute a coverability set of Petri nets that is more efficient in practice than the Karp&Miller procedure. Its optimisation relies on the ordering $\sqsubseteq$ that compares *pairs of markings*.

This algorithm has been discussed mainly from the theoretical point of view (albeit the practical evaluation we have provided at the end of the chapter). This means that several open questions remain regarding the (efficient) implementation of this algorithm. In particular, it remains to devise efficient data-structures to store and maintain transitively closed sets of pairs of markings, by means of their $\sqsubseteq$- maximal pairs only (see Section 6.3.6 for a more detailed discussion on these issues).

The lack of results about the expressive power of WSTS

Unlike other models of computations (such as finite or push-down automata), the study of the expressive powers of WSTS is still incomplete. The expressive power of PN, PN+T and PN+R on finite words and for finite sets of accepting states, has been fairly well characterised (see for instance [Pet81, Cia94, Jan86]). However, there is a lack of results when upward-closed sets of accepting configurations are considered, or for the general class of WSTS. Moreover, ω-languages have seldom been looked into.

In Chapter 7 and Chapter 8, we have partly completed these results:

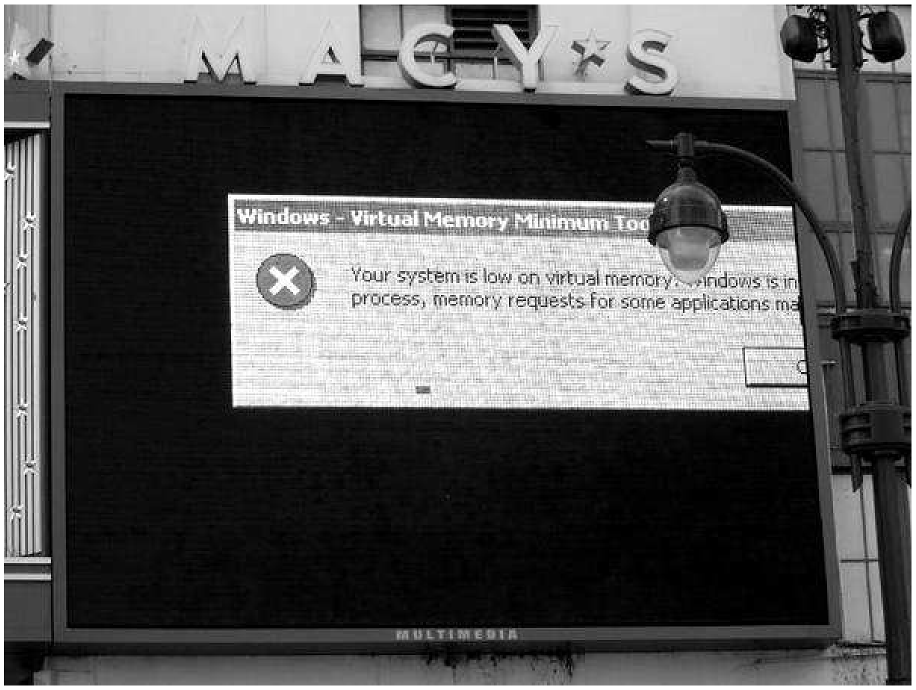
- As far as **ω-words** are concerned, we have studied and compared the respective expressive powers on ω-words of PN, PN+T, PN+NBA and PN+R, in Chapter 7. We have strictly separated the classes of languages that are definable by these models: PN+T are strictly more expressive that PN+NBA, which are in turn strictly more expressive than PN. As far as PN+R are concerned, they are strictly more expressive than PN+NBA. However, their relationship to PN+T still has to be characterised precisely. By allowing the use of silent transitions, one can simulate a PN+T thanks to a PN+R. The question remains open when silent transitions are not allowed.

- As far as **finite words** are concerned, we have defined and studied the class of *well-structured languages*, in Chapter 8. These languages are finite words languages that are accepted by WSTS when upward-closed sets of accepting states are considered. That class forms a full AFL and enjoys several positive properties, such as the decidability of emptiness. The peculiar characteristic of the sets of accepting states (upward-closed) seems also very naturally suited to WSTS, because they are monotonic.

  The main results of Chapter 8 are the *three pumping lemmata* (respectively on WSTS, PN and PN+NBA) that we have introduced. These lemmata allow us to obtain several new results on the classes WSL, L^G(PN), L^G(PN+NBA) and L^G(PN+T), and to (re-)prove, in a very easy way, classical results from the literature. In particular, we have been able to strictly separate the expressiveness of sub-classes of EPN: L^G(PN) is a strict subset of L^G(PN+NBA), which is a strict subset of L^G(PN+T). Unfortunately, as in the case of ω-languages, the relationship of L^G(PN+R) to L^G(PN+T) remains fuzzy (see Section 8.3.6 for a short discussion of this problem). This is thus still an open problem.

To conclude this work on WSTS, let us remark that these results would be rather useless without efficient procedures to *extract* models from the objects that have to be analysed. These can be programs (written in plain programming language such as C, C++, Ada,...), high-level description of systems (such as sequence charts), and so forth. A large effort has been recently carried out by the community of computer-aided verification in this direction. These efforts have produced tools such as Blast [HJMS03] or Slam [BR02]. They are able to analyse *directly* programs written in C, for instance, by means of powerful *abstraction methods* that extract infinite-state models from the source code.

Much effort, however, remains to be done in this direction. But this is clearly beyond the scope of this thesis...



Your system is low on virtual memory. Windows is in
process, memory requests for some applications ma

The large flat screens that display advertisement in front of some department stores are controlled by computers too. An unexpected crash of the controller can sometimes be embarrassing for the advertiser. This one displays a typical error message of the Microsoft Windows operating system. (Source: `http://www.fprintf.net/dang/blog/`)

Bibliography

[AAB99] Parosh Aziz ABDULLA, Aurore ANNICHINI and Ahmed BOUAJJANI. *Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol.* In Rance CLEAVELAND, ed., *TACAS*, volume 1579 of *Lecture Notes in Computer Science*, pp. 208–222. Springer, 1999. ISBN 3-540-65703-7.

[AABJ04] Parosh Aziz ABDULLA, Aurore ANNICHINI, Ahmed BOUAJJANI and Bengt JONSSON. *Using forward reachability analysis for verification of lossy channel system s. Form. Methods Syst. Des.*, 25(1):pp. 39–65, 2004. ISSN 0925-9856. doi: `http://dx.doi.org/10.1023/B:FORM.0000033962.51898.1a`.

[ABJ98] Parosh Aziz ABDULLA, Ahmed BOUAJJANI and Bengt JONSSON. *On-the-fly analysis of systems with unbounded, lossy fifo channels.* In Alan J. HU and Moshe Y. VARDI, eds., *CAV*, volume 1427 of *Lecture Notes in Computer Science*, pp. 305–318. Springer, 1998. ISBN 3-540-64608-6.

[ABS01] Aurore ANNICHINI, Ahmed BOUAJJANI and Mihaela SIGHIREANU. *Trex: A tool for reachability analysis of complex systems.* In BERRY *et al.* [BCF01], pp. 368–372.

[ACJT96] Parosh Aziz ABDULLA, Karlis CERANS, Bengt JONSSON and Yih-Kuen TSAY. *General Decidability Theorems for Infinite-state Systems.* In *Proceedings of the 11th Annual Symposium on Logic in Comuter Science (LICS'96)*, pp. 313–321. IEEE Computer Society Press, 1996.

[AD94] Rajeev ALUR and David L. DILL. *A Theory of Timed Automata. Theoretical Computer Science*, 126(2):pp. 183–236, 1994.

[ADMN04] Parosh Aziz ABDULLA, Johann DENEUX, Pritha MAHATA and Aletta NYLÉN. *Forward reachability analysis of timed petri nets.* In Yassine LAKHNECH and Sergio YOVINE, eds., *FORMATS/FTRTFT*, volume 3253 of *Lecture Notes in Computer Science*, pp. 343–362. Springer, 2004. ISBN 3-540-23167-6.

[AHK02] Rajeev ALUR, Thomas A. HENZINGER and Orna KUPFERMAN. *Alternating-time temporal logic.* *Journal of the ACM*, 49(5):pp. 672–713, 2002. ISSN 0004-5411. doi: `http://doi.acm.org/10.1145/585265.585270`.

[AJ93] Parosh Aziz ABDULLA and Bengt JONSSON. *Verifying programs with unreliable channels.* In *LICS*, pp. 160–170. IEEE Computer Society, 1993.

[Alh98] Sinan Si ALHIR. *UML in a Nutshell.* O'reilly& Associates, inc., 1998. ISBN 1-56592-448-7.

[Alu99] Rajeev ALUR. *Timed automata.* In Nicolas HALBWACHS and Doron PELED, eds., *CAV*, volume 1633 of *Lecture Notes in Computer Science*, pp. 8–22. Springer, 1999. ISBN 3-540-66202-2.

[AN01] Parosh Aziz ABDULLA and Aletta NYLÉN. *Timed petri nets and bqos.* In José Manuel COLOM and Maciej KOUTNY, eds., *ICATPN*, volume 2075 of *Lecture Notes in Computer Science*, pp. 53–70. Springer, 2001. ISBN 3-540-42252-8.

[AN02] Parosh Aziz ABDULLA and Aletta NYLÉN. *Undecidability of LTL for Timed Petri Nets.* In *Proceedings of the 4th International Workshop on Verification of Infinite-State Systems (INFINITY 2002).* 2002.

[AQR$^+$04] Tony ANDREWS, Shaz QADEER, Sriram K. RAJAMANI, Jakob REHOF and Yichen XIE. *Zing: A model checker for concurrent software.* In Rajeev ALUR and Doron PELED, eds., *CAV*, volume 3114 of *Lecture Notes in Computer Science*, pp. 484–487. Springer, 2004. ISBN 3-540-22342-8.

[Ari96] *ARIANE 5 flight 501 failure.* Technical report, `http://homepages.inf.ed.ac.uk/perdita/Book/ariane5rep.html`, 1996.

[Bar06] Sébastien BARDIN. *Vers un Model Checking avec Accélération Plate des Systèmes Hétérogènes.* Ph.D. thesis, ENS Cachan, France, 2006.

[BCF01] Gérard BERRY, Hubert COMON and Alain FINKEL, eds. *Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings*, volume 2102 of *Lecture Notes in Computer Science.* Springer, 2001. ISBN 3-540-42345-1.

[BCR01] Thomas BALL, Sagar CHAKI and Sriram K. RAJAMANI. *Parameterized verification of multithreaded software libraries.* In Tiziana MARGARIA and Wang YI, eds., *TACAS*, volume 2031 of *Lecture Notes in Computer Science*, pp. 158–173. Springer, 2001. ISBN 3-540-41865-2.

[BFLP03] Sébastien Bardin, Alain Finkel, Jérôme Leroux and Laure Petrucci. *Fast: Fast acceleration of symbolikc transition systems.* In Warren A. Hunt and Fabio Somenzi, eds., *CAV*, volume 2725 of *Lecture Notes in Computer Science*, pp. 118–121. Springer, 2003. ISBN 3-540-40524-0.

[BH05] Jesse D. Bingham and Alan J. Hu. *Empirically efficient verification for a class of infinite-state systems.* In Nicolas Halbwachs and Lenore D. Zuck, eds., *TACAS*, volume 3440 of *Lecture Notes in Computer Science*, pp. 77–92. Springer, 2005. ISBN 3-540-25333-5.

[Bin05] Jesse D. Bingham. *Model Checking Sequential Consistency and Parameterized Protocols.* Ph.D. thesis, The University of British Columbia, Canada, 2005.

[BM99] Ahmed Bouajjani and Richard Mayr. *Model checking lossy vector addition systems.* In Christoph Meinel and Sophie Tison, eds., *STACS*, volume 1563 of *Lecture Notes in Computer Science*, pp. 323–333. Springer, 1999.

[Boi99] B. Boigelot. *Symbolic Methods for Exploring Infinite State Spaces.* Ph.D. thesis, Université de Liège, 1999.

[Boo] Boost *graph library.* Home Page:
`http://www.boost.org/libs/graph/doc/index.html`.

[BR02] Thomas Ball and Sriram K. Rajamani. *The slam project: debugging system software via static analysis.* In *POPL '02: Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 1–3. ACM Press, New York, NY, USA, 2002. ISBN 1-58113-450-9. doi:`http://doi.acm.org/10.1145/503272.503274`.

[Bry86] Randal E. Bryant. *Graph-based Algorithms for Boolean Function Manipulation. IEEE Transaction on Computers*, C-35(8):pp. 667–691, 1986.

[BW95] Bernard Boigelot and Pierre Wolper. *An automata-theoretic approach to presburger arithmetic constraints (extended abstract).* In Alan Mycroft, ed., *SAS*, volume 983 of *Lecture Notes in Computer Science*, pp. 21–32. Springer, 1995. ISBN 3-540-60360-3.

[CDF$^+$05] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim Guldstrand Larsen and Didier Lime. *Efficient on-the-fly algorithms for the analysis of timed games.* In Martín Abadi and Luca de Alfaro, eds., *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pp. 66–80. Springer, 2005. ISBN 3-540-28309-9.

[CE81] Edmund M. Clarke and E. Allen Emerson. *Design and synthesis of synchronization skeletons using branching-time temporal logic.* In Dexter Kozen, ed., *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pp. 52–71. Springer, 1981. ISBN 3-540-11212-X.

[CES83] Edmund M. Clarke, E. Allen Emerson and A. Prasad Sistla. *Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach.* In *POPL*, pp. 117–126. 1983.

[CFI96] Gérard Cécé, Alain Finkel and S. Purushothaman Iyer. *Unreliable channels are easier to verify than perfect channels. Information and Computation*, 124(1):pp. 20–31, 1996.

[CGP99] Edmund M. Clarke, Orna Grumberg and Doron A. Peled. *Model Checking.* MIT Press, 1999. ISBN 0-262-03270-8.

[Cia94] Gianfranco Ciardo. *Petri nets with marking-dependent ar cardinality: Properties and analysis.* In Robert Valette, ed., *Application and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pp. 179–198. Springer, 1994. ISBN 3-540-58152-9.

[DB01] Giorgio Delzanno and Tefvik Bultan. *Constraint-based Verification of Client-server Protocols.* In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming (CP2001)*, volume 2239 of *Lecture Notes in Computer Science*, pp. 286–301. Srpinger, 2001.

[Del00] Giorgio Delzanno. *Automatic verification of parameterized cache coherence protocols.* In E. Allen Emerson and A. Prasad Sistla, eds., *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pp. 53–68. Springer, 2000. ISBN 3-540-67770-4.

[DEP99] Giorgio Delzanno, Javier Esparza and Andreas Podelski. *Constraint-based analysis of broadcast protocols.* In Jörg Flum and Mario Rodríguez-Artalejo, eds., *CSL*, volume 1683 of *Lecture Notes in Computer Science*, pp. 50–66. Springer, 1999. ISBN 3-540-66536-6.

[DFS98] Catherine Dufourd, Alain Finkel and Ph. Schnoebelen. *Reset nets between decidability and undecidability.* In Larsen *et al.* [LSW98], pp. 103–115.

[DR00] Giorgio Delzanno and Jean-François Raskin. *Symbolic representation of upward-closed sets.* In Susanne Graf and Michael I. Schwartzbach, eds., *TACAS*, volume 1785 of *Lecture Notes in Computer Science*, pp. 426–440. Springer, 2000. ISBN 3-540-67282-6.

[DRVB01] Giorgio Delzanno, Jean-François Raskin and Laurent Van Begin. *Attacking symbolic state explosion.* In Berry *et al.* [BCF01], pp. 298–310.

[DRVB02] Giorgio Delzanno, Jean-François Raskin and Laurent Van Begin. *Towards the automated verification of multithreaded java programs.* In Joost-Pieter Katoen and Perdita Stevens, eds., *TACAS*, volume 2280 of *Lecture Notes in Computer Science*, pp. 173–187. Springer, 2002. ISBN 3-540-43419-4.

[DRVB04] Giorgio Delzanno, Jean-François Raskin and Laurent Van Begin. *Covering sharing trees: a compact data structure for parameterized verification. STTT*, 5(2-3):pp. 268–297, 2004.

[EFM99] Javier Esparza, Alain Finkel and Richard Mayr. *On the Verification of Broadcast Protocols.* In *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS'99)*, pp. 352–359. IEEE Computer Society Press, 1999.

[EN98] E. Allen Emerson and Kedar S. Namjoshi. *On Model Checking for Non-deterministic Infinite-state Systems.* In *Proceedings of the 13th Annual Symposium on Logic in Computer Science (LICS '98)*, pp. 70–80. IEEE Computer Society Press, 1998.

[Esp94] Javier Esparza. *On the Decidabilty of Model Checking for Several mu-calculi and Petri Nets.* In *Proceedings of the 19th International Colloquium on Trees in Algebra and Programming (CAAP 1994)*, volume 787 of *LNCs*, pp. 115–129. Springer, 1994.

[Fas] Fast *web page.* `http://www.lsv.ens-cachan.fr/fast/example.php`.

[FGRVB05] Alain Finkel, Gilles Geeraerts, Jean-François Raskin and Laurent Van Begin. *On the omega-language expressive power of extended petri nets. Electronic Notes in Theoretical Computer Science*, 128(2):pp. 87–101, 2005.

[FGRVB06] Alain Finkel, Gilles Geeraerts, Jean-François Raskin and Laurent Van Begin. *On the omega-language expressive power of extended petri nets. Theoretical Computer Science*, 356(3):pp. 374–386, 2006.

[Fin90] Alain Finkel. *Reduction and Covering of Infinite Reachability Trees. Information and Computation*, 89(2):pp. 144–179, 1990.

[Fin91] Alain Finkel. *The minimal coverability graph for petri nets.* In Rozenberg [Roz93], pp. 210–243.

[Fla97] David Flanagan. *Java in a Nutshell (2nd edition).* O'reilly & Associates, inc., 1997. ISBN 1-56592-262-X.

[FRSB02] Alain Finkel, Jean-François Raskin, Mathias Samuelides and Laurent Van Begin. *Monotonic extensions of petri nets: Forward and backward search revisited. Electronic Notes in Theoretical Computer Science*, 68(6), 2002.

[FS01] Alain Finkel and Philippe Schnoebelen. *Well-structured transition systems everywhere! Theoretical Computer Science*, 256(1-2):pp. 63–92, 2001.

[Ger01] Rob Gerth. *Model checking if your life depends on it. a view from Intel's trenches.* In Matthew B. Dwyer, ed., *SPIN*, volume 2057 of *Lecture Notes in Computer Science*, p. 15. Springer, 2001. ISBN 3-540-42124-6.

[Gin75] Seymour Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages.* Elsevier Science Inc., 1975. ISBN 0444105867.

[God96] Partice Godefroid. *Partial–Order Methods for the Verification of Concurrent Systems: An Approach to the State–Explosion Problem*, volume 1032 of *Lecture Notes in Compute Science.* Springer–Verlag, 1996.

[GPVW95] Rob Gerth, Doron Peled, Moshe Vardi and Pierre Wolper. *Simple on–the–fly automatic verification of linear temporal logic.* In Chapman & Hall, ed., *Protocol Specification, Testing and Verification*, pp. 3–18. 1995.

[Gra97a] Bernd Grahlmann. *The pep tool.* In Orna Grumberg, ed., *CAV*, volume 1254 of *Lecture Notes in Computer Science*, pp. 440–443. Springer, 1997. ISBN 3-540-63166-6.

[Gra97b] Mark Grand. *Java Language Reference.* O'reilly, 1997. ISBN 1-56592-326-X.

[GRVB04] Gilles Geeraerts, Jean-François Raskin and Laurent Van Begin. *Expand, enlarge, and check: New algorithms for the coverability problem of wsts.* In Kamal Lodaya and Meena Mahajan, eds., *FSTTCS*, volume 3328 of *Lecture Notes in Computer Science*, pp. 287–298. Springer, 2004. ISBN 3-540-24058-6.

[GRVB05] Gilles Geeraerts, Jean-François Raskin and Laurent Van Begin. *Expand, enlarge and check... made efficient.* In Kousha Etessami and Sriram K. Rajamani, eds., *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pp. 394–407. Springer, 2005. ISBN 3-540-27231-3.

[GRVB06a] Pierre Ganty, Jean-François Raskin and Laurent Van Begin. *A complete abstract interpretation framework for coverability properties of WSTS.* In E. Allen Emerson and Kedar S. Namjoshi, eds., *VMCAI*, volume 3855 of *Lecture Notes in Computer Science*, pp. 49–64. Springer, 2006. ISBN 3-540-31139-4.

[GRVB06b] Gilles GEERAERTS, Jean-François RASKIN and Laurent VAN BEGIN. *Expand, enlarge and check: New algorithms for the coverability problem of WSTS. Journal of Computer and System Sciences*, 72(1):pp. 180–203, 2006.

[GRVB06c] Gilles GEERAERTS, Jean-François RASKIN and Laurent VAN BEGIN. *Well-structured languages. Submitted for publication*, 2006.

[GS92] Steven M. GERMAN and A. Prasad SISTLA. *Reasoning about Systems with Many Processes. Journal of ACM*, 39(3):pp. 675–735, 1992.

[Hen96] Thomas A. HENZINGER. *The theory of hybrid automata.* In *Proceedings of the 11th Symposium on Logic in Computer Science (LICS '96)*, p. 278. IEEE Computer Society, 1996. ISBN 0-8186-7463-6.

[HJMS03] Thomas A. HENZINGER, Ranjit JHALA, Rupak MAJUMDAR and Grégoire SUTRE. *Software verification with blast.* In Thomas BALL and Sriram K. RAJAMANI, eds., *SPIN*, volume 2648 of *Lecture Notes in Computer Science*, pp. 235–239. Springer, 2003. ISBN 3-540-40117-2.

[HKQ03] Thomas A. HENZINGER, Orna KUPFERMAN and Shaz QADEER. *From* pre*historic to* post*modern symbolic model checking. Formal Methods in System Design*, 23(3):pp. 303–327, 2003.

[HLP01] Klaus HAVELUND, Michael R. LOWRY and John PENIX. *Formal analysis of a space-craft controller using spin. IEEE Trans. Software Eng.*, 27(8):pp. 749–765, 2001.

[HMU01] John HOPCROFT, Rajeev MOTWANI and Jeffrey ULLMAN. *Introduction to Automata Theory, Languages, and Computation, second edition.* Addison-Wesley, 2001. ISBN 0201441241.

[HNRW06] Klaus HAVELUND, Manuel NÚÑEZ, Grigore ROSU and Burkhart WOLFF, eds. *Formal Approaches to Software Testing and Runtime Verification, First Combined International Workshops, FATES 2006 and RV 2006, Seattle, WA, USA, August 15-16, 2006, Revised Selected Papers*, volume 4262 of *Lecture Notes in Computer Science*. Springer, 2006. ISBN 3-540-49699-8.

[HP79] John E. HOPCROFT and Jean-Jacques PANSIOT. *On the reachability problem for* 5*-dimensional vector addition systems. Theoretical Computer Science*, 8(2):pp. 135–159, April 1979. ISSN 0304-3975.

[Imm81] Neil IMMERMAN. *Number of quantifiers is better than number of tape cells. Journal of Computer and System Sciences*, 22(3):pp. 384–406, 1981.

[INA] *Ina: Integrated net analyzer.* Home page: `http://www.informatik.hu-berlin.de/lehrstuehle/automaten/ina/manual.html`.

[Jac06] Daniel Jackson. *Dependable Software by Design. Scientific American*, 294(6), June 2006. ISSN 0036-8733.

[Jan86] Matthias Jantzen. *Language theory of petri nets.* In Wilfried Brauer, Wolfgang Reisig and Grzegorz Rozenberg, eds., *Advances in Petri Nets*, volume 254 of *Lecture Notes in Computer Science*, pp. 397–412. Springer, 1986. ISBN 3-540-17905-4.

[JK95] Bengt Jonsson and Lars Kempe. *Verifying safety properties of a class of infinite-state distributed algorithms.* In Pierre Wolper, ed., *CAV*, volume 939 of *Lecture Notes in Computer Science*, pp. 42–53. Springer, 1995. ISBN 3-540-60045-0.

[JM07] Ranjit Jhala and Rupak Majumdar. *Interprocedural analysis of asynchronous programs.* In Martin Hofmann and Matthias Felleisen, eds., *POPL*, pp. 339–350. ACM, 2007. ISBN 1-59593-575-4.

[KM69] Richard M. Karp and Raymond E. Miller. *Parallel Program Schemata. Journal of Computer and System Sciences*, 3:pp. 147–195, 1969.

[LAS] *The Liège Automata-based Symbolic Handler (*lash*).* Available at `http://www.montefiore.ulg.ac.be/~boigelot/research/lash`.

[Lea00] Douglas Lea. *Concurrent Programming in Java.* The Java Series. Addison Wesley, 2000. ISBN 0201695812.

[LL00] Michael Leuschel and Helko Lehmann. *Solving coverability problems of petri nets by partial deduction.* In *PPDP '00: Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming*, pp. 268–279. ACM Press, New York, NY, USA, 2000. ISBN 1-58113-265-4. doi: `http://doi.acm.org/10.1145/351268.351298`.

[LS98] Xinxin Liu and Scott A. Smolka. *Simple linear-time algorithms for minimal fixed points (extended abstract).* In Larsen *et al.* [LSW98], pp. 53–66.

[LSW98] Kim Guldstrand Larsen, Sven Skyum and Glynn Winskel, eds. *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*, volume 1443 of *Lecture Notes in Computer Science*. Springer, 1998. ISBN 3-540-64781-3.

[Lut95] Karsten LUTTGE. *Zustandsgraphen von Petri-Netzen*. Master's thesis, Humboldt-Universität zu Berlin, 1995.

[May84] Ernst W. MAYR. *An algorithm for the general petri net reachability problem. SIAM Journal of Computing*, 3(13):pp. 441–460, 1984.

[McM93] Kenneth L. MCMILLAN. *Symbolic Model Checking: an Approach to the State Explosion Problem*. Kluwer Academic Publishers, 1993. ISBN 0-7923-9380-5.

[Mer74] Philip M. MERLIN. *A study of the recoverability of computing systems*. Ph.D. thesis, University of California, Irvine, CA., 1974.

[Mil89] Robin MILNER. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. ISBN 0-13-115007-3.

[Min67] Marvin MINSKY. *Finite and Infinite Machines*. Englewood Cliffs, N.J., Prentice-Hall, 1967. ISBN 0131655639.

[NIS02] *The economic impacts of inadequate infrastructure for software testing (final report)*. Technical Report 02-3, National Institute of Standards and Technology, May 2002.

[NT93] Leveson NANCY and Clark S. TURNER. *An investigation of the Therac-25 accidents. IEEE Computer*, 26(7):pp. 18–41, July 1993. Updated version at `http://sunnyday.mit.edu/papers/therac.pdf`.

[OW99] Scott OAKS and Henry WONG. *Java Threads (Second Edition)*. O'Reilly & Associates, inc., 1999. ISBN 1565924185.

[Pel94] Doron PELED. *Combining partial order reductions with on-the-fly model-checking*. In David L. DILL, ed., *CAV*, volume 818 of *Lecture Notes in Computer Science*, pp. 377–390. Springer, 1994. ISBN 3-540-58179-0.

[Pet62] Carl Adam PETRI. *Kommunikation mit Automaten*. Ph.D. thesis, Technical University Darmstadt, 1962.

[Pet81] James L. PETERSON. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1981. ISBN 0136619835.

[Pre29] Mojźcesz PRESBURGER. *Über die Vollständigkeit eines gewissen systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt*. In *Comptes Rendus du premier congrès de Mathématiciens des Pays Slaves*, pp. 92–101. Warsaw, 1929.

[Pyt] PYTHON *web page*. `http://www.python.org`.

[PZ91] Louchka Popova-Zeugmann. *On time petri nets. Elektronische Informationsverarbeitung und Kybernetik*, 27(4):pp. 227–244, 1991.

[QS82] Jean-Pierre Queille and Joseph Sifakis. *Specification and verification of concurrent systems in CESAR.* In Mariangiola Dezani-Ciancaglini and Ugo Montanari, eds., *Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pp. 337–351. Springer, 1982. ISBN 3-540-11494-7.

[Ram74] Chander Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets.* Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA., 1974.

[Rei86] Wolfgang Reisig. *Petri Nets. An introduction.* Springer, 1986. ISBN 0-387-13723-8.

[Ric53] Henry G. Rice. *Classes of recursively enumerable sets and their decision problems. Transactions of the American Mathematical Society*, 74(2):pp. 358–366, March 1953.

[Roz93] Grzegorz Rozenberg, ed. *Advances in Petri Nets 1993, Papers from the 12th International Conference on Applications and Theory of Petri Nets, Gjern, Denmark, June 1991*, volume 674 of *Lecture Notes in Computer Science*. Springer, 1993. ISBN 3540566899.

[RSVB03] Jean-François Raskin, Mathias Samuelides and Laurent Van Begin. *Petri games are Monotonic but difficult to analyse.* Technical Report 508, Université Libre de Bruxelles, 2003.

[RV02] Alexandre Riazanov and Andrei Voronkov. *The design and implementation of vampire. AI Commun.*, 15(2-3):pp. 91–110, 2002.

[RVB04] Jean-François Raskin and Laurent Van Begin. *Petri nets with non-blocking arcs are difficult to analyze. Electronic Notes in Theoretical Computer Science*, 98:pp. 35–55, 2004.

[Sal73] Arto Salomaa. *Formal Languages.* Academic Press, 1973. ISBN 0-12-615750-2.

[Sal85] Arto Salomaa. *Computation and Automata*, volume 25 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, 1985. ISBN 0-521-30245-5.

[SLL01] Jeremy G. Siek, Lie-Quan Lee and Andrew Lumsdaine. *Boost Graph Library, The: User Guide and Reference Manual.* Addison Wesley Professional, 2001. ISBN 0201729148.

[STC96] Manuel Silva, Enrique Teruel and José Manuel Colom. *Linear algebraic and linear programming techniques for the analysis of place or transition net systems.* In Wolfgang Reisig and Grzegorz Rozenberg, eds., *Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pp. 309–373. Springer, 1996. ISBN 3-540-65306-6.

[Sti06] Gary Stix. *Send in the Terminator. Scientific American*, 294(12), December 2006. ISSN 0036-8733.

[Tur36] Alan M. Turing. *On computable numbers, with an application to the entscheidungsproblem.* In *Proceedings of the London Mathematical Society*, volume 42 of *2*. 1936.

[Val78] Rüdiger Valk. *On the computational power of extended petri nets.* In Józef Winkowski, ed., *MFCS*, volume 64 of *Lecture Notes in Computer Science*, pp. 526–535. Springer, 1978. ISBN 3-540-08921-7.

[Van03] Laurent Van Begin. *Efficient Verification of Counting Abstractions for Parametric systems.* Ph.D. thesis, Université Libre de Bruxelles, Belgium, 2003.

[VBvdA01] H. M. W. (Eric) Verbeek, Twan Basten and Wil M. P. van der Aalst. *Diagnosing workflow processes using woflan. Comput. J.*, 44(4):pp. 246–279, 2001.

[VRD03] Guido Van Rossum and Fred Drake, eds. *The* Python *Language Reference Manual.* Network Theory Ltd., 2003. ISBN 0954161785.

[VW86] Moshe Y. Vardi and Pierre Wolper. *An automata-theoretic approach to automatic program verification (preliminary report).* In *LICS*, pp. 332–344. IEEE Computer Society, 1986.

[Zam97] Denis Zampuniéris. *The Sharing Tree Data Structure: Theory and Applicationsin Formal Verification.* Ph.D. thesis, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, 1997.

[ZLC95] Denis Zampuniéris and Baudouin Le Charlier. *Efficient Handling of Large Sets of Tuples with Sharing Trees.* In *Proceedings of the Data Compressions Conference (DCC'95)*, p. 428. IEEE Computer Society Press, 1995.

Index