UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
Departamento de Sistemas Informáticos y Computación

TESIS DOCTORAL

Verificación de extensiones de Redes de Petri con precios, tiempo y múltiples instancias

Verification of priced and timed extensions of Petri Nets with multile instances

MEMORIA PARA OPTAR AL GRADO DE DOCTORA

PRESENTADA POR

María Rosa Martos Salgado

Director

Fernando Rosa Velardo

Madrid, 2016

Verificación de Extensiones de Redes de Petri con Precios, Tiempo y Múltiples Instancias

Verification of Priced and Timed Extensions of Petri Nets with Multiple Instances

Tesis Doctoral

María Rosa Martos Salgado

Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid
Noviembre de 2015

Trabajo dirigido por el Doctor Fernando Rosa Velardo

Verificación de Extensiones de Redes de Petri con Precios, Tiempo y Múltiples Instancias

Verification of Priced and Timed Extensions of Petri Nets with Multiple Instances

Memoria que presenta para optar al título de Doctor en Informática

María Rosa Martos Salgado

Dirigida por el doctor

Fernando Rosa Velardo

Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Noviembre de 2015

Verificación de Extensiones de Redes de Petri con Precios, Tiempo y Múltiples Instancias

Memoria presentada por María Rosa Martos Salgado para optar al grado de Doctor por la Universidad Complutense de Madrid, realizada bajo la dirección de D. Fernando Rosa Velardo (Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid).

Verification of Priced and Timed Extensions of Petri Nets with Multiple Instances

Report presented by María Rosa Martos Salgado to the Universidad Complutense de Madrid in order to apply for the Doctor's degree. This work has been supervised by Fernando Rosa Velardo (Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid).

Madrid, Noviembre de 2015

Agradecimientos

En primer lugar, quiero dar las gracias a mi director, Fernando Rosa, por haber guiado mis primeros pasos en el mundo de la investigación. Gracias por su paciencia y su disponibilidad. A menudo el entusiasmo con que una persona se emplea en una tarea depende mucho del entusiasmo que le insuflan sus maestros, y en este caso creo haber tenido un gran ejemplo.

Quiero agradecer a los informantes Ismael Rodríguez Laguna, Natalia Sidorova y Giorgio Delzanno el haber contribuido a la mejora de la tesis con sus constructivos comentarios y correcciones. Gracias también a la comisión de posgrado.

Gracias al DSIC, por ponerlo todo tan fácil y acogerme desde un primer momento. En particular, a David de Frutos y Narciso Martí, que me han brindado su ayuda siempre que la he necesitado. Gracias a los compañeros con los que he compartido pasillo en el aula 15, y en especial a Gabi, por animarnos el uno al otro.

Una de las experiencias más enriquecedoras que me ha dado la tesis son mis estancias en Islandia. Muchas gracias a las personas que me acogieron e hicieron que el tiempo que pasé allí fuera muy agradable, especialmente a Luca Aceto, Anna Ingólfsdóttir e Ignacio Fábregas. Gracias también a Guillermo y a los Reykjavik Raiders, por ayudarme a conocer lo maravillosa que es la gente islandesa.

Gracias a mis segundas familias, en las que siempre he encontrado el ánimo y apoyo que me hacía falta. A No Es Culpa Nuestra (galácticos, pinkis...), a mis dos equipos (mates y CRC) y a mis Jonsuis.

Por último, gracias a mi familia, y sobre todo a mi padre, que desde pequeña me enseñó a tener la confianza en mí necesaria para enfrentarme a cualquier reto.

Esta tesis ha sido financiada por la beca y contrato predoctoral de personal investigador en formación UCM: BE43/11 y por los proyectos DESAFIOS10 TIN2009-14599-C03-01, PROMETIDOS S2009/TIC-1465 y STRONGSOFT TIN2012-39391-C04-04.

Abstract

The model of Petri nets is a formal modeling language which is very suitable for the analysis and verification of infinite-state concurrent systems. In particular, due to its good decidability properties, it is very appropriate to study safety properties over such systems. However, Petri nets frequently lack the expressiveness to represent several essential characteristics of nowadays systems such as real time, real costs, or the managing of several parallel processes, each with an unbounded number of states. Several extensions of Petri nets have been defined and studied in the literature to fix these shortcomings. For example, Timed Petri nets [83, 10] deal with real time and ν-Petri nets [78] are able to represent an unbounded number of different infinite-state processes running concurrently. In this thesis we define new extensions which encompass these two characteristics, and study their decidability properties.

First, we define Timed ν-Petri nets by joining together Timed Petri nets and ν-Petri nets. The new model represents systems in which each process (also called instance) is represented by a different pure name, and it is endowed with an unbounded number of clocks. Then, a clock of an instance must satisfy certain given conditions (belonging to a given interval) in order to take part in the firing of a transition. Unfortunately, we prove that the verification of safety properties is undecidable for this model. In fact, it is undecidable even if we only consider two clocks per process. We restrict this model and define Locally-Synchronous ν-Petri nets by considering only one clock per instance, and successfully prove the decidability of safety properties for this model. Moreover, we study the expressiveness of Locally-Synchronous ν-Petri nets and prove that it is the most expressive non Turing-complete extension of Petri nets with respect to the languages they accept.

Next, we tackle the definition of a Timed-Priced model, by extending Locally-Synchronous ν-Petri nets with costs. We add the prices in the way of [2], by considering firing costs, produced by the executions of actions, and storage costs, produced by the storage and maintenance of resources while time elapses. Then,

we define the safety problem we would like to study: given a budget b, we say that a system is safe if we cannot reach a certain (upward-closed) set of states by spending more than b. Intuitively, b-safety ensures that we are not spending too much in order to reach certain states. Again, we obtain a positive decidability result for safety.

Finally, we apply the previous positive results for the verification of safety and soundness properties for priced and timed workflow nets in which several instances run concurrently in the same net, sharing some global resources. Before defining these new models of workflows, we extend the decidability results about soundness of the so-called resource-constrained workflow nets, by considering a more general definition of soundness. Then, we define two models of Timed-Priced Resource-Constrained Workflow nets: in the first model, we suppose that time elapses while executing actions, and therefore we relate costs to these executions, and to the storage while these actions are executed. In the second model time elapses in between the execution of actions, and therefore storage costs are produced then. In that way, we are able to compute the price of an instance in a run as the sum of the storage and the fired costs produced in the run. As we have multiple instances, each with its price, we can obtain the global price of a run in several ways depending on how we compute it. We consider several ways to compute a global price: the sum, the maximum, the average and the discounted sum of the prices of the instances. We study the decidability of safety in this setting, and prove its decidability for the sum, the maximum, the average and the finite discounted sum for the first model, and the maximum and the finite discounted sum for the second one. Regarding soundness, we prove that it is decidable for Timed-Priced Resource-Constrained Workflow nets with decidable safety and some additional hypothesis, as the soundness of its underlying workflow net without considering the restriction of the behavior caused by the sharing of resources.

Resumen

Las redes de Petri son un lenguaje formal muy adecuado para la modelización, análisis y verificación de sistemas concurrentes con infinitos estados. En particular, son muy apropiadas para estudiar las propiedades de seguridad de dichos sistemas, dadas sus buenas propiedades de decidibilidad. Sin embargo, en muchas ocasiones las redes de Petri carecen de la expresividad necesaria para representar algunas características fundamentales de los sistemas que se manejan hoy en día, como el manejo de tiempo real, costes reales, o la presencia de varios procesos con un número no acotado de estados ejecutándose en paralelo. En la literatura se han definido y estudiado algunas extensiones de las redes de Petri para la representación de las características anteriores. Por ejemplo, las "Redes de Petri Temporizadas" [83, 10](TPN) incluyen el manejo de tiempo real y las ν-redes de Petri [78](ν-PN) son capaces de representar un número no acotado de procesos con infinitos estados ejecutándose concurrentemente. En esta tesis definimos varias extensiones que reúnen estas dos características y estudiamos sus propiedades de decidibilidad.

En primer lugar definimos las "ν-Redes de Petri Temporizadas", que reúnen las características expresivas de las TPN y las ν-PN. Este nuevo modelo es capaz de representar sistemas con un número no acotado de procesos o instancias, donde cada proceso es representado por un nombre diferente, y tiene un número no acotado de relojes reales. En este modelo un reloj de una instancia debe satisfacer ciertas condiciones (pertenecer a un intervalo dado) para formar parte en el disparo de una transición. Desafortunadamente, demostramos que la verificación de propiedades de seguridad es indecidible para este modelo. De hecho, es indecidible incluso en el caso con solo dos relojes por instancia. Así pues, restringimos el modelo y definimos las "ν-PN Localmente Síncronas"(ν-$lsPN$), que manejan un solo reloj por instancia, para las que podemos probar que la verificación de propiedades de seguridad es decidible. Además, estudiamos la expresividad de las ν-$lsPN$, demostrando que son la extensión de las redes de Petri no Turing-

completa más expresiva con respecto a los lenguajes que es capaz de aceptar.

Una vez encontrado un formalismo apropiado para la modelización de sistemas con tiempo, nos disponemos a definir un nuevo modelo con tiempo y costes extendiendo las ν-$lsPN$, añadiendo los costes de la misma manera que se hace en [2]. De esta forma, consideramos dos tipos de coste: los costes de disparo, producidos por la ejecución de acciones, y los costes de almacenamiento, producidos por el almacenamiento y mantenimiento de recursos durante el paso del tiempo. Así, definimos el problema de seguridad que más tarde estudiamos: dado un presupuesto b, decimos que un sistema es seguro si no podemos alcanzar un conjunto (cerrado hacia arriba) de estados dado gastando más de b. Intuitivamente, esta propiedad asegura que no estamos gastando demasiado en alcanzar determinados estados. De nuevo, obtenemos un resultado positivo para la decidibilidad de esta propiedad.

Por último, aplicamos los resultados anteriores en la verificación de propiedades de seguridad y corrección para workflows (flujos de trabajo) con tiempo y costes en las que varias instancias son ejecutadas concurrentemente en la misma red, compartiendo ciertos recursos globales. Antes de definir los nuevos modelos con tiempo y costes, ampliamos los resultados ya existentes sobre "resource-constrained workflow nets" (rcwf, flujos de trabajo restringidos por los recursos), considerando una definición de corrección algo más general que la ya existente. Entonces, definimos dos modelos de rcwf con tiempo y precios: en el primer modelo suponemos que el tiempo pasa durante las ejecuciones de acciones, y por lo tanto los costes de almacenamiento se producen en el momento en el que se ejecutan acciones. En el segundo modelo, el tiempo pasa entre las diferentes ejecuciones de acciones, y por lo tanto los costes de almacenamiento se producen entre estas. De esta forma, podemos calcular el coste de una instancia en una ejecución sumando los costes de disparo y almacenamiento producidos por esta instancia durante la ejecución. Dado que en cada ejecución participan múltiples instancias con su coste asociado, podemos calcular el coste global de la ejecución de diferentes maneras. Así, consideramos diferentes formas de calcular este precio global. En particular, estudiamos la suma, el máximo, la media, y la suma ponderada de los precios de las instancias. De este modo, decimos que una red es segura si el precio global de ninguna ejecución válida supera el presupuesto dado. Finalmente, estudiamos la decidibilidad de la seguridad para los dos modelos anteriormente definidos y los diferentes predicados. Concretamente, probamos que la seguridad es decidible para la suma, el máximo, la media y la suma ponderada finita en el primer mo-

delo, y el máximo y la suma ponderada finita en el segundo modelo. Respecto a la corrección, demostramos que esta es decidible para rcwf con tiempo y precios cuya seguridad es decidible bajo algunas hipótesis adicionales, como que la red sin tiempo ni costes obtenida al eliminar las restricciones producidas por los recursos sea correcta.

Contents

Chapter 1

Introduction

1.1 State of the art

Nowadays software is everywhere. Transports, surgery, communications, banks and many other vital services depend on software, so that we need to rely on the systems we create to manage our lifes. Since non desired behaviors of software may result in the loss of money, resources, time or even lifes, we want to make sure that software systems really do what we need them to do. Several techniques, such as testing or formal verification, have risen for this purpose.

Testing consists on the repeated execution of the system under test, to prove that each of these executions satisfies the requirements that the system is supposed to meet, or to find the bugs that the system may contain. (Informal) testing is performed even before the first versions of the systems under test have been completed, when the coder tests his code in an informal way, frequently without a fixed procedure. Then, formal testing is usually performed by a tester team, which follows some formal procedure in order to select the test cases, run these tests and log the bugs they find. These formal procedures have been deeply studied in the literature, generating a wide variety of different techniques [68, 46, 45, 69, 39]. Even for very simple pieces of software, the amount of possible different tests may be incredibly large. The amount of tests that are performed depend on the time and resources that the testers are provided with. That is why in most of the cases, testing techniques use strategies to select the most convenient set of tests that we are able to perform. Even though, in most of the cases, testing cannot ensure the correctness of the system under test.

On the other hand, formal verification proves the correctness or incorrectness of a system with respect to a formal specification by using mathematical (formal)

techniques. One of the drawbacks of these methods is that a model which correctly represents the real system is required to apply the formal methods over it. We need to trust that this theoretical model behaves as the real system, so that we need formalisms which are expressive enough to deal with the representation of real systems.

Initially, software was created to be executed in an isolated way, without interacting with any other piece of software. However, in nowadays systems, several computations are usually executing concurrently, and interacting with each other. There are plenty of formal models for concurrent systems in the literature. These models should be appropriate to study the coordination, scheduling, protocols, orchestration or data exchangings of processes. Moreover, in distributed systems several components (or processes) run concurrently, communicating and coordinating their actions by using different mechanisms such as messages passing or resources sharing. Frequently related to this, formalisms should be able to deal with time too. Companies want to be sure not only that their systems work correctly, but also that they work in the correct timing, respecting the deadlines. Furthermore, timing is commonly related with concurrency, and delays take a fundamental role in protocols and orchestrations. Finally, systems frequently depend on the managing of resources and carry some costs. Hence, models dealing with quantitative measures are needed too.

Plenty of these formalisms have been defined and studied. For example, dealing with concurrency, we have process algebra [66, 87, 38, 48], which are defined to represent in a high-level manner the interactions and synchronizations of different independent processes, by applying algebraic laws, Petri nets, which provide a very graphical and intuitive view of distributed systems, MSR [20], which is a specification language for security protocols based on multiset rewriting, or MSR(C)[29, 30, 31] for parameterized concurrent systems, which are defined by combining multiset rewriting and constraints. There is also a great variety of timed formalisms. Among them, timed automata [12, 15] are an extension of finite automata in which a certain number of real-time clocks are considered, and the transitions which may or may not be executed in each state depend on the value of these clocks. Moreover, Petri nets have been extended to deal with time [83, 10] and quantitative aspects too [2].

1.1.1 Petri nets and extensions

Petri nets [72, 76, 71, 34] are a modeling language for the specification and verification of distributed systems. They were first defined in the thesis of Carl Adam Petri, in 1962 [72]. Since then, they have been deeply studied and diversely extended. One of the reasons of the success history of Petri nets may be their easiness, provided by their intuitive graphical representation. Moreover, they are expressive enough to model a great variety of systems. In particular, they are able to represent and easily analyze potentially unbounded systems, that is, systems with an unbounded number of states. Petri nets consist of *places*, represented by circles, and *transitions*, represented by boxes. In each place there may be a certain amount of tokens, which are represented by dots. The states of Petri nets are called markings, and they consist in the number of tokens at each place at a certain moment. The markings of a net may be changed by transitions. A transition can happen, or is enabled, if there are enough tokens in certain places, which are called the preconditions of the transition. Then, it can be fired so that tokens are removed from preconditions and added to certain places, called the postconditions of the transition. The preconditions of a transition are specified by arcs going from these places to the transition. Analogously, the postconditions of a transition are represented by arcs going from the transition to these places.

As the number of tokens in a certain place could grow unboundedly with the firings of transitions, Petri nets may represent systems with infinitely many different states, which gives them a greater expressive power than other formalisms, as finite automata. Despite this fact, many important problems are still decidable for Petri nets [34]. Among them are reachability, coverability and boundedness. Reachability [65, 58, 60, 63] is the problem of deciding if a given marking can be reached from the initial marking. Since the goal of the systems we model is frequently to reach a final state, this problem is sometimes important in order to verify that our systems behave as wished. On the other hand, coverability [55, 73], which is the problem of deciding if we can reach a marking which contains at least the same number of tokens in each place as a given marking, is more related to safety problems, that is, to ensure that we do not reach a set of "bad" states that we need to avoid. Hence, when we check coverability we would often like to obtain a negative answer. Boundedness [55, 73, 81] is the problem of deciding if there is a bounded number of reachable markings, and it is often checked to ensure we work with finite systems, when, for example, we are dealing with the consumption of resources.

Although Petri nets are very expressive compared to finite state formalisms, there are still some shortcomings that can be found when representing certain systems with them. For example, transitions cannot copy or transfer the content of some place to another, empty places or check the places for emptiness, missing the capability of representing some behaviors of the systems, like their resetting. Moreover, Petri nets cannot handle more sophisticated characteristics of systems like the representation of time or time constraints, the simulation of a potentially unbounded number of different copies of the same net executing concurrently or the parametric verification of systems in which an unknown number of processes take part, which may be fundamental to model the behavior of some systems. In order to fix these shortcomings, many extensions of Petri nets have been defined and studied in the literature. They provide Petri nets with very heterogeneous tools, such as different firing rules, more complex tokens, or time.

1.1.1.1 Firing rule extensions and token extensions

The firing rules of Petri nets add/remove a certain fixed amount of tokens to/from places. In the literature there are extensions of Petri nets that enrich this token game with more possibilities. For example, in *reset nets* [14], when a transition with a reset arc is fired, all the tokens in the place attached to this arc are removed. In *inhibitor nets* [42], some transitions may only be enabled when certain (inhibited) places are empty.

When adding expressiveness to our models, frequently some of the good decidability properties of Petri nets are lost. For example, nets with two or more inhibited places can simulate Minsky machines with two counters [67], which is a Turing-complete formalism, and therefore they are Turing complete, so that every interesting property becomes undecidable. More precisely, not only reachability, but also coverability and boundedness are undecidable for inhibitor nets. Reset nets are not Turing complete, though reachability and boundedness are not decidable for them.

Another way in which Petri nets can be extended is by enriching the tokens. Maybe the most popular extension of Petri nets is *coloured Petri nets* [49, 50], in which both the firing rules and the tokens are extended. In this extension, tokens carry values of a given data type, are distinguishable and may be modified according to any function annotated in the transitions. Again, coloured Petri nets are Turing complete. One of the reasons for the success of coloured Petri nets is the presence of the powerful tool CPN (http://cpntools.org/) [75, 51], which

provides a verifier for (bounded) coloured Petri nets.

There are simpler extensions that extend tokens, such as *Data Nets* [62] or *ν-Petri Nets* [78]. In Data Nets tokens are ordered and the conditions for firing a transition may depend on that order. For example, to enable a transitions, it may be required that the involved tokens are ordered in a concrete way, or that the tokens are equal. Moreover, data nets can perform whole-place operations like reseting places, or copying the tokens of some place to another. In ν-Petri nets, tokens carry unordered distinguishable names, sometimes called pure names in the literature [79, 101]. When firing a transition, it may be required that the involved tokens carry the same or a different name. Moreover, new names can be created fresh. The feature of fresh name creation provides another dimension of infinity: a ν-Petri net may have a potentially unbounded number of tokens of a potentially unbounded number of different names. When modeling systems, each name can represent a different process, which makes this formalism very suitable for the modeling of systems like bussiness processes and workflows that are composed of several processes which are in turn concurrent. For both extensions, reachability is not decidable, but other important properties, such as coverability, are decidable.

1.1.1.2 Timed extensions

As we mentioned before, time has become a crucial factor in system analysis, so many extensions of Petri nets dealing with time have appeared in the literature. The semantics of these extensions define not only the conditions and the effects of firing of transitions, but also how time elapses, and when the firings can happen. These extensions add time to Petri nets in several different ways [99]:

Discrete or continuous time: For timed models, there are two ways to define the nature of time elapsings: in terms of continuous delays or discrete delays. On the one hand, models with discrete delays can be more manageable to work with [23, 57, 25], but in some sense, they may lose some of the possible real behaviors of systems. On the other hand, continuous delays may seem more reasonable to model real time systems, but they sometimes contemplate behaviors which do not exist in the real world, such as Zeno behaviors, in which an unbounded number of events happen in a bounded period of time. There are several techniques for the verification and analysis of Petri nets with real time, such as region theory [8], which is based in the same technique for timed automata [11, 13], backward algorithms based in existential zones [10] or extrapolation operators [25], which are again based on a similar operator for timed automata [15].

Time relative to tokens, places or transitions: Several ways of adding time to Petri nets have been considered in terms of where to set the clocks, time constraints or restrictions. In some extensions, time is associated to tokens, that is, each token carries a clock, and the conditions to fire transitions are based on the age of the tokens that are involved in their firing [83, 10]. Other extensions associate a firing delay to transitions [74]. Also, there are works in which time is associated to places [88], so that a token must remain in a place a certain amount of time in order to be used for a firing.

Instantaneous or prolonged transitions: In most of the works, the firing of the transitions is instantaneous, that is, the tokens in preconditions are removed at the same time that the tokens are added to postconditions. However, there are models with prolonged transitions in which some time elapses between these two events.

Capacity: When considering prolonged transitions, it may be considered that a transition is being fired several times at the same moment. For example, if the firing of a transition takes three units of time, it may fire at some point, and be fired again less than three units of time after, so that it is being fired two times at this moment. The presence of a capacity k implies that the number of times a transition may be being fired at any given time is at most k. More precisely, there are three kinds of semantics regarding capacities: single server semantics, in which the capacity is one, multiple server semantics, in which there is a given capacity k, and infinite server semantics, in which there is no bound for the number of times a transition may be fired at a time.

Deterministic, non-deterministic or stochastic delays: In most of the first works on Petri nets with time, the time delays were set deterministically, that is, a fixed delay was associated to a place, transition or arc [74, 100]. However, in practical terms it may not make sense to fix these delays, since they usually represent the time that takes to accomplish some task. Therefore, two other ways of defining delays were defined consisting in adding some constraints to the delays (for example, belonging to an interval), or associating probabilistic distributions to delays.

Urgent or lazy semantics: There are also differences between models regarding the moment in which a transition may be fired. In models with urgent (or strong) semantics, a transition must be fired in a concrete period of time after being enabled (or even instantaneously). If it has not been fired during this period, this transition becomes urgent, and time cannot elapse anymore until the transition is

fired or disabled by the firing of other transitions. In models with lazy (or weak) semantics, transitions do not have to be fired at any concrete moment, even if the delay of time disables them due to the temporal restrictions.

One of the main models that have been deeply studied in the literature is *Timed Petri nets* [83, 10] (also called Timed-Arc Petri nets). It is a model with instantaneous transitions, non-deterministic delays and weak semantics, in which each token carries a clock, and the arcs going from places to transitions are labeled by time intervals. A transition is enabled if in each precondition there is a token whose clock fits in the corresponding interval. Then, when an enabled transition is fired, the clocks of the tokens which are added to postconditions are set to zero or more generally, to a value in a given interval. The model is not Turing complete, although reachability is not decidable for them. In fact, using the techniques in [18] it can be proved that Timed Petri nets is the most expressive model among the timed extensions of Petri nets without urgent semantics. TAPAAL (http://www.tapaal.net/) [52] is a tool for the analysis of bounded Timed Petri nets. It offers a graphical editor, a simulator and a model checker for a fragment of CTL. Moreover, it allows some other features like inhibitor arcs and invariants for the clocks of the tokens in the places.

Since plenty of systems depend on quantitative aspects, such as costs of human or material resources, it is important to study quantitative models, which take time into account. In [9, 2] Abdulla and Mayr defined Priced Timed Petri nets, which is a model which extends Timed Petri nets with costs. The model associates storage costs to places and firing costs to transitions. Intuitively, the storage cost of a place p represents the price of storing a token in place p per unit of time. The firing cost of a transition is the price of firing it, and it does not depend on time. Then, the authors study the costs of computations that reach a certain control state (or equivalently, that cover a certain marking). Although in general the minimum of the costs of all the executions does not exists, they prove that the infimum of the costs of computations that reach a certain control state is computable if all the costs are non-negative. However, when considering negative costs, the problem turns out to be undecidable.

1.1.2 Workflows

Initially, systems were created to execute individual and independent tasks. However, nowadays companies need to handle multiple non independent tasks, so the number and the complexity of their processes has increased. Therefore, they

require systems to manage the logistics of these bussiness processes. Workflows were created to represent the flows of work of companies or organizations, or more precisely, to model how tasks are accomplished in the correct time or order, using the right resources. Workflow processes are designed to manage similar cases (different instances of the same process), and they specify in what order tasks must be accomplished: each task must be executed after or before some other task. In this scope, a precondition for a task is a logical expression which may be evaluated to decide whether this task may be started or not. Analogously, the postcondition of a task is a logical expression which may be evaluated to decide whether a task is completed. These preconditions and postconditions often have to do with the tasks that have been already performed and the tasks than can start being performed, respectively. Moreover, each task may use some resources, which are typically, but not only, human resources (workers) or machines, and these resources may be shared by different instances of the processes.

The Workflow Management Coalition (WfMC) was founded in 1993 to support the development, education and promotion of workflow and business process management (BPM). Since then, it has created several process definition languages and a large glossary defining the main concepts in workflow management. However, this glossary provides rather non precise definitions, which lack formalization. For example, Fig. 1.1 shows the capture of the definition of workflow from the web of WfMC (http://www.wfmc.org/). It is clear that a more formal point of view is needed, not only to define the concepts and methodologies, but also to analyze and verify the workflows.

Petri nets are a very suitable formalism to represent workflows [93]. On the one hand, their simplicity and graphical nature make them easy to learn and to understand. On the other hand, their semantics have been well defined and there is a large background on verification and analysis for them. The idea of representing tasks by transitions; preconditions and postconditions by places, and cases or instances of processes by tokens is pretty intuitive. Moreover, the expressive power of Petri nets is enough to express how the cases of a workflow are routed through the tasks that must be accomplished in a concrete order [95]. Since the nineties, the properties, verification and analysis of workflow Petri nets have been widely studied in the literature [94, 95, 77].

Roughly speaking, a workflow Petri net is a Petri net with two special places, *in* and *out*. A token in place *in* represents an instance of the process that has been scheduled, and a token in *out* represents a finished instance. Moreover, there

are some additional conditions about the connectivity of places and transitions, so that all the conditions and tasks they represent are potentially able to take part in the management of the process.

In this setting, soundness is maybe the most interesting property to verify over workflows. Intuitively, we say that a workflow is sound if each possible execution can be finished correctly. More formally, a workflow net is sound if the following three conditions hold:

- Option to complete: From any marking which is reachable from the initial marking m_{in} with only one token in in, and empty elsewhere, we can reach the final marking m_{out} with only one token in out and empty anywhere else.

- Proper completion: If a marking m with some token in out is reached from m_{in}, then m is the final marking m_{out}.

- No dead transitions: For every transition t, there is a marking reachable from m_{in} which enables t.

It is well known that soundness for workflow nets is decidable [94]. To prove it, Aalst et al. define the extended net of a workflow net N, which is obtained by adding to N a transition which removes a token from out, and adds a token to in. They prove that a workflow Petri net is sound if and only if its extended net is bounded and live (from every reachable marking, every transition is able to be fired eventually). As checking both boundedness and liveness is decidable, soundness is decidable. However, soundness is undecidable for workflows endowed with reset arcs or inhibitor arcs [92].

Although this is the classical notion of soundness, other forms of soundness, such as the following ones, have been studied.

- A net is k-sound if from any marking reachable from the initial marking with k tokens in place in and empty elsewhere, the final marking with k tokens in out and empty elsewhere is reachable.

- 1-soundness is also called weak soundness, and corresponds to the first condition of the classical notion of soundness.

- A workflow net is up-to-k-sound if it is l-sound for all $1 \leq l \leq k$, and it is generally sound if it is k-sound for all $k \in \mathbb{N}$.

- Lazy soundness corresponds to the satisfaction of a weakened version of the properties "option to complete" and "proper completion" in which tokens may be left in the net as long as the place *out* is marked only once.

- A workflow net is relaxed sound if for each transition, there is a marking which enables t reaching a marking m from which m_{out} is reachable.

Both soundness and general soundness are decidable for workflow nets [94, 98]. However, both properties are undecidable if we consider them for workflow nets with reset or inhibitor arcs. Moreover, even weak, lazy and relaxed soundness are undecidable for workflows with reset arcs [92].

In the previous notions of soundness, the sharing of resources by several instances running the same workflow is not contemplated. It is natural to define a notion of soundness which takes it into account. In [44] van Hee et al. define and study resource-constrained workflow nets (rcwf nets), which are concurrent instances of workflow Petri nets, constrained by shared resources. Then, they define the soundness for them (also called dynamic soundness): intuitively, a rcwf net is sound if, provided with a sufficient amount of resources and no matter how many instances of the workflow are running concurrently, every instance is guaranteed to be able to finish correctly. Moreover, at the end of the run, the amount of available resources must be the same as in the beginning. In [53] the soundness of rcwf nets is proved to be decidable, by transforming the problem of checking the soundness of infinitely many bounded nets to checking the soundness of only one (unbounded) net. This is clearly a parameterized verification problem, since it is defined for workflows with arbitrarily many instances running in them.

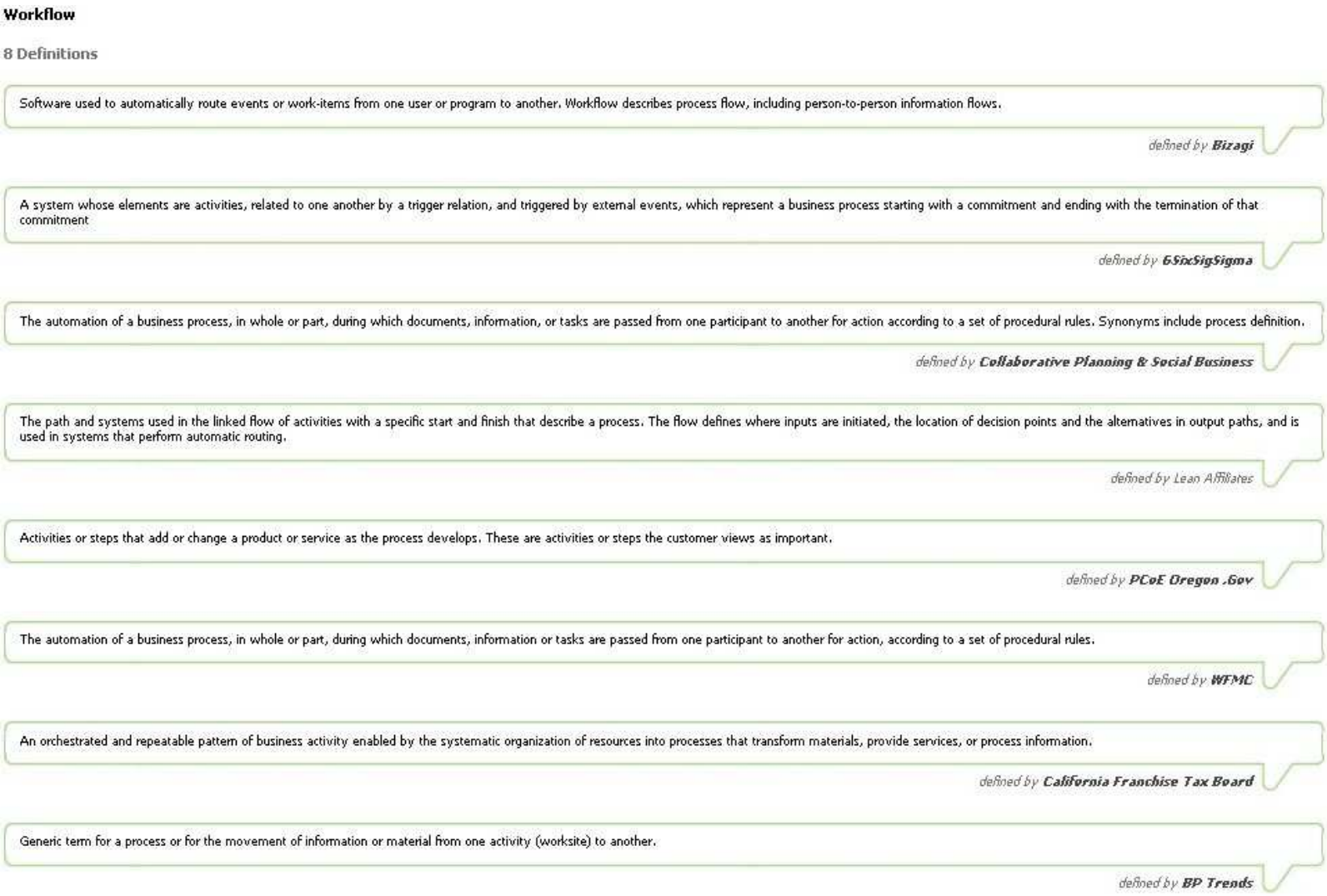


Figure 1.1: Workflow definition from WfMC glossary

1.2 Motivation and objectives

As mentioned before, we need to represent and study systems with several different characteristics: systems which depend on temporal issues, systems endowed with resources and costs, or systems in which several instances of a process run at the same time. In the scope of Petri nets, Priced Timed Petri nets [9] are defined in order to model systems in which costs depend on time. On the other hand, ν-Petri nets (ν-PN for short) [78] can be seen as Petri nets in which several instances of a process, represented by different names, run concurrently. Our goal is to define a model which encompasses these two capabilities.

When defining such a model, if we start from ν-PN, we have several possibilities regarding the way we add the clocks: we can consider a model in which each token carries a clock, and therefore there is a potentially unbounded number of clocks per instance; a model in which each instance carries only one clock, or a model with a fixed number of clocks per instance. In any case, we will label arcs by intervals, in the way of [9], so that the clocks of the tokens (or instances) in the preconditions of a transition must fit in the intervals of the incoming arcs of this transition in order to be fired. Moreover, when a transition is fired, we will change the clocks of the tokens/instances involved in the firing according to the outgoing arcs of the transition.

We would like to explore all of the previous options, and study their decidability properties and expressiveness. Although these new models would probably be more expressive than the two previous ones (Priced-Timed Petri nets and ν-Petri nets), we still want to retain good decidability properties, in order to be able to study some safety questions over the systems represented by our new model. In particular, it would be desirable that control-state reachability or coverability are still decidable. To perform this study, we need to define abstractions, in order to work with discrete models instead of continuous ones. In particular, we need to abstract time, in the way of [6, 5, 9], where an abstract model is defined by clustering the clocks in multisets with the same fractional parts, and ordering these multisets by the fractional parts they represent. Then, only the integer part of the clocks needs to be stored. Moreover, clocks which are older than the greater integer max in the labels of the arcs, are abstracted to $max+1$, since the concrete age of these clocks is not significant. In that way, a discrete model which represents the real time model is obtained, and can be studied by applying standard techniques, such as the framework of Well Structured Transition Systems.

Finally, if decidability is obtained, we would like to compare the expressiveness

of this new model to the expressiveness of some other existing models, as Timed Petri nets [83]. In order to compare the expressive power of different models, it is usual to study the languages generated by the models when associating labels to transitions, considering different accepting conditions. For example, given a marking m, the reachability and coverability languages of a model are the sets of words produced by the runs finishing in m or in a marking covering m, respectively. We will study the relative expressiveness of the different models by comparing the families of coverability languages they accept, as in [4, 40]. The reason why we choose to study the coverability languages instead of the reachability languages is because reachability is undecidable even for Timed Petri nets. Since our new models are expected to be at least as expressive as Timed Petri nets, considering reachability as our accepting condition is not suitable. In order to perform this task, we could apply the framework described in [16] and prove that our model is strictly more expressive than Timed Petri nets by comparing the size of their state spaces.

If the previous goal is achieved, we can put into practice the new models to study workflow nets. In particular, we can represent each instance in a resource-constrained workflow net [53] as a name in the style of ν-Petri nets, and add temporal constraints representing the time which is needed to complete a task, as done in [64]. Moreover, we can add costs to the executions of the workflow, as done in [9] for Timed Petri nets. Then, we want to define and study soundness properties for this model, that take into account time and costs. More precisely, our goals in the field of workflow nets are:

- To study the decidability of soundness of resource-constrained workflow nets in a more general way than done in [53, 44], where increasing the available resources is not allowed.

- To define a model of priced resource-constrained workflow nets which encompasses time and costs as done in [9] for Petri nets, that is, in a way such that the cost of the execution of an instance is computed as the sum of firing costs (the cost of performing an action) and storage costs (the cost of storing items while tasks are accomplished). More precisely, we first need to define the cost of firing a transition t from a concrete marking m, which is the sum of its firing costs and its storage costs when fired. Then, the price of an instance in an execution is the sum of the costs of the firing of transitions by this instance.

- To define a soundness problem in which a net is sound if none of its executions spends more than a given budget. As each instance is endowed with a price we can collect every price in a set of prices and extract from it a global price in several different ways. For example, we could consider the sum of the price of each instance, the maximum or the average. Our goal is to define a general frame by defining *priced predicates* over sets of prices like "the sum of the prices is under a given budget", "the maximum of the prices is under a given budget" or "the average of the prices is under a given budget". Then, we will define soundness parametrized by a price predicate: a workflow net is sound if the price of each of its executions satisfies the price predicate.

- To study the previous soundness problems, taking advantage of the properties of the basic model we first defined.

1.3 Our contribution

We firstly define a new model called *Timed ν-Petri nets*, by enriching ν-Petri nets with time in the way of [83, 10]. In this new extension, the arcs of the net are labeled by intervals, and each token is endowed both with a name and a (real time) clock. Then, in order to fire a transition t, the clocks of the tokens in the preconditions of t must fit in the intervals of the arcs going to t. Moreover, when t is fired, the clocks of the tokens which are added to postconditions are set to a value in the intervals labeling the arcs going from t to these postconditions. We prove that Timed ν-Petri nets are Turing-complete, so that, unfortunately, the coverability problem is undecidable. We do it by simulating an extension of Petri nets called ν-RN [79]. Basically, a ν-RN system is just a collection of ν-PN that can synchronize with each other, and that can create replicas of themselves (for more details see [79]). The main idea of this proof is how to encode the different instances of a ν-RN N by means of a ν-TPN N'. We do it by forcing all tokens in N' that belong to the same instance of N to have the same age. Then, the firing of transitions in which a single instance takes part is done by requiring that all tokens involved in the firing are of age exactly 1. Moreover, new instances are created by creating tokens with an age different from the age of every other token. We will see that the construction is lossy, in the sense that tokens that become older than 1 become useless. However, this does not affect control-state reachability. We think that the reduction from ν-RN is interesting by itself: this

technique may be applicable in other cases to perform reductions from models with time to models with (maybe ordered) instances. In fact, later we prove that decidability of control-state reachability is undecidable even for ν-*TPN* with only two tokens (and clocks) per instance by reducing the same (undecidable) problem for Timed Petri nets with two clocks per token to it.

As each color in a ν-*PN* may represent a different process running in the net, it seems natural and interesting to study a model in which each of these processes is endowed with a single clock instead of one clock per token. We define *Locally-Synchronous ν-Petri nets* (ν-*lsPN* for short), which is a restriction of Timed ν-Petri nets in which all the tokens of the same name share a clock. Moreover, we find a positive decidability result: we prove that coverability is decidable by making a discrete abstraction of the model and proving that this abstraction fits in the frame of Well Structured Transition Systems for which, under some light additional conditions, coverability is decidable. The abstraction is based in the theory of regions used in [6, 10] to represent the state space of Timed Petri nets. Basically, in order to represent the markings and the clocks of Locally-Synchronous ν-Petri nets instances, we order them in multisets according to the following rules:

- All the instances with an age older than the maximum max of the bounds in the intervals labeling arcs in the net are represented in a multiset A_∞ by a pair consisting of their markings and the natural number $max + 1$.
- The rest of the instances are added to multisets $A_0, \ldots, A_n$ of instances with the same fractional part of their ages. The instances are represented by a pair consisting in their markings and the natural part of their ages.
- The instances represented in A_0 have an age represented by a natural number. Moreover, if $i < j$ then the fractional part of the age of the instances in j is greater than the ones in i.

In that way, we are able to represent the markings of our net in a discrete manner, and we define a model which simulates Locally-Synchronous ν-Petri nets. Finally, we prove that this model is a Well Structured Transition System, and hence coverability is decidable for it.

As mentioned in the objectives, we study the expressive power of this new model. We define classes of Locally-Synchronous ν-Petri nets depending on the number of unbounded places in the nets. Then, we study the expressiveness of the different classes by comparing the coverability languages they accept. First, we

prove that the class of bounded Locally-Synchronous ν-Petri nets has the same expressive power as Timed Petri nets, by performing direct reductions. Then, we apply the framework based on ordinals developed in [16] (which basically proves that under certain hypotheses, a bigger state space implies having more expressiveness) to our classes of Locally-Synchronous ν-Petri nets, proving that the expressive power of the models strictly increases with the number of unbounded places.

Once we have defined a suitable timed model with still decidable interesting properties, the next step according to our planning is to define a timed and priced model(ν-$PTdPN$). Naturally, we do it by extending ν-$lsPN$ with the cost model, which consists in a function $Cost$ which assigns natural intervals to places and transitions. Intuitively, the storage cost of a place p ($Cost(p)$) is the price of storing tokens in p per unit of time. The firing cost of a transition t ($Cost(t)$) is the price of firing t. In order to study the decidability of control-state reachability, once again we need to define an abstraction of the model in order to reduce the problem to control-state reachability in a discrete (untimed) model. This time, we define an abstraction in the way the authors do in [2] for Priced Timed Petri nets. Now, we still classify instances by the fractional parts of their clocks, forgetting about their concrete values. However, we order the multisets of instances in a different way: a region is of the form $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$, where c is the accumulated cost of the run, and for some $\delta \leq 1/2$, if $i < 0$, A_i represents a set of instances whose clocks have the same fractional part, which is greater than $1 - \delta$ and if $i > 0$, A_i represents a set of instances whose clocks have the same fractional part, which is smaller than δ. Then we prove that in order to decide control-state reachability, we only need to consider regions with infinitesimally small δ, by applying the techniques in [17, 2], in which the feasible computations are represented in a polyhedron which can be described by a totally unimodular matrix, for which the coordinates of its vertices are integers. Finally, we prove that the abstraction we build is a Well Structured Transition system, and again, we apply this framework to obtain decidability of the control-state reachability problem.

Before applying this model to workflow nets, we focus on resource-constrained workflow nets, without time or costs, in order to complete the previous works in [44, 53]. For this purpose, we express resource-constrained workflow nets in terms of ν-Petri nets, representing each process executing the workflow by a different instance in the ν-Petri net. As in previous works, we study decidability

of dynamic soundness. However, we allow the creation of new resources in the executions of the net. We prove undecidability of this problem by a non-trivial reduction to reachability for ν-Petri nets. Then, we study decidability in a restricted but very natural case. Basically, we assume that each instance is sound when endowed with infinitely many resources. We prove that dynamic soundness is decidable in this case.

Then, we define (still untimed) Priced Resource-Constrained Workflow nets. We add costs modeled as tuples of natural numbers to the previous model and define firing costs as the price of performing tasks (or firing transitions in our model), and storage costs as the price of storing materials or resources while a task is performed (keeping tokens in places when transitions are fired). In that way, although time is not managed explicitly, we consider that in this model time elapses while performing actions, that is, while firing transitions. Then, the price of the execution of an instance is the sum of the firing costs and storage costs involving its firings of transitions and storage of tokens. As required in the objectives, we define a framework in order to define the price of a complete execution. More precisely, we give a definition of soundness based on price predicates which depend on a given budget b, such as "the sum of the prices of the instances keeps under b", "the more expensive instance does not cost more than b" or "the average of the costs of the instances does not reach b". More precisely, we say that a net is sound considering a price predicate Φ and a budget b if from every reachable state, the final state can be reached satisfying $\Phi(b)$. We prove that this "priced" soundness is decidable for the three previous predicates by reducing it to coverability. Basically, we represent the costs as tokens which are added to a special place p_b in a lossy manner, since we do not have transfer arcs to add the storage costs depending on the number of the tokens in places. Then, the net is not sound for a budget b if a marking with b or more tokens in p_b is reachable. Moreover, we study the relations between the different price predicates, that is, we study if the soundness of a net for a price predicate implies the soundness of the net for each of the other price predicates.

Finally, we define a model which encompasses the representation of time and costs over resource-constrained workflow nets. We do it by restricting the priced-timed extension of ν-Petri nets we define previously to nets with the form of a resource-constrained workflow net. In that way we are able to consider the previous price model, with firing and storage costs, but now storage costs depend on time. Therefore, now we consider that time elapses between the firing of

discrete transitions, unlike in the previous extension. However, instead of defining the price of a run as for ν-$PTdPN$, we do it like for Priced Resource-Constrained workflow nets, in which the price of a run is the set of prices of its instances. Then, after slightly adapting the parameterized framework we considered before to define soundness, we tackle the decidability of the different price predicates for this new extension, by taking advantage of the decidability of control-state reachability for ν-$PTdPN$ previously proved.

1.4 Contents

The following chapters are organized as follows:

- Chapter 2 introduces the main basic concepts and notations used in the rest of this thesis.
- Chapter 3 defines the different timed extensions of ν-Petri nets that are considered in this thesis. Moreover, decidability of coverability and expressiveness is studied for these new models.
- Chapter 4 defines an extension of ν-Petri nets with time and costs depending on time. Then, control-state reachability is proved to be decidable for this extension.
- Chapter 5 continues the previous works about Resource-Constrained Workflow nets, giving a more general definition of the problem of dynamic soundness, and studying its decidability.
- Chapter 6 defines Priced Resource-Constrained Workflow nets and Priced-Timed Resource-Constrained Workflow nets. Moreover, a new concept of "priced" soundness, which is parameterized on the way we calculate the cost of each execution, is defined and studied.

1.5 Publications

We list the publications which are related to this thesis.

- Dynamic soundness in Resource-Constrained Workflow Nets.
  María Martos-Salgado, Fernando Rosa-Velardo.
  In 13th IFIP WG 6.1 International Conference on Formal Methods for Open

Object-based Distributed Systems, FMOODS'11, and 30th IFIP WG 6.1 International Conference on FORmal TEchniques for Networked and Distributed Systems, FORTE'11. LNCS 6722, pp. 259-273. Springer, 2011.

- Cost Soundness for Priced Resource-Constrained Workflow nets.
  María Martos-Salgado, Fernando Rosa-Velardo.
  In 33rd International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, PETRI NETS 2012. LNCS 7347, pp. 108-127. Springer, 2012.

- Safety and soundness for priced resource-constrained workflow nets.
  María Martos-Salgado, Fernando Rosa-Velardo.
  Fundamenta Informaticae, vol. 131(1), pp. 55-80. IOS Press, 2014.

- Expressiveness of Dynamic Networks of Timed Petri Nets.
  María Martos-Salgado, Fernando Rosa-Velardo.
  In 8th International Conference on Language and Automata Theory and Applications, LATA 2014. LNCS vol. 8370, pp. 516-527. Springer, 2014.

- Dynamic Networks of Timed Petri Nets.
  María Martos-Salgado, Fernando Rosa-Velardo.
  In 35th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, PETRI NETS 2014. LNCS 8489, pp. 294-313. Springer, 2014.

Chapter 2

Preliminaries

In this chapter we present the basic concepts and set the notations we use in this thesis. All the definitions and results we present here are already well established in the literature.

2.1 Multisets, words and quasiorders

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ and $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$ be the naturals completed with their limit ω. We write $v[i]$ to denote the i^{th} component of $v \in \mathbb{N}_\omega^k$ and $\mathbf{0} = (0, \ldots, 0)$. Given $n \in \mathbb{N}$, we denote $n^+ = \{1, 2, \ldots, n\}$, $n^* = \{0, 1, 2, \ldots, n\}$ and $n^*_\infty = n^* \cup \{\infty\}$. We denote open, closed and mixed intervals of real numbers as (a, c), $[a, b]$ and $[a, c)$ or $(a, b]$, respectively, where $a, b \in \mathbb{N}$ and $c \in \mathbb{N} \cup \{\infty\}$. The set of intervals is denoted by $\mathcal{I}$. Let $\mathbb{R}_{\geq 0} = [0, \infty)$ and for each $x \in \mathbb{R}_{\geq 0}$ we denote by $\lfloor x \rfloor$ and $\mathit{frct}(x)$ the integer and the fractional part of x, respectively. A multiset m over a (maybe infinite) set X is a mapping $m : X \to \mathbb{N}$. Given a multiset m over X, we say that the multiplicity of an element $x \in X$ is $m(x)$, and we call $\{x \in X \mid m(x) > 0\}$ the support of m, denoted by $S(m)$. We say m is a finite multiset if $S(m)$ is finite, and then, we define $|m| = \sum_{x \in X} m(x)$. We denote by $X^\oplus$ the set of finite multisets over X. Given $m_1, m_2 \in X^\oplus$, we define $m_1 + m_2 \in X^\oplus$ by $(m_1 + m_2)(x) = m_1(x) + m_2(x)$ and $m_1 \subseteq m_2$ if for each $x \in X$, $m_1(x) \leq m_2(x)$. If $m_1 \subseteq m_2$ we can define $m_2 - m_1 \in X^\oplus$ by $(m_2 - m_1)(x) = m_2(x) - m_1(x)$. We denote by $\emptyset$ the empty multiset, that is, the multiset such that for each $x \in X$, $\emptyset(x) = 0$. Sometimes we use set notation for multisets with repetitions, to account for multiplicities.

A quasi ordering $\leq$ is a reflexive and transitive relation on a set X. If $\leq$ is also antisymmetric, then it is a partial ordering (po for short). Given a quasi ordering

$\leq$ over X and $x, x' \in X$, we may write $x \geq x'$ whenever $x' \leq x$. A well quasi ordering (well partial ordering) is a quasi ordering (partial ordering, respectively) such that, for any infinite sequence $x_1, x_2, x_3, \ldots \in X$, there are $i, j \in \mathbb{N}$ with $i < j$ and $x_i < x_j$. Given a quasi ordering $\leq$ on X, and X_1 a subset of X (denoted by $X_1 \subseteq X$), we denote $\uparrow X_1 = \{x \in X \mid \exists x' \in X_1, x' \leq x\}$ the upward-closure of X_1 and $\downarrow X_1 = \{x \in X \mid \exists x' \in X_1, x' \geq x\}$ the downward-closure of X_1. For $x \in X$ we write $\uparrow x$ and $\downarrow x$ instead of $\uparrow\{x\}$ and $\downarrow\{x\}$, respectively. An upward-closed subset $U \subseteq X$ is a subset such that, for each $u \in U$, if there is $x \in X$ with $u \leq x$ then $x \in U$. Therefore, if U is an upward-closed subset, then $\uparrow U = U$ and for every $x \in X$, $\uparrow x$ is an upward-closed set. A basis of an upward closed set U is a subset $B \subseteq U$ such that $U = \bigcup_{x \in B} \uparrow x = \uparrow B$. Analogously, a downward-closed subset $D \subseteq X$ is a subset such that, for each $d \in D$, if there is $x \in X$ with $d \geq x$ then $x \in D$, so if D is a downward-closed subset, then $\downarrow D = D$ and for every $x \in X$, $\downarrow x$ is an downward-closed set. Moreover, a basis of an downward closed set D is a subset $B \subseteq D$ such that $D = \bigcup_{x \in B} \downarrow x = \downarrow B$.

Given a po $\leq$ over a set X, we define the po $\leq^{\oplus}$ over $X^{\oplus}$ as $\{x_1, \ldots, x_n\} \leq^{\oplus} \{y_1, \ldots, y_m\}$ if there is an injection $h : n^+ \to m^+$ such that $x_i \leq y_{h(i)}$ for each $i \in n^+$. If $(X, \leq)$ is a well partial ordering (wpo for short) then so is $(X^{\oplus}, \leq^{\oplus})$ [47].

Given a set X, any $u = x_1 \cdots x_n$ with $n \geq 0$ and $x_i \in X$ for all $i \in n^+$ is a (finite) *word* over X, and for each $i \in n^+$ we denote $u(i) = x_i$ and $|u| = n$. We denote by $X^{\circledast}$ the set of finite words over X. If $n = 0$ then u is the empty word, denoted by ϵ. If X is a wpo then so is $X^{\circledast}$ [47] ordered by $\leq^{\circledast}$, defined as $x_1 \ldots x_n \leq^{\circledast} y_1 \ldots y_m$ if there is a strictly increasing mapping $h : n^+ \to m^+$ such that $x_i \leq y_{h(i)}$ for each $i \in n^+$.

2.2 Transition systems

Definition 2.2.1 (Labelled transition system) *A* labelled transition system *(LTS for short) is a tuple $\langle S, \Sigma, \to, s_0 \rangle$ where S is a set of states, Σ is a finite alphabet, $s_0 \in S$ is the initial state and $\to \subseteq (S \times \Sigma \times S)$ is the transition relation.*

When convenient, we will omit the initial state from the definition of the considered LTSs. We will write $s \xrightarrow{a} s'$ instead of $(s, a, s') \in \to$ and sometimes we may omit the labels, that is, we may write $s \to s'$ whenever there is $a \in \Sigma$ such that $s \xrightarrow{a} s'$. If there are $s_1, \ldots, s_n \in S$ and $a_1, \ldots, a_{n-1} \in \Sigma$ such that $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-1}} s_n$, then we write $s_1 \xrightarrow{w} s_n$, where $w = a_1 a_2 \ldots a_{n-1}$. Such a sequence of firings is called a path. We write $s \to^* s'$ if there is $w \in \Sigma^*$ such that

$s \xrightarrow{w} s'$. Given a state $s \in S$, we denote by $Succ(s)$ (respectively $Pred(s)$) the set $\{s' \mid s \to s'\}$ of immediate successors of s ($\{s' \mid s' \to s\}$, the set of immediate predecessors of s, respectively). We say that a state s is reachable if $s_0 \to^* s$. Given an LTS $TS = \langle S, \Sigma, \to, s_0 \rangle$ and $S_F \subseteq S$, we define the language of TS as $\mathcal{L}(TS, S_F) = \{w \in \Sigma^* \mid \exists s \in S_F, s_0 \xrightarrow{w} s\}$. A finite automaton is a tuple $\langle S, \Sigma, \to, s_0, s_f \rangle$, where s_f is a final state and $\langle S, \Sigma, \to, s_0 \rangle$ is an LTS in which the sets of states and transitions are finite. Hence, the language of an automaton A is $\mathcal{L}(A) = \{w \in \Sigma^* \mid s_0 \xrightarrow{w} s_f\}$.

Now, we present some problems that have been studied for LTS in literature, which we will use in this thesis.

Definition 2.2.2 (Coverability, boundedness and termination) *Given an LTS $TS = \langle S, \Sigma, \to, s_0 \rangle$ endowed with an ordering $\leq$ over S:*

The coverability problem *is that of deciding if, given state s of TS, there is a reachable state s' such that $s \leq s'$.*

The boundedness problem *is that of deciding if there is $n \in \mathbb{N}$ such that $|\{s \mid s \text{ is reachable from } s_0\}| < n$, that is, if there is a bounded number of reachable states of TS.*

The termination problem *is that of deciding if there is an infinite path $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$.*

An LTS $\langle S, \Sigma, \to, s_0 \rangle$ is monotone with respect to an order $\leq$ if for every $s_1, s_2, s_1' \in S$ such that $s_1 \to s_1'$ and $s_1 \leq s_2$, there is an $s_2' \in S$ such that $s_2 \to s_2'$ and $s_1' \leq s_2'$. Finally, given two states s and s' we say that s covers s' if $s \geq s'$. Intuitively, this means that a state s which is greater or equal than another state s' can always evolve as s', getting greater or equal states.

2.2.1 Well-structured transition systems

Many of the decidability results we consider in this thesis, are due to Dickson's lemma [33] and the monotonicity of Petri nets and some of its extensions. In [35], Finkel defined *Well Structured Transition Systems*, which are essentially systems which satisfy these two properties.

Definition 2.2.3 (Well-structured transition system) *A labelled transition system $TS = \langle S, \Sigma, \to, s_0 \rangle$ is a well-structured transition system with respect to an ordering $\leq$ on S if TS is monotone with respect to $\leq$ and $\leq$ is a well-quasi ordering.*

The decidability of coverability, boundedness and termination for WSTS have been deeply studied in literature. Some (not very strong) additional conditions are required in order to obtain the decidability of the previous problems. These conditions are the following ones.

Definition 2.2.4 (Effective *Pred*-basis, compatibility) *Given a WSTS TS:*

TS has effective *Pred*-basis *if there is an algorithm which accepts a state s of TS and returns a finite basis of $\uparrow Pred(\uparrow s)$.*

TS has transitive compatibility *if for all s, s' and t states with $s \leq t$ and $s \to s'$, there is a non-empty path $t \to t_1 \to \ldots \to t_n$ with $s' \leq t_n$.*

TS has strict compatibility *if for all s, s' and t states with $s < t$ and $s \to s'$, there is a path $t \to^* t'$, with $s' \leq t'$.*

TS has reflexive compatibility *if for all s, s' and t with $s \leq t$ and $s \to s'$, either $s' \leq t$ or $t \to s''$ with $s'' \geq s'$.*

In order to prove the decidability of boundedness and termination for WSTS, Finkel et al. use the so-called *tree-saturation methods*. These methods consist in representing all possible computations of the system in a tree structure, called the *finite reachability tree* [36]. More precisely, each node of the reachability tree is labelled by a state of the system, and the root n_i is labelled by the initial state. Then, each node is live or dead. A node n is dead if the state it represents does not have any successor, or if there is a node n' in the path from n_i to n such that the state represented by n covers the one represented by n'. Then, the node does not have any child. Otherwise, the node is live, and it has a child for each successor of the state it represents. The reachability tree is finite and effectively computable for WSTSs with computable $\leq$ and successors, and although compatibility is not required for building it, when we have compatibility the reachability tree contains the answer of some of the previous problems. In particular:

- The boundedness problem is decidable for WSTS with computable *Succ*, decidable partial ordering and strict transitive compatibility.
- The termination problem is decidable for WSTS with computable *Succ*, decidable ordering and transitive compatibility.
- The coverability problem is decidable for WSTS with computable *Succ*, decidable ordering and reflexive compatibility.

Abdulla et al. use another approach for the decidability of coverability, called *set-saturation method* [3, 7], which was generalized by Finkel et al. later [37]. It

is a backwards reachability algorithm, which consists in, given an upward-closed set of states I (which is the set of states covering the state that we want to check for its coverability), computing $Pred^*(I)$, which is the limit of the sequence $I = I_0 \subseteq I_1 \subseteq I_2 \subseteq \ldots$, where $I_{n+1} = I_n \cup Pred(I_n)$. This limit is computable if we have effective $Pred$-basis, and in this case, coverability is decidable [37].

Therefore, the following results are obtained by applying the previous techniques.

Proposition 2.2.1 *[35, 36, 37]*

Coverability is decidable for WSTS with effective Pred-basis.

Boundedness is decidable for WSTS with strict transitive compatibility, a decidable well partial ordering $\leq$ and computable Succ.

Termination is decidable for WSTS with transitive compatibility, decidable $\leq$ and computable Succ.

WSTSs can be seen as acceptors of languages in order to study their relative expressive power. For this purpose, we need to choose an accepting condition to define the languages we want to use to compare families of WSTS. Several conditions, such as reachability, coverability or non condition (trace languages), have been considered in literature. The class of reachability languages is not a good criterion for our purpose, since it corresponds to the set of recursively enumerable languages for most of the extensions of Petri nets we are going to consider. In our case, we study the relative expressiveness of classes of WSTS by comparing the coverability languages they accept, that is, we will consider coverability as accepting condition as in [40, 4]. Therefore, the languages we consider consist in words formed by computations finishing in states which cover a given final state.

Given a WSTS TS with initial state s_0, provided with a quasi ordering $\leq$ and a final state s, the covering language is defined by $L(TS, s_0, s) = \{w \in \Sigma^{\circledast} \mid s_0 \xrightarrow{w} s' \text{ with } s \leq s'\}$. Then, given two classes of WSTS $\mathbf{S}_1$ and $\mathbf{S}_2$, we write $\mathbf{S}_1 \preceq \mathbf{S}_2$ if for each $TS_1 \in \mathbf{S}_1$, there is $TS_2 \in \mathbf{S}_2$ with $L(TS_1) = L(TS_2)$. We write $\mathbf{S}_1 \simeq \mathbf{S}_2$ when $\mathbf{S}_1 \preceq \mathbf{S}_2$ and $\mathbf{S}_2 \preceq \mathbf{S}_1$; and $\mathbf{S}_1 \prec \mathbf{S}_2$ if $\mathbf{S}_1 \preceq \mathbf{S}_2$ and $\mathbf{S}_2 \not\preceq \mathbf{S}_1$.

2.3 Petri nets and extensions

In this section we formally present the syntax and the semantics of Petri nets and the extensions we use in this thesis.

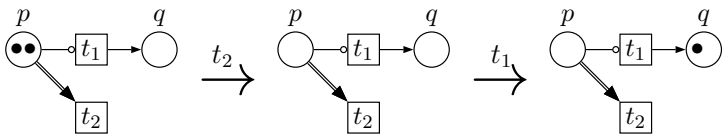


Figure 2.1: The firing in a Petri net

Definition 2.3.1 (Petri net) *A* Petri net *(PN for short) is a tuple* $\langle P, T, F, R, I\rangle$ *where* P *and* T *are finite disjoint sets of places and transitions,* $F \subseteq (P \times T) \cup (T \times P)$*,* $R \in P \times T$ *and* $I \in P \times T$*.*

We call the elements of F, R and I arcs. The elements of R are the reset arcs, and the elements of I are the inhibitor arcs. A PN $\langle P, T, F, R, I\rangle$ is a Place/Transition net (P/T net for short) if $R = I = \emptyset$ (and we denote it by $\langle P, T, F\rangle$), a reset net if $I = \emptyset$ and it is an inhibitor net if $R = \emptyset$. In figures, we represent places as circles, transitions as rectangles, arcs as arrows, reset arcs as double arrows and inhibitor arcs as arcs with a circle as arrowhead. A marking is an element of $P^{\oplus}$. Intuitively, the number of appearances of a place p in a marking m, represents the number of tokens in p at the considered marking. Given $t \in T$, ${}^\bullet t = \{p \in P \mid (p, t) \in F\}$ and $t^\bullet = \{p \in P \mid (t, p) \in F\}$.

Given a PN $\langle P, T, F, R, I\rangle$, a marking m and a transition t, we say that t is enabled if ${}^\bullet t \subseteq m$ and $m(p) = 0$ whenever $(p, t) \in I$, that is, it is enabled if there are enough tokens in the places corresponding to the incoming arcs of t, and if the places with an inhibitor arc going to t are empty. Then, t can be fired, reaching a marking m' such that, for each $p \in P$:

- $m'(p) = t^\bullet(p)$ if $(p, t) \in R$.
- $m'(p) = (m(p) -^\bullet t(p)) + t^\bullet(p)$ otherwise.

Example 2.3.1 *The left-hand side of fig. 2.1 shows a Petri net with two tokens in place* p*. Therefore, transition* t_1 *is not enabled at this marking, since it has an inhibitor arc from* p*. However, transition* t_2 *can be fired, reaching the marking depicted in the central part of the figure, in which the place* p *has been emptied by the reset arc. Therefore,* t_1 *is now enabled, and can be fired, reaching a new marking depicted in the right-hand side of the figure, in which a token has been added to* q*.*

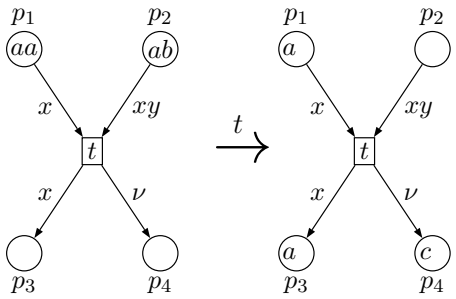


Figure 2.2: Firing of a transition in a ν-PN

2.3.1 ν-Petri nets

For the definitions concerning ν-Petri nets we fix an infinite set Id of names, an infinite set Var of variables and a subset of special variables $\Upsilon \subset Var$ for fresh name creation. Moreover, in the rest of this thesis we denote variables in Υ as $\nu, \nu_1, \nu_2, \ldots$

Definition 2.3.2 (ν-Petri net) *[78] A* ν-Petri net *(ν-PN for short) is a tuple $\langle P, T, F, H\rangle$ where P and T are finite disjoint sets of places and transitions, and $F, H : T \to (P \times Var)^{\oplus}$ are the input and the output functions, respectively.*

We denote $Var(t) = \{x \in Var \mid \exists p \in P, (p, x) \in F(t) + H(t)\}$, $Var(p) = \{x \in Var \mid \exists t \in T, (p, x) \in F(t) + H(t)\}$, and if $(p, x) \in F(t)$ we say that there is an arc from p to t labelled by x. Analogously, if $(p, x) \in H(t)$, we say that there is an arc from t to p labelled by x. A marking of a ν-PN is an element m of $(Id \times P)^{\oplus}$, and if $(a, p) \in m$, we say that there is a token of name a in place p. A mode is an injection $\sigma : Var \to Id$. Modes are extended homomorphically to $(P \times Var)^{\oplus}$, that is, given $M \in (P \times Var)^{\oplus}$, $x \in Var$ and $a \in Id$ with $\sigma(x) = a$, we define $\sigma(M)(a, p) = M(p, x)$. We say that a transition $t \in T$ is enabled at a marking m with mode σ if $\sigma(F(t)) \subseteq m$ and for any $\nu \in \Upsilon$, $p \in P$, $(\nu, p) \notin m$. Then, t can be fired, reaching a new marking $m' = (m - \sigma(F(t))) + \sigma(H(t))$ and we denote this by $m \xrightarrow{t} m'$.

Example 2.3.2 *Focus on the ν-PN depicted in the left-hand side of figure 2.2, with $(p_1, x), (p_2, x), (p_2, y) \in F(t)$ and $(p_3, x), (p_4, \nu) \in H(t)$. Transition t is enabled at the depicted marking, with a mode σ such that $\sigma(x) = a$, $\sigma(y) = b$ and $\sigma(\nu) = c$. Then, it can be fired, reaching the marking depicted in the right-hand side. Note that the name of the token added to p_4 (c) is fresh, since ν is in the set of special variables for name creation.*

2.3.2 Timed Petri nets

Now we present the model of Timed Petri nets, in which most of the models introduced in this thesis are based.

Definition 2.3.3 (Timed Petri net) *[83, 10] A* timed Petri net *(TPN for short) is a tuple $\langle P, T, F, H\rangle$, where P and T are finite disjoint sets of places and transitions, and $F, H : P \times T \to \mathcal{I}^{\oplus}$.*

A marking m of a Timed Petri net is an element of $(P \times \mathbb{R}_{\geq 0})^{\oplus}$. Intuitively, $(p, r) \in m$ means that there is a token of age r in place p at the marking m. Abusing notation, we denote $m(p)$ the multiset of ages of tokens in p at the marking m.

The marking of a TPN may evolve in two ways: by firing discrete transitions and by firing timed transitions, that is, elapsing time. Given a marking $m = \{(p_1, c_1), \ldots, (p_n, c_n)\}$ and $d \geq 0$, we write $m \xrightarrow{d} m'$, where $m' = \{(p_1, c_1 + d), \ldots, (p_n, c_n + d)\}$, that is, the marking in which every token is d units of time older. We say that a transition t is enabled at a marking m if for each $p \in P$ with $F(p, t) = \{I_1, \ldots, I_n\}$, there is $In_p = \{c_1, \ldots, c_n\} \subseteq m(p)$ such that $c_i \in I_i$ for each $i \in \{1, \ldots, n\}$. Then, t can be fired from m, reaching the new marking m' such that for each $p \in P$ with $H(p, t) = \{J_1, \ldots, J_l\}$, $m'(p) = (m(p) - In_p) + Out_p$, where $Out_p = \{j_1, \ldots, j_l\}$ with $j_i \in J_i$ for each $i \in \{1, \ldots, l\}$. We denote this by $m \xrightarrow{t} m'$. We write $m \to m'$ if there is $d \geq 0$ with $m \xrightarrow{d} m'$ or there is $t \in T$ with $m \xrightarrow{t} m'$.

As markings of TPNs are multisets over $P \times \mathbb{R}_{\geq 0}$, the corresponding order over multisets can be considered for them, as well as the coverability problem. In [10] Abdulla and Nylén prove that the coverability problem is decidable for TPN.

Chapter 3

Timed Nets

3.1 Timed ν-Petri nets

Our main goal in this chapter is to define an extension of Petri nets combining two orthogonal capabilities: the representation of different instances of the same net, which can be created fresh and synchronize with other instances, and the management of time. We start from a model with the first capability, ν-Petri nets. Then, we need to consider a model of time to extend ν-PN. As exposed in Chapter 1, there are plenty of ways of extending Petri nets with time. Our first attempt is to extend ν-PN in the way of Timed Petri Nets [10], which is the most expressive of these models for which coverability is still decidable. Therefore, we provide each token (which carries a name) with a real clock, and label the incoming and outcoming arcs of transitions by intervals (besides the variables).

Definition 3.1.1 (Timed ν-Petri nets) *Let Var be a set of variables and $\Upsilon \subset Var$ a subset of special variables. A* Timed ν-Petri net *(ν-TPN for short) is a tuple $N = \langle P, T, In, Out, Time \rangle$, where:*

- *P and T are finite disjoint sets of places and transitions, respectively,*
- *$In : T \to (P \times Var \times \mathcal{I})^{\oplus}$ is the input function,*
- *$Out : T \to (P \times Var \times \mathcal{I})^{\oplus}$ is the output function.*

Given a transition $t \in T$, we take $Var(t)$ as the set of variables adjacent to t, that is, $Var(t) = \{x \in Var \mid \exists p \in P, I \in \mathcal{I},\ (p, x, I) \in In(t) + Out(t)\}$. In figures, for each $(p, x, I) \in In(t)$ we draw an arc from p to t, labeled by x, I (and analogously for postconditions).

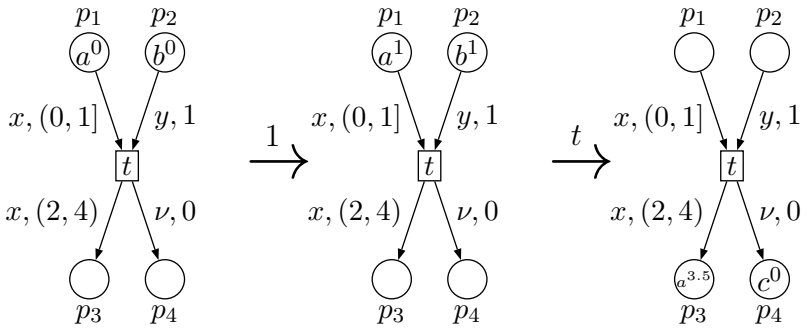


Figure 3.1: Firing of transitions in a ν-*TPN*

Example 3.1.1 *Fig. 3.1 shows a ν-TPN with $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t\}$, $In(t) = \{(p_1, x, (0, 1]), (p_2, y, [1, 1])\}$ and $Out(t) = \{(p_3, x, (2, 4)), (p_4, \nu, [0, 0])\}$. Therefore, $Var(t) = \{x, y, \nu\}$.*

Now, we define a semantics for this syntax. Each token in a marking is provided with a name, a clock and the place where it is. Then, a marking is the multiset of these tokens. For the formal definition of marking we consider a fixed infinite set Id of names.

Definition 3.1.2 (Markings) *A* token *of a ν-TPN is an element of $P \times Id \times \mathbb{R}_{\geq 0}$. A* marking *is a finite multiset of tokens.*

We write $p(a, r)$ instead of (p, a, r) to denote tokens. Intuitively, $p(a, r)$ is a token in p, carrying the name a, with clock value r. We use M, M', M_1,... to range over markings. We say M *marks* $p \in P$ if there are $a \in Id$ and $r \in \mathbb{R}_{\geq 0}$ such that $p(a, r) \in M$. We denote $Id(M) = \{a \mid \exists p, r,\ p(a, r) \in M\}$. In figures, tokens are depicted as names with its age as a superscript. We assume $\bullet \in Id$, so that black tokens can appear in markings as in ordinary Petri nets. If an arc is not labeled by any variable we assume that the token involved is $\bullet$. Moreover, in figures we do not write the interval $[0, \infty)$. Hence, ordinary notations in Petri nets can be used.

Let us define how the markings may evolve in the executions. As expected, they may change in two different ways: the elapsing of time (continuous or timed transitions) and the firing of transitions (discrete transitions). In the following definition, time elapsing is accomplished by simply adding the same amount of time to each token in the net.

Definition 3.1.3 (Firing of timed transitions) *Given a marking $M = \{p_1(a_1, r_1), \ldots, p_n(a_n, r_n)\}$ and a delay $d \in \mathbb{R}_{\geq 0}$, we write M^{+d} to denote the*

marking $\{p_1(a_1, r_1 + d), \ldots, p_n(a_n, r_n + d)\}$ *in which the value of the clocks of all tokens has increased by* d. *Then we write* $M \xrightarrow{d} M^{+d}$.

In order to fire a transition $t \in T$, we assign an identifier to each of the variables in $Var(t)$, in such a way that the identifiers assigned to special variables in Υ are fresh names (not in the previous marking). Moreover, we need to ensure that the preconditions are fulfilled, that is, if we assign the identifier a to variable x, then for each $(p, x, I) \in In(t)$ there is a token $p(a, r)$ in the current marking such that $r \in I$. Finally, the corresponding tokens are removed/added from/to the net according to In, Out and the assignation we have chosen.

Definition 3.1.4 (Firing of discrete transitions) *Let* $t \in T$ *be a transition with* $In(t) = \{p_1(x_1, I_1), \ldots, p_n(x_n, I_n)\}$ *and* $Out(t) = \{q_1(y_1, J_1), \ldots, q_m(y_m, J_m)\}$. *We say* t *is* enabled *or* can be fired *in marking* M, *evolving to* M', *and we denote it by* $M \xrightarrow{t} M'$, *if there is an injection* $\sigma : Var(t) \to Id$, $r_1, \ldots, r_n \in \mathbb{R}_{\geq 0}$ *and* $r'_1, \ldots, r'_m \in \mathbb{R}_{\geq 0}$ *such that:*

- $r_i \in I_i$ *for all* $i \in n^+$ *and* $r'_j \in J_j$ *for all* $j \in m^+$,
- $\sigma(\nu) \notin Id(M)$ *for all* $\nu \in \Upsilon$,
- $\{p_1(\sigma(x_1), r_1), \ldots, p_n(\sigma(x_n), r_n)\} \subseteq M$,
- $M' = (M - \{p_1(\sigma(x_1), r_1), \ldots, p_n(\sigma(x_n), r_n)\}) + \{q_1(\sigma(y_1), r'_1), \ldots, q_m(\sigma(y_m), r'_m)\}$.

Putting together the two previous definitions we obtain the semantics of ν-TPN, so that we write $M \to M'$ if $M \xrightarrow{t} M'$ for some $t \in T$ or $M \xrightarrow{d} M'$ for some $d \in \mathbb{R}_{\geq 0}$.

Example 3.1.2 *The nets in Fig. 3.1 show the same* ν-TPN *with three different markings. In the first marking the transition* t *is not enabled, since the age of the only token in* p_2 *is not in* $[1, 1]$. *However, after a delay of one unit of time,* t *becomes enabled, and can be fired reaching, for example, the marking depicted in the right. If we call* M_1, M_2 *and* M_3 *the markings represented in the first, second and third nets, respectively,* $M_1 \xrightarrow{1} M_1^{+1} = M_2$ *and* $M_2 \xrightarrow{t} M_3$ *with mode* σ, *where* $\sigma(x) = a$, $\sigma(y) = b$ *and* $\sigma(\nu) = c$.

Notice that we are defining a weak semantics, in which time elapsings can happen even if they disable transitions. For instance, from M_1 in Fig. 3.1 two units of time can elapse, which disables the firing of t forever.

Now that we have defined our model and its semantics, we focus on solving the decidability of the control-state reachability problem.

Definition 3.1.5 *The* control-state reachability *problem is that of deciding, given a ν-TPNN and a place p of N, whether p is marked in some reachable marking.*

As the previous definition does not depend on the time model, it will apply to the rest of the models in the paper. We use this terminology, even if places in ν-TPN are not necessarily control-states. In fact, if we consider that $m \in (P \times Id)^{\oplus}$ is coverable if there is a reachable marking M such that its tokens cover m in an untimed manner (as in ν-PN), then control-state reachability is equivalent to coverability: control-state reachability is in fact coverability of the marking with only one token in the control-state place, and coverability of a marking m can easily be reduced to control-state reachability by adding to the net a control-state place p and a transition with tokens in marking m as precondition and p as postcondition (without caring about the ages of the tokens).

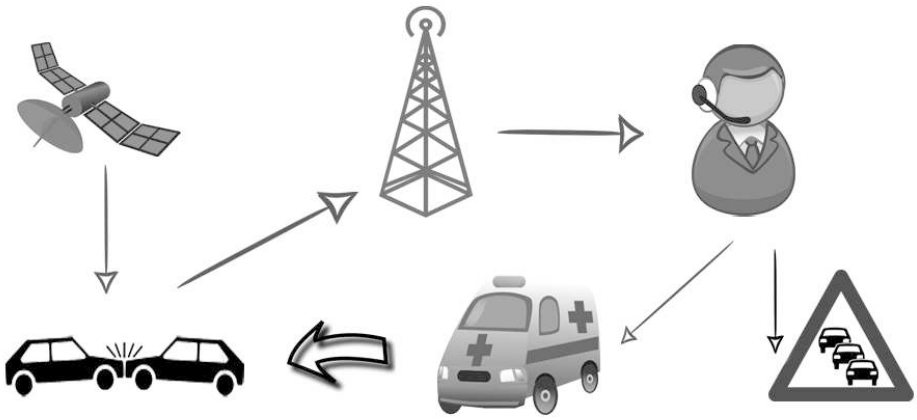


Figure 3.2: How eCall works?

Let us illustrate the previous definitions with an example.

Example 3.1.3 *It is well known that in case of a crash, the time of reaction of the emergency services is fundamental to save lifes.* 75% *of the deaths caused by an accident happen in the "golden hour", the first hour after the crash. The European Union has voted that from April of 2018 all the cars must be equiped with the so-called eCall system (http://www.imobilitysupport.eu/), which is a system*

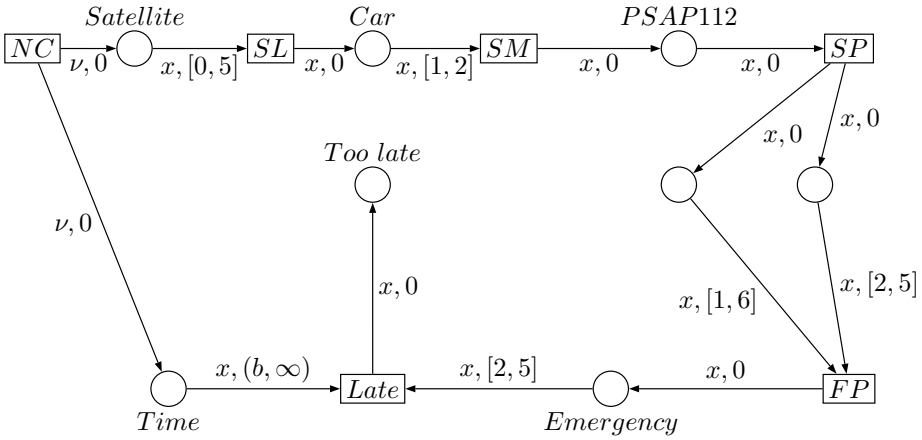


Figure 3.3: A ν-*TPN* representing the eCall system

that automatically calls the nearest emergency centre in case of a crash, sending a "minimum set of data" (MSD), which includes the exact location of the accident. Hence, in a short time the emergency services know that an accident has happened and where it is.

Fig. 3.2 shows how eCall works in a simplified way. When the crash happens, a satellite indicates the precise location of the car. Then, the MSD is sent from the car to the Public Safety Answering Point 112 (PSAP112), where an operator can see the location of the incident, call the emergency services and give information of the accident to the traffic information and management centre. Then, an ambulance is sent to the location of the accident.

The ν-TPN in Fig. 3.3 represents the eCall system. For simplicity, we write $x, 0$ instead of $x, [0, 0]$ in the labels of the arcs. Each name in the net represents a different accident. The firing of NC creates a new name which represents a new accident. A token of this new name is set to place time. The age of this token will represent the time that has elapsed since the corresponding crash. Transition Late can only fire if more than b units of time have elapsed since the accident and the ambulance has arrived to the location at this moment (where b is the maximum amount of time we consider acceptable to get the ambulance). Hence, we want to ensure that place Too Late cannot be marked, that is, we are interested in solving the control-state reachability problem for place Too Late. The different steps of the processing of the information of the accident with their time constraints are simulated by the rest of the transitions of the net.

Note that as we are not considering urgency, some tokens can get stuck in the net if they become too old. For example, if there is a token older than 5 in place

satellite it gets stuck in this place. However, this does not affect the property we want to verify, since Late can only be marked by instances which fulfill the time constraints of the arcs.

3.1.1 Undecidability of safety properties for ν-*TPN*

We are going to prove that control-state reachability is undecidable for ν-*TPN* by giving a reduction from a Turing-complete model. Despite the fact that there are many well-known Turing complete formalisms, such as Minsky or Turing Machines, for simplicity, we prefer to perform a reduction from a model which is closer to ν-*TPN*. In particular, we choose ν-Replicated net systems [101, 79], which are collections of ν-PN which can synchronize with each other and create new instances of the net. Let us formally introduce this model before giving the proof of undecidability.

3.1.1.1 ν-replicated net systems

Let us fix an arbitrary set $\mathcal{S}$ of service names and let $Sync = \{s?, s! \mid s \in \mathcal{S}\}$. We will use the elements of *Sync* for the synchronizations, in the way of *CCS*.

Definition 3.1.6 (ν-*RN* systems [101, 79]) *A* ν-*RN* system *is a tuple* $N = \langle P, T, F, H, \lambda \rangle$*, where:*

- $\langle P, T, F, H \rangle$ *is a* ν-*PN,*
- $\lambda : T \to \mathcal{L}$ *assigns a label to each transition,*

where $\mathcal{L} = (\mathcal{S} \cup Sync) \times (P \times Id)^{\oplus}$*. An* instance *of* N *is an element of* $(P \times Id)^{\oplus}$*. A* marking *of* N *is a multiset of instances of* N*.*

Intuitively, when two transitions are labelled by $s!$ and $s?$ respectively, for some $s \in \mathcal{S}$, they can synchronize their firings, creating a new instance of the net. We write $Var(t)$ to denote the set of arcs adjacent to t. For two instances M and M' we write $M \xrightarrow{t(\sigma)} M'$ if M can reach M' after the firing of t with mode σ, following the semantics of ν-*PN*. We write $Id(\mathcal{M})$ to denote the set of names that appear in marking $\mathcal{M}$. We identify any marking $\mathcal{M}$ with $\mathcal{M} + \{\emptyset\}$, so that empty instances, ie, instances without tokens, can be removed.

Definition 3.1.7 (Firing of autonomous transitions) *Let* $t \in T$ *be such that* $\lambda(t) = (s, \overline{M})$*, and* M *and* M' *be two instances such that* $M \xrightarrow{t(\sigma)} M'$ *with* $\sigma(\nu) \notin$

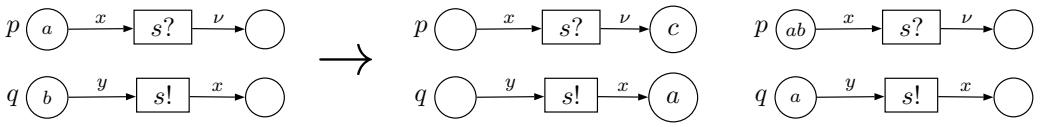


Figure 3.4: Synchronous firing in a *ν-RN* system.

$Id(\overline{M})$ *for* $\nu \in \Upsilon$. *Then* $\{M\} + \mathcal{M} \xrightarrow{t} \{M', \overline{M}\} + \mathcal{M}$ *for any marking* $\mathcal{M}$ *such that* $\sigma(\nu) \notin Id(\mathcal{M})$ *for* $\nu \in \Upsilon$.

Definition 3.1.8 (Firing of synchronizing transitions) *Let* $t_1, t_2 \in T$ *be such that* $\lambda(t_1) = (s?, \overline{M}_1)$ *and* $\lambda(t_2) = (s!, \overline{M}_2)$ *for some* $s \in \mathcal{S}$, *and let* M_1, M_1', M_2 *and* M_2' *be instances such that* $M_i \xrightarrow{t_i(\sigma_i)} M_i'$ *with* $\sigma_1(x) = \sigma_2(x)$ *for all* $x \in Var(t_1) \cap Var(t_2)$ *and* $\sigma(\nu) \notin Id(\overline{M}_i)$ *for* $i = 1, 2$.
Then $\{M_1, M_2\} + \mathcal{M} \xrightarrow{(t_1,t_2)} \{M_1', M_2', \overline{M}_1, \overline{M}_2\} + \mathcal{M}$ *for any marking* $\mathcal{M}$ *such that* $\sigma(\nu) \notin Id(\mathcal{M})$ *for* $\nu \in \Upsilon$.

Notice that when $\overline{M}$, $\overline{M}_1$ or $\overline{M}_2$ are empty then no instance is created. For more details about *ν-RN* systems, see [101, 79].

Example 3.1.4 *Figure 3.4 shows the firing of two compatible transitions* t_1 *and* t_2, *which create a new instance* M *with* $M(p) = \{a, b\}$ *and* $M(q) = \{a\}$.

Definition 3.1.9 (Control-state reachability) *We define the* control-state reachability problem *as that of deciding, given a ν-RN system* N *and a place* p *of* N, *whether there is a reachable marking containing an instance* M *such that* $(p, a) \in M$ *for some* $a \in Id$.

In [79] *ν-RN* systems are proved to be Turing-complete. Moreover, termination for Turing machines can be easily reduced to control-state reachability for *ν-RN*, which is therefore undecidable.

3.1.1.2 Proof of undecidability

In order to simplify the proof of undecidability of control-state reachability for *ν-TPN*, we can safely assume that only autonomous transitions can create new instances in a *ν-RN* system (and only one in each firing). Moreover, the new instances will all have the same marking, consisting of a token in a single place (denoted by p_0). Indeed, such *ν-RN* are enough to simulate Turing machines, as shown in [79].

Proposition 3.1.1 *Control-state reachability is undecidable for ν-TPN.*

Proof: We reduce control-state reachability for ν-RN systems to our problem. Given a ν-RN $N = \langle P, T, In, Out, \lambda \rangle$, we build a ν-TPN $N' = \langle P', T', In', Out' \rangle$ which simulates it. In particular, we build N' such that $P \subset P'$, and a place $p \in P$ can be marked in N iff it can be marked in N'.

Intuitively, we represent each instance of N by a multiset of tokens with the same clock value in N'. The construction guarantees that all the transitions in N' use only tokens with clocks set to 1. Hence, tokens with clocks older than 1 are dead tokens, that cannot be used for the firing of transitions. In order to allow instances not to become dead, we will add transitions that reset tokens with clock 1 to 0. These transitions may not reset every token with clock 1, in which case some tokens are lost (after the elapsing of time). Therefore, in some simulations some tokens are lost, but there are also perfect simulations in which no tokens are lost. In this sense our simulation is lossy, though it preserves control-state reachability, since losing tokens can only remove behavior (no spurious behavior is introduced). We also guarantee in our construction that we do not merge instances, that is, that no two tokens with different clock values may end up having the same clock value.

Executions in N' simulate executions of N in two steps: In the first step N' creates an unbounded number of tokens with different clock values, which represent all the instances that may take part in the simulation. Again, there are executions which may get stuck because we do not generate all the tokens we need in this first phase. However, if the control place is marked in some execution r of N, then there are executions of N' which generate enough tokens in the first step to simulate r. The second step is the simulation itself. We consider in N' two places s_1 and s_2 (marked in mutual exclusion) to specify in which of the two steps the simulation currently is.

Step 1 (creation of instances): In the first step, depicted in Fig. 3.5, we repeatedly fire a transition *new*, which creates new tokens with clock 0 in place *ins*. The clock of each token in *ins* will represent a different instance of N, so that we need to ensure that they are all different. We do that by forcing some time elapsing between two consecutive firings of *new*, by demanding that the token in s_1 is strictly older than 0 when *new* is fired (and setting it back to 0). Initially, there is only one token in place s_1, with clock 0.

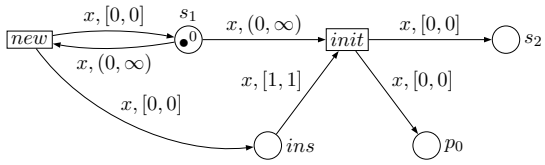


Figure 3.5: Creation of instances

The firing of a transition *init* concludes step 1, by moving the token in s_1 to s_2 when the token in s_1 has a non-null clock. It also sets the initial marking of N, by taking a token of clock value 1 from *ins* and putting it in p_0, with clock 0. Notice that this guarantees that the clock value of the token in the initial instance is different from all the clock values of the tokens in *ins*.

Step 2 (simulation of transitions): As mentioned before, only tokens with clocks between 0 and 1 (both included) are valid tokens, that represent a token in some instance. Step 1 guarantees that at the beginning of step 2 there are no two tokens having clocks set to 0 and 1, respectively. Moreover, at any point in step 2, two tokens in P with clocks 0 and 1 belong to the same instance. Now we show how we reset the clock of tokens, and how we simulate the firing of autonomous transitions (possibly creating a fresh instance), and the synchronization of two compatible transitions.

Resetting tokens: In order to be able to perform perfect (non-lossy) simulations, we need to be able to reset the clock of tokens with value 1. For that purpose, for each place $p \in P'$ we add a transition t_p which takes from p a token of clock 1 and puts it back with clock set to 0.[1] Formally, $In'(t_p) = \{(p, x, [1, 1])\}$ and $Out'(t_p) = \{(p, x, [0, 0])\}$. Notice that this is correct because before resetting there are no tokens with clock set to 0.

Simulation of the firing of an autonomous transition: The simulation of the (autonomous) transition $t \in T$ is simply achieved by demanding that the clock of all tokens involved in the firing is set to 1. Thus, we consider $t \in T'$, and we attach the interval $[1, 1]$ to every arc adjacent to t. More precisely, if $(p, x) \in In(t)$ then $(p, x, [1, 1]) \in In'(t)$ (and analogously for postconditions). We also add place s_2 as pre/postcondition of t. Moreover, if t creates a fresh instance, it puts a token in a new place *act*. Intuitively, we store in *act* a token (of any age) for each instance that the simulation has created, but that has not been initialized yet. In order to initialize new instances, we add a new transition t_{set}, which takes a token from

[1] It is enough to reset places in which the clock is meaningful, unlike e.g. s_2.

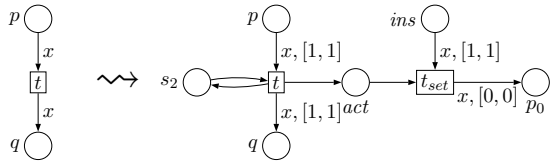


Figure 3.6: Simulation of the firing of t, assuming t creates a fresh instance

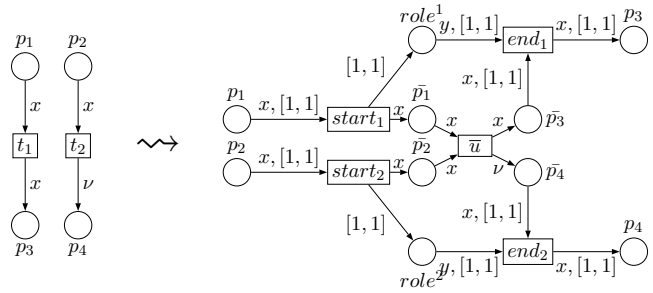


Figure 3.7: Synchronizing transitions

act and a token with clock value 1 from *ins*, and puts a token in p_0 with clock set to 0, analogously as *init* (see Fig.3.6). Again, notice that when there is a token with clock value 1 in *ins* there is no token with clock 0 in the whole net, so that we are correctly creating the new instance.

Simulation of synchronizing transition: Let us see how we simulate the firing of $u = (t_1, t_2) \in T \times T$, where t_1 and t_2 are two compatible transitions according to λ(see Fig. 3.7). We simulate u by means of the consecutive firing of transitions $start_u^1$, $start_u^2$, $\overline{u}$, end_u^1 and end_u^2 in T'. We guarantee (thanks to s_2 and new control places, not shown in Fig. 3.7) that these transitions can only be fired in the order shown, and that $start_u^1$ can only be fired when there is a token in s_2 (no simultaneous simulations of firings can take place).

Let us consider in P' new places, $role^1$ and $role^2$ (whose content can also be reseted, as explained above), and for each $p \in P$ let us consider $\overline{p} \in P'$. The firing of $start_u^1$ removes the tokens from the preconditions p of t_1 with clock value 1 and puts them in the corresponding $\overline{p}$ (with any value for the clock). More precisely, if $(p, x) \in In(t_1)$ then $(p, x, [1, 1]) \in In'(start_u^1)$ and $(\overline{p}, x, [0, \infty)) \in Out'(start_u^1)$. Moreover, a token (with any name, e.g. a black token) is added to $role^1$ with clock value 1. The case of $start_u^2$ is analogous.

The firing of $\overline{u}$ simulates the firing of u (that is, the simultaneous firing of t_1 and t_2) in the overlined places. More precisely, if $(p, x) \in In(t_i)$ for $i \in \{1, 2\}$ then $(\overline{p}, x, [0, \infty)) \in In'(\overline{u})$ (and analogously for postconditions). In particular, it

checks that names in different places are matched according to the variables in the arcs, and new names are created if needed. Notice that if the names selected by $start_u^1$ and $start_u^2$ do not match then $\overline{u}$ is disabled. Hence, our simulation may introduce deadlocks, though it still preserves control-state reachability. Notice also that this firing can take place independently of the clocks of the tokens involved.

Finally, transitions end_u^1 and end_u^2 set the clocks of the tokens involved in the firing of u to their correct values. For that purpose, end_u^i takes the token from $role^i$ with clock value 1, and for every p postcondition of t_i it takes the token in $\overline{p}$ and puts it in p with clock value 1. More precisely, for $i = 1, 2$, $(role^i, y, [1, 1]) \in In'(t_i)$ (where y is a *fresh* variable), and if $(p, x) \in Out(t_i)$ then $(\overline{p}, x, [0, \infty)) \in In'(end_u^i)$ and $(p, x, [1, 1]) \in Out'(end_u^i)$.

The previous simulation preserves control-state reachability. Indeed, if p is marked by some execution of N, then that execution can be perfectly simulated, ending up in a marking that marks p. Conversely, if p is marked by some execution of N', by construction that execution corresponds to the simulation of some execution of N which also marks p (possibly with more tokens, if some were lost).

□

We have proved that control-state reachability is undecidable for ν-*TPN* by performing a reduction in which, basically, we use clocks to represent instances. We think that this technique may be generalizable to perform reductions between models with time and models with (maybe ordered) instances.

Next, we prove that control-state reachability is undecidable even for ν-*TPN* with two tokens (and therefore two clocks) per instance. We prove it by performing a reduction from the same problem for Timed Petri nets with two clocks per token, which is undecidable [1]. Let us first define such model:

Definition 3.1.10 (Timed Petri net with two clocks per token) *A timed Petri net with two clocks per token (2TdPN for short) is a tuple $\langle P, T, F, H\rangle$ where P and T are two finite disjoint sets of places and transitions, respectively, and $F, H : P \times T \to (\mathcal{I} \times \mathcal{I})^{\oplus}$.*

The semantics of *2TdPN* is similar to the semantics of *TdPN*. In fact, the only difference between them is the presence of a second clock per token, which is managed exactly as the other one. Hence, a marking m of a *2TdPN* is an element of $(P \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0})^{\oplus}$. Intuitively, $(p, r, s) \in m$ means that there is a token with clocks r and s in place p at the marking m. In a notation abuse, we

denote by $m(p)$ the multiset of pairs of clocks of tokens in p at the marking m. Again, the marking of a *2TdPN* may evolve in two ways: by elapsing time and firing discrete transitions. Given a marking $m = \{(p_1, r_1, s_1), \ldots, (p_n, r_n, s_n)\}$ and $d \geq 0$, we write $m \xrightarrow{d} m'$, where $m' = \{(p_1, r_1 + d, s_1 + d), \ldots, (p_n, r_n + d, s_n + d)\}$, that is, the marking in which the two clocks of every token are d units of time older. A transition t is enabled at a marking m if for each $p \in P$ with $F(p,t) = \{(I_1^1, I_1^2), \ldots, (I_n^1, I_n^2)\}$, there is $In_p = \{(r_1, s_1), \ldots, (r_l, s_l)\} \subseteq m(p)$ such that for each $i \in \{1, \ldots, l\}$, $r_i^1 \in I_i^1$ and $r_i^2 \in I_i^2$. Then, t can be fired from m, reaching a new marking m' such that for each $p \in P$ with $H(p,t) = \{(J_1^1, J_1^2), \ldots, (J_l^1, J_l^2)\}$, $m'(p) = (m(p) - In_p) + Out_p$, where $Out_p = \{(j_1^1, j_1^2), \ldots, (j_l^1, j_l^2)\}$ with $j_i^1 \in J_i^1$ and $j_i^2 \in J_i^2$ for each $i \in \{1, \ldots, l\}$, and denote this by $m \xrightarrow{t} m'$. Finally, we write $m \to m'$ if there is $d \geq 0$ with $m \xrightarrow{d} m'$ or there is $t \in T$ with $m \xrightarrow{t} m'$.

The control-state reachability problem for *2TdPN* is defined in the same way as for *ν-TPN*, that is, given a *2TdPN* with initial marking m_0 and a place p, the control-state reachability problem is that of deciding if there is a marking which is reachable from m_0 and marks the place p. Undecidability of control-state reachability problem is proved in [1], by defining a reduction from the control-state reachability problem for 2-counter machines.

Now, we are ready to prove that control-state reachability is undecidable for *ν-TPN* with only two tokens per instance.

Proposition 3.1.2 *Given a 2TdPN N with initial marking m_0 and a place p of N, there is a ν-TPN N' with an initial marking m_0', and a place p' of N' such that p is reachable from m_0 in N if and only if p' is reachable from m_0' in N'. Moreover, all the markings of N' that are reachable from m_0' have at most two tokens per instance.*

Proof: Given a *2TdPN* $N = \langle P, T, F, H \rangle$ we define the corresponding *ν-TPN* $N' = \langle P', T', In, Out \rangle$ with:

- $P' = \{p^1, p^2 \mid p \in P\}$ and $T' = T$.
- For each $t \in T$, $In(t) = \biguplus_{p \in P} \biguplus_{(I_i^1, I_i^2) \in F(p,t)} \{(p^1, x_i, I_i^1)\} \cup \{(p^2, x_i, I_i^2)\}$.
- Analogously, for each $t \in T$, $Out(t) = \biguplus_{p \in P} \biguplus_{(I_i^1, I_i^2) \in H(p,t)} \{(p, x_i, I_i^1)\} \cup \{(p, x_i, I_i^2)\}$.

Moreover, given a marking $m = \{(p_1, r_1, s_1), \ldots, (p_n, r_n, s_n)\}$ of N, we define the corresponding marking of N' as $m' = \{p_1^1(a_1, r_1), p_2^1(a_2, r_2), \ldots, p_n^1(a_n, r_n)\} + \{p_1^2(a_1, s_1), p_2^2(a_2, s_2), \ldots, p_n^2(a_n, s_n)\}$.

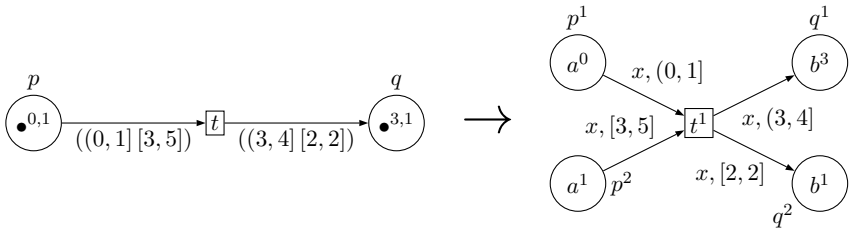


Figure 3.8: From *2TdPN* to *ν-TPN*.

Intuitively, we represent the first and the second clock of the tokens of N in the first and second copy of the places of N in N', respectively, as depicted in Fig. 3.8. In fact, it is trivial to prove that $m_1 \to m_2$ in N if and only if $m'_1 \to m'_2$ in N'. Hence, p of N is reachable from m_0 if and only if the place p^1 (or p^2) is reachable from m'_0.

□

Therefore, we obtain the next corollary:

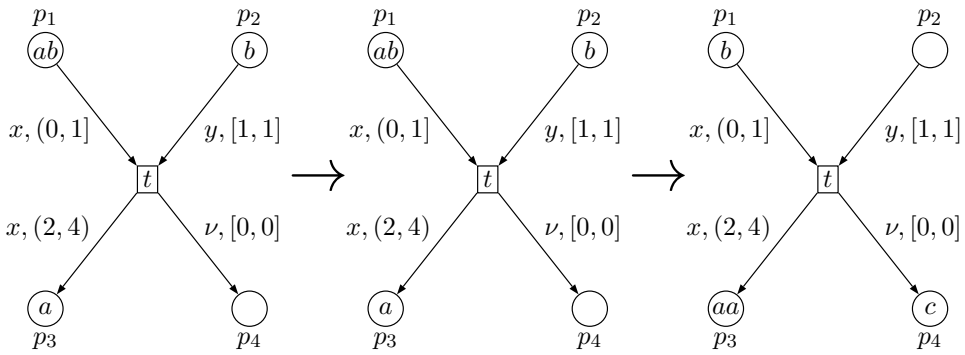
Corollary 3.1.3 *Control-state reachability is undecidable for ν-TPN with only two tokens per instance.*

3.2 Locally synchronous ν-Petri nets

In this section we define and study a new timed extension of ν-Petri nets which differs from the previous one in the number of clocks: this time, our model is locally synchronous, meaning that all the tokens of each instance share one clock. The timed restrictions of the firings are expressed in the same way as *ν-TPN*. Hence, this new model could be expressed as a syntactic restriction of the previous one, by selecting a distinguished token per instance, which would carry the age of the instance. However, we prefer to define *locally synchronous ν-Petri nets* from scratch, thus obtaining more manageable notations to be used in the forthcoming proofs. Again, we fix a set of variables *Var* and $\Upsilon \subset Var$ a subset of special variables.

Definition 3.2.1 (Locally synchronous ν-PN) *A locally synchronous ν-Petri net (ν-lsPN for short) is a tuple* $N = \langle P, T, In, Out, Time \rangle$ *where:*

- *P and T are finite disjoint sets of places and transitions, respectively,*



$M_1 = a{:}(\{p_1, p_3\}, 0),\ b{:}(\{p_1, p_2\}, 0.5)$ $M_2 = a{:}(\{p_1, p_3\}, 0.5),\ b{:}(\{p_1, p_2\}, 1)$ $M_3 = a{:}(\{p_3, p_3\}, 3), b{:}(\{p_1\}, 1),\ c{:}(\{p_4\}, 0)$

Figure 3.9: Firing of a transition in a ν-*lsPN*.

- *for each* $t \in T$, $In_t, Out_t : Var \to P^{\oplus}$ *are the input and output functions of* T, *respectively,*
- *for each* $t \in T$, $Time_t : Var \to \mathcal{I} \times \mathcal{I}$ *is the time constraints function of* T.

We denote by $Time_t^1(x)$ and $Time_t^2(x)$ the first and the second component of $Time(x)$, respectively. Again, for each $t \in T$ we define $Var(t) = \{x \in Var \mid In_t(x) + Out_t(x) \neq \emptyset\}$, that is, the set of variables labelling arcs adjacent to t. Moreover, we split this set into the subsets $nfVar(t) = Var(t) \backslash \Upsilon$ and $fVar(t) = Var(t) \cap \Upsilon$. For example, in any of the nets depicted in Fig. 3.9, $Var(t) = \{x, y, \nu\}$, $fVar(t) = \{\nu\}$ and $nfVar(t) = \{x, y\}$.

In order to define the markings for ν-*lsPN*, we fix a set Id of names.

Definition 3.2.2 (Markings of ν-*lsPN*) *A marking* M *of a* ν-*lsPN is an expression of the form* $a_1{:}(m_1, r_1), \ldots, a_n{:}(m_n, r_n)$, *where* $Id(M) = \{a_1, ..., a_n\} \subset Id$ *are pairwise different names, and for each* $i \in n^+$, $\emptyset \neq m_i \in P^{\oplus}$ *and* $r_i \in \mathbb{R}_{\geq 0}$.

We call *instances* the elements of the form $a{:}(m, r)$, and we say that $a{:}(m, r)$ is an instance with name a, tokens according to m and age (or clock) r. We treat markings as multisets of instances, and we use $M, M', \ldots$ to range over them. We assume that each m_i is not empty and we say that a marking M marks a place $p \in P$ if there is $a{:}(m, r) \in M$ such that $p \in m$.

Now, we define the semantics of the model. As in the previous one, markings may evolve in two ways: the elapsing of time and the firing of transitions.

Definition 3.2.3 (Firing of timed transitions) *Given* $M = a_1 : (m_1, r_1), ..., a_n : (m_n, r_n)$ *and* $d \in \mathbb{R}_{\geq 0}$, *we write* M^{+d} *to denote the marking* $a_1 : (m_1, r_1 + d), ..., a_n : (m_n, r_n + d)$, *in which the clock of every instance has increased by* d. *We write* $M \xrightarrow{d} M^{+d}$.

In order to define the firing of discrete transitions, we need to be able to add and remove instances $a:(m,r)$ with empty m, obtained when removing the preconditions of a transition. For that purpose, we say that M' is an $\emptyset$-expansion of a marking M (or M is an $\emptyset$-contraction of M') if M' is obtained by adding instances $a : (\emptyset, r)$ to M.

Definition 3.2.4 (Firing of discrete transitions) *Let* $t \in T$ *with* $nfVar(t) = \{x_1, ..., x_n\}$ *and* $fVar(t) = \{\nu_1, ..., \nu_k\}$. *We say* t *is* enabled *at marking* M *if:*

- $M = a_1 : (m_1, r_1), ..., a_n : (m_n, r_n) + \overline{M}$, *for some* $\overline{M}$,
- *for each* $i \in n^+$, $In_t(x_i) \subseteq m_i$ *and* $r_i \in Time_t^1(x_i)$.

Then, t *can be* fired, *and taking*

- $\{b_1, ..., b_k\}$ *pairwise different names not in* $Id(M)$,
- $m'_i = (m_i - In_t(x_i)) + Out_t(x_i)$ *for all* $i \in n^+$,
- $m''_j = Out_t(\nu_j)$ *for all* $j \in k^+$,
- r'_i *any value in* $Time_t^2(x_i)$, *for all* $i \in n^+$,
- r''_j *any value in* $Time_t^2(\nu_j)$, *for all* $j \in k^+$,

we can reach M', *denoted by* $M \xrightarrow{t} M'$, *where* M' *is the* $\emptyset$*-contraction of* $a_1 : (m'_1, r'_1), ..., a_n{:}(m'_n, r'_n), b_1{:}(m''_1, r''_1), ..., b_k{:}(m''_k, r''_k) + \overline{M}$

In order to understand how the firing of transitions works, focus on Fig. 3.9, which depicts a ν-*lsPN* with three different markings. In the first marking M_1 the transition t is not fireable, because no instance with a clock value in $[1, 1]$ has a token in place p_2. However, after waiting 0.5 units of time (that is, after firing $M_1 \xrightarrow{0.5} M_2$), the marking M_2 is reached, and t becomes enabled. Then, we can fire t by assigning a to x, b to y and c to ν thus reaching, for example, the marking M_3 in the figure.

Let us give a more realistic example of a ν-*lsPN* representing the Fischer's protocol, to show how ν-*lsPN* can represent real life processes.

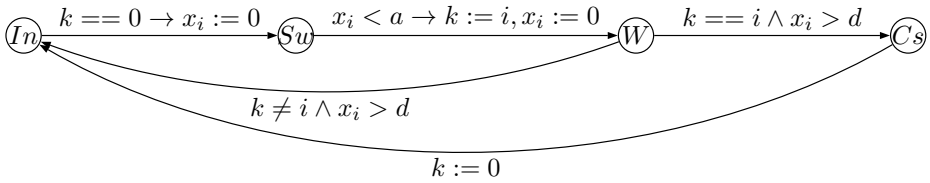


Figure 3.10: Timed automaton modeling the i-th process of Fischer's mutual exclusion protocol

Example 3.2.1 Fischer's protocol:

We model a parameterized version of Fischer's protocol for mutual exclusion, which considers n processes p_i (where n is a parameter), each of those endowed with a real clock x_i. Moreover, a shared integer variable $k \in \{1, \ldots, n\}$ is considered, in order to set the turn for entering the critical section.

Each process p_i can be modeled by the timed automaton of Fig. 3.10, and behaves as follows:

```
1  repeat
2    non critical section                7    until k==i;
3    repeat                              8    critical section;
4      await k==0;                       9    k:=0;
5      k:=i;                             10   non critical section
6      delay(d);                         11 until false;
```

Process p_i repeatedly tries to enter the critical section (state In). For that purpose, it waits until $k = 0$, which means that no other process is in the critical section (state Sw). Then, it sets $k := i$, to ask for permission to enter (state W). After a delay of d units of time, if k is still i, the process enters the critical section (state Cs), setting $k = 0$ when it leaves. Otherwise, it repeats lines $4-6$. In order to make the algorithm satisfy the mutual exclusion property, it is important to fix a proper delay d, greater than the time a each process takes to execute line 5. Then in Fig. 3.10 we take $a < d$.

Let us define our model: We consider the net depicted in Fig 3.11. Intuitively, each token in places In, Sw, W and Cs represents a different instance. The variable k is represented by a place k that contains a black token if $k = 0$ or a token with the identifier that changed its last value. When a transition t is fired, if there are two different variables $x, y \in Var(t)$, then the names of the tokens associated to x and y in the firing are different (hence checking if $k == i$). Notice the transition new*, that can create any number of processes in their initial state. To prove mutual exclusion, we have to prove that no marking with two tokens in place Cs can be reached, which can be easily reduced to control-state reachability.*

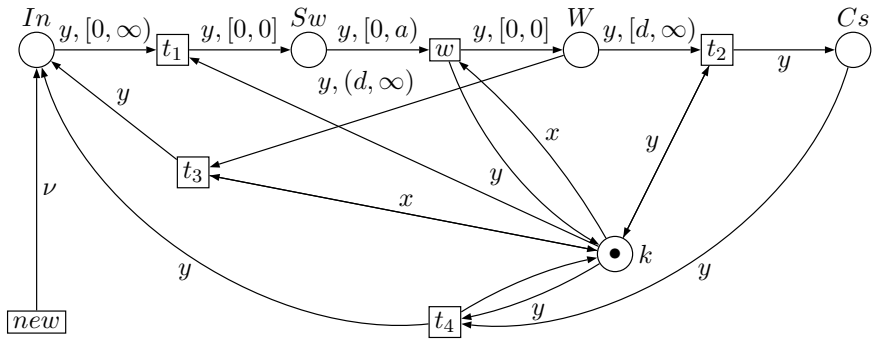


Figure 3.11: Fischer's mutual exclusion protocol as a *ν-lsPN*

Note that the timed automaton in Fig. 3.10 modeling Fischer's protocol and our parametric model are very similar. In [5], the authors model Fischer's protocol using Timed Petri nets. As they do not use colors, they need to use a counting abstraction (which considers the state space of each process and the shared variable), and the obtained model is far more complicated than ours.

3.2.1 Decidability of control-state reachability for *ν-lsPN*

Let us recall that the control-state reachability problem for *ν-lsPN* is defined analogously as for *ν-TPN*, as reachability of the set of markings which mark a control place p. In order to prove its decidability, we first need to define a more manageable model, equivalent to *ν-lsPN*, up to control-state reachability. Note that our model is infinite in three dimensions: It may reach markings with an unbounded number of instances, each of which is potentially unbounded and endowed with a clock in an uncountable domain. Moreover, any marking has infinitely-many possible successors due to time delays, and therefore, the transition system induced by a *ν-lsPN* is not finitary. We use the theory of regions [6, 10] to obtain a model with finitary transition system over a countable domain, which is equivalent to *ν-lsPN*, up to control-state reachability. Moreover, the model we define will be a $WSTS$, thus obtaining the decidability of the control-state reachability problem by reducing it to a coverability problem.

In this section, we fix a *ν-lsPN* $N = \langle P, T, In, Out, Time \rangle$ and we denote by max the maximum integer bound appearing in $Time$.

Definition 3.2.5 (Regions) *A* region *is an expression of the form* $A_0 * A_1 * \ldots * A_n * A_\infty$ *with* $n \geq 0$*, where* $A_i \in (P^\oplus \times I_i)^\oplus$ *for every* $i \in n^*_\infty$ *and* $I_0 = max^*$*,* $I_i = (max - 1)^*$ *for* $i \in n^+$ *and* $I_\infty = \{max + 1\}$*. We write* $|R| = \sum_{i \in n^*_\infty} |A_i|$*.*

As we want to obtain a discrete domain of states, we cannot keep all the information about the (real) ages of the instances. In fact, we only need to retain the integer part of the age of the instances, and the order between the fractional parts of the instances because, despite time is continuous, the limits of the intervals are natural numbers. Moreover, we do not need to keep the concrete names of the instances and the concrete age of instances older than max, since this information is not useful from the point of view of control-state reachability. Therefore, we represent the information of each instance $a:(m,r)$ by the pair $(m,\lfloor r \rfloor)$ which consist of the multiset of places m and a natural number representing the integer part of its age $\lfloor r \rfloor$. Then, in order to represent each marking by a region, we split the pairs representing its instances into three multisets:

- The multiset M_0 of instances with integer age lower than max,
- the multiset M_∞ of instances older than max and
- a multiset M_m with the rest of instances.

Finally, we put the pairs representing instances in M_0 and M_∞ in A_0 and A_∞ respectively, and we keep in $A_1, \ldots A_n$ the pairs representing instances in M_m, ordered according to the fractional part of their ages. More formally:

Definition 3.2.6 (Region of a marking) *Let M be a marking. We define the region $R_M = A_0 * A^{x_1} * ... * A^{x_n} * A_\infty$ where:*

- $|R_M| = |M|$, $x_1, ..., x_n \in (0,1)$ *and* $i < j$ *iff* $x_i < x_j$,
- $A_0 = \{(m,r) \mid a{:}(m,r) \in M,\ r \in \max^*\}$,
- $A^x = \{(m,\lfloor r \rfloor) \mid a{:}(m,r) \in M, r < \max, frct(r) = x\}$,
- $A_\infty = \{(m, \max+1) \mid a{:}(m,r) \in M,\ r > \max\}$.

Let us illustrate the previous definition by an example:

Example 3.2.2 *The first part of Fig. 3.12 represents the marking $M = a:(pq,1.5),\ b:(p,2),\ c:(ppq,4.5),\ d:(q,2.3),\ e:(qq,0.3)$. Suppose that $max = 3$. Then, we split the instances in the three multisets depicted in the middle of the figure. Finally, we lose the information about the names of the instances and the fractional part of the ages. Moreover, we split the multiset M_m into the two multisets A_1, which consists of the instances b and c with fractional part of the age 0.3, and A_2, with the instance a which has fractional part of the age 0.5.*

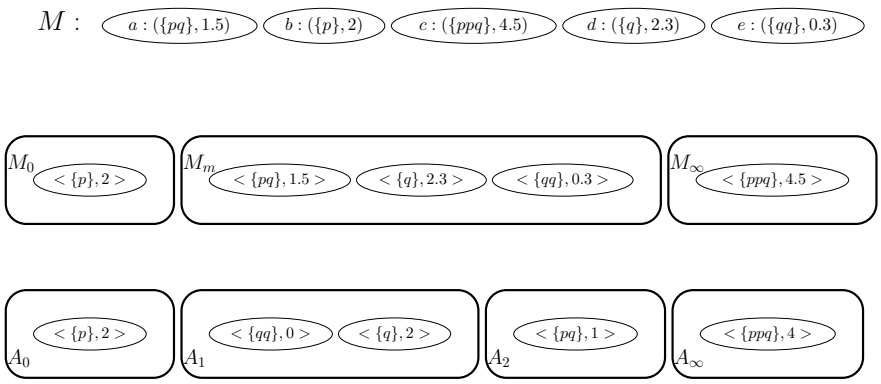


Figure 3.12: The region associated to a marking

Now we define the transition system over regions induced by a *ν-lsPN*. Let us fix some notations: Given $A \in (P^{\oplus} \times \mathcal{I})^{\oplus}$, we define $A^{<} = \{(m,r) \in A \mid r < max\}$, $A^{=} = \{(m, r+1) \in A \mid r = max\}$ and $A^{+1} = \{(m, r+1) \mid (m,r) \in A)\}$. Finally, for the next definitions we consider a region $R = A_0 * A_1 * \ldots A_n * A_\infty$.

Time elapsing: There are two ways in which time may elapse in regions. If $A_0 \neq \emptyset$, the region may evolve to $\emptyset * A_0^{<} * A_1 * \ldots * A_n * (A_\infty + A_0^{=})$, which corresponds to a small elapsing of time that makes all the instances in A_0 have a non-integer clock value, and so that the instances in A_n do not reach an integer value. Notice that instances in A_0 with clock max are added to A_∞. Otherwise, when $A_0 = \emptyset$, the region may evolve to $A_n^{+1} * A_1 * \ldots * A_{n-1} * A_\infty$, which represents an elapsing of time that causes the instances in A_n (those with a higher fractional part) to reach the next integer part. Formally:

Definition 3.2.7 (Time elapsing for regions) *Let $R = A_0 * A_1 * \ldots * A_n * A_\infty$ be a region. We write $R \overset{\delta}{\twoheadrightarrow} R'$, where*

$$R' = \begin{cases} \emptyset * A_0^{<} * A_1 * \ldots * A_n * (A_\infty + A_0^{=}) & \text{if } A_0 \neq \emptyset \\ A_n^{+1} * A_1 * \ldots * A_{n-1} * A_\infty & \text{otherwise} \end{cases}$$

Example 3.2.3 *Let us consider the region $R = A_0 * A_1 * A_2 * A_\infty$ depicted in Fig. 3.13 and suppose that $max = 3$. As $A_0 \neq \emptyset$, the elapsing we take is of the first type in the definition. Hence, we obtain a region with $A_0 = \emptyset$. From this new region, time elapses in the second way, and therefore the multiset A_3^{+1} becomes A_0.*

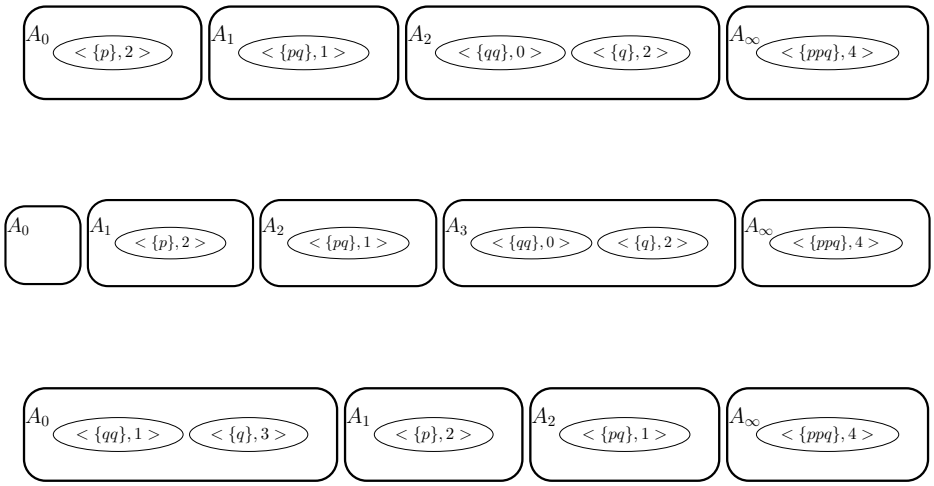


Figure 3.13: Time elapsing for regions

Firing of transitions: In order to define the firing of transitions for regions we first need to define $\emptyset$-expansions/contractions for them.

Definition 3.2.8 ($\emptyset$-expansion/contraction) *We say R' is an* $\emptyset$-expansion *of a region $R = A_0 * A_1 * \ldots * A_n * A_\infty$ (or R is the* $\emptyset$-contraction *of R') if R' is of the form $A'_0 * u_0 * A'_1 * u_1 * \ldots * A'_n * u_n * A'_\infty$ and for each $i \in \mathbb{N}$:*

- *$A'_i = A_i + B_i$ with $m = \emptyset$ for all $(m, r) \in B_i$,*
- *$u_i = B^i_1 * \ldots * B^i_{k_i}$ with $k_i \geq 0$ and $m = \emptyset$ for all $(m, r) \in B^i_j$.*

Intuitively, we obtain an $\emptyset$-expansion of a region R by adding to R pairs with empty marking in new multisets B^i_j which we insert in R, or in the multisets A_i in R.

Example 3.2.4 *Fig. 3.14 shows a region R and one of its $\emptyset$-expansions R'. Note that R' can be built from R by adding the two multisets B^0_1 and B^0_2 with pairs representing instances with empty markings and the pair $< \{\emptyset\}, 3 >$ to A_1.*

Intuitively, in order to fire a discrete transition from a region R, it must be enabled, which means that we must be able to assign to each variable $x \in Var(t)$ a different pair (m, r) in some multiset A_i of R, in such a way that $In_t(x) \subseteq m$ and the clock of the instance which represents (m, r) is in $Time^1_t(x)$. In order

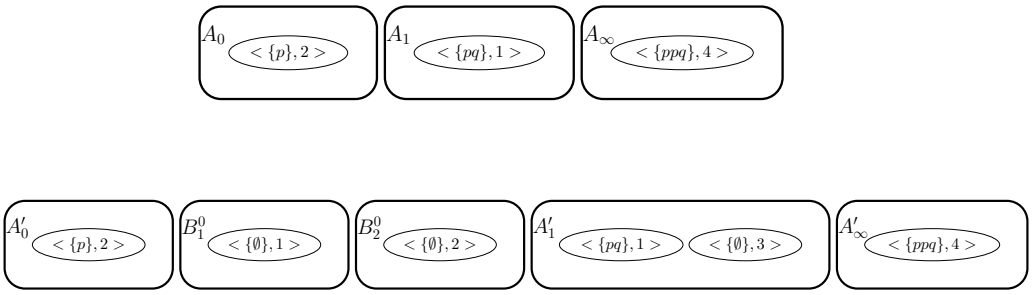


Figure 3.14: $\emptyset$-expansion of a region

to improve readability, we define a predicate *match* which encompasses these two requirements.

Definition 3.2.9 (Match predicate) *Given $a = (m, r) \in (P^{\oplus} \times \mathbb{R}_{\geq 0})$ and $\alpha = (m', I) \in P^{\oplus} \times \mathcal{I}$, the predicate $match(a, \alpha)$ holds if $m' \subseteq m$ and $r \in I$.*

If a transition is enabled then it can be fired, reaching a new region in which we update the markings of the pairs assigned to each variable according to In_t and Out_t, and we update the clocks of the pair according to $Time^2_t$. Moreover, we possibly need to remove some of the pairs we have chosen from some A_i they are in, and put them in a different A_j, according to one of the possible clocks they may represent. Finally, for each $\nu \in \Upsilon$, we put a new pair $(Out_t(\nu), r)$ in a proper (and maybe new) multiset of the region. In order to make the previous assignments, we define *modes* for regions. For any interval I, we call *left closure of I* the result of replacing the left delimiter of I by a closed one (for instance, the left closure of (a, b) is $[a, b)$). Moreover, in the rest of this chapter, given $t \in T$, we denote $l_t = |Var(t)|$.

Intuitively, in order to define the modes for the firings of discrete transitions in *ν-lsPN*, we are going to define three functions τ_1, τ_2 and τ_3 which assign to each variable of t a pair of the region to perform the firing, the (integer) clock value to which we update the clock of this pair and the new position in the region that the pair takes, respectively. For this purpose we enumerate the pairs in the multisets of the region (or more precisely, an $\emptyset$-expansion of the region), that is, for each A_i in the region, we denote $A_i = \{(m_{i1}, r_{i1}), \ldots, (m_{ik}, r_{ik})\}$, taking the first l_t pairs, the ones that are assigned to variables by τ_1. Moreover, for each $j \in k^+$ we denote $a_j = (m_{ij}, a_{ij})$.

Definition 3.2.10 *Let $t \in T$ be a transition and $A_0 * A_1 * \ldots * A_n * A_\infty$ be an $\emptyset$-expansion of a region R. A mode for t and R is any tuple $\tau = (\tau_1, \tau_2, \tau_3)$ where $\tau_1 : Var(t) \to (n^*_\infty \times l^+_t)$ is an injection, and $\tau_2 : Var(t) \to (\max+1)^*$ and $\tau_3 : Var(t) \to n^*_\infty \cup (n^* \times l^+_t)$ are mappings such that:*

- *For all $x \in Var(t)$, $\tau_2(x)$ is in the left closure of $Time^2_t(x)$,*
- *if $\tau_2(x) > \max$ then $\tau_3(x) = \infty$,*
- *if $Time^2_t(x) = (a, b]$ or $Time^2_t(x) = (a, b)$ and $\tau_2(x) = a$ then $\tau_3(x) \neq 0$.*

The first condition above ensures that the integers we choose to update the clocks are correct according to $Time^2_t(x)$. We need to consider the left closure of $Time^2_t(x)$ because in regions we do not represent the fractional parts of the clocks, and therefore, if $a + \epsilon < a + 1$ is the clock of a new created instance, which belongs to the interval $(a, b) = Time^2_t(x)$, then this clock is represented by a in the corresponding region, which belongs to the left closure of $Time^2_t(x)$, but not to $Time^2_t(x)$. The second condition makes sure that pairs with clocks older than max are stored in A_∞. Finally, the third condition ensures that the newly created pairs representing instances with a clock value of integer part a, that must not be exactly a, are not stored in A_0. Let us now define the firing of transitions for regions.

Definition 3.2.11 *We say a transition t is enabled at a region R if there is an $\emptyset$-expansion $A_0 * A_1 * \ldots * A_n * A_\infty$ of R and a mode $\tau = (\tau_1, \tau_2, \tau_3)$ such that for each $i \in n^*_\infty$, $\bar{A}_i = \{(m_{ij}, r_{ij}) \mid \tau_1(x) = (i, j)\} \subseteq A_i$ and for each $x \in Var(t)$ with $\tau_1(x) = (i, j)$:*

- *If $x \in \Upsilon$, then $m_{ij} = \emptyset$,*
- *If $i \in \{0, \infty\}$ then $match((m_{ij}, r_{ij}), (In_t(x), Time^1_t(x)))$ and*
- *$match((m_{ij}, r_{ij} + 0.5), (In_t(x), Time^1_t(x)))$ otherwise.*

Then, we define $m'_{ij} = (m_{ij} - In_t(x)) + Out_t(x)$ and take for all $k \in n^*$ and $b \in l^+_t$:

- *$B_k = A_k - \bar{A}_k$,*
- *$D_k = \{(m'_{ij}, r) \mid \exists x \in Var(t)$ with $\tau_1(x) = (i, j), \tau_2(x) = r, \tau_3(x) = k\}$,*
- *$C_k = B_k + D_k$ and*
- *$C_{kb} = \{(m'_{ij}, r) \mid \exists x \in Var(t)$ with $\tau_1(x) = (i, j), \tau_2(x) = r, \tau_3(x) = (k, b)\}$.*

Then, we write $R \overset{t}{\twoheadrightarrow} R'$*, where* R' *is the* $\emptyset$*-contraction of* $C_0 * C_{01} * \ldots * C_{0l_t} * C_1 * C_{11} * \ldots * C_{1l_t} * \ldots * C_n * C_{n1} * \ldots * C_{nl_t} * C_\infty$.

Intuitively, for each $i \in n^*_\infty$, $\bar{A}_i$ represents the multiset of instances selected by τ_1 from the multiset A_i which take part in the firing. We require $match((m_{ij}, r_{ij} + 0.5), (In_t(x), Time^1_t(x)))$ if $i \notin \{0, 1\}$ because in this case, the age of the instances represented in A_i is not natural, and therefore, if r is the age of such an instance, $r + 0.5$ must be in the corresponding interval in order to to be selected for the firing. Therefore, B_i is the multiset obtained after removing from A_i the instances corresponding to the preconditions. Finally, C_i represents B_i after adding the postconditions, and the C_{ij}s represent multisets of instances which we assign to clocks with fractional parts not appearing in R. Note that between two C_is we add l_t C_{ij}s. This is so to handle the case in which all the instances update their clocks to different values with a fractional part between the ones represented by C_i and C_{i+1}.

Example 3.2.5 *The top of Fig. 3.15 shows a region* R *for* $\max = 3$*, and the values of the functions* In, Out *and* $Time$ *for a transition* t*. Let us consider the mode* $\tau = (\tau_1, \tau_2, \tau_3)$ *where:*

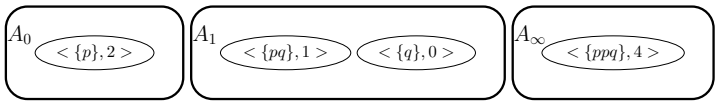
- $\tau_1(x) = (1, 1)$, $\tau_2(x) = 0$, $\tau_3(x) = 0$,
- $\tau_1(y) = (\infty, 1)$, $\tau_2(y) = 4$, $\tau_3(y) = \infty$,
- $\tau_1(\nu_1) = (0, 2)$, $\tau_2(\nu_1) = 2$, $\tau_3(\nu_1) = 0$,
- $\tau_1(\nu_2) = (0, 3)$, $\tau_2(\nu_2) = 2$, $\tau_3(\nu_2) = (1, 1)$.

Transition t *can be fired from* R *considering the mode* τ*. With the previous notations, we have:*

- $\bar{A}_0 = \emptyset$*, so* $B_0 = A_0$ *and* $C_0 = \{(\{p\}, 2), (\{p\}, 2)\}$,
- $\bar{A}_1 = \{(\{pq\}, 1)\}$*, so* $B_1 = A_1 - \{(\{pq\}, 1)\} = \{(\{q\}, 0)\} = C_1$,
- $\bar{A}_\infty = \{(\{ppq\}, 4)\}$*, so* $B_\infty = \emptyset$ *and* $C_\infty = \{(\{pqq\}, 4)\}$ *and*
- $C_{11} = \{(\{pq\}, 2)\}$.

Therefore, we obtain the region depicted in the bottom of Fig. 3.15.

Let $\overset{\Delta}{\twoheadrightarrow}$ be the reflexive and transitive closure of $\overset{\delta}{\twoheadrightarrow}$ and $\twoheadrightarrow = \overset{\Delta}{\twoheadrightarrow} \cup \bigcup_{t \in T} \overset{t}{\twoheadrightarrow}$.



$$In_t(x) = \{p, q\}, \quad Time_t(x) = ((0, 2], [0, \infty)), \quad Out_t(x) = \emptyset$$

$$In_t(y) = \{p\}, \quad Time_t(y) = ([3, \infty), [2, \infty), \quad Out_t(y) = \{q\}$$

$$In_t(\nu_1) = \emptyset, \quad Time_t(\nu_1) = ([0, \infty), [2, 2]), \quad Out_t(\nu_1) = \{p\}$$

$$In_t(\nu_2) = \emptyset, \quad Time_t(\nu_2) = ([0, \infty), [1, 3]), \quad Out_t(\nu_2) = \{p, q\}$$

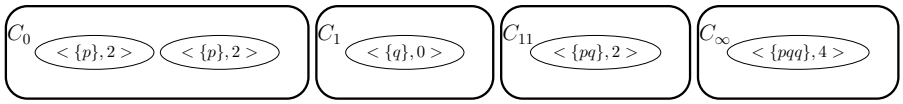


Figure 3.15: Firing of a discrete transition from a region

3.2.1.1 Correctness of the simulation

Now that we have defined the semantics of regions, we are ready to prove that this new model simulates the behavior of ν-*lsPN* properly.

Proposition 3.2.1 *The following relations between* $\rightarrow$ *and* $\twoheadrightarrow$ *hold:*

- *If* $M \rightarrow^* M'$ *then* $R_M \twoheadrightarrow^* R_{M'}$,
- *If* $R_M \twoheadrightarrow^* R'$ *then there is* M' *with* $R' = R_{M'}$ *and* $M \rightarrow^* M'$.

Before giving the proof of the previous proposition, let us fix some notations: We denote $\mathcal{C}(M) = \{r \mid a : (m, r) \in M\} \in {\mathbb{R}_{\geq 0}}^{\oplus}$. Moreover, for simplicity, we sometimes extend firings of transitions in regions, to firings in $\emptyset$-expansions of regions in the natural way.

In order to improve readability, we are going to split the proof of the previous proposition in the following nine lemmas. Their proofs are rather technical, so if the reader is not interested in these quite technical details, we recommend to skip them. However, there is a point which would be interesting to focus on, which is

how we manage the elapsings of time. In Def. 3.2.7 there are two ways in which time may elapse, depending on the region we consider. Both ways correspond to a small elapse, of less than a unit of time. However, we need to be able to represent longer elapsings. Prop. 3.2.1 can be proved because $\overset{\Delta}{\twoheadrightarrow}$ is defined as the reflexive and transitive closure of $\overset{\delta}{\twoheadrightarrow}$, and therefore, we can concatenate as many small elapsings as we need, in order to represent longer elapsings of time.

We consider the continuous and the discrete firings separately for both parts of the proposition. Moreover, we split the proof for the simulation of continuous firings in several lemmas, considering the different cases of continuous firing sepatatedly. The first lemma proves that a small delay from a marking with instances of a natural age lower than max can be simulated by a continuous firing for regions of the first type. By "small" delay we mean that no instance with a non integer age reaches or exceeds the next integer age.

Lemma 3.2.2 *Let M be a marking such that $\mathcal{C}(M) \cap \ \max^* \neq \emptyset$ and $\epsilon = \max\{frct(r) \mid r \in \mathcal{C}(M), r < \max\}$. If $0 < d < 1 - \epsilon$ then $R_M \overset{\delta}{\twoheadrightarrow} R_{M^{+d}}$. Moreover, $\mathcal{C}(M^{+d}) \cap \ max^* = \emptyset$.*

Proof: Suppose that $R_M = A_0 * A^{x_1} * ... * A^{x_n} * A_\infty$, where $0 < x_1 < ... < x_n < 1$ are the fractional parts of the ages of the instances younger than max in M. Then, as $x_n = \epsilon = \max\{frct(r) \mid r \in \mathcal{C}(M)\}$, $0 < d < 1 - \epsilon$ and $\mathcal{C}(M) \cap \mathbb{N} \neq \emptyset$, the fractional parts of the ages of the instances younger than max in M^{+d} are $d, x_1 + d, ..., x_n + d$ (with $x_n + d < 1$). For each $i \in n^+$, the instances and markings with fractional parts of its ages $x_i + d$ in M^{+d} are the same as the ones in M with fractional parts of its ages x_i. Moreover, the instances and markings with fractional parts of its ages d in M^{+d} are the instances with natural ages younger than max in M. Therefore, $R_{M^{+d}} = \emptyset * A^d * A^{x_1+d} * ... * A^{x_n+d} * A'_\infty$ as defined in Def. 3.2.6, where in A'_∞ are represented the instances in A_∞ and the instances in A_0 with age max, and in A^d are represented the instances in A_0 which are younger than max. By the first case of Def. 3.2.7, we have that $R_M \overset{\delta}{\twoheadrightarrow} R'$, where $R' = \emptyset * A_0^< * A_{x_1} * ... * A_{x_n} * (A_\infty + A_0^=) = \emptyset * A^d * A^{x_1+d} * ... * A^{x_n+d} * A'_\infty = R_{M^{+d}}$.

□

Next, we consider the case of a small delay from a marking without instances of a natural age lower than max, and prove that the region of the marking we obtain is the same as the region of the initial marking.

Lemma 3.2.3 *Let M be a marking such that $\mathcal{C}(M) \cap \max^* = \emptyset$ and $\epsilon = \max\{frct(r) \mid r \in \mathcal{C}(M), r < \max\}$. If $d < 1 - \epsilon$ then $R_M = R_{M^{+d}}$.*

Proof: Suppose that $R_M = \emptyset * A^{x_1} * ... * A^{x_n} * A_\infty$, where $0 < x_1 < ... < x_n < 1$ are the fractional parts of the ages of the instances younger than max in M. Then, as $x_n = \epsilon = \max\{frct(r) \mid r \in \mathcal{C}(M)\}$, $d < 1 - \epsilon$ and $\mathcal{C}(M) \cap \mathbb{N} = \emptyset$, the fractional parts of the ages of the instances younger than max in M^{+d} are $x_1 + d, ..., x_n + d < 1$, and moreover, for each $i \in n^+$, the instances and markings with fractional parts of its ages $x_i + d$ in M^{+d} are the same as the ones in M with fractional parts of its ages x_i. Therefore, by the definition of region, $R_{M^{+d}} = \emptyset * A^{x_1+d} * ... * A^{x_n+d} * A_\infty = \emptyset * A^{x_1} * ... * A^{x_n} * A_\infty = R_M$.

□

Now we consider the remaining case of time elapsing.

Lemma 3.2.4 *Let M be a marking such that $\mathcal{C}(M) \cap \max^* = \emptyset$ and $\epsilon = \max\{frct(r) \mid r \in \mathcal{C}(M), r < \max\}$. If $d = 1 - \epsilon$ then $R_M \overset{\delta}{\twoheadrightarrow} R_{M^{+d}}$. Moreover, if $\{r \in \mathcal{C}(M) \mid r < \max, r \notin \mathbb{N}\} \neq \emptyset$, then $\mathcal{C}(M^{+d}) \cap \max^* \neq \emptyset$.*

Proof: Suppose that $R_M = \emptyset * A^{x_1} * ... * A^{x_n} * A_\infty$, where $x_1, ..., x_n \in (0, 1)$ with $i < j$ iff $x_i < x_j$, are the fractional parts of the ages of the instances younger than max in M. Then, as $x_n = \epsilon = \max\{frct(r) \mid r \in \mathcal{C}(M)\}$, $d = 1 - \epsilon$ and $\mathcal{C}(M) \cap \max^* = \emptyset$, the fractional parts of the ages of the instances younger than max in M^{+d} are $0, x_1 + d, ..., x_{n-1} + d$. For each $i \in (n-1)^+$, the instances and markings with fractional parts of its ages $x_i + d$ in M^{+d} are the same as the ones in M with fractional parts of its ages x_i. Moreover, the instances with natural ages lower than max $+1$ in M^{+d} are the instances with ages with fractional part x_n in M. Therefore, $R_{M^{+d}} = A_0 * A^{x_1+d} * ... * A^{x_{n-1}+d} * A_\infty$ as defined in Def. 3.2.6. By the second case of Def. 3.2.7, we have that $R_M \overset{\delta}{\twoheadrightarrow} R'$, where $R' = A_n^{+1} * A_{x_1} * ... * A_{x_{n-1}} * A_\infty = A_0 * A^{x_1+d} * ... * A^{x_{n-1}+d} * A_\infty = R_{M^{+d}}$.

□

In the following lemma we use the previous ones in order to prove that regions can simulate delays of 1 unit of time.

Lemma 3.2.5 *Let M be a marking of a ν-lsPN. Then $R_M \overset{\Delta}{\twoheadrightarrow} R_{M^{+1}}$.*

Proof: First, let us suppose that $\mathcal{C}(M) \cap \max^* \neq \emptyset$. Let $R_M = A_0 * A_1 * ... * A_n * A_\infty$ and let x_i be the fractional part of the ages of instances represented in A_i for $i \in n^*$ (so $x_0 = 0$) and take $x_{n+1} = 1$. Then $x_0 < x_1 < ... < x_n < x_{n+1}$. We define $\epsilon_i = (x_{i+1} - x_i)/2$ for $i \in n^*$. Let $M_{n+1} = M$, $M'_{i+1} = M_{i+1}^{+\epsilon_i}$ for $i \in n^*$ and $M_{i-1} = (M'_i)^{+\epsilon_i}$ for $i \in (n+1)^+$. Then we have $M = M_{n+1} \overset{\epsilon_n}{\twoheadrightarrow} M'_{n+1} \overset{\epsilon_n}{\twoheadrightarrow} M_n \overset{\epsilon_{n-1}}{\twoheadrightarrow} ... \overset{\epsilon_1}{\twoheadrightarrow} M_1 \overset{\epsilon_0}{\twoheadrightarrow} M'_1 \overset{\epsilon_0}{\twoheadrightarrow} M_0$. Notice that $\sum_{i \in n^*} 2\epsilon_i = 1$, so that $M_0 = M^{+1}$. It also holds that $\mathcal{C}(M_i) \cap \mathbb{N} \neq \emptyset$ for all $i \in (n+1)^*$ and $\mathcal{C}(M'_i) \cap \mathbb{N} = \emptyset$ for all $i \in (n+1)^+$. Moreover, the maximum fractional part of the reals in M_{i+1} is $1 - 2\epsilon_i$ for $i \in n^+$, and that of M'_{i+1} is $1 - \epsilon_i$ for $i \in n^*$. Then M_{i+1} and ϵ_i are in the hypothesis of Lemma 3.2.2, and M'_{i+1} and ϵ_i in the ones of Lemma 3.2.4. Therefore, $R_{M_i} \overset{\delta}{\twoheadrightarrow} R_{M'_i}$ for $i \in (n+1)^+$ and $R_{M'_{i+1}} \overset{\delta}{\twoheadrightarrow} R_{M_i}$ for $i \in n^*$, so that $R_M = R_{M_{n+1}} \overset{\Delta}{\twoheadrightarrow} R_{M_0} = R_{M^{+1}}$.

Now, let us suppose that $\mathcal{C}(M) \cap \max^* = \emptyset$. If $\{r \in \mathcal{C}(M) \leq max\} = \emptyset$, then $R_M = R_{M^{+1}}$, so we are done (all the instances in the marking are represented in A_∞ with age $\max + 1$). Otherwise, there is an instance with clock $r < max$, and therefore, R_M is of the form $\emptyset * A_1 * ... * A_n * A_\infty$, with $n > 0$ and $A_n \neq \emptyset$. Therefore, by applying lemma 3.2.4, if $\epsilon = \max\{frct(r) \mid r \in \mathcal{C}(M), r < \max\}$ and $d = 1 - \epsilon$ then $R_M \overset{\delta}{\twoheadrightarrow} R_{M^{+d}}$, with $\mathcal{C}(M^{+d}) \cap \ \max^* \neq \emptyset$, reaching a region of the case which has been analyzed previously (with $A_0 \neq \emptyset$), except for the delay $+d$. Therefore, by a similar reasoning we can prove that $R_M \overset{\Delta}{\twoheadrightarrow} R_{M^{+1}}$.

□

Finally, we use the previous lemmas to prove that the relation $\overset{\Delta}{\twoheadrightarrow}$ for regions simulates any continuous firing in a ν-*lsPN*.

Lemma 3.2.6 *Given two markings M and M' and $d > 0$, if $M \overset{d}{\rightarrow} M'$ then $R_M \overset{\Delta}{\twoheadrightarrow} R_{M'}$.*

Proof: Let $d > 1$ (the other case is easier) and $M \overset{d}{\rightarrow} M'$. Then, we have that $M \overset{1}{\rightarrow} M^{+1} \overset{1}{\rightarrow} M^{+2} \overset{1}{\rightarrow} \ldots \overset{1}{\rightarrow} M^{+\lfloor d \rfloor} \xrightarrow{frct(d)} M'$. Because of Lemma 3.2.5 we know that $R_M \overset{\Delta}{\twoheadrightarrow} R_{M^{+1}} \overset{\Delta}{\twoheadrightarrow} R_{M^{+2}} \overset{\Delta}{\twoheadrightarrow} \ldots \overset{\Delta}{\twoheadrightarrow} R_{M^{+\lfloor d \rfloor}}$. Therefore, we only need to prove that $R_{M^{+\lfloor d \rfloor}} \overset{\Delta}{\twoheadrightarrow} R_{M'}$. As in Lemma 3.2.5, let $R_{M^{+\lfloor d \rfloor}} = A_0 * A_1 * ... * A_n * A_\infty$ and let x_i be the fractional part of the ages of instances represented in A_i for $i \in n^*$, $x_{n+1} = 1$, $\epsilon_i = (x_{i+1} - x_i)/2$ for $i \in n^*$, $M_{n+1} = M^{+\lfloor d \rfloor}$, $M'_i = M_i^{+\epsilon_i}$ and $M_{i-1} = (M'_i)^{+\epsilon_i}$. Now, we select k such that $1 - x_k \leq frct(d)$ and $1 - x_{k-1} > frct(d)$, and we define $y = x_k - (1 - frct(d))$ and $M_y = M_k^{+y}$. Note that $\sum_{i=k}^{n} 2 * \epsilon + y =$

$(1-x_k)+x_k-(1-frct(d)) = frct(d)$, so that $M_y = M^{+\lfloor d \rfloor + frct(d)} = M'$. Repeating a similar reasoning as for the proof of Lemma 3.2.5, we can conclude that $R_{M^{+\lfloor d \rfloor}} = R_{M_{n+1}} \overset{\Delta}{\twoheadrightarrow} R_{M_y} = R_{M'}$ and therefore $R_M \overset{\Delta}{\twoheadrightarrow} R_{M'}$.

□

Next, we consider the discrete firing of a transition from a marking, and we prove that it can be simulated by the firing of a transition from the corresponding region.

Lemma 3.2.7 *Given two markings M and M', if $M \xrightarrow{t} M'$ then $R_M \overset{t}{\twoheadrightarrow} R_{M'}$.*

Proof: Let us suppose that $M \xrightarrow{t} M'$ and without loss of generality, let us suppose that for each $t \in T$ and for each $\nu \in fVar(t)$, $In_t(\nu) = \emptyset$. Then, if $nfVar(t) = \{x_1, ..., x_{n_1}\}$ and $fVar(t) = \{x_{n_1+1}, ..., x_{n_2}\}$ then we have $M = a_1 : (m_1, r_1), ..., a_{n_1} : (m_{n_1}, r_{n_1}) + \overline{M}$ and for each $i \in n_1^+$, (1) $In_t(x_i) \subseteq m_i$ and (2) $r_i \in Time_t^1(x_i)$. Moreover, if $R_M = A_0 * A_1 * \ldots * A_n * A_\infty$, let $R_M^\emptyset = A_0 * A_1 * \ldots * A_n * A_{n+1} * A_\infty$ be the $\emptyset$-expansion of R_M, where A_{n+1} only contains n_2 empty instances. We denote $A_i = \{(m_{i1}^\emptyset, r_{i1}^\emptyset), \ldots, (m_{ik_i}^\emptyset, r_{ik_i}^\emptyset)\}$ for $i \in n_\infty^*$ and $L = max\{k_i \mid i \in n_\infty^*\}$.

Let us define a mode $\tau = (\tau_1, \tau_2, \tau_3)$ for firing t from R_M, obtaining a region R'_M. First of all, we define τ_1, and prove that t is enabled at R_M with a mode with τ_1 as its first component. Let $\Phi_1 : n_1^+ \to n_\infty^* \times L$, be the function which associates each $i \in n_1^+$ to the location of the pair which represents the instance a_i in R_M. We define the mode τ in such a way that the instances in $\overline{M}$ will not be selected by τ_1, and therefore, they will remain in the same $A_i s$ after firing t in the region. We define τ_1 such that $\tau_1(x_i) = \Phi_1(i)$ if $x_i \in nfVar(t)$ and $\tau_1(x_i) = (n+1, i)$ otherwise. Then, we have that, for each $x \in Var(t)$:

- If $x \in \Upsilon$ then $\tau_1(x) = (n+1, i)$, and $m_{n+1,i} = \emptyset$.
- If $\tau_1(x) = (i, j)$, then $In_t(x) \subseteq m_{ij}^\emptyset$, because of (1).
- $r_{ij} \in Time_t^1(x)$ if $i \in \{0, \infty\}$, and $r_{ij} + 0.5 \in Time_t^1(x)$, otherwise, because of (2).
- If $x \notin \Upsilon$, $\tau_1(x) = (i, j)$, and $i \in \{0, \infty\}$ then $match((m_{ij}^\emptyset, r_{ij}), (In_t(x), Time_t^1(x)))$, because of the two previous points.
- Analogously, if $x \notin \Upsilon$, $\tau_1(x) = (i, j)$, and $i \notin \{0, \infty\}$ then $match((m_{ij}^\emptyset, r_{ij} + 0.5), (In_t(x), Time_t^1(x)))$.

Therefore, t is enabled at R_M. Now, we prove that we can fire it in such a way that we reach $R_{M'}$. For that purpose, we first need to define the proper functions τ_2 and τ_3 of the mode for the firing.

Let us call $\Phi_2 : Var(t) \to \mathbb{R}_{\geq 0}$ the function which associates each $x_i \in Var(t)$ to the age to which the instance a_i of M is updated in the firing. Then, we define τ_2 such that, for each $x \in Var(t)$, $\tau_2(x) = \lfloor \Phi_2(x) \rfloor$ if $\Phi_2(x) \leq max$ and $\tau_2(x) = max + 1$ otherwise. With this definition, for each $x \in Var(t)$, $\tau_2(x)$ is in the left closure of $Time_t^2(x)$, as Def. 3.2.10 demands. Finally, we define τ_3 such that:

- If $\tau_2(x) = n + 1$ then $\tau_3(x) = \infty$ (and therefore, the second condition demanded by Def. 3.2.10 is met), else
- if $\tau_2(x) = \Phi_2(x)$, that is, if $\Phi_2(x) \in \mathbb{N}$ then $\tau_3(x) = 0$, else
- if $frct(\Phi_2(x))$ is a fractional part which has names represented in A_i then $\tau_3(x) = i$, else
- if A_i represents the names with the greatest fractional part f lower than $frct(\Phi_2(x))$, and $\Phi_2(x)$ has the j^{th} fractional part greater than f of $\{frct(\Phi_2(x_i)) \mid x_i \in Var(t)\}$, then $\tau_3(x) = (i, j)$.

The third condition of Def. 3.2.10 holds for the mode τ we have defined, because if $Time_t^2(x) = (a, b]$ or $Time_t^2(x) = (a, b)$ and $\tau_2(x) = a$, then $\tau_2(x) \neq \Phi_2(x)$ and therefore $\tau_3(x) \neq 0$.

Now, we prove that the region R that we reach by firing t from R_M with mode τ is $R_{M'}$. We do it by proving that A'_i is in $R_{M'}$ iff it is in R.

We analyze different cases:

- First, we consider A'_0 of $R_{M'}$.
  $A'_0 = \biguplus_{a \in Id}\{(m, r) \mid a : (m, r) \in \bar{M}, r \in \mathbb{N}, r \leq max\} + \biguplus_{k \in n_1^+}\{(m, r) \mid \Phi_1(k) = (i, j), m = (m_{ij}^{\emptyset} - In_t(x_k)) + Out_t(x_k), \Phi_2(x_k) = r\} + \biguplus_{k \in \{n_1+1,\ldots,n_2\}}\{(m, r) \mid m = Out_t(x_k), \Phi_2(x_k) = r, r \in \mathbb{N}, r \leq max\} = \{(m_{0j}^{\emptyset}, r_{0j}^{\emptyset}) \in A_0 \mid \nexists x \in Var(t) \text{ with } \tau_1(x) = (0, j)\} + \biguplus_{x \in nfVar(t) \mid \tau_3(x) = 0}\{(m, r) \mid \tau_1(x) = (i, j), m = (m_{ij}^{\emptyset} - In_t(x)) + Out_t(x), \tau_2(x) = r\} + \biguplus_{\nu \in fVar(t)}\{(m, r) \mid m = Out_t(\nu), \tau_2(\nu) = r, \tau_3(\nu) = 0\}$ which is the first multiset in R.
- Now, we consider A'_∞ of $R_{M'}$.
  $A'_\infty = \biguplus_{a \in Id}\{(m, max + 1) \mid a : (m, r) \in \bar{M}, r > max\} +$

$\biguplus_{k\in n_1^+}\{(m, max{+}1) \mid \Phi_1(k) = (i,j), m = (m_{ij}^{\emptyset} - In_t(x_k)) + Out_t(x_k), \Phi_2(x_k) = r, r > max\} + \biguplus_{k\in\{n_1+1,\ldots,n_2\}}\{(m, max+1) \mid m = Out_t(x_k), \Phi_2(x_k) = r, r > max\} = \{(m_{\infty j}^{\emptyset}, max+1) \in A_\infty \mid \nexists x \in Var(t)$ with $\tau_1(x) = (\infty, j)\} + \biguplus_{x\in nfVar(t)}\{(m, max+1) \mid \tau_1(x) = (i,j), m = (m_{ij}^{\emptyset} - In_t(x)) + Out_t(x), \tau_2(x) = max{+}1\} + \biguplus_{\nu\in fVar(t)}\{(m, max{+}1) \mid m = Out_t(\nu), \tau_2(\nu) = max+1\}$, which is the last set in R.

- Now, let us consider the case in which A is a set of $R_{M'}$ which represents instances in M with the fractional part of the age ρ. Moreover, let us suppose A_k is the set of R_M which represents instances with this fractional part of the age. Then $A = \biguplus_{a\in Id}\{(m,r) \mid a : (m,r') \in \bar{M}, r = \lfloor r' \rfloor, frct(r') = \rho, r' \leq max\} + \biguplus_{k'\in n_1^+}\{(m,r) \mid \Phi_1(k') = (i,j), m = (m_{ij}^{\emptyset} - In_t(x_{k'})) + Out_t(x_{k'}), \Phi_2(x_{k'}) = r', r = \lfloor r' \rfloor, frct(r') = \rho, r' \leq max\} + \biguplus_{k'\in\{n_1+1,\ldots,n_2\}}\{(m,r) \mid m = Out_t(x_{k'}), \Phi_2(x_{k'}) = r', r = \lfloor r' \rfloor, frct(r') = \rho, r' \leq max\} = \{(m_{kj}^{\emptyset}, r_{kj}^{\emptyset}) \in A_k \mid \nexists x \in Var(t)$ with $\tau_1(x) = (k,j)\} + \biguplus_{x\in nfVar(t)}\{(m,r) \mid \tau_1(x) = (i,j), m = (m_{ij}^{\emptyset} - In_t(x)) + Out_t(x), \tau_2(x) = r, \tau_3(x) = k\} + \biguplus_{\nu\in fVar(t)}\{(m,r) \mid m = Out_t(\nu), \tau_2(\nu) = r, \tau_3(\nu) = k\}$, which is in R.

- Finally, we consider the case in which A is a set of R'_M which represents instances with fractional part of the age ρ different to all the ones in M. Then, $A = \biguplus_{k\in n_1^+}\{(m,r) \mid \Phi_1(k) = (i,j), m = (m_{ij}^{\emptyset} - In_t(x_k)) + Out_t(x_k), \Phi_2(x_k) = r', r = \lfloor r' \rfloor, frct(r') = \rho, r' \leq max\} + \biguplus_{k\in\{n_1+1,\ldots,n_2\}}\{(m,r) \mid m = Out_t(x_k), \Phi_2(x_k) = r', r = \lfloor r' \rfloor, frct(r') = \rho, r' \leq max\} = \biguplus_{x\in nfVar(t)}\{(m,r) \mid \tau_1(x) = (i,j), m = (m_{ij}^{\emptyset} - In_t(x)) + Out_t(x), \tau_2(x) = r, \tau_3(x) = (k1, k2)$ where A_{k1} represents the names with the greatest fractional part f lower than $\Phi_2(x_k)$, and $\Phi_2(x)$ has the $k2^{th}$ fractional part greater than f of $\{frct(\Phi_2(x_i)) \mid x_i \in Var(t)\}\} + \biguplus_{\nu\in fVar(t)}\{(m,r) \mid m = Out_t(\nu), \tau_2(\nu) = r, \tau_3(\nu) = (k1, k2)$ where A_{k1} represents the names with the greatest fractional part f lower than $\Phi_2(x_k)$, and $\Phi_2(\nu)$ has the $k2^{th}$ fractional part greater than f of $\{frct(\Phi_2(x_i)) \mid x_i \in Var(t)\}\}$, which is in R.

We have considered all the multisets, both in $R_{M'}$ and in R. Finally, note that the order of the sets of R correspond to the order of the corresponding A'_i of R'_M. That is because we have defined τ, in such a way that we order the different sets depending on the fractional part of $r''_g s$ younger than max, setting the instances older than max in A'_∞, as in R'_M.

□

So far, we have proved that any firing from a marking in a *ν-lsPN* can be simulated by a firing from its corresponding region. Now, we prove that each firing from a region of a marking represents some firing in the original *ν-lsPN*. First, we focus on the discrete firings.

Lemma 3.2.8 *Given a marking M, if $R_M \overset{t}{\twoheadrightarrow} R'$ then there is M' with $R' = R_{M'}$ and $M \overset{t}{\rightarrow} M'$.*

Proof: Let us suppose that $R_M \overset{t}{\twoheadrightarrow} R'$ with mode $\tau = (\tau_1, \tau_2, \tau_3)$. We define a marking M' with $R' = R_{M'}$, and then we prove that $M \overset{t}{\rightarrow} M'$. Let us suppose $R_M = A_0 * A_1 * \ldots * A_n * A_\infty$ and $R' = A'_0 * A'_1 * \ldots * A'_{n'} * A'_\infty$. We define M' as the marking such that:

- For each $(m, r) \in A'_0$, $a{:}(m, r) \in M'$.
- For each $(m, max + 1) \in A'_\infty$, $a : (m, r) \in M'$, where r is the age of the corresponding instance $a{:}(m, r) \in M$ if $(m, max+1)$ is in A_∞, and $max+1$ otherwise.
- Analogously, we consider $(m, r) \in A'_i$, where A'_i is a multiset obtained in the firing of t from the multiset A of R_M which represents instances with age of fractional part $\rho(A'_i)$. Then, $a{:}(m, r') \in M'$, where $r' = r + \rho(A'_i)$.
- Finally, for each A'_i which is a new multiset obtained in the firing of t, we define some $\rho(A'_i) \in (0, 1)$ such that if $j < i < j'$ then $\rho(A'_j) < \rho(A'_i) < \rho(A'_{j'})$. Then, for each $(m, r) \in A'_i$, $a{:}(m, r') \in M'$, where $r' = r + \rho(A'_i)$.

Due to the way we have defined M, taking the pairs representing instances and building the corresponding instances, it is clear that $R' = R_{M'}$. Now, we prove that t is enabled at M and $M \overset{t}{\rightarrow} M'$. We know that t is enabled at R_M with mode τ, and therefore, for each $x \in nfVar(t)$ with $\tau_1(x) = (i, j)$:

- $In_t(x) \subseteq m_{ij}$ and
- $r_{ij} \in Time^1_t(x)$ if $i \in \{0, \infty\}$, and $r_{ij} + 0.5 \in Time^1_t(x)$.

Then, if $nfVar(t) = \{x_1, ..., x_n\}$, we can rename $M = a_1 : (m_1, r_1), ..., a_n : (m_n, r_n) + \overline{M}$, in such a way that for each $i \in n^+$, $In_t(x_i) \subseteq m_i$ and $r_i \in Time^1_t(x_i)$, where m_i is m_{jk} of R_M if $\tau_1(x_i) = (j, k)$. Therefore, t is enabled at M, so there is (at least) a marking M'' such that $M \overset{t}{\rightarrow} M''$. We prove that we can

obtain a marking M'' from the firing such that for each instance in M'', the same instance is in M', and therefore, as the number of instances of M' and M'' are the same (because of the definition of firing of transition for region), $M'' = M'$. Let $a{:}(m, r) \in M''$. We consider different cases:

- If $a{:}(m, r) \in \overline{M}$ then it is in M too. Let us suppose that $(m_{ij}, r_{ij}) \in A_i$ is the pair which represents this instance in R_M. Then, because of how we have renamed M, there is not $x \in Var$ with $\tau_1(x) = (i, j)$, and therefore $(m_{ij}, r_{ij}) \in A'_k$, where A'_k is the set of R' which represents instances that remain in a certain A_k after firing. Therefore, $a{:}(m, r) \in M'$, because of the third point in the definition of M'.

- Suppose $a{:}(m, r) \notin \overline{M}$. Then, $a{:}(m, r)$ is associated to some $x_i \in Var(t)$ in the firing, that is, there is x_i such that $a = a_i$ and $m = (m_i - In_t(x_i)) + Out_t(x_i)$ if $x_i \in nfVar(t)$ and $m = Out_t(x_i)$ if $x_i \in fVar(t)$. We analyze the first case, and suppose $\tau_3(x_i) \neq \infty$ (the other cases are analogous). If $\tau_1(x_i) = (j, k)$ then $(m_{jk}, r_{jk}) \in R_M$, where $m_{jk} = m_i$ and $r_{jk} = \lfloor r_i \rfloor$. Then, after firing t from R_M with mode τ, we have that $(m, \tau_2(x_i)) \in A'_{\tau_3(x)}$ (with the notations of the firings for R'). Therefore, because of how we have defined M', the instance $a{:}(m, r)$ is in M' (note that $r = \tau_2(x_i) + \rho(A'_{\tau_3(x)})$, where $\rho(A'_{\tau_3(x)})$ represents the fractional part of the ages of the instances represented in $A'_{\tau_3(x)}$.

Therefore, $M'' = M'$, so $M \xrightarrow{t} M'$.

□

Now, we focus on continuous firings. First, we prove that each "small" continuous firing $R_M \overset{\delta}{\twoheadrightarrow} R'$ from the region which represents a marking M, represents a certain small elapsing of time from M in a proper way.

Lemma 3.2.9 *Given a marking M, if $R_M \overset{\delta}{\twoheadrightarrow} R'$ then there is M' with $R' = R_{M'}$ and $M \xrightarrow{d} M'$ for some $d \in (0, 1)$.*

Proof: Let us suppose that $R_M \overset{\delta}{\twoheadrightarrow} R'$. Let us first consider the case in which $R_M = \emptyset * A^{x_1} * ... * A^{x_n} * A_\infty$, where x_i is the fractional part of the age of the instances represented in A^{x_i}. Then, $R' = A^{x_n+1} * A^{x_1} * ... * A^{x_{n-1}} * A_\infty$. Let $d = 1 - x_n$. Let M' be a marking such that $M \xrightarrow{d} M'$. We are going to prove that $R' = R_{M'}$. By the definition of region of a marking, $R_{M'} = A'_0 * A^{x'_1} * ... * A^{x'_{n-1}} * A'_\infty$ where:

- $A'_0 = \biguplus_{a\in Id}\{(m,r) \mid a:(m,r) \in M',\ r \in \max^*\} = \biguplus_{a\in Id}\{(m,r) \mid a{:}(m,r-d) \in M,\ r \in \max^*\} = A_n^{+1}$.
- $A^{x'_i} = \biguplus_{a\in Id}\{(m,\lfloor r\rfloor) \mid a:(m,r) \in M', r < \max, frct(r) = x'_i\} = \biguplus_{a\in Id}\{(m,\lfloor r\rfloor) \mid a{:}(m,r-d) \in M, r-d < \max, frct(r-d) = x_i\} = A^{x_i}$, for each $i \in (n-1)^+$.
- $A'_\infty = \biguplus_{a\in Id}\{(m,\max+1) \mid a:(m,r) \in M',\ r > \max\} = \biguplus_{a\in Id}\{(m,\max+1) \mid a{:}(m,r-d) \in M,\ r-d > \max\} = A_\infty$.

Now, we consider the case in which $R_M = A_0 * A^{x_1} * ... * A^{x_n} * A_\infty$ and $A_0 \neq \emptyset$. Then, $R' = \emptyset * A_0^{<} * A^{x_1} * ... * A^{x_n} * (A_\infty + A_0^{=})$. Let $0 < d < 1 - x_n$, and let M' be a marking such that $M \xrightarrow{d} M'$. Again, we are going to prove that $R' = R_{M'}$. By the definition of region of a marking, $R_{M'} = A'_0 * A^{x'_1} * ... * A^{x'_n} * A'_\infty$ where:

- $A'_0 = \biguplus_{a\in Id}\{(m,r) \mid a:(m,r) \in M',\ r \in \max^*\} = \biguplus_{a\in Id}\{(m,r) \mid a{:}(m,r-d) \in M',\ r \in \max^*\} = \emptyset$.
- $A^{x'_1} = \biguplus_{a\in Id}\{(m,\lfloor r\rfloor) \mid a:(m,r) \in M', r < \max, frct(r) = x'_1\} = \biguplus_{a\in Id}\{(m,\lfloor r\rfloor) \mid a{:}(m,r-d) \in M, r < \max, frct(r) = x'_1\} = A_0^{<}$.
- $A^{x'_i} = \biguplus_{a\in Id}\{(m,\lfloor r\rfloor) \mid a:(m,r) \in M', r < \max, frct(r) = x'_i\} = \biguplus_{a\in Id}\{(m,\lfloor r\rfloor) \mid a{:}(m,r-d) \in M, r-d < \max, frct(r-d) = x_i\} = A^{x_i}$, for each $i \in \{2,\ldots,n\}$.
- $A_\infty = \biguplus_{a\in Id}\{(m,\max+1) \mid a:(m,r) \in M',\ r > \max\} = \biguplus_{a\in Id}\{(m,\max+1) \mid a:(m,r) \in M,\ r-d > \max\} \cup \{(m,\max+1) \mid a:(m,r) \in M,\ r = \max\} = A_\infty + A_0^{=}$.

□

Finally, we prove that any continuous firing from a region of a marking M represents a continuous firing from M, that is, it reaches the region of a marking which can be reached from M after a certain delay.

Lemma 3.2.10 *Given a marking M, if $R_M \overset{\Delta}{\twoheadrightarrow} R'$ there is M' with $R' = R_{M'}$ and $M \xrightarrow{d} M'$ for some $d \in \mathbb{R}_{\geq 0}$.*

Proof: Suppose that $R_M \overset{\Delta}{\twoheadrightarrow} R'$. As $\overset{\Delta}{\twoheadrightarrow}$ is the reflexive and transitive closure of $\overset{\delta}{\twoheadrightarrow}$, we have that $R_M = R_0 \overset{\delta}{\twoheadrightarrow} R_1 \overset{\delta}{\twoheadrightarrow} \ldots \overset{\delta}{\twoheadrightarrow} R_k = R'$. We can prove by an inductive reasoning that for each $i \in n^+$, there exist M_i, d_i such that $R_i = R_{M_i}$ and $M_{i-1} \xrightarrow{d_i}$

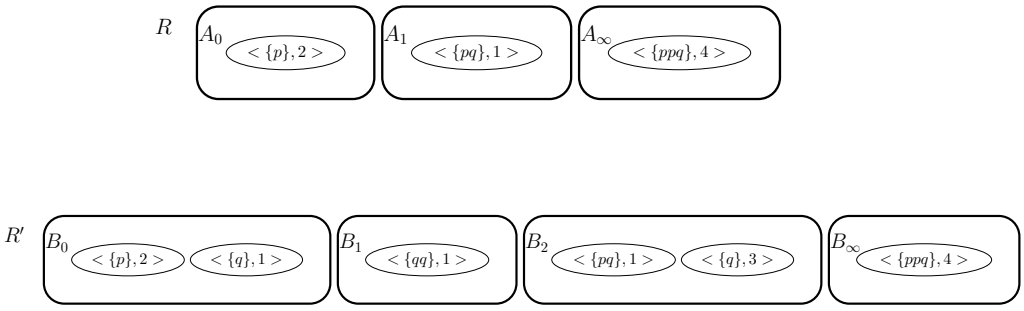


Figure 3.16: $R \sqsubseteq R'$

M_i, by applying the previous claim to each R_i. Therefore, we have that $M = M_0 \xrightarrow{d_1} M_1 \xrightarrow{d_2} \ldots \xrightarrow{d_k} M_k$ and $R' = R_k = R_{M_k}$. Therefore, if we take $d = \sum_{i \in k^+} d_i$, then we have that $M \xrightarrow{d} M_k$ and $R' = R_{M_k}$

□

As $\twoheadrightarrow = \overset{\Delta}{\twoheadrightarrow} \cup \bigcup_{t \in T} \overset{t}{\twoheadrightarrow}$, Prop. 3.2.1 easily follows from the previous lemmas. Hence, we have proved that the model we have defined simulates the behavior of *ν-lsPN*.

3.2.1.2 Coverability for regions is decidable

In order to reduce control-state reachability for *ν-lsPN* to some problem over regions, we first need to define an order over them.

Definition 3.2.12 (Order over regions) *We define* $(m, r) \leq (m', r')$ *iff* $m \subseteq m'$ *and* $r = r'$*. Then, we define* $A_0 * A_1 * \ldots * A_n * A_\infty \sqsubseteq B_0 * B_1 * \ldots * B_m * B_\infty$ *iff* $A_0 \leq^{\oplus} B_0$*,* $A_\infty \leq^{\oplus} B_\infty$ *and* $A_1 \ldots A_n \leq^{\oplus \circledast} B_1 \ldots B_m$*.*

Example 3.2.6 *Fig. 3.16 shows two regions* R *and* R'*. Note that* $A_0 \subseteq B_0$*,* $A_1 \subseteq B_2$ *and* $A_\infty \subseteq B_\infty$*. Hence,* $R \sqsubseteq R'$*.*

Notice that we are using the word order induced by the multiset order, and therefore $\sqsubseteq$ is a decidable wpo. Formally:

Proposition 3.2.11 $\sqsubseteq$ *is a decidable wpo.*

Proof: In the first place, $\sqsubseteq$ is trivially decidable. To prove that it is a wpo, let us remark that a region $R = A_0 * A_1 * ... * A_n * A_\infty$ can be seen as an element of $X = X^{\oplus}_{max^*} \times (X^{\oplus}_{(max-1)^*})^{\circledast} \times X^{\oplus}_{\{max+1\}}$, where for every $I \subseteq (max+1)^*$, $X_I = P^{\oplus} \times I$. Indeed, $A_0 \in X^{\oplus}_{max^*}$, $A_\infty \in X^{\oplus}_{\{max+1\}}$ and $u = A_1 * ... * A_n$ can be seen as a word over $X^{\oplus}_{(max-1)^*}$. Therefore, $\sqsubseteq$ is just the standard order in X, as defined in the preliminaries. Then, $\sqsubseteq$ is a wpo because it is built from wpos (finite sets with equality[2]) using operators that preserve well-orders (multisets, words and the product).

□

The order $\sqsubseteq$ we have defined induces a coverability problem in our transition system with regions as states. Let us recall that in this case the coverability problem is that of deciding if, given region R, there is a reachable region R' such that $R \leq R'$. Let us see that we can reduce control-state reachability for ν-*lsPN* to this problem.

Proposition 3.2.12 *Given $p \in P$ we can compute a set of regions $\mathcal{R}_p$ such that there is a reachable marking that marks p iff $\uparrow\mathcal{R}_p$ can be reached.*

Proof: Let $R_0^r = \{(\{p\}, r)\} * \emptyset$ for each $r \in \max^*_\infty$, $R_\infty = \emptyset * \{(\{p\}, \max+1)\}$ and $R^r = \emptyset * \{(\{p\}, r)\} * \emptyset$ for $r \in (\max-1)^*$. Let us see that $\mathcal{R}_p = \{R_0^r \mid r \in \max^*_\infty\} \cup \{R^r \mid r \in (\max-1)^*\} \cup \{R_\infty\}$ satisfies the thesis. First, let us assume that $M_0 \to^* M$ with $a : (m, r) \in M$ with $p \in m$. By Prop. 3.2.1 we have $R_{M_0} \twoheadrightarrow^* R_M = A_0 * A_1 * ... * A_n * A_\infty$. Let us distinguish cases for $r \in \mathbb{R}_{\geq 0}$.

- If $r \in \max^*$ then by Def. 3.2.6, $(m, r) \in A_0$ and $R_0^r \sqsubseteq R_M$.
- If $r > \max$ also by Def. 3.2.6 we have $(m, \max+1) \in A_\infty$, so that $R_\infty \sqsubseteq R_M$.
- If $r \leq \max$ and $r \notin \mathbb{N}$, we have $(m, \lfloor r \rfloor) \in A_i$ for some $i \in n^+$, so that $R^{\lfloor r \rfloor} \sqsubseteq R_M$.

In any of the previous cases, $R_M \in \uparrow\mathcal{R}_p$.

Conversely, let us assume that $R_{M_0} \twoheadrightarrow^* R$ with $R \in \uparrow\mathcal{R}_p$. By Prop. 3.2.1 there is M reachable such that $R = R_M$. Since $R_M \in \uparrow\mathcal{R}_p$ there is $R' \in \mathcal{R}_p$ such that $R' \sqsubseteq R$. Let us denote $R = A_0 * A_1 * ... * A_n * A_\infty$. Analogously as in the converse implication, and using again Def. 3.2.6, we distinguish cases over R':

[2]Multiset containment is the multiset order induced by the equality.

- Suppose that $R' = R'_0$, for some $r \in \max^*_\infty$. As $R' \sqsubseteq R$, $\{(\{p\}, r)\} \leq^\oplus A_0$. Therefore, there is $(m, r) \in A_0$ such that m marks p. Then, by Def. 3.2.6, there is $a \in Id$ such that $a{:}(m, r) \in M$ and M marks p.

- Now, suppose that $R' = R_\infty$. As $R' \sqsubseteq R$, $\{(\{p\}, \max+1)\} \leq^\oplus A_\infty$. Therefore, there is $(m, \max+1) \in A_\infty$ such that m marks p. By Def. 3.2.6, there are $a \in Id$ and $r > \max$ such that $a{:}(m, r) \in M$ and M marks p.

- Finally, let $R' = R^r$ for some $r \in \max^*_\infty$. As $R' \sqsubseteq R$, there is $i \in \mathbb{N}$ such that $\{(\{p\}, r)\} \leq^\oplus A_i$. Then, there is $(m, r) \in A_i$ such that m marks p. Hence, by Def. 3.2.6, there are $a \in Id$ and $\epsilon \in (0, 1)$ such that $a{:}(m, r+\epsilon) \in M$ and M marks p.

In any of the previous cases, we obtain that $a : (m, r) \in M$ for some m with $p \in m$, so we conclude.

□

Therefore, in order to prove the decidability of control-state reachability for ν-*lsPN*, we just need to prove that coverability is decidable for the transition system over regions we have defined. Let us recall from the preliminaries that coverability is decidable for WSTS with effective *Pred*-basis. Hence, since we have proved that $\sqsubseteq$ is a decidable wpo, it only remains to prove that the transition relation is compatible with the order, and that the effective *Pred*-basis property holds, in order to prove that coverability for the transition system over regions induced by a ν-*lsPN* is decidable. We start by showing that the transition relation is compatible with $\sqsubseteq$ by proving that $\overset{\Delta}{\twoheadrightarrow}$ and $\overset{t}{\twoheadrightarrow}$ are compatible with $\sqsubseteq$, and considering the union of both relations.

We start by proving the compatibility for a small delay δ.

Lemma 3.2.13 *If $R_1 \overset{\delta}{\twoheadrightarrow} R_2$ and $R_1 \sqsubseteq R'_1$ then there is R'_2 such that $R'_1 \sqsubseteq R'_2$ and $R'_1 \overset{\Delta}{\twoheadrightarrow} R'_2$.*

Proof: Let $R_1 = A_0 * A_1 * ... * A_n * A_\infty$ and $R'_1 = B_0 * u_0 * B_1 * ... * B_n * u_n * B_\infty$ with $A_i \leq^\oplus B_i$.

First, we assume that $A_0 \neq \emptyset$, so that $B_0 \neq \emptyset$. By Def. 3.2.7 we have $R_2 = \emptyset * A_0^< * A_1 * ... * A_n * (A_\infty + A_0^=)$ and since $B_0 \neq \emptyset$ we also have $R'_1 \overset{\delta}{\twoheadrightarrow} R'_2 = \emptyset * B_0^< * u_0 * B_1 * ... * B_n * u_n * (B_\infty + B_0^=)$. Since $A_0 \leq B_0$ we also have $A_0^< \leq^\oplus B_0^<$, $A_0^= \leq^\oplus B_0^=$ and thus $(A_\infty + A_0^=) \leq^\oplus (B_\infty + B_0^=)$. Then $R'_1 \sqsubseteq R'_2$.

Now, let us assume that $A_0 = \emptyset$, so that $R_2 = A_n^{+1} * A_1 * ... * A_{n-1} * A_\infty$. We also assume that $B_0 \neq \emptyset$ (the other case is slightly more simple). If $u_n = C_1 * ... * C_k$ then $R_1' \overset{\Delta}{\twoheadrightarrow}{}^* R_2' = B_n^{+1} * (C_1^{+1})^< * ... * (C_k^{+1})^< * B_0^< * u_0 * B_1 * u_1 * ... * u_{n-1} * (B_\infty + B_0^= + (C_1^{+1})^= + ... + (C_k^{+1})^=)$ in $2k+2$ steps. As $A_n^{+1} \leq^\oplus B_n^{+1}$, $A_\infty \leq^\oplus B_\infty + B_0^= + (C_1^{+1})^= + ... + (C_k^{+1})^=$ and $A_i \leq^\oplus B_i$, we have $R_2 \sqsubseteq R_2'$.

□

Since $\overset{\Delta}{\twoheadrightarrow}$ is the reflexive and transitive closure of $\overset{\delta}{\twoheadrightarrow}$, we obtain the compatibility for Δ as a corollary of the previous lemma.

Corollary 3.2.14 *If $R_1 \overset{\Delta}{\twoheadrightarrow} R_2$ and $R_1 \sqsubseteq R_1'$ then there is R_2' such that $R_1' \sqsubseteq R_2'$ and $R_1' \overset{\Delta}{\twoheadrightarrow} R_2'$.*

In order to prove compatibility for discrete transitions, we first need to prove a lemma concerning the $\emptyset$-expansions of the regions we consider.

Lemma 3.2.15 *Let R and R' be two regions such that $R \sqsubseteq R'$. If $A_0 * A_1 * ... * A_n * A_\infty$ is an $\emptyset$-expansion of R, then there is an $\emptyset$-expansion of R' of the form $B_0 * u_0 * B_1 * u_1 * ... * u_{n-1} * B_n * u_n * B_\infty$ such that for all $i \in n_\infty^*$:*

- $A_i \leq^\oplus B_i$,
- $(\emptyset, r) \in A_i$ *iff* $(\emptyset, r) \in B_i$.

Proof: Indeed, the $\emptyset$-expansion of R' in the lemma can be obtained by adding to R' the same empty instances added to R in order to obtain $A_0 * A_1 * ... * A_n * A_\infty$.

□

We are ready to tackle the compatibility for discrete transitions.

Lemma 3.2.16 *If $R_1 \overset{t}{\twoheadrightarrow} R_1'$ and $R_1 \sqsubseteq R_2$ then there is R_2' such that $R_2 \sqsubseteq R_2'$ and $R_2 \overset{t}{\twoheadrightarrow} R_2'$.*

Proof: Assume $R_1 \overset{t}{\twoheadrightarrow} R_1'$ with mode τ . Let $A_0 * A_1 * \ldots * A_n * A_\infty$ be the $\emptyset$-expansion of R_1 in the firing of t. By the previous lemma, there is an $\emptyset$-expansion of R_2 of the form $A_0^2 * u_0 * A_1^2 * u_1 * ... * u_{n-1} * A_n^2 * u_n * A_\infty^2$ such that $A_i \leq^\oplus A_i^2$ and $(\emptyset, r) \in A_i$ iff $(\emptyset, r) \in A_i^2$ for all i. Without loss of generality, we can suppose that for each $i \in n_\infty^*$, if $A_i = \{(m_1, r_1), \ldots, (m_{n'}, r_{n'})\}$ and $A_i^2 = \{(m_1^2, r_1^2), \ldots, (m_{n''}^2, r_{n''}^2)\}$ then, for each $j \in n'^+$, $m_{ij} \subseteq m_{ij}^2$ and $r_{ij} = r_{ij}^2$. Let us

see that we can fire t from R_2 with mode τ, obtaining R_2' greater than R_1'. Note that mode τ for R_2 is an abuse of notation, since we are forgetting about the u_is. However, we prefer to keep this notation in order to avoid renamings. First, we prove that t is enabled at R_2 with mode τ. Let $x \in Var(t)$ with $\tau_1(x) = (i, j)$.

- If $x \in \Upsilon$, then $m_{ij} = \emptyset$, and therefore, $m^2_{ij} = \emptyset$ (because of how we have defined the $\emptyset$-expansion of R_2).
- $In_t(x) \subseteq m_{ij} \subseteq m^2_{ij}$.
- As $r_{ij} = r^2_{ij}$, the conditions for r^2_{ij} hold trivially.

Therefore, t is enabled at R_2. Let us see that $R_2 \overset{t}{\twoheadrightarrow} R_2' \leq R_1'$. Suppose $R_1' = C_0 * C_{01} * \ldots * C_1 * \ldots * C_{n'} * C_{n'1} * \ldots * C_\infty$ and $R_2' = C^2_0 * C^2_{01} * \ldots * u_1 * \ldots * C^2_1 * \ldots * C^2_n * C^2_{n1} * \ldots * u_n * \ldots * C^2_\infty$, as in the definition of firings of transitions for regions. Again, this is an abuse of notation, because we are forgetting about the us. This could be fixed by defining another τ_3', by simply doing a renumeration. However, for the ease of understanding, we prefer to keep using τ_3 and only consider the C^2_is (in fact, the us do not take part in the firing). We prove that for each index i there is $C^2_{i'}$ with $C_i \leq^{\oplus} C^2_{i'}$. Let i be one of the indices in R_1'. We prove that for each $(m'_{ij}, r'_{ij}) \in C_i$, $(m^{2'}_{ij}, r^{2'}_{ij})$ is such that $m'_{ij} \subseteq m^{2'}_{ij}$ and $r'_{ij} = r^{2'}_{ij}$. We consider two cases:

- Suppose that there is not x with $\tau_3(x) = (i, j)$. Then, $(m'_{ij}, r'_{ij}) \in A_i$, and therefore there is $(m^{2'}_{ij}, r^{2'}_{ij}) \in A^2_i$, with $m'_{ij} \subseteq m^{2'}_{ij}$ and $r^{2'}_{ij} = r'_{ij}$. Moreover, as there is not x with $\tau_3(x) = (i, j)$, $(m^{2'}_{ij}, r^{2'}_{ij}) \in C^2_i$.
- Suppose that there is x with $\tau_3(x) = (i, j)$. Then, if $\tau_1(x) = (k, l)$ then $m'_{ij} = (m_{kl} - In_t(x)) + Out_t(x) \subseteq (m^2_{kl} - In_t(x)) + Out_t(x) = m^{2'}_{ij}$. Moreover, $r'_{ij} = \tau_2(x) = r^{2'}_{ij}$.

□

Finally, we obtain compatibility as a corollary of Lemma 3.2.14 and Lemma 3.2.16.

Corollary 3.2.17 $\twoheadrightarrow$ is compatible with $\sqsubseteq$.

We have proved that the transition systems over regions endowed with the order we have defined are Well Structured Transition Systems. Now, we focus on proving that the effective $Pred$-basis property holds for them. For that purpose, we need to prove that we are able to compute $\min(\uparrow Pre(\uparrow R))$ for any region R. We do it by defining a function $\overline{Pre}$ to compute the predecessors of a region. We split Pre into $Pre_\Delta(R) = \{R' \mid R' \overset{\Delta}{\twoheadrightarrow} R\}$ and $Pre_t(R) = \{R' \mid R' \overset{t}{\twoheadrightarrow} R\}$, and we define $\overline{Pre}_\Delta$ and $\overline{Pre}_t$ for each $t \in T$, so that $Pre_\Delta(\uparrow R) = \uparrow \overline{Pre}_\Delta(R)$ and $Pre_t(\uparrow R) = \uparrow \overline{Pre}_t(R)$. The details of the proof are rather technical, and hence, we prefer not to include them in this chapter, and to put them instead in Appendix A.

Proposition 3.2.18 *$\twoheadrightarrow$ has effective $Pred$-basis.*

Therefore, we have proved that the transition system induced by the regions of a *ν-lsPN* is a Well Structured Transition System, with effective $Pred$-basis. Hence, coverability is decidable for it. Finally, we are able to obtain the decidability of control-state reachability for *ν-lsPN* as a corollary of the previous result and Prop. 3.2.12.

Corollary 3.2.19 *Control-state reachability is decidable for ν-lsPN.*

3.3 Expressiveness

In the previous sections we have studied several extensions of ν-Petri nets, proving that if we provide each instance with a clock in the way of Timed Petri nets, the model we obtain (*ν-lsPN*) has decidable control-state reachability. More precisely, working with regions, we showed that this model belongs to the class of WSTS, for which coverability is decidable. Moreover, we have proved that control-state reachability is undecidable for any model in which we associate more than one clock to each instance in the way of Timed Petri nets.

Several works [40, 4, 16, 32] study the languages generated by the labelled transition systems of extensions of Petri Nets, by associating a label with each transition. Several acceptance conditions, like reachability, coverability or no condition, may be considered. These languages are commonly used to compare the expressiveness of different models.

In [40] coverability languages (those obtained with coverability as acceptance condition) are proposed as a measure to compare the expressiveness of WSTS. In [4, 40, 32, 16] Petri nets (PN), Petri nets with transfers and resets (AWN),

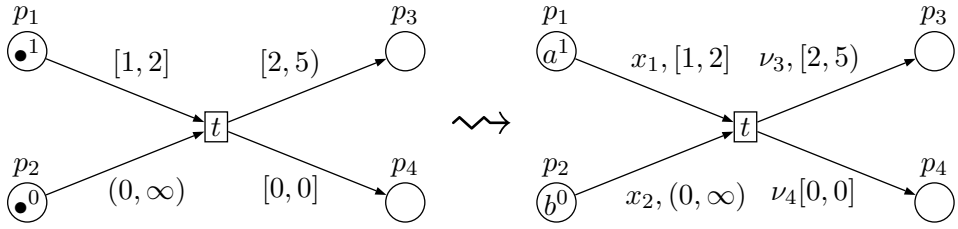


Figure 3.17: Illustrating Prop. 3.3.1

ν-*PN* and Data Nets (DN), an extension of ν-*PN* with ordered data, are compared, proving the following strict relations:

$$PN \prec AWN \prec \nu\text{-}PN \prec DN \approx TdPN$$

In this section, we compare the expressiveness of ν-*lsPN* with the expressiveness of other well-structured models by comparing the coverability languages they accept. First, we study bounded ν-*lsPN*, and we prove that they accept the same coverability languages as *TdPN*, and then we consider the general case. In order to analyze the expressiveness of ν-*lsPN*we partition that class into $\bigcup_{k\geq 0} \nu\text{-}lsPN_k$, where ν-$lsPN_k$ denotes the class of ν-*lsPN* with at most k unbounded places. Alternatively, we could consider the class with exactly k unbounded places, though we claim these two classes are equivalent with respect to coverability languages. A place p is *bounded* if there is some $b \in \mathbb{N}$ such that every instance $a : (m, r)$ satisfies $m(p) \leq b$ in every reachable marking. This is actually an undecidable problem [80], though this is not important for the study of expressiveness. If a ν-*lsPN* has P as set of unbounded places and m places bounded by b, we can represent each instance as an element of $Q \times P^{\oplus}$ with $Q = \{0, \ldots, b\}^m$.

3.3.1 Bounded ν-*lsPN*

First, we see that ν-$lsPN_0$, that is, bounded ν-*lsPN*, is at least as expressive as *TdPN*. For that purpose, given a *TdPN* we are going to build a ν-$lsPN_0$ which accepts the same coverability language.

Proposition 3.3.1 *TdPN* $\preceq \nu$-$lsPN_0$

Proof: Given a $TdPN N = \langle P, T, F, H\rangle$ we build a ν-$lsPN_0$ $N' = \langle P', T', In, Out, Time\rangle$ such that $\mathcal{L}(N) = \mathcal{L}(N')$. The net N' has the same sets of

places and transitions (with the same labels) as N, respectively, that is, $P' = P$ and $T' = T$. We simulate a token in p with age r by an instance with a single token in p, and with age r. Each transition is simulated by a transition (with the same label, hence accepting the same language) that removes instances/tokens with clocks with the proper values and creates fresh instances, again with clocks with the proper values.

Therefore, for each input arc of a transition t from place p_i of N, labelled by an interval I_j, we add to N' an arc labelled by (x_{ij}, I_j), as depicted in Fig. 3.17. Analogously, for each output place p_i of a transition t of N labelled by an interval I'_j, we add to N' an arc labelled by (ν_{ij}, I'_j) (to represent the creation of new tokens, we create new instances, in order to guarantee the net is bounded). Formally, for each place p_i and transition t of N, with $F(p_i, t) = \{I_1, \ldots, I_n\}$, for each $j \in n^+$ we have $In_t(x_{ij}) = \{p_i\}$, $Out_t(x_{ij}) = \emptyset$ and $Time_t(x_{ij}) = I_j$. Moreover, for each place p_i and transition t of N, with $H(p_i, t) = \{I_1, \ldots, I_n\}$, for each $j \in n^+$ we have $Out_t(\nu_{ij}) = \{p_i\}$ and $Time_t(\nu_{ij}) = I_j$.

If the initial marking of the *TdPN* is $\{(p_1, r_1), \ldots, (p_n, r_n)\}$ we consider $m_0 = a_1{:}(p_1, r_1), \ldots, a_n{:}(p_n, r_n)$ (for arbitrary $a_1, \ldots, a_n$) as initial marking of the ν-*lsPN* (and analogously for the final marking). Each instance in N' has exactly one token in any marking of any run beginning from m_0, so that $N' \in \nu\text{-}lsPN_0$. It is easy to see that every run of N' can be simulated by the same run in N', except for the fact that the tokens of N' are all coloured by different names.

□

Now we prove the converse of the previous result by performing a standard counting abstraction over the markings.

Proposition 3.3.2 $\nu\text{-}lsPN_0 \preceq TdPN$.

Proof: Let $N \in \nu\text{-}lsPN_0$. As N is bounded, we can define a finite set of control places $Q = \{q \in P^{\oplus} \mid$ there are a reachable marking $M, r \in \mathbb{R}_{\geq 0}$ and $a \in Id$ such that $M = a{:}(q, r) + \overline{M}$ for some $\overline{M}\}$. Then, each instance in each reachable marking can be represented by a token in control-state $q \in Q$ carrying the value of its clock $r \in \mathbb{R}_{\geq 0}$. We perform a standard counting abstraction (see Fig. 3.18): we build a *TdPN* with Q as set of places, so that each token (with a clock value) in a place q represents an instance in state q (with the same clock value). Then, a transition $t \in T$ is simulated by consuming a token from q (with a legal clock value) for each instance that has to be in state q in order to fire

t in N (and analogously for postconditions). That is, for each transition t with $nfVar(t) = \{x_1, \ldots, x_n\}$ and $fVar(t) = \{\nu_1, \ldots, \nu_m\}$ and for each pair (Q_1, Q_2) $Q_1 = \{q_1^1, \ldots, q_n^1\}, Q_2 = \{q_1^2, \ldots, q_n^2, q_{n+1}^2, \ldots, q_{n+m}^2\} \subseteq Q$, such that:

- for each $i \in n^+$, $q_i^1 \in Q$,
- for each $i \in n + m^+$, $q_i^2 \in Q$,
- for each $i \in n^+$ $In_t(x_i) \subseteq q_i^1$,
- for each $i \in n^+$ $q_i^2 = (q_i^1 - In_t(x_i)) + Out_t(x_i)$ (q_i^2 may be $\emptyset$),
- for each $i \in m^+$ $q_{n+i}^2 = Out_t(\nu_i)$,

we add to N' a transition $t_{Q_1Q_2}$ such that:

- for each $i \in n^+$, $F(q_i^1, t_{Q_1Q_2}) = \{Time_t^1(x_i)\}$,
- for each $i \in n^+$, $H(q_i^2, t_{Q_1Q_2}) = \{Time_t^2(x_i)\}$ and
- for each $i \in m^+$, $H(q_{n+i}^2, t_{Q_1Q_2}) = \{Time_t^2(\nu_i)\}$.

The concrete names of the tokens in ν-*lsPN* are abstract, meaning that they only need to be remembered in order to compare tokens of the same or different instances. As we have one token marking a control place representing each instance of N in our simulation, this ensures that we do not merge tokens representing different instances. Moreover, the constraints over transitions are set just as in the original net. Hence, for each pair of reachable markings M_1 and M_2 of N, such that $M_1 \xrightarrow{t} M_2$, we have that $M_1 = a_1{:}(q_1^1, r_1^1), \ldots, a_i{:}(q_i^1, r_i^1) + \overline{M}$, $M_2 = b_1{:}(q_1^2, r_1^2), \ldots, b_{i+j}{:}(q_{i+j}^2, r_{i+j}^2) + \overline{M}$ and if $Q1 = \{q_1^1, \ldots, q_i^1\}$ and $Q2 = \{q_1^2, \ldots, q_j^2\}$, then we have the corresponding firing in N': $\{(q_1^1, r_1^1), \ldots, (q_i^1, r_i^1)\} + \overline{M}' \xrightarrow{t_{Q_1Q_2}} \{(q_1^2, r_1^2), \ldots, (q_j^2, r_j^2)\} + \overline{M}'$, where $\overline{M}'$ is the marking of N' representing $\overline{M}$.

□

As a corollary, we obtain that *TdPN* and ν-$lsPN_0$ admit the same coverability languages.

Corollary 3.3.3 $TdPN \simeq \nu\text{-}lsPN_0$

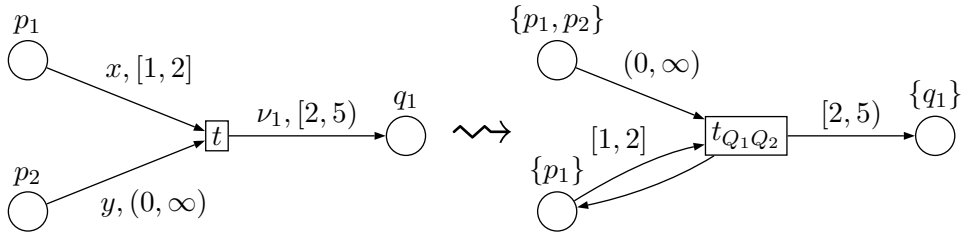


Figure 3.18: Illustrating Prop. 3.3.2, when the initial marking of the ν-*lsPN* in the left-side is $M = a : (\{p_1, p_2\}, 1), b : (\{p_1\}, 2), c : (\{q_1\}, 0)$. We consider $Q_1 = \{\{p_1, p_2\}, \{p_1\}\}$ and $Q_2 = \{\{p_1\}, \{q_1\}\}$.

3.3.2 Expressiveness of general ν-*lsPN*

In the previous section we have proved that ν-$lsPN_0$ and *TdPN* have the same expressiveness concerning the coverability languages they admit. Now we prove that ν-*lsPN* $\not\preceq$ *TdPN*, that is, that the language of some ν-*lsPN* cannot be obtained as the language of any *TdPN*. We do it indirectly, by proving that ν-$lsPN_k \prec \nu$-$lsPN_{k+1}$, which by the previous corollary in particular implies that *TdPN* $\prec$ ν-$lsPN_1$. Clearly, ν-$lsPN_k \preceq \nu$-$lsPN_{k+1}$ holds. Let us prove that ν-$lsPN_{k+1} \not\preceq \nu$-$lsPN_k$. In order to give this proof, we are going to use a framework previously defined by Bonnet et al.

Framework: In [16] a framework for the strict comparison of WSTS is developed, in order to compare the relative expressiveness of classes of WSTS based on their state spaces. More precisely, the framework is mainly based in two concepts: *reflexions* and *witness languages*.

Definition 3.3.1 (Reflexion) *Let $(X, \leq_X)$ and $(Y, \leq_Y)$ be two well partial ordered sets. A mapping $\varphi : X \to Y$ is a reflection if $\varphi(x) \leq_Y \varphi(x')$ implies $x \leq_X x'$ for all $x, x' \in X$.*

We write $X \sqsubseteq_{refl} Y$ if there is a reflection from X to Y. A reflection is an *isomorphism* if it is bijective and $x \leq_X x'$ implies $\varphi(x) \leq_Y \varphi(x')$ (note that the symmetric property of monotonicity is the reflexion). We extend the relation $\sqsubseteq_{refl}$ to classes of wpo by $\mathbf{X} \sqsubseteq_{refl} \mathbf{X}'$ if for any $X \in \mathbf{X}$ there is $X' \in \mathbf{X}'$ such that $X \sqsubseteq_{refl} X'$.

Given an alphabet $\Sigma = \{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$, we consider a disjoint copy $\overline{\Sigma} = \{\overline{\mathbf{a}_1}, \ldots, \overline{\mathbf{a}_k}\}$.

This notation is extended to words by $\overline{\omega} = \overline{\mathbf{a}_1} \dots \overline{\mathbf{a}_k}$ if $\omega = \mathbf{a}_1 \dots \mathbf{a}_k \in \Sigma^{\circledast}$ and analogously, if L is a language, we define $\overline{L} = \{\overline{\omega} \mid \omega \in L\}$.

Definition 3.3.2 *Let Σ be a finite alphabet. A* Σ-representation *of a wpo X is any surjective partial function $\gamma : \Sigma^{\circledast} \to X$. For a Σ-representation γ of X, we define $L_\gamma = \{u\overline{v} \mid u, v \in dom(\gamma)$ and $\gamma(v) \leq \gamma(u)\}$, where we denote by $dom(\gamma)$ the domain of γ, and then we say L_γ is a witness of X.*

Intuitively, given $x \in X$, each $\omega \in \Sigma^{\circledast}$ with $\gamma(\omega) = x$ is a representation of x. Therefore, the fact that a WSTS can recognize the language L_γ witnesses that in some sense it can represent the structure of X: it can accept all words starting in some $u \in \Sigma^{\circledast}$, followed by some $\overline{v} \in \overline{\Sigma}^{\circledast}$, representing that $\gamma(v) \leq \gamma(u)$.

Example 3.3.1 *Let $X = Q \times \mathbb{N}$, with Q finite, with its standard order $\leq$ ($(q, n) \leq (p, m)$ iff $q = p$ and $n \leq m$). Taking $\Sigma = \{a\} \cup Q$, a Σ-representation of X is $\gamma : \Sigma^{\circledast} \to X$ with $\gamma(qa^n) = (q, n)$, so $L_\gamma = \{qa^n\overline{q}\overline{a}^m \mid m \leq n\}$ is a witness of X.*

Finally, in order to apply the framework to two classes of WSTS, we need to prove that both classes are able to accept encodings of their own state space. We call these classes *self-witnessing*. Formally, if $\mathbf{S}$ is a class of WSTS whose state spaces are included in the class of wpos $\mathbf{X}$, $(\mathbf{X}, \mathbf{S})$ is *self-witnessing* if, for all $X \in \mathbf{X}$, there is $S \in \mathbf{S}$ that recognizes a witness of X.

Proposition 3.3.4 ([16]) *Let $(\mathbf{X}, \mathbf{S})$ and $(\mathbf{X}', \mathbf{S}')$ be self-witnessing WSTS classes. If $\mathbf{S} \preceq \mathbf{S}'$ then $\mathbf{X} \sqsubseteq_{refl} \mathbf{X}'$.*

We use the previous result to prove non-inclusions of the families of coverability languages of *ν-lsPN*$_n$. Now, we define the state space of *ν-lsPN* by using regions, in terms of standard set constructions, like products, multisets or words.

Definition 3.3.3 *We define $\mathbf{X}_k$ as the class of sets $X^{\oplus}_{\max^*} \times (X^{\oplus}_{(\max -1)^*})^{\circledast} \times X^{\oplus}_{\{\max +1\}}$ for some $max \in \mathbb{N}$ and P, Q finite sets with $|P| = k$, where for every $I \subseteq (\max +1)^*$, $X_I = Q \times P^{\oplus} \times I$.*

Note that we can represent each region of a *ν-lsPN* as an element of some $X \in \mathbf{X}_k$. If $R = A_0 * A_1 * \ldots * A_n * A_\infty$ is a region, A_0 is represented by an element of $X^{\oplus}_{\max^*}$, $A_1 * \ldots * A_n$ by a word over the alphabet $X^{\oplus}_{(\max -1)^*}$ and A_∞ by an element of $X^{\oplus}_{\{\max +1\}}$. In particular, each $(m, r) \in A_i$ is represented by some $(q, \{p_1, \dots, p_n\}, k) \in Q \times P^{\oplus} \times I$, where $q \in Q$ is a control state corresponding to the bounded part of m.

For $X \in \mathbf{X}_k$, we will write Q_X, P_X and max_X to refer to Q, P and max as above (or just Q, P and max, abusing notation). Now, we show the order we use in $\mathbf{X}_k$:

- As $\mathbf{X}_k$ is the class of sets of $X^{\oplus}_{\max^*} \times (X^{\oplus}_{(\max -1)^*})^{\circledast} \times X^{\oplus}_{\{\max +1\}}$ we use the standard order over the Cartesian product to define the order over the class. That is, given $X, X' \in X^{\oplus}_{\max^*}$, $\omega, \omega' \in (X^{\oplus}_{(\max -1)^*})^{\circledast}$ and $Z, Z' \in X^{\oplus}_{\{\max +1\}}$, we say that $(X, \omega, Z) \leq^c (X', \omega', Z')$ if $X \leq X'$, $\omega \leq^* \omega'$ and $Z \leq Z'$.
- We order the elements in $(X^{\oplus}_{(\max -1)^*})^{\circledast}$ by the standard order for words $\leq^*$. That is, given two words $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$, we say that $a_1, \ldots, a_n \leq b_1, \ldots, b_m$ if there is a strictly increasing mapping $\phi : n^+ \to m^+$ such that, for each $i \in n^+$, $a_i \leq b_{\phi(i)}$.
- Again, if $I \subseteq (\max +1)^*$, we define the order over X_I as the standard order over the Cartesian product of $Q \times P^{\oplus} \times I$, $\leq^c$.
- Finally, we use the equality over Q, the standard multiset order defined in Sec. 2.1 over $P^{\oplus}$ and the standard order for naturals over I.

Example 3.3.2 *Let us recall the region $R = A_0 * A_1 * A_2 * A_\infty$ in Ex. 3.2.2, depicted in Fig. 3.12, where $A_0 = \{(\{p\}, 2)\}$, $A_1 = \{(\{qq\}, 0)\ ,\ (\{q\}, 2)\}$, $A_2 = \{(\{pq\}, 1)\}$ and $A_\infty = \{(\{ppq\}, 4)\}$. Suppose that $max = 3$ and q is the only bounded place, bounded by 2. We can take $Q = \{q_0, q_1, q_2\}$, where q_i is a state representing an instance with i tokens in q, and $P = \{p\}$. Then, we can write the region R as:*

$$(\{(q_0, \{p\}, 2)\}, \{(q_2, \emptyset, 0), (q_1, \emptyset, 2)\}\{(q_1, \{p\}, 1)\}, \{(q_1, \{p^2\}, 4)\})$$

which belongs to $X^{\oplus}_{\max^*} \times (X^{\oplus}_{(\max -1)^*})^{\circledast} \times X^{\oplus}_{\{\max +1\}}$, for $max = 3$.

In order to apply Prop. 3.3.4 to prove $\nu\text{-}lsPN_{k+1} \not\preceq \nu\text{-}lsPN_k$ we have to see that every $(\mathbf{X}_k, \nu\text{-}lsPN_k)$ is self-witnessing and that $\mathbf{X}_{k+1} \not\sqsubseteq_{refl} \mathbf{X}_k$. Given $X \in \mathbf{X}_k$, we define a Σ-representation γ_X of X, and therefore, we obtain a witness L_{γ_X} of X. We first need to define three auxiliary functions γ^I_1, γ^I_2 and γ^I_3. Intuitively, γ_1, γ_2 and γ_3 are in charge of the creation of tuples representing instances, multisets and words, respectively. More precisely, the encoding of an instance $(q, \{p_1, \ldots, p_n\}, k)$ will be the word $qp_1 \ldots p_n k$. Given $u_1, \ldots, u_n$ encodings of pairs, the encoding of the multiset of these pairs will be $u_1 \# \ldots \# u_n$. Finally, $v_1 * \ldots * v_n$ will be the encoding of the word formed by the multisets represented by $v_1, \ldots, v_n$.

Definition 3.3.4 *Given $X \in \mathbf{X}_k$, let $\Sigma = Q \cup P \cup (\max+1)^* \cup \{*, \#, \&\}$. We define $\gamma_1^I : \Sigma^{\circledast} \to X_I$, $\gamma_2^I : \Sigma^{\circledast} \to X_I^{\oplus}$ and $\gamma_3^I : \Sigma^{\circledast} \to (X_I^{\oplus})^{\circledast}$ as follows:*

- $\gamma_1^I(qp_1 \ldots p_n k) = (q, \{p_1, \ldots, p_n\}, k)$, *with $p_i \in P$, $q \in Q$ and $k \in I$,*
- $\gamma_2^I(u_1 \# \ldots \# u_n) = \{\gamma_1^I(u_1), \ldots, \gamma_1^I(u_n)\}$, *with $u_i \in dom(\gamma_1^I)$ for every i,*
- $\gamma_3^I(v_1 * \ldots * v_n) = \gamma_2^I(v_1) \ldots \gamma_2^I(v_n)$, *where $v_i \in dom(\gamma_2^I)$ for every i.*

Finally, we define the partial function $\gamma_X : \Sigma^{\circledast} \to X$ as

$$\gamma_X(u\&v\&w) = (\gamma_2^{max^*}(u), \gamma_3^{(max-1)^*}(v), \gamma_2^{\{max+1\}}(w))$$

γ_X is surjective, so that it is a Σ-representation of X and L_γ is a witness of X. Although not explicitly mentioned in the results from [16] shown above, if $X \in \mathbf{X}_{k+1}$ and $\mathbf{X}_{k+1} \not\sqsubseteq_{refl} \mathbf{X}_k$ then L_{γ_X} proves that $\nu\text{-}lsPN_{k+1} \not\preceq \nu\text{-}lsPN_k$.

Proposition 3.3.5 *$(\mathbf{X}_k, \nu\text{-}lsPN_k)$ is self-witnessing.*

Proof: Given $X \in \mathbf{X}_k$ we have to prove that there is $N \in \nu\text{-}lsPN_k$ such that $L(N) = L_{\gamma_X}$. Notice that N must have at most k unbounded places. N operates in two phases: the first phase generates u with $\gamma(u) = R = A_0 * A_1 * \ldots * A_n * A_\infty$, and the second one recognizes any $\overline{v}$ with $\gamma(v) \leq R$. In turn, each of the phases has three consecutive sub-phases, dealing with A_0, $A_1 * \ldots * A_n$ and A_∞, respectively. We use control places to move from one subphase to the next (with transitions labeled by & or $\overline{\&}$). In order to differentiate between phases, we say that we generate words in the first one, but we recognize them in the second.

We explain the generation of $A_1 * \ldots * A_n$, which is depicted in Fig. 3.19 (the other phases are simpler). In the figure, we omit the labels of the arcs labelled by $[0, \infty)$. Let $A_i = \{(q_1^i, m_1^i, r_1^i), \ldots, (q_{n_i}^i, m_{n_i}^i, r_{n_i}^i)\}$. We use a different name to represent each process instance. Moreover, instances in the same A_i have the same age. We use a place *now* that holds the name (with age 0) of the instance currently being generated. For a given i, we start by firing a transition of the form t_q (labelled by some $q \in Q$) which copies the name in *now* to a place q. This firing is followed by the (possibly multiple) firing of transitions t_p (labelled by some $p \in P$), each copying the name in *now* to a place p. These firings are followed by the firing of some t_r (labelled by $r \in (\max -1)^*$), which copies the name in *now* to a place r.

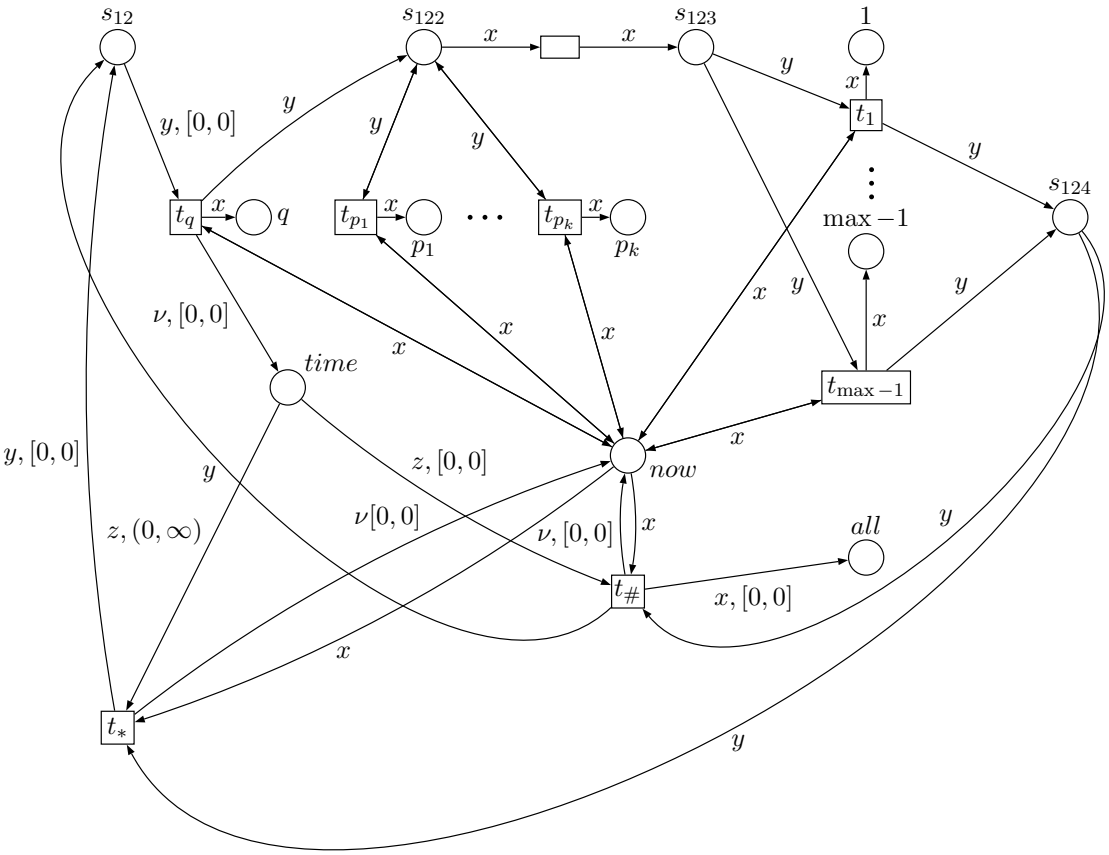


Figure 3.19: Generation of the encoding of $A_1 * \ldots * A_n$

Therefore, a word u with $\gamma_1^{(\max -1)^*}(u) = (q_1^i, m_1^i, k_1^i) \in X_{(\max -1)^*}$ can be produced. Next, there are two options: the immediate firing of a discrete transition, or an elapsing of time, leading to the generation of the next A_i:

If no time has elapsed from the previous firing of some transition $q \in Q$, the name in *now* can be moved to a place *all*, and replaced by a fresh name, with age 0 by one of the following sequence of firings:

- By firing transition $t_\#$, labeled by #, which sets a token of age 0 in place s_{12}.
- By firing a transition t'_* labeled by $*$ (omitted from the figure), which sets a token of age 0 in an auxiliary place, in order to force to wait some time to fire another transition $t''_\#$ which finally sets a token of age 0 in place s_{12}. This sequence is fired in order to start to generate the encoding of a new element in $X^{\oplus}_{(\max -1)^*}$.

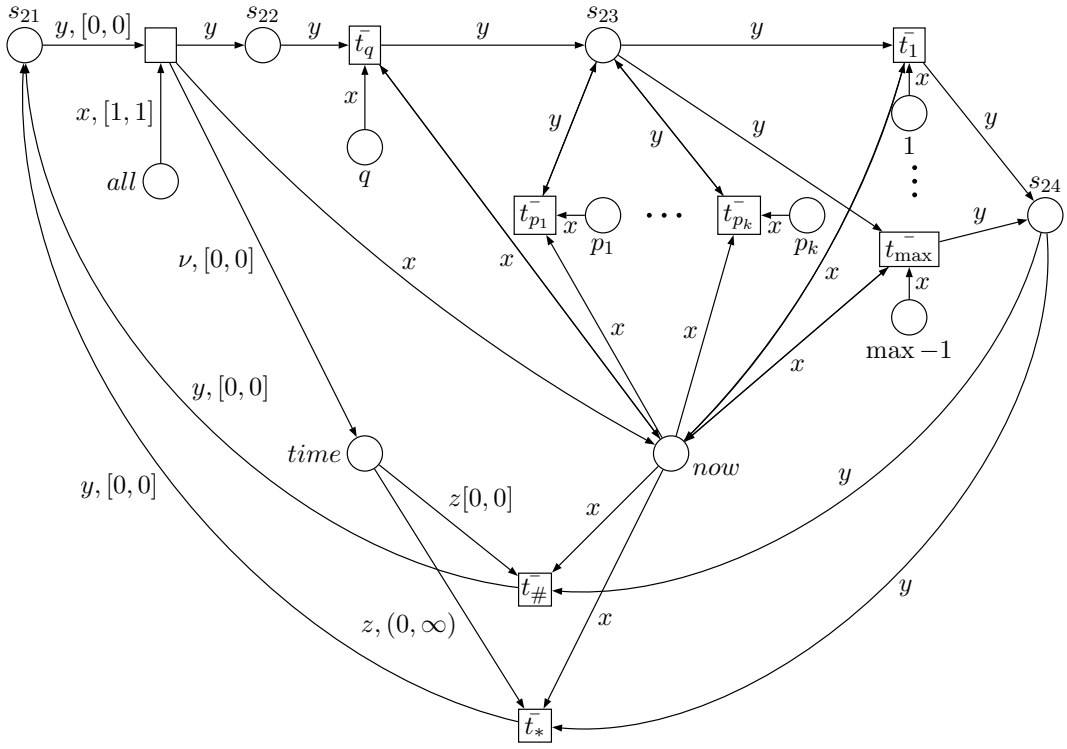


Figure 3.20: Generation of the encoding of $\bar{A}'_1 * \ldots * \bar{A}'_k$

Otherwise, some time has elapsed and hence we are forced to "forget" the instance in *now*, and start the generation of the next element A_i in $X^{\oplus}_{(\max-1)^*}$, by firing transition t_*.

These actions can be repeated to generate (the encoding of) any element in $X^{\oplus}_{(\max-1)^*}$.

After this phase, there is any number of pairwise different names (each representing an instance) in *all*, some of which have the same age (representing those instances in the same A_i). Moreover, for any name a in *all*, a may belong to some of the places in P (possibly repeated), and exactly to one place q and one place k.

The transitions in the second phase (the recognizing phase) demand that the age of the instances involved is exactly 1. Moreover, they are all labeled with symbols in $\overline{\Sigma}$.

This phase starts by taking any name in *all* with age 1 and putting it back to *now*. Then, a transition of the form $\bar{t}_q$ (labelled by $\overline{q}$) can remove a token from q

with the same name as the one in *now*. Then, transitions of the form $\overline{t}_p$ (labeled by $\overline{p}$) can be fired, each consuming a name from p matching the name in *now*. At any point, a transition of the form $\overline{t}_k$, labeled by $\overline{k}$ (with $k \in (\max)^*$) can be fired, which consumes from k a name matching the one in *now*. Thus, if the current name represented an instance (q, m, k) then (an encoding of) any (q, m', k) with $m' \subseteq m$ can be recognized. If no time has elapsed since the last firing of the first transition of this phase, the name in *now* can be consumed by firing transition $\overline{t}_{\#}$ labeled by $\overline{\#}$ in order to start recognizing the next instance.

At any point of the recognizing of some instance time can elapse. Then, when we finish recognizing some instance, $\overline{t}_*$ can be fired, labeled by $\overline{*}$, with the same effect as $\overline{t}_{\#}$. In that case, in order to start recognizing instances in the next A_i, another instance in *now* may have reached age 1, and hence, we start recognizing it. Notice that when time elapses, all the names with age greater than 1 are lost (the encodings of the instances they represent cannot be recognized). This is consistent with the fact that we must recognize (the encoding of) a state which is *less or equal* than the one we generated. Notice also that even in the first phase, names with ages older than 1 become garbage. However, it is possible to generate all the names in the first phase with an age smaller than 1, so that the same state can be recognized.

Even though the order between instances is not preserved within each A_i (this is not demanded by the order in $X^{\oplus}_{(\max -1)^*}$), this order *is* preserved between different A_i's, because older instances reach the age of 1 before. To conclude, we consider as final marking the one with a token in the control-state marked in the second phase (the recognizing one). Note that the built net has $max = 1$, that is, the maximum integer bound in the intervals labelling the arcs of the net we build is 1.

□

Thus, to apply Prop. 3.3.4 it only remains to see that $\mathbf{X}_{k+1} \not\sqsubseteq_{refl} \mathbf{X}_k$. In order to prove it we use ordinal theory (see Prop. 3.3.6 below). Ordinal theory was introduced by Cantor in 1883 [19], and intuitively, it is used to measure the "length" of well-ordered sets, which is called the *order type* of the set.

Let us explain the needed concepts about ordinals (for more details see [16]). Each ordinal α is equal to the set of ordinals $\{\beta \mid \beta < \alpha\}$ below it, and the class of ordinals is totally ordered by inclusion. For example, the ordinal 30 is the order type of all the ordinals below it, that is, the ordinals between 0 and 29. Hence, it is identified as the set $\{0, \ldots, 29\}$. Every total well order $(X, \leq)$ is isomorphic to

a unique ordinal $ot(X, \leq)$, called the *order type* of X. In the context of ordinals, we define $0 = \emptyset$, $n = \{0, \ldots, n-1\}$ and $\omega = \mathbb{N}$, ordered by the usual order. The ordinals below ϵ_0 (those bounded by a tower $\omega^{\omega^{\cdot^{\cdot^{\omega}}}}$) can be represented by the hierarchy of ordinals in Cantor Normal Form (CNF), recursively given by $C_0 = \{0\}$, and $C_{n+1} = \{\omega^{\alpha_1} + \ldots + \omega^{\alpha_p} \mid p \in \mathbb{N}, \alpha_1, \ldots, \alpha_p \in C_n \text{ and } \alpha_1 \geq \ldots \geq \alpha_p\}$ ordered by $\omega^{\alpha_1} + \cdots + \omega^{\alpha_p} \leq \omega^{\alpha'_1} + \cdots + \omega^{\alpha'_q}$ iff $(\alpha_1, \ldots, \alpha_p) \leq_{lex} (\alpha'_1, \ldots, \alpha'_q)$, where $\leq_{lex}$ is the lexicographic order. For example, given a finite set P, the order type of $P^{\oplus}$ is $\omega^{|P|}$, which is in Cantor Normal form.

Each ordinal below ϵ_0 has a unique CNF. We abbreviate $\alpha + \overset{k}{\cdots} + \alpha = \alpha * k$. A *linearization* of a po $\leq$ is a total order $\leq'$ st $x \leq y \Rightarrow x \leq' y$. A linearization of a wpo is well and total, hence isomorphic to an ordinal. The *maximal order type* of $(X, \leq)$ is $ot(X, \leq) = sup\ \{ot(X, \leq') \mid \leq' \text{ linearization of } \leq\}$.

The following result states that we can prove $\mathbf{X}_{k+1} \not\sqsubseteq_{refl} \mathbf{X}_k$ by comparing their ordinal types.

Proposition 3.3.6 ([103]) *For X and Y wpos, if $X \sqsubseteq_{refl} Y$ then $ot(X) \leq ot(Y)$.*

Using [28, 84, 103], we can compute the order type of products, domains of finite words or finite multisets. In particular, we will need the following result.

Lemma 3.3.7 *For every Q, P and I finite, $ot((Q \times P^{\oplus} \times I)^{\oplus\circledast}) = \omega^{\omega^{\omega^{|P|} * |Q| * |I|}}$. In particular, $ot(P^{\oplus\oplus\circledast}) = \omega^{\omega^{\omega^{|P|}}}$.*

Proof: Let X be any wpo with $\omega \leq ot(X) < \varepsilon_0$ and Y a finite set. In [28, 84] it is proved that $ot(X \times Y) = ot(X) * |Y|$ and $ot(X^{\circledast}) = \omega^{\omega^{ot(X)}}$, and [103] proves that $ot(X^{\oplus}) = \omega^{ot(X)}$. Moreover, $P^{\oplus}$ is isomorphic to $\mathbb{N}^{|P|}$, so that $ot(P^{\oplus}) = ot(\mathbb{N}^{|P|}) = \omega^{|P|}$. This allows us to compute the ordinals in the lemma.

□

Now, we apply the previous results in order to prove that the expressiveness of *ν-lsPN* grows with the number of unbounded places.

Proposition 3.3.8 *ν-lsPN$_k \prec \nu$-lsPN$_{k+1}$ for each $k \geq 0$.*

Proof: Trivially *ν-lsPN*$_k \preceq$ *ν-lsPN*$_{k+1}$ holds. In order to prove *ν-lsPN*$_{k+1} \not\preceq$ *ν-lsPN*$_k$ it is enough to apply Prop. 3.3.4, since both classes are self-witnessing (Prop. 3.3.5). Let us see that $\mathbf{X}_{k+1} \not\sqsubseteq_{refl} \mathbf{X}_k$. We consider only the part of the

state spaces composed of words (of multisets), the one playing the relevant part. Let us take $X_{k+1} = P^{\oplus\oplus *} \in \mathbf{X}_{k+1}$ (i.e., with only one control-state and $max = 1$), so that $|P| = k + 1$. For any $X_k \in \mathbf{X}_k$ we have that $X_k = (Q \times P'^{\oplus} \times I)^{\oplus *}$ for some $max \in \mathbb{N}$ and finite P' and Q with $|P'| = k$. By the previous lemma, $ot(X_{k+1}) = \omega^{\omega^{\omega^{k+1}}}$ and $ot(X_k) = \omega^{\omega^{\omega^{k}*|I|*|Q|}}$, which satisfy $ot(X_{k+1}) \not\leq ot(X_k)$. Since this is true for any $X_k \in \mathbf{X}_k$ we have that $\mathbf{X}_{k+1} \not\sqsubseteq_{refl} \mathbf{X}_k$.

□

Finally, we obtain the following result as a corollary.

Corollary 3.3.9 *TdPN* $\simeq \nu$*-lsPN*$_0 \prec \nu$*-lsPN*$_1 \prec \nu$*-lsPN*$_2 \prec \ldots \prec \nu$*-lsPN*

Hence, we have completed the picture at the beginning of this section, obtaining:

$$PN \prec AWN \prec \nu\text{-}PN \prec DN \simeq TdPN \simeq \nu\text{-}lsPN_0 \prec \nu\text{-}lsPN_1 \prec \ldots \prec \nu\text{-}lsPN$$

In particular, this implies that ν*-lsPN* is the most expressive class of all the *WSTS* classes whose expressiveness have been compared up to coverability languages, up to our knowledge.

Despite we have not focused on the complexity issues about the safety problems we are studying, note that, since ν*-lsPN* are more expressive than Data Nets or TPN, we can already obtain a lower bound for coverability and termination at level $F_{\omega^{\omega^{\omega}}}$ [43] in the fast-growing hierarchy. It would be interesting to know if this lower bound is tight, though we may expect it is not, due to the higher order types of the state space in ν*-lsPN*.

Chapter 4

Priced-Timed Nets

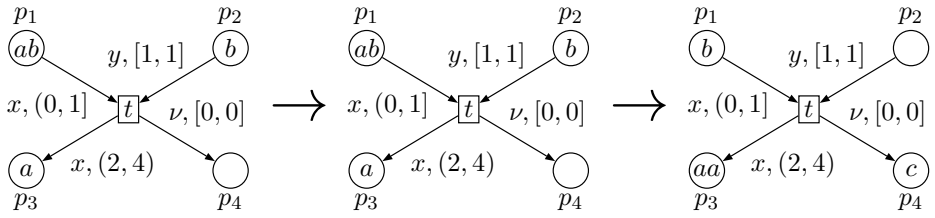
The goal of this chapter is to define a model to represent timed systems whose runs carry an associated cost in which different instances intervene, and to define and study safety properties over the prices of these runs. As in the previous chapter we found that the nets with more than one clock per instance had undecidable control-state reachability, we take ν-*lsPN* as the timed model in which we base our new extension. We define a costs model based on the one in [2]. We add the costs by considering a function *Cost* which assigns natural prices to places and transitions. Then, we will consider that each run produces two kinds of costs: storage costs and firing costs. Storage costs are associated to places, and are produced in the delays. These costs represent the ones produced by the storing and conservation of materials/resources. The costs associated to transitions are the firing costs, and represent the costs of performing actions. Then, the total price of a run will be the sum of the firing and storage costs it produces.

In such priced systems, we often want to remain under a given budget. Hence, the safety problem we will define consists in determining if every run which reaches a given set of final states keeps under this given budget. We will study the decidability of this problem by defining an abstraction which is equivalent to our new model, as done in [2], and proving that this abstraction belongs to the framework of Well Structured Transition Systems, as done in the previous chapter.

4.1 Priced-timed ν-Petri nets

First of all, let us define our priced and timed model.

Definition 4.1.1 (Priced-timed ν-Petri nets) *A* priced-timed ν-Petri net



$M_1 = a{:}(\{p_1p_3\}, 0),\ b{:}(\{p_1p_2\}, 0.5)$ $\quad M_2 = a{:}(\{p_1p_3\}, 0.5),\ b{:}(\{p_1p_2\}, 1)$ $\quad M_3 = a{:}(\{p_3p_3\}, 3), b{:}(\{p_1\}, 1),\ c{:}(\{p_4\}, 0)$

Figure 4.1: Firing of a transition in a ν-*PTdPN*.

(ν-PTdPN for short) is a tuple $N = \langle P, T, In, Out, Time, Cost\rangle$ *where* $\langle P, T, In, Out, Time\rangle$ *is a ν-lsPN called the* underlying *ν-lsPN and* $Cost : P \cup T \to \mathbb{N}$.

The behavior of a ν-*PTdPN* is given by the semantics of its underlying ν-*lsPN*. Therefore, the costs function does not change any behavior, it is only used to compute the prices of the runs. Hence, control-state reachability as defined for ν-*lsPN* is still decidable, and it is an interesting property to verify over ν-*PTdPN*. However, we would like to study safety properties dealing with prices too.

The price of a run is defined as the sum of the firing costs (the price of firing discrete transitions) and storage costs (the price of storing tokens in places while timed transitions are fired). Let us formally define the price of a run.

Definition 4.1.2 (Price of a run) *The cost of firing a discrete transition $t \in T$ is defined as*

$$Cost(M \xrightarrow{t} M') = Cost(t).$$

The cost of firing a timed transition with delay $d \in \mathbb{R}_{\geq 0}$ is defined as

$$Cost(M \xrightarrow{d} M') = d * \textstyle\sum_{p\in P} \sum_{i=1}^{n} m_i(p) * Cost(p),$$

where $M = \{a_1{:}(m_1, r_1), \ldots, a_n{:}(m_n, r_n)\}$.

Let $\pi = M_1 \xrightarrow{l_1} \ldots \xrightarrow{l_{n-1}} M_n$ *be a run of a ν-PTdPN. We define the price of π as*

$$Cost(\pi) = \textstyle\sum_{i=1}^{n-1} Cost(M_i \xrightarrow{l_i} M_{i+1}).$$

Let us illustrate the previous definition by an example:

Example 4.1.1 *We consider the ν-PTdPN defined by the underlying ν-lsPN depicted on Fig. 4.1, and the function Cost with $Cost(p_1) = Cost(t) = 2$, $Cost(p_2) = 1$ and $Cost(n) = 0$ otherwise. Then, the price of the firing $M_1 \xrightarrow{0.5} M_2$, depicted in the left-hand side of the figure is $0.5 * ((1 * 2) + (1 * 2) + (1 * 1)) = 2.5$. The price of the second firing $M_2 \xrightarrow{t} M_3$ is $Cost(t) = 2$. Hence, the price of the complete run is 4.5.*

In order to define the safety problem for *ν-PTdPN* we first need to define an order over its markings.

Definition 4.1.3 (Order over markings) *Given a marking M of a ν-PTdPN, let us call $Mk(M)$ the multiset over $P^{\oplus} \times \mathbb{R}_{\geq 0}$ with*

$$Mk(M)(m, r) = |\{a \in Id \mid a{:}(m, r) \in M\}|.$$

Then, given two markings M and M', we say that $M \ll M'$ if $Mk(M) \subseteq Mk(M')$, where $\subseteq$ is the standard order over multisets.

For example, $M_k(M_3) = \{(\{p_3, p_3\}, 3), (\{p_1\}, 1), (\{p_4\}, 0)\}$, where M_3 is the third marking in Fig 4.1.

Now we define the safety problem. Intuitively, a *ν-PTdPN* is b-safe for an initial marking M_0 and a final marking M_f if each run from M_0 to a marking which covers M_f costs less that b. Without loss of generality, we will suppose that all the instances in the initial marking have age 0.

Definition 4.1.4 (Safety problem) *Let $b \in \mathbb{N}$ and M_0, M_f be two markings of a ν-PTdPN N, where all the instances in M_0 have age 0 and all the instances in M_f have integer age. We say that N is b-safe for the pair of markings M_0, M_f, if for each run $\pi = M_0 \to \cdots \to M$ starting from M_0 and reaching $M \gg M_f$, $Cost(\pi) < b$.*

In [2] Abdulla and Mayr study the Cost-Threshold problem for Priced-Timed Petri nets, which consists in deciding whether a given marking can be covered in some run without spending more than a given budget. For this purpose, they prove that to solve the problem it is enough to consider computations where the ages of the clocks are arbitrarily close to an integer. Despite we are not trying to solve the same problem, we will use the same ideas to prove decidability of safety. Without loss of generality, in this section we suppose that $M_0 = a{:}(\{p_0\}, 0)$, that is, that the initial marking has only one token of age 0 in an special place p_0.

Given a marking M, note that we can make the following decomposition in a unique way:

- $M = M_{-n} + \ldots + M_{-1} + M_0 + M_1 + \ldots + M_m$.
- If $a{:}(m,r) \in M_i$ and $i < 0$, then $frct(r) \geq 1/2$. Analogously, if $a{:}(m,r) \in M_i$ and $i > 0$, then $frct(r) < 1/2$. Finally, if $a{:}(m,r) \in M_0$ then $frct(r) = 0$.
- $a:(m,r) \in M_i$ and $a':(m',r') \in M_j$, then $frct(r) = frct(r')$ if and only if $i = j$, and if $-n \leq i < j < 0$ or $0 \leq i < j \leq m$ then $frct(r) < frct(r')$.
- If $i \neq 0$ then $M_i \neq \emptyset$.

Example 4.1.2 *Let* $M = a:(\{p\}, 0.3), b:(\{pq\}, 2.2), c:(\{qq\}, 4), d:(\{p\}, 0.6), e:(\{q\}, 2.6)$. *The previous decomposition would be* $M = M_{-1} + M_0 + M_1 + M_2$, *with* $M_{-1} = d{:}(\{p\}, 0.6), e{:}(\{q\}, 2.6)$, $M_0 = c{:}(\{qq\}, 4)$, $M_1 = b{:}(\{pq\}, 2.2)$ *and* $M_2 = a{:}(\{p\}, 0.3)$.

Definition 4.1.5 (Detailed transition) *Let* $M = M_{-n} + \ldots + M_{-1} + M_0 + M_1 + \ldots + M_m$ *be a marking of a* ν*-PTdPN decomposed as previously defined, and* ϵ *the fractional part of the clocks of the instances in* M_{-1}*, if it exists, or* $1/2$ *otherwise. A firing of a timed transition* $M \xrightarrow{d} M'$ *is detailed if at most one fractional part of the clocks of the instances changes in a way that makes it reach or exceed the next integer value, that is,* $0 < d \leq 1 - \epsilon$ *or* $M_0 = \emptyset$ *and* $d = \epsilon$.

Each timed transition in a run can be decomposed in several firings of detailed timed transitions. Moreover, the price of the firing of the timed transition is equal to the sum of the prices of all the detailed transitions. Hence, from now on, in order to study the safety problem for a ν*-PTdPN* N, we only need to consider the runs of N in which all the timed transitions are detailed. This is similar to the cases of the proofs of lemmas 3.2.2-3.2.6 of previous section, in which we only need to consider the timed transitions from markings in which there are instances with natural age lower than max and no other instance reaches or exceeds the next integer age, or in which there are not instances with natural age lower than max and only the instances lower than max with the greatest fractional part of the age in the marking reach the next natural age.

Now, we formally define the concept of "computations where the ages of the clocks are arbitrarily close to an integer" and prove that it is enough to consider such computations in order to solve our safety problem, in the way of [2].

Definition 4.1.6 (δ-form) *Let $0 < \delta \leq 1/5$.*

- *An instance $a{:}(m, r)$ is in δ-form if $frct(r) < \delta$ or $frct(r) \geq 1 - \delta$.*
- *A marking is in δ-form if all its instances are in δ-form.*
- *A discrete transition is fired in δ-form if the instances created by the firing, and the instances whose clocks are changed by the firing are in δ-form.*
- *A run π is in δ-form if every discrete transitions in π is fired in δ-form and for each timed transition $M \xrightarrow{d} M'$ in π, $0 < d < \delta$ or $1 - \delta < d < 1$.*

We require $\delta \leq 1/5$ in order to ensure that the cases $frct(r) < \delta$ and $frct(r) \geq 1 - \delta$ do not overlap, and even after a delay $d \leq 1/5$, if $\delta' = \delta + d \leq 1/5$, the cases $frct(r) < \delta'$ and $frct(r) \geq 1 - \delta'$ still do not overlap.

Proposition 4.1.1 *Let $\pi = M_0 \xrightarrow{l_0} \ldots \xrightarrow{l_{n-1}} M_n$ be a run of a ν-PTdPN, with $\mathcal{M}_f$ an upward-closed set of markings such that $M_n \in \mathcal{M}_f$. For each $0 < \delta < 1/5$ there is a run $\pi' = M_0 \xrightarrow{l'_0} M'_1 \xrightarrow{l'_1} \ldots \xrightarrow{l_{n'-1}} M'_n$ in δ-form such that:*

- *$M'_n \in \mathcal{M}_f$,*
- *$Cost(\pi) \leq Cost(\pi')$,*
- *for each $i \in n^*$, $|M_i| = |M'_i|$, and*
- *if π is detailed, then π' is detailed.*

Proof: Let nt be the number of timed transitions in π, for $i \in nt^+$, let d_i be the delay of the i^{th} timed transition in π and let $0 < \delta < 1/5$. Let nc be the number of times the value of a clock of an instance is changed in π, including the case of the creation of new instances. Let us fix an arbitrary order over this "newly updated" or "newly created" clocks, and call them $c_1, \ldots, c_{nc} \in \mathbb{R}_{\geq 0}$. Let $a_1, \ldots, a_{nc}$ be the instances these clocks belong to. Note that this (multi)set of transitions may have repetitions. In that case, without loss of generality, we suppose that each instance is renamed when the value of its clock is changed by the firing of a discrete transition. We can simulate these renamings in a lossy way, preserving reachability.

We consider the set S of all the runs with the same structure as π, that is, the same transitions and modes as π, but maybe modifying the values of $d_1, \ldots, d_{nt}$ (the time delays) and $c_1, \ldots, c_{nc}$. These runs have the same length as π, and the

size of each marking is preserved too. Moreover, if π is detailed, a run π' in S is detailed too. Next, we find a run π' in S in δ-form with $Cost(\pi) \leq Cost(\pi')$.

The set of tuples $(d_1, \ldots, d_{nt}, c_1, \ldots, c_{nc})$ for which such a run is in S is defined by a set of inequations that depend on the intervals labelling transition. Each of the following conditions must be expressed by this set of equations:

- The time always advances, that is, $d_i > 0$.
- If an instance a_i is created by an output arc with interval $[a, b]$, then we have $a \leq c_i \leq b$. The case of (half) open intervals is analogous, using strict inequalities.
- Analogously, if the clock of an instance a_i is changed by an output arc with interval $[a, b]$, then we have $a \leq c_i \leq b$. Again, the case of (half) open intervals is analogous, using strict inequalities.
- Suppose a_i is an instance such that some of its tokens are input of some discrete transition t via an arc labelled by $[a, b]$. Let $d_k, d_{k+1}, \ldots, d_{k+l}$ be the delays between the setting of the clock c_i and the firing of t. Then, we have $a \leq c_i + d_k + d_{k+1} + \ldots + d_{k+l} \leq b$.

The previous inequations describe a polyhedron PH which contains all possible tuples such that the runs they represent are in S. The tuple corresponding to π is in S, and therefore PH is nonempty, so we can build the closure $\overline{PH}$ of the polyhedron by substituting the strict inequalities by normal ones. Let $v = (d_1, \ldots, d_{nt}, c_1, \ldots, c_{nc})$ be a vector of variables. Then, $\overline{PH}$ can be described by the inequation $M * v \leq c$, where c is a vector of integers and M is a matrix. M has the same form as the $PTPN$ matrices defined in [2]. Moreover, in Theorem 23 of [2] it is proved that the coordinates of the vertices of a polyhedron which is defined by such a matrix and a vector c of integers, are integers. Therefore, the coordinates of $\overline{PH}$ are integers.

Since the $Cost$ function is linear in $d_1, \ldots, d_{nt}$, the supreme of the prices of the runs in S is represented by a vertex of $\overline{PH}$, which has integer coordinates by [2]. Therefore, we can get arbitrarily close to the supreme of the costs giving to $d_1, \ldots, d_{nt}, c_1, \ldots, c_{nc}$ values arbitrarily close to integers. Therefore, for every π, there is π' in S with the same structure as π, in δ-form (that is, arbitrarily close to integers), such that $Cost(\pi) \leq Cost(\pi')$.

□

Hence, we obtain the following corollary.

Corollary 4.1.2 *For every $0 < \delta < 1/5$, if $M_0 \xrightarrow{\pi} M_n$ is a run of a ν-PTdPN N, there is a run $M_0 \xrightarrow{\pi'} M'_n$ in δ-form such that $Cost(\pi) \leq Cost(\pi')$ and $M_n \ll M'_n$.*

Hence, given a ν-*PTdPN* N, in order to solve the safety problem for N, that we have defined previously, we only need to analyze the runs in δ-form, for some $0 < \delta < 1/5$.

4.2 Abstract ν-*PTdPN*

Now we define an abstraction of ν-*PTdPN* merging the way of ordering the clocks in [2] and the way of abstracting instances in Chapter 3. The runs of the new model will represent runs of ν-*PTdPN*s in δ-form, for infinitesimally small δ. Moreover, the states of the model (called priced regions), will represent the accumulated cost of the run. Then, as in the case of ν-*lsPN*, we will prove that the new abstraction is a $WSTS$ and that we can solve the safety problem for ν-*PTdPN* by reducing it to a coverability problem for the abstraction. The new abstract model has the same syntax as ν-*PTdPN*: for each ν-*PTdPN* $N = \langle P, T, In, Out, Time, Cost\rangle$, we define the corresponding *abstract* ν-*PTdPN* $N' = \langle P, In, Out, Time, Cost\rangle$, denoted by ν-$aPTdPN(N)$. Now, we define the semantics of the abstract model.

Definition 4.2.1 (Priced regions) *A* priced region *is a tuple of the form $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m\rangle$, with $c, n, m \in \mathbb{N}$ and $A_i \in (P^{\oplus} \times (max+1)^*)^{\oplus}$ for $i \in \{-n, \ldots, m\}$.*

Intuitively, in a priced region, if $i < 0$, A_i represents a set of instances whose clocks have the same fractional part (as in Chapter 3), which is greater than $1-\delta$. Analogously, if $i > 0$, A_i represents a set of instances whose clocks have the same fractional part, which is smaller than δ. Moreover, if $i < j < 0$ or $0 < i < j$, the clocks of the instances in A_i have fractional part smaller than the ones in the instances in A_j. Finally, A_0 represents the instances whose clocks are natural numbers. The number c represents the accumulated cost of the run in the abstract model. For simplicity, we suppose that $A_i \neq \emptyset$ if $i \neq 0$.

Definition 4.2.2 (Priced region of a marking) *Given a marking of a ν-PTdPN $M = M_{-n} + \ldots + M_{-1} + M_0 + M_1 + \ldots + M_m$ decomposed in its*

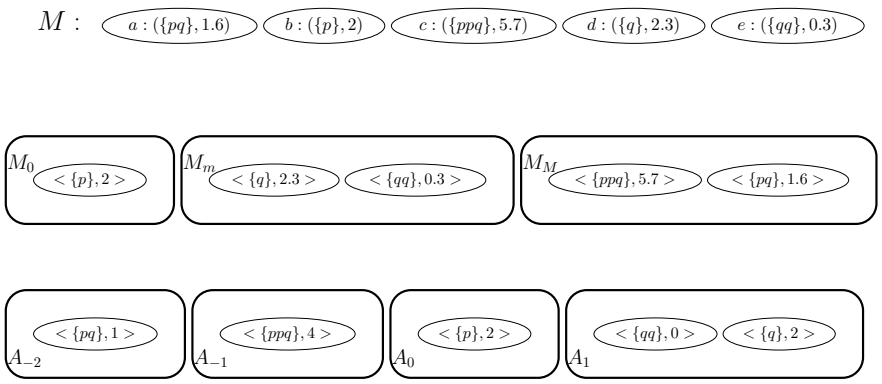


Figure 4.2: The abstract region associated to a marking

fractional parts as previously defined, we define the priced region

$$PR(M) = \langle 0, A_{-n} * \ldots * A_{-1}, A_0, A_1 * A_m \rangle$$

where for every $i \in \{-n, \ldots, m\}$, $A_i((m,n)) = |\{a \in Id \mid \exists r \in \mathbb{R}_{\geq 0}$ *with* $n = \lfloor r \rfloor, a:(m,r) \in M_i\}|$ *for* $n \leq \max$, *and* $A_i((m, max+1)) = |\{a \in Id \mid \exists r > \max, a{:}(m,r) \in M_i\}|$.

Let us illustrate the previous definition with an example.

Example 4.2.1 *The upper part of Fig. 4.2 represents a marking of a* ν*-PTdPN with* $max = 3$*. The first step we take to build its corresponding abstract region is to sort the instances by the fractional part of their clocks, forgetting the concrete names: we consider the multiset* M_0 *of instances with natural age (b), the multiset* M_m *of instances with fractional part of the age lower than* 0.5 *(d and e) and the multiset* M_M *of instances with fractional part of the age greater than* 0.5 *(a and c). Finally, we forget about the concrete fractional parts of the ages, we change the age of the only instance older than max (c) to* $max+1$*, and we build multisets of instances with the same fractional part of the age, setting first the instances in* M_M*, then the instance in* M_0 *and finally the instances in* M_m*.*

Now, we define the transition relation $\rightarrowtail$ for ν-$aPTdPN$. Note that in addition to managing the abstraction of the markings, we need to calculate the increment of (abstract) costs in each transition. We first consider the firing of abstract

timed transitions, so we need to handle the addition of storage costs. Given $A \in (P^{\oplus} \times (\max+1)^*)^{\oplus}$, we denote $A^{+1} \in (P^{\oplus} \times (max+1)^*)^{\oplus}$ defined by: $A^{+1}((m, r+1)) = A((m,r))$ if $0 < r \leq \max$ and $A^{+1}((m, \max+1)) = A((m, \max+1)) + A((m, \max))$ otherwise. Note that as in the abstraction for ν-*lsPN*, all the ages greater than max are represented by $\max+1$.

Definition 4.2.3 (Firing of timed transitions in ν-*aPTdPN*) *We consider four kinds of abstract timed transitions $\overset{i}{\rightarrowtail}$:*

- $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \overset{1}{\rightarrowtail} \langle c, A_{-n} * \ldots * A_{-1}, \emptyset, A_0 * A_1 * A_m \rangle$,
- $\langle c, A_{-n} * \ldots * A_{-1}, \emptyset, A_1 * \ldots * A_m \rangle \overset{2}{\rightarrowtail} \langle c, A_{-n} * \ldots A_{-2}, A_{-1}^{+1}, A_1 * \ldots * A_m \rangle$,
- $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \overset{3}{\rightarrowtail} \langle c', A_{-n}^{+1} * \ldots * A_{-1}^{+1} * A_0 * \ldots * A_k, \emptyset, A_{k+1}^{+1} * \ldots * A_m^{+1} \rangle$, *with* $c' = c + \sum_{i=-n}^{m} \sum_{(m',r) \in A_i} A_i(m', r) * \sum_{p \in P} m'(p) * Cost(p)$.
- $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \overset{4}{\rightarrowtail} \langle c', A_{-n}^{+1} * \ldots * A_{-1}^{+1} * A_0 * \ldots * A_k, A_{k+1}^{+1}, A_{k+2}^{+1} * \ldots * A_m^{+1} \rangle$, *with* $c' = c + \sum_{i=-n}^{m} \sum_{(m',r) \in A_i} A_i(m', r) * \sum_{p \in P} m'(p) * Cost(p)$.

Moreover, we define $\overset{\Delta}{\rightarrowtail}$ as the reflexive and transitive closure of $\overset{1}{\rightarrowtail} \cup \overset{2}{\rightarrowtail} \cup \overset{3}{\rightarrowtail} \cup \overset{4}{\rightarrowtail}$.

The first and second types of firings represent infinitesimally small delays (lower than the considered δ) so we consider that the price of the run is not incremented. In the first type the instances in A_0 change to a non-integer value but the instances in A_{-1} do not reach or exceed the next integer. In the second type $A_0 = \emptyset$, and instances in A_{-1} reach the next integer age.

The third and fourth types of firings represent delays which are close to one (greater than $1 - \delta$) so we consider that the price of the run is incremented by $c' = c + \sum_{i=-n}^{m} \sum_{(m',r) \in A_i} A_i(m', r) * \sum_{p \in P} m'(p) * Cost(p)$, that is, the storage costs corresponding to one unit of time. In the third type only the clocks of the instances in $A_0 * \ldots * A_k$ do not exceed the next integer, and no instance reaches an integer age. In the fourth type only the clocks of the instances in $A_0 * \ldots * A_{k+1}$ do not exceed the next integer, and the instances represented by A_{k+1} reach the next integer age.

Example 4.2.2 *Let us consider the same ν-PTdPN as in the previous example, with $max = 3$. The first and second regions depicted on Fig. 4.3 represent a firing of a timed transition of type* 1. *Hence, the multiset A_0 becomes A_1, and A_0 gets*

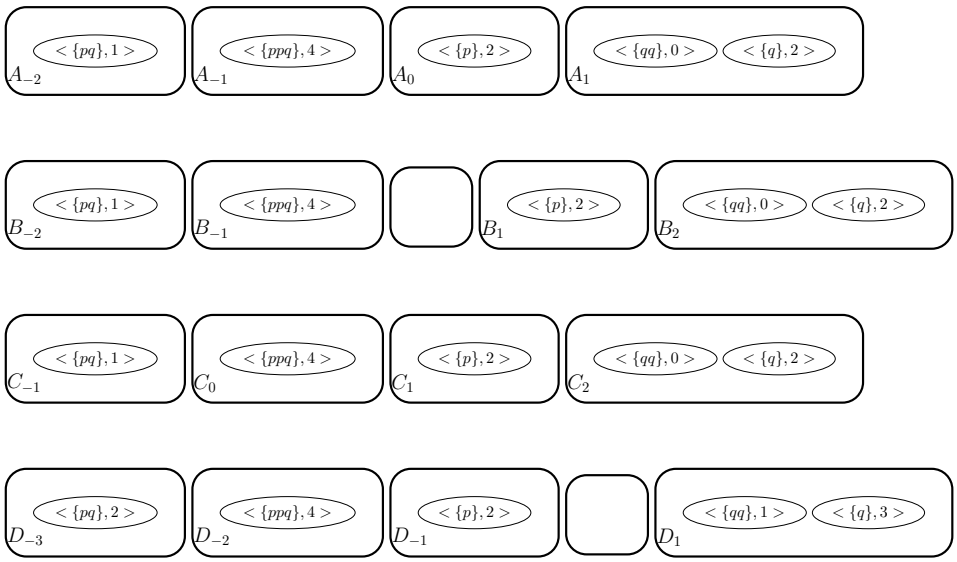


Figure 4.3: Time elapsing for regions

empty. Then, there is a time elapsing of the second kind from the second region to the third one, in which the instance in A_{-1} goes to A_0. Note that the age of this instance is not incremented, since it is already greater than max. Finally, a firing of type 3 is represented by the third and fourth regions, in which all the instances but the ones in A_0 and A_1 exceed their next integer age, and no instance reaches exactly the next integer age.

Now, we define the firings of discrete transitions. In order to make this definition simpler, we consider the predicate *match* defined in the previous chapter, which relates the abstract representation of instances (pairs of $P^{\oplus} \times \mathbb{R}_{\geq 0}$) with the inputs and outputs of the transitions.

Moreover, if $a = (m, r) \in (P^{\oplus} \times \mathbb{R}_{\geq 0})$ we denote $a^{+\epsilon} = (m, r + \epsilon)$. Finally, we are ready to define the firing of discrete transitions in ν-*aPTdPN*.

Definition 4.2.4 (Firing of discrete transitions in ν-*aPTdPN*) *Let $t \in T$. We have $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \overset{t}{\rightarrowtail} \langle c', B_{-n'} * \ldots * B_{-1}, B_0, B_1 * \ldots * B_{m'} \rangle$ if for each $i \in \{-n, \ldots, m\}$, there are $A_i^I, A_i^{rest}, O, A_0^O, O^\nu, A_0^\nu \in (P^{\oplus} \times (max + 1)^*)$ such that, for each $0 < \epsilon < 1$ the following conditions hold:*

- $c' = c + Cost(t)$.

- $A_i = A_i^I + A_i^{rest}$.
- *There are two bijections* $f : nfVar(t) \to \biguplus_{i\in\{-n,\dots,m\}} A_i^I$ *and* $h : nfVar(t) \to O + A_0^O$ *such that, if* $f(x) = (m, r)$ *and* $h(x) = (m', r')$ *then:*
  - *If* $f(x)$ *is selected from* A_0 *then* $match(f(x), (In_t(x), Time_t^1(x)))$ *and* $match(f(x)^{+\epsilon}, (In_t(x), Time_t^1(x)))$ *otherwise.*
  - $m' = (m - In_t(x)) + Out_t(x)$ *and if* $h(x)$ *is selected from* A_0^O *then* $r' \in Time_t^2(x)$ *and* $r' + \epsilon \in Time_t^2(x)$ *otherwise.*
- *There is a bijection* $h' : fVar(t) \to O^\nu + A_0^\nu$ *such that, if* $h'(\nu) = (m, r)$ *then:*
  - $m = Out_t(\nu)$,
  - *If* $h'(\nu)$ *is selected from* A_0^ν *then* $r \in Time_t^2(\nu)$ *and* $r + \epsilon \in Time_t^2(\nu)$ *otherwise.*
- *There is a strictly monotone injection* $\varphi : \{i \in \{-m, \dots, n\} \mid A_i^{rest} \neq \emptyset\} \cup \{0\} \to \{-m', \dots, n'\}$ *with* $\varphi(0) = 0$ *such that:*
  - *For all* $i \in \{-n, \dots, -1, 1, \dots, m\}$, $A_i^{rest} \subseteq B_{\varphi(i)}$.
  - $B_0 = A_0^{rest} + A_0^O + A_0^\nu$.
  - $\biguplus_{i\in\{-n,\dots,-1,1,\dots,m\}} B_i = (\biguplus_{i\in\{-n,\dots,-1,1,\dots,m\}} A_i^{rest}) + O + O^\nu$.

The first condition of the previous definition makes the cost represented by the initial region be incremented by the cost of firing t. The second condition expresses that the multisets A_i in the initial region are split into two parts: the instances A_i^I which are assigned to some variable in the firing, and those A_i^{rest} which keep unchanged. In the third and fourth conditions the injection f selects the appropriate pairs to be associated to the incoming variables of t (which fit into the conditions of In and $Time^1$), and express how the outcoming instances, selected by the injection h and h' from O and A_0^O, must be. Note that if an instance which takes part in the firing is not selected from A_0, we add ϵ to the age when checking if it is in the corresponding interval $Time_t^1(x)$. This is because the natural number in the pair of the selected instance corresponds to the natural part of the age of the instance in the non abstract net, that has fractional part greater than 0. Finally, the last condition basically expresses that the order of the remaining multisets in the new region must not change (because φ is monotone), and that the new region is obtained from the old one by removing the multisets of the region which represent instances assigned to preconditions, and adding

to some of the multisets (or maybe to new ones), the pairs representing new or changed instances, which are in O and A_0^O.

Then, we define the transition relation $\rightarrowtail$ over ν-$aPTdPN$ as $\rightarrowtail = \overset{\Delta}{\rightarrowtail} \cup \bigcup_{t\in T} \overset{t}{\rightarrowtail}$.

4.2.1 Correctness of the simulation

In this section we prove that the previous abstraction simulates ν-$PTdPN$ in a correct way. More precisely, we will prove that given a ν-$PTdPN$ N with an initial marking M_0 and a final marking M_f, the supremum of the prices of the runs which begin in M_0 and finish in a marking greater than M_f in N and in its ν-$aPTdPN$ coincide. This condition is enough to reduce the safety problem for ν-$PTdPN$ to an analogous problem for ν-$aPTdPN$. We consider several lemmas in order to prove the previous result.

In the rest of this chapter, given a region $R = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m\rangle$, we denote $R_{+x} = \langle c+x, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m\rangle$. Let us first focus on the case of a discrete transition.

Lemma 4.2.1 *Given a marking M of a ν-$PTdPN$ in δ-form, for some $\delta \leq 1/5$, and $c \in \mathbb{R}_{\geq 0}$, there is a firing of a discrete transition in δ-form $M \xrightarrow{t} M'$ if and only if $PR(M)_{+c} \overset{t}{\rightarrowtail} PR(M')_{+(c+Cost(t))}$.*

Proof: Let $M_{-n} + \ldots + M_{-1} + M_0 + M_1 + \ldots + M_m$ be the decomposition of M as defined previously, and $PR(M)_{+c} = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * A_m\rangle$. We prove both implications.

First, suppose there is a firing of a discrete transition in δ-form $M \xrightarrow{t} M'$. Note that if $PR(M)_{+c} \overset{t}{\rightarrowtail} R$ for some region R then the first component of R will be $c + Cost(t)$, by Def. 4.2.4. Hence, we focus on proving that this transition can be performed, and the rest of the components of R are as required. Let us consider $M = a_1 : (m_1, r_1), ..., a_n : (m_n, r_j) + \overline{M}$ as in Def. 3.2.4, and let us call $\tilde{M} = a_1 : (m_1, r_1), ..., a_n : (m_n, r_j)$. Then, for each $k \in \{-n \ldots m\}$ we can decompose $M_k = \tilde{M}_k + \overline{M}_k$, where $\tilde{M}_k \ll \tilde{M}$ and $\overline{M}_k \ll \overline{M}$. Let us define, for each $k \in \{-n \ldots m\}$, the multisets $A_k^I, A_k^{rest} \in (P^{\oplus} \times \mathbb{N})^{\oplus}$ for the abstract firing, with:

- $A_k^I((m,n)) = \left|\{a \in Id \mid \exists r \leq \max \text{ with } n = \lfloor r \rfloor, a{:}(m,r) \in \tilde{M}_k\}\right|$ if $n \leq \max$, $A_k^I((m, max+1)) = \left|\{a \in Id \mid \exists r > \max, a{:}(m,r) \in \tilde{M}_k\}\right|$,

- $A_k^{rest}((m,n)) = |\{a \in Id \mid \exists r \leq \max \text{ with } n = \lfloor r \rfloor, a:(m,r) \in \overline{M}_k\}|$ if $n \leq \max$ and $A_k^{rest}((m,max+1)) = |\{a \in Id \mid \exists r > \max, a{:}(m,r) \in \overline{M}_k\}|$.

Note that $A_k = A_k^I + A_k^{rest}$. Now, if for each instance $a_i{:}(m_i,r_i) \in \tilde{M}$, and m_i' and r_i' are the ones defined in the firing of t, then we define

$$O = \biguplus_{i \in j^+ | r_i' \notin \mathbb{N}} (m_i', \lfloor r_i' \rfloor)$$

and

$$A_0^O = \biguplus_{i \in j^+ | r_i' \in \mathbb{N}} (m_i', \lfloor r_i' \rfloor).$$

Analogously, if $fVar = \{\nu_1, \ldots, \nu_{j'}\}$, for each $i \in j'^+$ if we consider m_i'' and r_i'' as in the firing of t, then we define

$$O^\nu = \biguplus_{i \in j'^+ | r_i'' \notin \mathbb{N}} (m_i'', \lfloor r_i'' \rfloor)$$

and

$$A_0^\nu = \biguplus_{i \in j'^+ | r_i'' \in \mathbb{N}} (m_i'', \lfloor r_i'' \rfloor).$$

Now, we only have to prove that the five conditions in Def.4.2.4 hold. The first and second conditions have already been proved. Let us check the three remaining ones:

- First, we define f: Suppose that $nfVar(t) = \{x_1, \ldots, x_j\}$ as in Def. 3.2.4. Then, for each $i \in j^+$, we define $f(x_i) = (m_i, \lfloor r_i \rfloor)$, that is, we assign to x_i the pair corresponding to (m_i, r_i) in the abstract net. If $f(x)$ is selected from A_0 then $\lfloor r_i \rfloor = r_i$ and by the second point in Def. 3.2.4, $match(f(x_i), (In_t(x_i), Time_t^1(x_i)))$ holds. Otherwise, $r_i = \lfloor r_i \rfloor + \epsilon$ for some $0 < \epsilon < 1$ and hence $match(f(x_i)^{+\epsilon}, (In_t(x_i), Time_t^1(x_i)))$ holds.

  Now we define h in an analogous way: for each $i \in j^+$, we define $h(x_i) = (m_i', \lfloor r_i' \rfloor)$, that is, we assign to each x_i the pair representing (m_i', r_i') in O or A_0^O. Then, by the fourth point in Def. 3.2.4, $m_i' = (m - In_t(x_i)) + H_t(x_i)$. Moreover, if $h(x_i)$ is selected from A_0^O then $r_i' = \lfloor r_i' \rfloor$ and therefore, by the sixth point in Def. 3.2.4, $r_i' \in Time_t^2(x_i)$. Otherwise, $r_i' = \lfloor r_i' \rfloor + \epsilon$ for some $0 < \epsilon < 1$ and hence $\lfloor r_i' \rfloor + \epsilon \in Time_t^2(x_i)$.

- For each $i \in j'^+$, we define $h'(\nu_i) = (m_i'', \lfloor r_i'' \rfloor)$, where (m_i'', r_i'') is as defined

in Def. 3.2.4. Therefore, by the fifth point in Def. 3.2.4, $m''_i = Out(\nu_i)$. Moreover, if $h'(\nu_i)$ is selected from A''_0 then $r''_i = \lfloor r''_i \rfloor$ and, by the last point in Def. 3.2.4, $\lfloor r''_i \rfloor \in Time^2_t(\nu_i)$. Otherwise, there is $0 < \epsilon < 1$ with $r''_i = \lfloor r''_i \rfloor + \epsilon$, and hence $\lfloor r''_i \rfloor + \epsilon \in Time^2_t(\nu_i)$.

- Then, we define $B_0 = A^{rest}_0 + A^O_0 + A^\nu_0$, ensuring the second condition. Moreover, the mutisets $\{(m'_i, r'_i) \mid x_i \in nfVar(t)\}$ and $\{(m''_i, r''_i) \mid \nu_i \in fVar(t)\}$ can be decomposed into increasing fractional parts in an unique way, as done for M. Let us suppose that these two decompositions are $M'_{-n'} * \ldots * M'_{-1}, M'_0, M'_1 * \ldots * M'_{m'}$ and $M''_{-n''}, * \ldots * M''_{-1}, M''_0, M''_1 * \ldots * M''_{m''}$, respectively. Then, we consider the abstractions of these decompositions, $A'_{-n'} * \ldots * A'_{-1}, \emptyset, A'_1 * \ldots * A'_{m'}$ and $A''_{-n''} * \ldots * A''_{-1}, \emptyset, A''_1 * \ldots * A''_{m''}$, by only taking the integer part of the ages, and discarding the pairs with integer age which are represented in A^O_0 and A^ν_0. Note that $\biguplus_{i \in \{-n', m'\}} A'_i = O$ and $\biguplus_{i \in \{-n'', m''\}} A''_i = O^\nu$.

  Then, we can merge these two decompositions with the sequence $A^{rest}_{-n} * \ldots * A^{rest}_{-1}, A^{rest}_0, A^{rest}_1 * \ldots * A^{rest}_m$, by summing the multisets representing instances with the same fractional part of the age, obtaining a sequence $B_{-n'} * \ldots * B_{-1}, B_0, B_1 * \ldots * B_{m'}$ for which there is a strictly monotone injection $\varphi : \{i \in \{-m, \ldots, n\} \mid A_i \neq \emptyset\} \to \{-m', \ldots, n'\}$ with $f(0) = 0$ such that for each $i \in \{-n, \ldots, -1, 1, \ldots, m\}$, $A^{rest}_i \subseteq B_{\varphi(i)}$ and $\biguplus_{i \neq 0} B_i = (\biguplus_{i \neq 0} A^{rest}_i) + O + O^\nu$.

Now we prove the other implication. Let us suppose that there are two markings of a ν-*PTdPN* M and M' with $PR(M)_{+c} \overset{t}{\rightarrowtail} PR(M')_{+(c+Cost(t))}$. For each $i \in \{-n, \ldots m\}$, we consider the decomposition in Def. 4.2.4, $A_i = A^I_i + A^{rest}_i$. Moreover, let us suppose that $PR(M') = \langle c + Cost(t), B_{-n'} * \ldots * B_{-1}, B_0, B_1 * \ldots * B_{m'} \rangle$. In order to prove that we can fire t from M, we take $\overline{M} = \biguplus_{i \in \{-n, \ldots, m\}} M^{rest}_i$, where A^{rest}_i are the pairs corresponding to the abstraction of M^{rest}_i, and we denote $A^{rest} = \biguplus_{i \in \{-n, \ldots, m\}} A^{rest}_i$. Suppose that $nfVar(t) = \{x_1, \ldots, x_j\}$. Then, for each $i \in j^+$, we denote $a_i{:}(m_i, r_i)$ the instance of M such that $f(x_i) = (m_i, \lfloor r_i \rfloor)$ is the pair representing it in $PR(M)$. Then, by the third point of Def. 4.2.4, we have that $i \in j^+$, $In_t(x_i) \subseteq m_i$ and $r_i(t) \in Time^1_t(x_i)$. Hence, t is enabled at M.

Let us show that $M \overset{t}{\to} M'$. First, we focus on the instances with natural age. We have that $B_0 = A^{rest}_0 + A^O_0 + A^\nu_0$. Hence, by the third point of Def. 4.2.4, each selected instance with integer age in M' assigned whose abstraction in the abstract firing is assigned to a non-free variable x_i by f (and hence it is in A^O_0) is

of the form $a_i{:}(m'_i, r'_i)$, where $m'_i = (m_i - In_t(x_i)) + Out_t(x_i)$ and $r'_i \in Time^2_t(x_i)$. Note that such values can be taken for the firing $M \xrightarrow{t} M'$, by the fourth and sixth points of Def. 3.2.4. Analogously, by the fourth point of Def. 4.2.4, each selected instance with integer age in M' whose abstraction in the abstract firing is assigned to a free variable ν_i by h' (and hence it is in A^ν_0) is of the form $b_i{:}(m''_i, r''_i)$, where $m''_i = Out_t(\nu_i)$, $r''_i \in Time^2_t(\nu_i)$ and b_i is a new name which is not in $Id(M)$. Again, such values can be taken for the firing of $M \xrightarrow{t} M'$, by the fifth and seventh points of Def. 3.2.4. Hence, if we do not take any more instances with natural ages in the firing, we ensure that the instances with natural ages that we obtained after firing t from M in that way are exactly represented in B_0.

Now, we focus on the rest of the instances of M'. Let $a_i : (m'_i, r'_i)$ be an instance in M' such that $a_i \in Id(M)$. Then, by the third point of Def. 4.2.4, $m'_i = (m_i - In_t(x_i)) + Out_t(x_i)$ and $\lfloor r'_i \rfloor + \epsilon \in Time^2_t(x_i)$. As $r'_i \notin \mathbb{N}$, $r'_i \in Time^2_t(x_i)$. Hence, by the fourth and sixth points of Def. 3.2.4, we can take these values of m'_i and r'_i for the firing of t from M. Analogously, if $a_i : (m'_i, r'_i)$ is an instance in M' such that $a_i \notin Id(M)$, then by the third point of Def. 4.2.4, $m''_i = m_i + Out_t(\nu_i)$ and $\lfloor r'_i \rfloor + \epsilon \in Time^2_t(x_i)$. Again, as $r'_i \notin \mathbb{N}$, $r'_i \in Time^2_t(x_i)$. Hence, by the fifth and seventh points of Def. 3.2.4, we can take these values of m''_i and r''_i for the firing of t from M. Therefore, we have obtained all the instances of M' by firing t from M. Moreover, note that all the m'_is and m''_is in the firing from M have been defined. Hence, $M \xrightarrow{t} M'$.

Finally, it is easy to see that if M is in δ-form then M' can be chosen in δ-form. As the ages of the instances represented by A^{rest}_i in the firing do not change, we only have to ensure that the fractional part of the newly created ages can be lower than δ or greater than $1-\delta$. Moreover, the instances whose ages are newly created and have fractional part as some of the instances in M, or are in between the fractional parts of the ages of two instances of M both greater than $1-\delta$ or lower than δ, are trivially in δ-form. Hence, if $M = M_{-n}+\ldots+M_{-1}+M_0+M_1+\ldots+M_m$, then we only need to focus on the newly created ages which are larger than the ones in $M_1, \ldots, M_m$ and lower than any of the ones in $M_{-n}, \ldots, M_{-1}$. First, note that since the timing restrictions are defined as intervals with natural bounds, the concrete fractional part of the taken ages do not affect the fulfilling of the restrictions (only whether they are 0 or not). Hence, whenever we have a fractional part of the age greater than $1/2$, we can take a fractional part greater than $1-\delta$ instead, and whenever we have a fractional part of the age lower than $1/2$, we can take a fractional part lower than δ instead. Moreover, note that if α is

the maximum fractional part in $M_1, \ldots, M_m$, then $\alpha < \delta$, because M is in δ-form. Hence, there is space for an unbounded number of instances with different fractional part of the age in (α, δ). Analogously, if β is the minimum fractional part in $M_{-n}, \ldots, M_{-1}$, then $\beta > 1 - \delta$, because M is in δ-form. Hence, there is space for an unbounded number of instances with different fractional parts of their ages in $(1 - \delta, 1 - \beta)$.

□

Now we focus on the correction of the simulation of the timed transitions. We consider two different cases: the "small" delays close to zero, lower than δ, and the "big" delays close to one and greater than $1 - \delta$. We first analyze the first case.

Lemma 4.2.2 *Given a marking M of a ν-PTdPN in δ-form for some $\delta \leq 1/5$, $d \in (0, \delta)$ and $c \in \mathbb{R}_{\geq 0}$, there is a detailed transition $M \xrightarrow{d} M^{+d}$ iff there is an abstract timed transition $PR(M)_{+c} \overset{1}{\rightarrowtail} PR(M^{+d})_{+c}$ or $PR(M)_{+c} \overset{2}{\rightarrowtail} PR(M^{+d})_{+c}$.*

Proof: Let $M_{-n} + \ldots + M_{-1} + M_0 + M_1 + \ldots + M_m$ be the decomposition of M as defined previously, ϵ be the fractional part of the ages of the instances in $M-1$ and $M' = M_{-n}^{+d} + \ldots + M_{-1}^{+d} + M_0^{+d} + M_1^{+d} + \ldots + M_m^{+d}$. Moreover, let $PR(M)_{+c} = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$. If M is in δ-form then $0 < 1 - \epsilon < \delta$. As the transition $M \xrightarrow{d} M^{+d}$ is detailed, either $d < 1 - \epsilon$ or $d = 1 - \epsilon$ and $M_0 = \emptyset$. We analyze both cases:

- If $d < 1 - \epsilon$ then the fractional part of the ages of the instances in M_{-1}^{+d} is $\epsilon + d$, with $1 - \delta < \epsilon + d < 1$. Moreover, the fractional part of the instances in M_0^{+d} is d, with $0 < d < \delta$. Therefore, we can apply the first case of Def. 4.2.3, getting that $PR(M)_{+c} = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \overset{1}{\rightarrowtail} \langle c, A_{-n} * \ldots * A_{-1}, \emptyset, A_0, A_1 * \ldots * A_m \rangle = PR(M')_{+c}$ if and only if $M \xrightarrow{d} M^{+d}$.

- If $d = 1 - \epsilon$ and $M_0 = \emptyset$ then the fractional part of the ages of the instances in M_{-1}^{+d} is $frct(\epsilon + (1 - \epsilon)) = 0$, that is, these instances reach the next integer age. Therefore, we can apply the second case of Def. 4.2.3 to obtain that $PR(M)_{+c} = \langle c, A_{-n} * \ldots * A_{-1}, \emptyset, A_1 * \ldots * A_m \rangle \overset{2}{\rightarrowtail} \langle c, A_{-n} * \ldots * A_{-2}, A_{-1}^{+1}, A_0 * A_1 * \ldots * A_m \rangle = PR(M')_{+c}$ if and only if $M \xrightarrow{d} M^{+d}$.

□

Finally, let us analyze what happens with the delays greater than $1-\delta$.

Lemma 4.2.3 *Given a marking M of a ν-PTdPN in δ-form for some $\delta \leq 1/5$, $d \in (1-\delta, 1)$ and $c \in \mathbb{R}_{\geq 0}$, there is a detailed transition $M \xrightarrow{d} M^{+d}$ if and only if there is an abstract timed transition as defined in the third or the fourth types of Def. 4.2.3 such that $PR(M)_{+c} \xrightarrow{d} PR(M^{+d})_{+c'}$, where $c' = c + \sum_{a \in Id(M)} \sum_{m \mid \exists r \in \mathbb{R}_{\geq 0} \text{ with } a:(m,r) \in M} \sum_{p \in P} m(p) * Cost(p)$.*

Proof: Again, let $M_{-n}+\ldots+M_{-1}+M_0+M_1+\ldots+M_m$ be the decomposition of M as defined previously. This time, let ϵ_k be the fractional part of the ages of the instances in M_k, for $k \in m^*$. As M is in δ-form, $0 \leq \epsilon_k < \delta$. Moreover, let $PR(M)_{+c} = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$. Then, there must be $k \in m^+$ such that either $d = 1 - \epsilon_k$ or $d \in (1-\epsilon_{k-1}, 1-\epsilon_k)$. Let us analyze both cases:

- If there is $k \in m^+$ with $d = 1 - \epsilon_k$ then the fractional part of the ages of the tokens in M_k after the firing is $frct(\epsilon_k + 1 - \epsilon_k) = 0$, that is, these instances reach the next integer age. Moreover, the instances in $M_{-n}, \ldots, M_{-1}$ and $M_{k+1}, \ldots, M_m$ exceed the next integer age, while the instances in $M_0, \ldots, M_{k-1}$ do not reach the next integer age, and are slightly below it after the firing. Hence, if we apply the fourth kind of firing of Def. 4.2.3, we get that $PR(M)_{+c} = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \overset{4}{\hookrightarrow} \langle c', A_{-n}^{+1} * \ldots * A_{-1}^{+1} * A_0 * \ldots * A_{k-1}, A_k, A_{k+1}^{+1} * \ldots * A_m^{+1} \rangle = PR(M')_{+c'}$, where $c' = c + \sum_{i=-n}^{m} \sum_{(m',r) \in A_i} A_i(m', r) * \sum_{p \in P} m'(p) * Cost(p) = c + \sum_{a \in Id(M)} \sum_{m \mid \exists r \in \mathbb{R}_{\geq 0} \text{ with } a:(m,r) \in M} \sum_{p \in P} m(p) * Cost(p)$, if and only if $M \xrightarrow{d} M^{+d}$.

- Finally, if there is $k \in m^+$ with $d \in (1-\epsilon_{k-1}, 1-\epsilon_k)$ then no instance has fractional part of age 0 after the firing. Moreover, the instances in $M_k, \ldots, M_m$ and $M_{-n}, \ldots, M_{-1}$ exceed then next integer age, while the rest of the instances do not reach the next integer age, and keep slightly under it. Hence, if we apply the third kind of firing of Def. 4.2.3, we get $PR(M)_{+c} = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \overset{3}{\hookrightarrow} \langle c', A_{-n}^{+1} * \ldots * A_{-1}^{+1} * A_0 * \ldots * A_{k-1}, \emptyset, A_k^{+1} * \ldots * A_m^{+1} \rangle = PR(M')_{+c'}$, where $c' = c + \sum_{i=-n}^{m} \sum_{(m',r) \in A_i} A_i(m', r) * \sum_{p \in P} m'(p) * Cost(p) = c + \sum_{a \in Id(M)} \sum_{m \mid \exists r \in \mathbb{R}_{\geq 0} \text{ with } a:(m,r) \in M} \sum_{p \in P} m(p) * Cost(p)$, if and only if $M \xrightarrow{d} M^{+d}$.

□

Now we focus on how the abstraction simulates the costs of the runs.

Lemma 4.2.4 *Let M_0 be a marking of a ν-PTdPN with all the instances of age 0. For each run $\pi = M_0 \to M_1 \to \ldots \to M_f$ in detailed δ-form, with $f * \delta \leq 1/5$, there is a run $PR(M_0) \rightarrowtail PR(M_1)_{+c_1} \rightarrowtail \ldots \rightarrowtail PR(M_f)_{+c_f}$ such that:*

$$|cost(\pi) - c_f| \leq f * \delta * (max_{0 \leq i \leq f}(\sum_{a_j:(m_j,r_j) \in M_i} |m_j|)) * (max_{p \in P} Cost(p)).$$

Conversely, for each run $\langle 0, \emptyset, A_0, \emptyset \rangle = R_0 \rightarrowtail R_1 \rightarrowtail \ldots \rightarrowtail R_f = \langle c, B_{-n} * \ldots * B_{-1}, B_0, B_1 * \ldots * B_m \rangle$ and every $\delta \leq 1/5$, there is a run $\pi' = M_0 \to M_1 \to \ldots \to M_f$ with $R_i = PR(M_i)_{+c_i}$ for each $i \in f^*$ and $c_i \in \mathbb{R}_{\geq 0}$ such that:

$$\left|c - cost(\pi')\right| \leq f * \delta * (max_{0 \leq i \leq f}(\sum_{(m_j,r_j) \text{ in } R_i} |m_j|)) * (max_{p \in P} Cost(p)).$$

Proof: Let $\pi = M_0 \to M_1 \to \ldots \to M_f$ be a run of a ν-PTdPN in detailed δ-form, with $f * \delta \leq 1/5$. As π is in detailed form, the previous lemmas can be applied. Hence, there is a corresponding abstract computation $\pi' = PR(M_0) \rightarrowtail PR(M_1)_{+c_1} \rightarrowtail \ldots \rightarrowtail PR(M_f)_{+c_f}$. Note that the price of firing a transition t in a ν-PTdPN is exactly the price which is added to the first component of the corresponding region in the abstract run when firing t. Hence, we only need to focus on the differences of costs in the timed transitions. As π is in δ-form, each timed transition $M_i \xrightarrow{d} M_{i+1}$ in the run has $d \in (0, \delta)$ or $d \in (1 - \delta, 1)$. Moreover, as the run begins with all instances of age 0, the fractional part of the age of each instance in a marking M_i of the run is less than $i * \delta \leq f * \delta < 1/5$ far from the nearest integer. Then, for every timed transition $M_i \xrightarrow{d} M_{i+1}$ of the run, we have

$$\left|Cost(M_i \xrightarrow{d} M_{i+1}) - (c_{i+1} - c_i)\right| \leq \delta * (\sum_{a_j:(m_j,r_j) \in M_i} |m_j|) * (max_{p \in P} Cost(p)).$$

Therefore, we have:

$$|cost(\pi) - c_f| \leq f * \delta * (max_{0 \leq i \leq f}(\sum_{a_j:(m_j,r_j) \in M_i} |m_j|)) * (max_{p \in P} Cost(p)).$$

Conversely, let us consider the abstract run $\langle 0, \emptyset, A_0, \emptyset \rangle = R_0 \rightarrowtail R_1 \rightarrowtail \ldots \rightarrowtail R_f = \langle c, B_{-n} * \ldots * B_{-1}, B_0, B_1 * \ldots * B_m \rangle$, and let $M_0 = PR(R_0)$. Note that all the instances of M_0 have age 0. We can use the previous lemmas to build the run $\pi' = M_0 \to M_1 \to \ldots \to M_f$ we need. For each $0 \leq i \leq f$, let $\delta_i = \delta * 2^{i-f}$. Then, by the previous lemmas, we can build a run π' for which the following properties

hold:

- For each $0 \leq i \leq f$, $R_i = PR(M_i)$.
- M_i is in δ_i-form. We prove it by induction. For M_0 the property holds trivially, since all the instances in this marking have age 0. Suppose that M_i is in δ_i form. Then, from the previous lemmas we obtain that if $M_i \to M_{i+1}$ is a timed transition $\xrightarrow{d}$, then either we have $d \in (0, \delta_i)$ or $d \in (1-\delta, 1)$. Hence, M_{i+1} is in $2 * \delta_i = \delta_{i+1}$-form.

Again, we only need to analyze the difference of prices of the timed transitions. For every timed transition $M_i \xrightarrow{d} M_{i+1}$ of π', if c_i and c_{i+1} are the first components of R_i and R_{i+1} respectively, we have

$$\Big|Cost(M_i \xrightarrow{d} Mi+1) - (c_{i+1} - c_i)\Big| \leq \delta_i * (\sum_{(m_j, r_j) \text{ in } R_i} |m_j|) * (max_{p \in P} Cost(p)).$$

Hence, we obtain:

$$\big|c - cost(\pi')\big| \leq f * \delta * (max_{0 \leq i \leq f}(\sum_{(m_j, r_j) \text{ in } R_i} |m_j|)) * (max_{p \in P} Cost(p)).$$

□

Finally, we can prove that the abstraction we have defined properly works to reduce the price safety problem for ν-*PTdPN* to a coverability problem for ν-*aPTdPN*.

Proposition 4.2.5 *Given two markings M_0, M_f of a ν-PTdPN N, where all the instances in M_0 have age 0,*

$$sup\{Cost(\pi) \mid M_0 \xrightarrow{\pi} M \gg M_f\} =$$

$$sup\{c \mid PR(M_0) \rightarrowtail^* PR(M) = \langle c, A_{-n'} * \ldots * A_{-1}, A_0, A_1 * A_m \rangle \text{ with } M \gg M_f\}.$$

Proof: We denote $S = sup\{Cost(\pi) \mid M_0 \xrightarrow{\pi} M \gg M_f\}$ and $S' = sup\{c \mid PR(M_0) \rightarrowtail^* PR(M) = \langle c, A_{-n'} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$ with $M \gg M_f\}$. First, let us prove that $S \not< S'$. We need to prove that for each run of N finishing in a marking which covers M_f with cost c, there is a run in the abstract net that finish in a region corresponding to a marking which covers M_f and has a price greater than or equal to c. Let π_ϵ be a run such that $M_0 \xrightarrow{\pi_\epsilon} M$,

with $M \gg M_f$ and $S - Cost(\pi_\epsilon) = \epsilon$. We can assume that π_ϵ is in detailed form. Let $\pi_\epsilon = M_0 \to M_1 \to \ldots \to M_n$, and $\delta_\epsilon = min\{1/(5 * n), \epsilon/(n * (max_{0 \leq i \leq n}(\sum_{a_j:(m_j,r_j) \in M_i} |m_j|)) * (max_{p \in P} Cost(p)))\}$. By Prop. 4.1.1, there is a run $M_0 \xrightarrow{\pi'_\epsilon} M'_n \gg M_f$ of N in detailed δ_ϵ-form such that:

- $Cost(\pi'_\epsilon) \geq Cost(\pi_\epsilon)$ and
- for each $i \in n^*$, $|M_i| = |M'_i|$.

Hence, $S - Cost(\pi'_\epsilon) \leq \epsilon$. Moreover, by Lemma 4.2.4, there is an abstract run $\pi''_\epsilon = PR(M_0) \rightarrowtail PR(M'_1)_{+c_1} \rightarrowtail \ldots \rightarrowtail PR(M'_n)_{+c_n}$ with $|Cost(\pi'_\epsilon) - c_n| \leq n * \delta_\epsilon * (max_{0 \leq i \leq n}(\sum_{a_j:(m_j,r_j) \in M'_i} |m_j|)) * (max_{p \in P} Cost(p)) \leq \epsilon$. Hence, $S - Cost(\pi''_\epsilon) \leq \epsilon$ and therefore $S \not< S'$.

Now we prove that $S' \not< S$. Let $\pi_\epsilon = RM(M_0) \overset{\pi_\epsilon}{\rightarrowtail} PR(M_n)_{+c_n} = \langle c, A_{-n'} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$ with $M_n \gg M_f$ be a run of the abstract net such that $S' - c = \epsilon$ (note that $c = c_n$). Again, let us define $\delta_\epsilon = min\{1/(5 * n), \epsilon/(n * (max_{0 \leq i \leq n}(\sum_{(m_j,r_j) \text{ in } R_i} |m_j|)) * (max_{p \in P} Cost(p))\}$. By Lemma 4.2.4, there is a run $\pi'_\epsilon = M_0 \to \ldots \to M_n$ in detailed δ_ϵ-form such that $|c - cost(\pi'_\epsilon)| \leq n * \delta_\epsilon * (max_{0 \leq i \leq n}(\sum_{(m_j,r_j) \text{ in } R_i} |m_j|)) * (max_{p \in P} Cost(p))$. Hence, $S' - Cost(\pi'_\epsilon) \leq \epsilon$, and therefore $S' \not< S$.

Therefore, we obtain that $S = S'$.

□

4.2.2 Coverability for ν-$aPTdPN$ is decidable

In this section we are going to define an order over priced regions which induces a (priced) coverability problem for ν-$aPTdPN$, in order to reduce priced safety for ν-$PTdPN$ to this problem. Then, we will prove that coverability is decidable for ν-$aPTdPN$ by applying the framework of Well Structured Transition Systems, as we did for ν-$lsPN$ in the previous chapter. We first define the order for ν-$aPTdPN$.

Definition 4.2.5 (Order over priced regions) *We define* $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \sqsubseteq^{Pr} \langle c', B_{-n'} * \ldots * B_{-1}, B_0, B_1 * \ldots * B_{m'} \rangle$ *iff* $A_0 \leq^{\oplus} B_0$, $A_{-n} * \ldots * A_{-1} \leq^{\oplus \circledast} B_{-n'} * \ldots * B_{-1}$, $A_1 * \ldots * A_m \leq^{\oplus \circledast} B_1 * \ldots * B_{m'}$ *and* $c \leq c'$.

This order induces a coverability problem in the transition system generated by a ν-$aPTdPN$ with priced regions as states. Note that if $b \in \mathbb{N}$ and M_0, M_f are

two markings of a ν-*PTdPN* N, where all the instances in M_0 have age 0, we can reduce b-safety of N for the pair of markings M_0, M_f to the coverability problem induced by $\sqsubseteq^{Pr}$ for the initial region $PR(M_0)$ and the final region $PR(M_f)_{+b}$. Indeed, N is b-safe for M_0, M_f if and only if for each run $\pi = M_0 \to \cdots \to M$ starting from M_0 and reaching $M \gg M_f$, $Cost(\pi) < b$. By Prop. 4.2.5, this is equivalent to: $sup\{Cost(\pi) \mid M_0 \xrightarrow{\pi} M \gg M_f\} = sup\{c \mid PR(M_0) \rightarrowtail^* PR(M) = \langle c, A_{-n'} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$ for some $M \gg M_f\} < b$. Hence, N is b-safe for M_0, M_f if and only if the region $PR(M_f)_{+b}$ is not coverable from $PR(M_0)$.

Hence, we focus on proving the decidability of coverability for ν-*aPTdPN*. First of all, we need to prove that $\sqsubseteq^{Pr}$ is a decidable wqo.

Proposition 4.2.6 *$\sqsubseteq^{Pr}$ is a decidable wqo.*

Proof: First, $\sqsubseteq^{Pr}$ is trivially decidable. Moreover, note that a priced region $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$ can be seen an element of $X = \mathbb{N} \times (X^{\oplus}_{(max+1)^*})^{\circledast} \times X^{\oplus}_{(max+1)^*} \times (X^{\oplus}_{(max+1)^*})^{\circledast}$, where $X_{(max+1)^*} = P^{\oplus} \times (max+1)^*$ because $A_0 \in X^{\oplus}_{(max+1)^*}$ and $A_{-n} * \ldots * A_{-1}$ and $A_1 * \ldots * A_m$ can be seen as words over $X^{\oplus}_{(max+1)^*}$. Hence, $\sqsubseteq^{Pr}$ is the standard order in X, as defined in the preliminaries. Then, $\sqsubseteq^{Pr}$ is a wpo because it is built from wpos using operators that preserve well-orders.

□

Now we prove that $\rightarrowtail$ is compatible with $\sqsubseteq^{Pr}$. We are going to split this proof in several lemmas, handling the discrete and the continuous firings separately, and then merging them. Before starting with the continuous case, given a word $w = A_1 * \ldots * A_n$ over $(P^{\oplus} \times (max+1)^*)^{\oplus}$ and $m \in \mathbb{N}$, let us define $w^{+m} = A_1^{+m} * \ldots * A_n^{+m}$.

Lemma 4.2.7 *Given $i \in \{1,2,3,4\}$ and R_1, R_2 and R_1' three priced regions, if $R_1 \overset{i}{\rightarrowtail} R_2$ and $R_1 \sqsubseteq^{Pr} R_1'$ then there is a priced region R_2' such that $R_1' \rightarrowtail^* R_2'$ and $R_2 \sqsubseteq^{Pr} R_2'$.*

Proof: Suppose that $R_1 = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$ and $R_1' = \langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} * a_{-1}, A'_0, a_0 * A'_1 * a_1 * \ldots * A'_m * a_m \rangle$, with $c' \geq c$ and for each $j \in \{-n-1, \ldots, m\}$, $A_j \leq^{\oplus} A'_j$ and $a_j \in ((P^{\oplus} \times (max+1)^*)^{\oplus})^{\circledast}$. Let us analyze the four cases:

- First, suppose that $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle \overset{1}{\rightarrowtail} \langle c, A_{-n} * \ldots * A_{-1}, \emptyset, A_0 * A_1 * A_m \rangle = R_2$. Then $\langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} *$

$a_{-1}, A'_0, a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle \overset{1}{\rightarrowtail} \langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} * a_{-1}, \emptyset, A'_0 * a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle \sqsupseteq^{Pr} R_2$.

- If $\langle c, A_{-n} * \ldots * A_{-1}, \emptyset, A_1 * \ldots * A_m\rangle \overset{2}{\rightarrowtail} \langle c, A_{-n} * \ldots A_{-2}, A_{-1}^{+1}, A_1 * A_m\rangle = R_2$, then one of the two following cases hold:
  - If $A'_0 = \emptyset$ then, by concatenating firings of the second and the first kind, we get $\langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} * a_{-1}, \emptyset, a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle \rightarrowtail^* \langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * a_{-2}, {A'_{-1}}^{+1}, a_{-1}^{+1} * a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle \sqsupseteq^{Pr} R_2$.
  - If $A'_0 \neq \emptyset$ then, by concatenating firings of the first and the second kind, we get $\langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} * a_{-1}, A'_0, a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle \rightarrowtail^* \langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * a_{-2}, {A'_{-1}}^{+1}, a_{-1}^{+1} * A'_0 * a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle \sqsupseteq^{Pr} R_2$.
- Now, suppose $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m\rangle \overset{3}{\rightarrowtail} \langle c'', A_{-n}^{+1} * \ldots * A_{-1}^{+1} * A_0 * \ldots * A_k, \emptyset, A_{k+1}^{+1} * \ldots * A_m^{+1}\rangle = R_2$, with $c'' = c + \sum_{i=-n}^{m} \sum_{(m',r)\in A_i} A_i(m', r) * \sum_{p\in P} m'(p) * Cost(p)$. Then, we have $\langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} * a_{-1}, A'_0, a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle \overset{3}{\rightarrowtail} \langle c''', a_{-n-1}^{+1} * {A'_{-n}}^{+1} * a_{-n}^{+1} * \ldots * {A'_{-1}}^{+1} * a_{-1}^{+1} * A'_0 * a_0 * \ldots * A'_k, \emptyset, a_k^{+1} * {A'_{k+1}}^{+1} * \ldots * {A'_m}^{+1} * a_m^{+1}\rangle = R'_2$, with $c''' = c' + \sum_{i=-n}^{m} \sum_{(m',r)\in A_i} A_i(m', r) * \sum_{p\in P} m'(p) * Cost(p) + \sum_{k=-n-1}^{m} \sum_{B_i\in a_k} \sum_{(m',r)\in B_i} B_i(m', r) * \sum_{p\in P} m'(p) * Cost(p)$, and hence $R'_2 \sqsupseteq^{Pr} R_2$.
- Finally, suppose that $\langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m\rangle \overset{4}{\rightarrowtail} \langle c', A_{-n}^{+1} * \ldots * A_{-1}^{+1} * A_0 * \ldots * A_k, A_{k+1}^{+1}, A_{k+2}^{+1} * \ldots * A_m^{+1}\rangle$, with $c'' = c + \sum_{i=-n}^{m} \sum_{\{m|\exists r,(m,r)\in A_i\}} \sum_{p\in P} m(p) * Cost(p)$. Then, $\langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} * a_{-1}, A'_0, a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle \overset{4}{\rightarrowtail} \langle c''', a_{-n-1}^{+1} * {A'_{-n}}^{+1} * a_{-n}^{+1} * \ldots * {A'_{-1}}^{+1} * a_{-1}^{+1} * A'_0 * a_0 * \ldots * A'_k * a_k, {A'_{k+1}}^{+1}, a_{k+1}^{+1} \ldots * {A'_m}^{+1} * a_m^{+1}\rangle = R'_2$, with $c''' = c' + \sum_{i=-n}^{m} \sum_{(m',r)\in A_i} A_i(m', r) * \sum_{p\in P} m'(p) * Cost(p) + \sum_{k=-n-1}^{m} \sum_{B_i\in a_k} \sum_{(m',r)\in B_i} B_i(m', r) * \sum_{p\in P} m'(p) * Cost(p)$, and hence $R'_2 \sqsupseteq^{Pr} R_2$.

□

Now we focus on the firings of discrete transitions.

Lemma 4.2.8 *Given $t \in T$ and R_1, R_2 and R'_1 three priced regions, if $R_1 \overset{t}{\rightarrowtail} R_2$ and $R_1 \sqsubseteq^{Pr} R'_1$ then there is a priced region R'_2 such that $R'_1 \overset{t}{\rightarrowtail} R'_2$ and $R_2 \sqsubseteq^{Pr} R'_2$.*

Proof: Suppose that $R_1 = \langle c, A_{-n}*\ldots*A_{-1}, A_0, A_1*\ldots*A_m\rangle \overset{t}{\rightarrowtail} \langle c'', B_{-n'}*\ldots* B_{-1}, B_0, B_1*\ldots*B_{m'}\rangle = R_2$ and $R_1' = \langle c', a_{-n-1}*A'_{-n}*a_{-n}*\ldots*A'_{-1}*a_{-1}, A'_0, a_0* A'_1*a_1*\ldots*A'_m*a_m\rangle$, with $c' \geq c$ and for each $j \in \{-n-1,\ldots,m\}$, $A_j \leq^{\oplus} A'_j$ and $a_j \in ((P^{\oplus} \times (max+1)^*)^{\oplus})^{\circledast}$. We define the components firing $R_1' = \langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} * a_{-1}, A'_0, a_0 * A'_1 * a_1 \ldots * A'_m * a_m\rangle \rightarrowtail \langle c''', \omega_1, B'_0, \omega_2\rangle = R_2'$, in order to get $R_2' \sqsupseteq^{Pr} R_2$.

First, note that $c'' = c + Cost(t)$, and $c''' = c' + Cost(t)$. Hence, as $c \leq c'$, we have $c'' \leq c'''$. Now, suppose f, h, h' and φ are the injections defined for the firing of t from R_1. Let us define the injections for the firing from R_1':

- For each $i \in \{-n-1,\ldots,m\}$ and $A' \in a_i$, we define $A'^{rest} = A'$ and $A'^{I} = \emptyset$. Moreover, for each $i \in \{-n,\ldots,m\}$ there is an injection $\xi_i : A_i \rightarrow A'_i$ such that, if $(m,r) \in A_i$, then $(m,r) \leq \xi_i((m,r))$. Then, we define $A_i'^{I} = \biguplus_{(m,r)\in A_i}\{\xi_i((m,r))\}$ and $A_i'^{rest} = A'_i - A_i'^{I}$.

- We define $f' : nfVar(t) \rightarrow \biguplus_{i\in\{-n,\ldots,m\}} A_i'^{I}$ such that, if $x \in nfVar(t)$, then $f'(x) = (m',r')$ if $f(x) = (m,r)$ and $(m',r') = \xi_i(m,r)$ for some $i \in \{-n,\ldots,m\}$. In this case, note that $m \subseteq m'$, and hence, if $f'(x)$ is selected from A'_0 then $match(f'(x), (In_t(x), Time_t^1(x)))$ holds, or $match(f'(x)^{+\epsilon}, (In_t(x), Time_t^1(x)))$ holds otherwise.

  If $f(x)$ is taken from A_i, $h(x) = (m_1, r_1)$, and $\xi_i(f(x)) = (m', r')$ then we define $h''(x) = (m_2, r_2)$ with $m_2 = (m' - In_t(x)) + Out_t(x) \supseteq m_1$ and $r_2 = r_1$. Hence, we take $O' = \biguplus_{i\in\{-n,\ldots,m\}} \biguplus_{\{(m_1,r_1)\in O \mid \exists x\in nfVar, h(x)=(m_1,r_1) \text{ and } f(x)\in A_i\}} \{(m_2,r_1) \mid \xi_i(f(x)) = (m',r') \text{ and } m_2 = (m' - In_t(x)) + Out_t(x)\}$. Analogously, $A_0^{O'} = \biguplus_{i\in\{-n,\ldots,m\}} \biguplus_{\{(m_1,r_1)\in A_0^O \mid \exists x\in nfVar, h(x)=(m_1,r_1) \text{ and } f(x)\in A_i\}} \{(m_2,r_1) \mid \xi_i(f(x)) = (m',r') \text{ and } m_2 = (m' - In_t(x)) + Out_t(x)\}$. Note that as we take the same clocks as in the firing from R_1, the timing conditions concerning $h'(x)$ trivially hold. Moreover, note that in the previous definitions of O' and $A_0^{O'}$, it holds that $m_1 \subseteq m_2$.

- We consider the same multisets O^{ν} and A_0^{ν} and the injection h' of the firing of t from R_1. Hence, the conditions over them trivially hold.

- Let us denote $R_2' = \langle c', b_{-n'-1} * B'_{-n'} * b_{-n'} * \ldots * B'_{-1} * b_{-1}, B'_0, b_0 * B'_1 * b_1 * \ldots * B'_{m'} b_{m'}\rangle$, with $b_i = B'_{i1} * \ldots * B'_{ik_i}$ for $i \in \{-n'-1,\ldots,m'\}$

  Moreover, we denote $a_i = A'_{i1} * \ldots * A'_{il_i}$ for $i \in \{-n-1,\ldots,m\}$. Then, we define φ' (by abusing notation, since it is defined over natural numbers and pairs of natural numbers) such that:

- For all $i \in \{-n, \ldots, m\}$ such that $A_i^{rest} \neq \emptyset$, $\varphi'(i) = \varphi(i)$,
- if $\varphi(i) = j$, then φ' assigns the multisets in a_i to b_j, in the order they are, that is, for each $i' \in l_i^+$, $\varphi(ii') = ji'$.

Note that $\varphi'(0) = \varphi(0) = 0$. Moreover, with this assignation, which is clearly a monotone injection, we can define R_2' such that:

- For all $i \in \{-n, \ldots, m\}$ such that $A_i'^{rest} \neq \emptyset$, $A_i'^{rest} \subseteq B'_{\varphi(i)}$. Moreover, for each $A'_{ij} \in a_i$, $A'_{ij} = B'_{\varphi(ij)}$,
- $B'_0 = A_0'^{rest} + A_0^{O'} + A_0^{\nu}$,
- $\biguplus_{i \in (\{-n', \ldots, -1, 1, \ldots, m'\})} (B'_i + \biguplus_{j \in k_i^+} B'_{ij}) = (\biguplus_{i \in (\{-n', \ldots, -1, 1, \ldots, m'\})} (A'_i + \biguplus_{j \in l_i^+} A_{ij}'^{rest})) + O' + O^{\nu}$.

Hence, the firing is well defined, according to Def.4.2.4. Moreover, as $A_i^{rest} \leq^{\oplus} A_i'^{rest}$ for each $i \in \{-n, \ldots, m\}$, $O \leq^{\oplus} O'$ and $A_0^O \leq^{\oplus} A_0^{O'}$, if we define the assignation of the elements in O' to the different B'_i as done with the elements in O to the multisets B_i, it holds that $R_2 \sqsubseteq^{Pr} R_2'$.

□

Hence, by combining the two previous lemmas, we obtain compatibility.

Corollary 4.2.9 $\rightarrowtail$ is compatible with $\sqsubseteq^{Pr}$.

At this point, we have proved that ν-*aPTdPN* with the order $\sqsubseteq^{Pr}$ belong to the framework of Well Structured Transition Systems. Hence, in order to prove decidability of coverability, it only remains to prove that $\rightarrowtail$ has effective *Pred*-basis, that is, that we can compute $\min(\uparrow Pre(\uparrow R))$ for any region R. We handle the priced and timed transitions separately. Therefore, as done for ν-*lsPN*, we split *Pre* into $Pre_\Delta(R) = \{R' \mid \exists i \in 4^+ R' \overset{i}{\rightarrowtail} R\}$ and $Pre_t(R) = \{R' \mid R' \overset{t}{\rightarrowtail} R\}$, and we define $\overline{Pre}_\Delta$ and $\overline{Pre}_t$ for each $t \in T$, so that $Pre_\Delta(\uparrow R) = \uparrow \overline{Pre}_\Delta(R)$ and $Pre_t(\uparrow R) = \uparrow \overline{Pre}_t(R)$. The details of the proof are quite technical, so we prefer to omit them from this chapter. The interested reader can see these details in Appendix B.

Proposition 4.2.10 $\rightarrowtail$ *has effective Pred-basis.*

We have proved that the transition system defined by ν-*aPTdPN* with the order $\sqsubseteq^{Pr}$ belongs to the frame of Well Structured Transition Systems and has

effective *Pred*-basis. Hence, the coverability problem for ν-*aPTdPN* is decidable. Therefore, as a corollary of Prop. 4.2.5, we finally obtain the decidability of priced safety for ν-*PTdPN*.

Corollary 4.2.11 *Priced Safety is decidable for ν-PTdPN.*

Chapter 5

Resource Constrained Workflow Nets

Workflow nets have been widely used to model business processes and a rich theory for their analysis and verification has been developed [97, 92]. Basically, a workflow net is a Petri net with two special places: *in* and *out*. A run of a workflow net represents a process, which begins in a state represented by the marking with a token in *in* and empty elsewhere, and (hopefully) finishes in a state represented by a token in *out* and empty elsewhere. The fundamental verification problem for workflows is the soundness problem, which consists in deciding if the net can always reach this final state properly. In particular, the runs of sound workflows do not have deadlocks or livelocks. In [92] soundness is proved to be decidable for workflow nets.

Recent works study an extension of wf-nets, called resource-constrained wf-nets (rcwf-nets) [53, 44], which are wf-nets in which some places are dynamic (representing the processes that execute the workflow) and some are static (representing some shared resources). Following a terminology from OOP, a rcwf-net can be seen as the definition of a class, with its local and static attributes, represented by dynamic and static places, respectively. Then, the rcwf-net can be instantiated several times, but every instance must share the tokens in static places.

Even if a single instance of a rcwf-net is sound, several instances could deadlock because of static places. In [44] the authors define dynamic soundness, which essentially amounts to the condition stating that any number of instances running simultaneously can always reach the final state, in which all the tasks have been completed.

In both works, the authors consider rcwf-nets that do not create or consume static resources, that is, rcwf-nets that always return a global resource after using it. In particular, the behavior of a single instance of a rcwf-net is such that the number of tokens in the static places in the initial and final markings coincide. Under this assumption, the number of tokens in the static places is bounded by the number of tokens in the initial marking. The authors prove in [44] that dynamic soundness is decidable whenever there is only a single static place, that is, whenever there is a single type of global resources. Recently, [53] further studies the problem of dynamic soundness, extending the previous result to rcwf-nets with any number of static places, but considering a fixed number of initial resources (unlike in [44], in which the existence of a minimal number of resources for which the rcwf-net is sound is part of the problem). Under these assumptions, it is enough for the authors to study the absence of deadlocks.

As a formalism to model resource constrained business processes, it would be interesting to add prices and time to rcwf-nets, since in many occasions there are fundamental requirements regarding the timings and the costs of these systems, such as finishing each task before a deadline, or without expending too much money. That is why in the following chapter we will use some of the models defined in the previous sections in order to add costs and time to rcwf-nets. However, before that, we want to continue the study of dynamic soundness (without costs) under some more general conditions. In particular, we study the problem of dynamic soundness for rcwf-nets with any number of static places, and without restricting their behavior so that instances can terminate their task having created new global resources or having consumed some.

Before tackling dynamic soundness, let us fix some notations and the definition of workflow, for which we use the definition in [53].

Definition 5.0.6 (Workflow Petri net) *A workflow Petri net (shortly a wf-net) is a P/T net $N = (P, T, F)$ such that:*

- *there are $in, out \in P$ with ${}^\bullet in = \emptyset$ and $out^\bullet = \emptyset$,*
- *for each $p \in P \setminus \{in, out\}$, ${}^\bullet p \neq \emptyset$ and $p^\bullet \neq \emptyset$.*

The second condition intuitively states that all the places contribute to the completion of the task. Since we can always force that condition to be satisfied, we will from now on ignore it. Most works consider another additional condition for the definition of workflow net: for each $n \in P \cup T$ there is a path $in =$

$n_1, n_2, \ldots, n_j = n, \ldots, n_k = out$ with $F(n_i, n_{i+1}) > 0$ for $1 \leq i \leq k-1$. We will call this condition the *path property*, and we prefer to consider it apart from the definition of workflow in order to be able to talk about workflows with or without the path property, and study decidability issues about soundness for both of them. We denote by m_{in} the marking of N with a single token in in, and empty elsewhere. Analogously, m_{out} is the marking of N with a single token in out and empty elsewhere. There are several definitions of soundness of wf-nets in the literature. We will use one called weak soundness in [92]. A wf-net is *weakly sound* if for any marking reachable from m_{in} it is possible to reach m_{out}.

5.1 Asynchronous ν-Petri nets

As mentioned before, one can intuitively see each name in a ν-PN as a different process running in the net. Then, we can see a firing of a transition in which different names are involved as a synchronization between the corresponding processes.

Next, we prove that we can assume that actually each process can only synchronize with a global shared memory, so that a synchronization between two processes must be achieved via this shared memory. Technically, we will use ordinary black tokens to represent this global memory, and names to represent processes.

Definition 5.1.1 (Asynchronous ν-PN) *An asynchronous ν-PN is a ν-PN (P, T, F, H) such that:*

- *for each $t \in T$, either $Var(t) \subseteq \{\nu, \epsilon\}$ or $Var(t) \subseteq \{x, \epsilon\}$,*
- *for each $p \in P$, either $Var(p) = \{x\}$ or $Var(p) = \{\epsilon\}$.*

We call static places those $p \in P$ with $Var(p) = \{\epsilon\}$, and dynamic places those $p \in P$ with $Var(p) = \{x\}$. We will write $P = P_S \cup P_D$, with P_S the set of static places and P_D the set of dynamic places. Thus, we will disallow a situation in which $x, y \in Var(t)$ with $x \neq y$. Let us now see that asynchronous ν-PN can simulate ν-PN so that reachability is preserved.

Proposition 5.1.1 *Let N be a ν-PN, and m_0 a marking of N. There is an asynchronous ν-PN N' and a marking m'_0 of N' such that $m_0 \rightarrow^* \emptyset$ iff $m'_0 \rightarrow^* \emptyset$.*

Proof: Let $N = \langle P, T, F, H \rangle$ be a ν-PN with initial marking m_0. We build an asynchronous ν-PN $N' = \langle P', T', F', H' \rangle$ which simulates N as required. We simulate the firing of each transition $t \in T$ in an isolated way, by the sequential firing of several transitions satisfying the requirement in the definition of asynchronous ν-PN. For that purpose, we have a set of control places with plain tokens, whose adjacent arcs will be labelled by ϵ, and the rest of the arcs, involving instances of the original net, will be labelled by x. We consider that $P \subseteq P'$, so a marking m' of N' simulates the restriction of this marking to the places P in N. Before explaining how we simulate transitions, we want to guarantee that the simulation of a transition is done atomically, that is, whenever such a simulation is started, no other simulation can start until the former has finished. For that purpose, we add a place p_0, which will be marked whenever the simulation of a transition is allowed to start. Now, let us consider $t \in T$, and assume that $Var(t) = \{x_1, \ldots, x_n\}$ and for each $i \in n^+$ we have $F(t) = \{(p_{11}, x_1), \ldots, (p_{1k_1}, x_1) \ldots (p_{n1}, x_n), \ldots, (p_{nk_n}, x_n)\}$ and analogously $H(t) = \{(q_{11}, x_1), \ldots, (q_{1l_1}, x_1) \ldots (q_{n1}, x_n), \ldots, (q_{nl_n}, x_n)\}$. Then, we add to T' transitions $t_1, \ldots, t_n$, each one of them managing the consumption/addition of the tokens of a different instance. Moreover, for each $i \in (n-1)^+$ we add a place pt_i which is postcondition of t_i and precondition of pt_{i+1}. The cases of t_1 and t_n are handled apart because they are postcondition and precondition of p_0 instead of any pt_i. Formally, for each $i \in \{2, \ldots, n-1\}$ we have:

- $F(t_i) = \{(pt_{i-1}, \epsilon)\} + \{(p_{i1}, x), \ldots, (p_{ik_i}, x)\}$,
- $H(t_i) = \{(pt_i, \epsilon)\} + \{(q_{i1}, x), \ldots, (q_{il_i}, x)\}$.

Analogously, we have $F(t_1) = \{(p_0, \epsilon)\} + \{(p_{11}, x), \ldots, (p_{1k_1}, x)\}$, $H(t_1) = \{(pt_1, \epsilon)\} + \{(q_{11}, x), \ldots, (q_{1l_1}, x)\}$ and $F(t_n) = \{(pt_{n-1}, \epsilon)\} + \{(p_{n1}, x), \ldots, (p_{nk_n}, x)\}$, $H(t_n) = \{(p_0, \epsilon)\} + \{(q_{n1}, x), \ldots, (q_{nl_n}, x)\}$.

In this way, we simulate the firing of each transition t by the firings of transitions $t_1, \ldots, t_n$ and hence, if we start from the marking of N' $m_0 + \{(\bullet, p_0)\}$, we can simulate the firings in all the runs of N. Notice that this simulation can introduce deadlocks (for instance, when we fire t_1 but we cannot continue with t_2 due to absence of tokens), but it does preserve reachability.

□

Fig. 5.1 illustrates the previous construction when $Var(t) = \{x, y, \nu\}$.

As reachability is undecidable for ν-PN we obtain the following corollary.

Corollary 5.1.2 *Reachability of $\emptyset$ is undecidable for asynchronous ν-PN.*

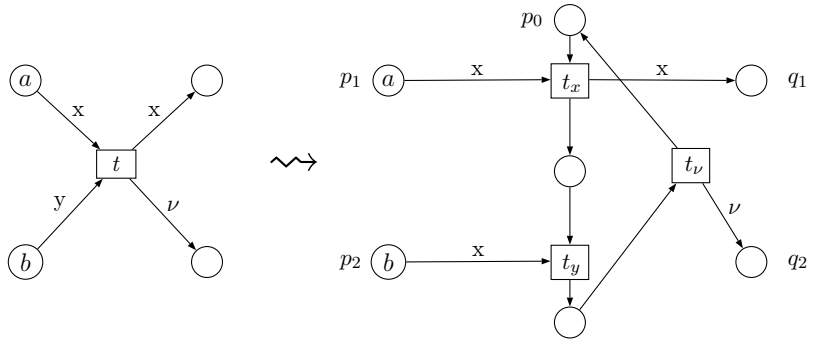


Figure 5.1: Simulation of a transition t

5.2 Resource-constrained workflow nets

In [44] rcwf-nets are defined. We propose here a slightly different presentation of rcwf-nets, though equivalent. We directly define rcwf-nets using asynchronous ν-PN, in order to shorten the gap between rcwf-nets and ν-PN, and use its (un)decidability properties to obtain results about dynamic soundness.

Given a ν-PN $N = (P, T, F, H)$ and $x \in Var$ we define the P/T net $N_x = (P, T, F_x)$, where for each $p \in P$ and $t \in T$, $F_x(p,t) = F(t)(p,x)$ and $F_x(t,p) = H(t)(p,x)$. Moreover, for $Q \subseteq P$, we define $F|_Q$ as the multiset over $Q \times T$ such that, for each $q \in Q$ and $t \in T$, $F|_Q(q,t) = F(t)(q,x)$. Intuitively, we select the arcs of the ν-PN restricted to the ones adjacent to Q and labelled by x. Analogously, we define $H|_Q \in (T \times Q)^{\oplus}$ such that, for each $q \in Q$ and $t \in T$, $H|_Q(t,q) = H(t)(q,x)$. We are now ready to define rcwf-nets.

Definition 5.2.1 (Resource-constrained Wf-net) *A* resource-constrained wf-net *(or rcwf-net for short) is an asynchronous ν-PN $N = (P, T, F, H)$ such that:*

- *for all $t \in T$, and $\nu \in \Upsilon$, $\nu \notin Var(t)$,*
- *$N_p = (P_D, T, F|_{P_D} + H|_{P_D})_x$ is a wf-net.*

N_p is the P/T net obtained by removing static places, which we call *production net* of N. Then, a rcwf-net is an asynchronous ν-PN that does not create new tokens (because no variable $\nu \in \Upsilon$ labels any arc) and such that its production net is a wf-net. In particular, it contains two special places *in* and *out* given by the definition of wf-nets. When there is no confusion we will simple refer to these places as *in* and *out*, respectively. In figures, we represent static places by bold circles, and dynamic places by normal circles. Moreover, for simplicity, sometimes

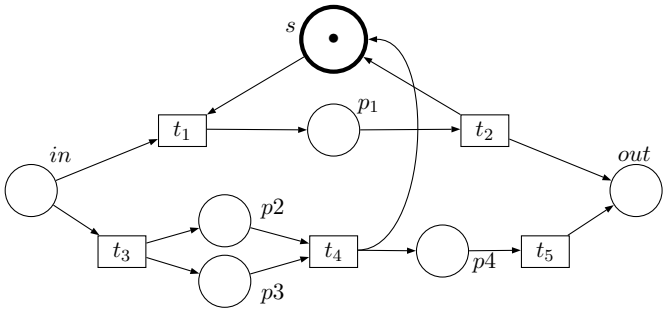


Figure 5.2: A proper rcwf-net N

we omit the labels x and ϵ from the arcs. For example, Fig. 5.2 shows a rcwf-net with only one static place s.

Definition 5.2.2 (Initial marking of a rcwf-net) *Let $N = (P, T, F, H)$ be a rcwf-net and $m_0 \in P_S^{\oplus}$. For any $k \geq 0$, we define m_0^k, as the marking of N given by:*

- *$m_0^k(s)$ contains $m_0(s)$ black tokens, for each $s \in P_S$,*
- *$m_0^k(in)$ contains k pairwise different names,*
- *$m_0^k(d)$ is empty for every $d \in P_D \setminus \{in\}$.*

Moreover, for m_0^k we define the set of final markings $\mathcal{M}_{out}^k$ that contain the same k names in out, and empty in the rest of the dynamic places.

Notice that in the final markings we are not fixing the amount of tokens in static places, unlike in [44, 53].

Definition 5.2.3 (Dynamic soundness) *Let $N = (P, T, F)$ be a rcwf-net and $m_0 \in P_S^{\oplus}$. We say N is dynamically sound for m_0 if for each $k \geq 0$ and for each m reachable from m_0^k, we can reach some marking in $\mathcal{M}_{out}^k$.*

5.3 Undecidability result

In this section we prove undecidability of dynamic soundness for rcwf-nets by reducing reachability for asynchronous ν-PN, which is undecidable, to it.

For this purpose, given an asynchronous ν-PN N, an initial marking m_0 of N (which we can assume to contain a single token in a given place i), we are going to construct a rcwf-net N' which is dynamically sound if and only if the empty

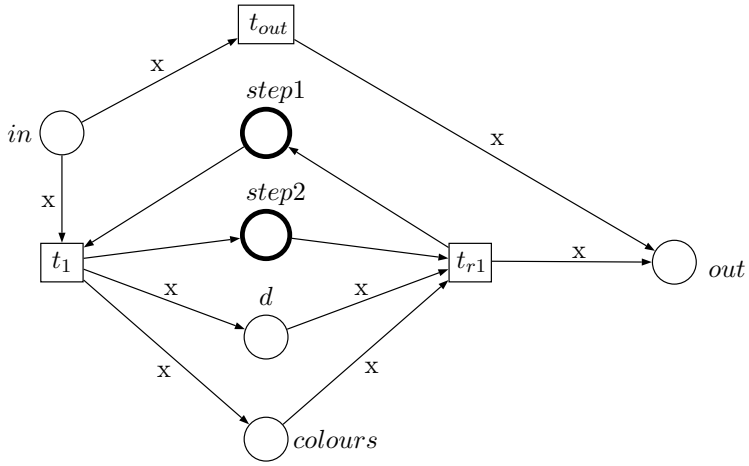


Figure 5.3: Step 1

marking is not reachable from m_0. Intuitively, the runs of N' will be divided into four steps: In the first step, the net gets ready for the simulation; in the second step, the initial marking m_0 of N is set; the third step simulates N; and finally, the last step is intuitively used to force that $\emptyset$ is not reachable if and only if N' is dynamically sound.

Let us explain with detail the four steps. In order to control in which step we are, we consider four static places $step1$, $step2$, $step3$ and $step4$, that will be marked in mutual exclusion. Initially, $step1$ is marked.

5.3.1 Step 1: getting ready

First of all, as we want to build a rcwf-net, we add two special places in and out. We add a transition t_{out} which can move a token from in to out. This transition does not have any other precondition, so that it can be fired in any of the steps.

We will also consider two dynamic places, d and $colours$. The purpose of d will be explained in the last step. The place $colours$ will store all the colours that we will use in the simulation of N, so that each transition in the construction which takes a token from in, will add it to $colours$. We store all the colours in order to be able to add them to out even if N consume all the tokens of some color. We need the place $colours$ because N could erase some names, but we cannot do this in N' without being dynamically unsound.

In this first step, a transition t_1 is fired, removing a token from in and adding it to the two dynamic places d and $colours$. The purpose of placing a token in d will be explained later, in the last step. It also moves the token from $step1$ to

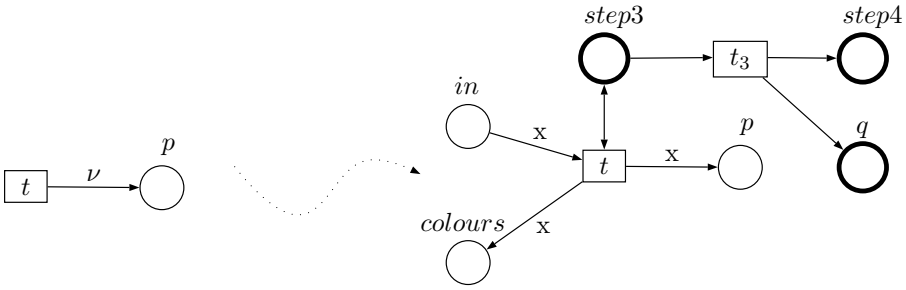


Figure 5.4: Step 3

*step*2, thus moving on to the next step.

Finally, we need the firing of t_1 to be "reversible" (for the case in which we have a single name in *in*). Therefore, we add a new transition t_{r1} which moves a token from *step*2 to *step*1, removes the tokens in *colours* and d, and adds a token of the same color to *out* (not to *in*, since it cannot have incoming arcs). Fig. 5.3 illustrates the first step.

5.3.2 Step 2: setting the initial marking

In order to simulate the behavior of N, we consider in N' the set of places of N. In this step we set the initial marking, which consists only of a name in the place of N that we call i. Therefore, we take a token from *in* and put it both in i and in *colours*. Moreover, we move the token from *step*2 to *step*3.

5.3.3 Step 3: simulating N

In this step we simulate the behavior of N. Since N is an asynchronous ν-PN, it only uses variables x, ν and ϵ. Since N' is a rcwf-net, we have to simulate the creation of new names without using ν. We do it analogously as in the previous steps, by taking from *in* a name whenever one must be created, and placing it both in *colours* and whatever places pointed by arcs labeled by ν. Since all the names contained in the place *in* are different, this is a correct simulation of the creation of a fresh name.

It may be the case that at some point there are no more tokens in the place *in*, so that no more name creations can be simulated. Therefore, a run of N' with k different names in the place *in* simulates a run of N in which at most k names are used (actually, $k-1$ because of the name that is put in d). Notice that the dynamic soundness has to do with the behavior of a rcwf-net from any initial

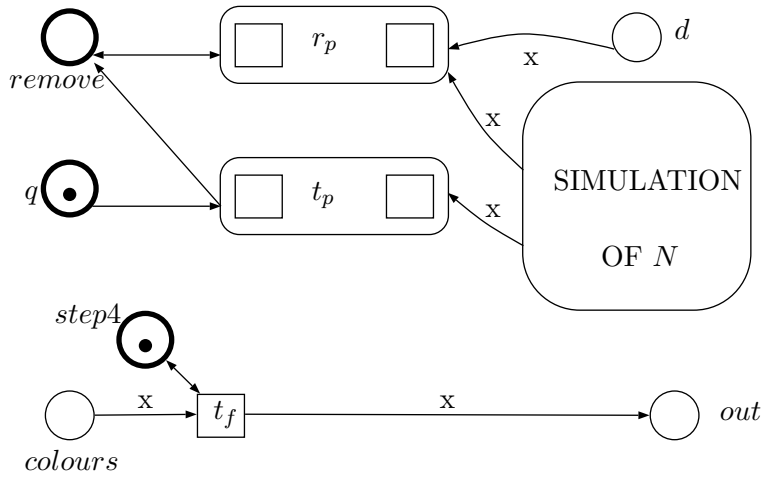


Figure 5.5: Step 4

marking, so that all the behaviors of N will be considered.

In this step we add *step*3 both as precondition and postcondition of any transition in N, so that transitions in N can only be fired in this step. At any point, we can fire a transition t_3 that moves the token from *step*3 to *step*4, thus finishing the simulation of N. Moreover, it also puts a black token in a new static place q, whose purpose we will explain later. Fig. 5.4 shows the simulation of a transition with a ν.

5.3.4 Step 4: reducing reachability to dynamic soundness

When the fourth step starts, there is a name in d, a black token in *step*4 (which will stay there until the end of the execution of N') and in q, the set of names that have been used along the execution of the rcwf-net is stored in *colours* and the places of N are marked with a marking which is reachable in N.

We add a transition t_f, which can move all the tokens from *colours* to *out*, and with *step*4 both as precondition and postcondition, so that it cannot be fired until this step starts.

We want to force N' to be dynamically unsound whenever $\emptyset$ is reachable. Since we can move names directly from *in* to *out*, we need to build a marking from which it is not possible to remove names from places different from *out*.

We add to N' a transition t_p for each place p of N. When q is marked, there is a choice between all the transitions t_p, each of which removes a token from p,

and puts a black token in a static place *remove*. Intuitively, we are only able to fire some t_p if the current marking of N is not $\emptyset$. Otherwise, if t_3 was fired exactly from $\emptyset$, then no transition t_p can be fired.

If we are able to fire some t_p then we have a token in *remove*. In that case, we can fire transitions r_p for each dynamic place p (different from *colours*, *in* and *out*), that removes a token from p, and puts the token back to *remove*. Therefore, if *remove* is marked, we can empty every dynamic place different from *colours*, *in* and *out*. In particular, the firing of r_d is the only way to remove the token in d. Fig. 5.5 sketches how the fourth step is performed.

5.3.5 Undecidability

Now we are ready to prove that the previous construction reduces reachability for asynchronous ν-PN to dynamic soundness for rcwf-nets.

Proposition 5.3.1 *Given a ν-PN N with initial marking m_0, the rcwf-net N' built is dynamically sound if and only if $\emptyset$ is not reachable from m_0 in N.*

Proof: First, let us suppose that $\emptyset$ is reachable from m_0 in N. Let n be the number of different names created in some run that reaches $\emptyset$. If we consider the net N' with $n+1$ or more instances (that is, with at least $n+1$ different names in the place *in*), then we can reach a marking m' of N' in which the places of N are unmarked, the names that have been used in the computation are stored in *colours*, d is marked by a color and *step*4 and q are marked with black tokens. From this marking, we cannot fire any of the t_p transitions, and therefore, we cannot remove the token from q. Therefore, *remove* cannot be marked, which is the only way in which the name in d can be removed. Summing up, from the initial marking with $n+1$ different names in *in* we have reached a marking from which we cannot reach a final marking of N' (that in which the only marked dynamic place is *out*), so that N' is not dynamically sound.

Conversely, let us suppose that $\emptyset$ is not reachable. We have to prove that for each $k \geq 0$ and for each m reachable from m_0^k, we can reach some marking in $\mathcal{M}_{out}^k$. Let us consider several cases, depending on which step the considered marking is in.

- If *step*1 is marked in m then all the names are either in the place *in* or in *out*. Therefore, we can fire t_{out} repeatedly, transferring all the tokens in *in* to *out*, and we are done.

- If $step2$ is marked in m we can fire t_{r1}, reaching a marking in which $step1$ is marked, so we can apply the previous case.
- If $step3$ is marked in m we can fire t_3, reaching a marking in which $step4$ is marked. We discuss this case next.
- If $step4$ is marked in m we can fire t_f repeatedly, putting all the names that have been used by the construction in out, thus emptying $colours$. Moreover, we can fire t_{out} repeatedly, moving all the tokens which remain in in to out. Therefore, all the tokens that initially were set in in, are set in out, so we only have to prove that we can empty the other dynamic places. If $step4$ is marked then there must be a token in q or $remove$. If the token is in q, since $\emptyset$ is not reachable, there is some name in some place p of N. Therefore, we can fire the transition t_p from m, reaching a marking in which $remove$ is marked. Finally, if $remove$ is marked in m, we can remove all the tokens from the dynamic places different from $colours$, in and out, reaching the desired marking.

□

The previous result proves that reachability of the empty marking in asynchronous ν-PN, which is undecidable, can be reduced to dynamic soundness for rcwf-nets. Therefore, we finally obtain the following result:

Corollary 5.3.2 *Dynamic soundness is undecidable for rcwf-nets.*

5.4 Decidability of dynamic soundness for a subclass of rcwf-nets

We have proved that dynamic soundness is undecidable in general. However, if we consider more restrictive requirements for our rcwf-nets, dynamic soundness turns decidable. In the literature, several notions of rcwf-nets and soundness have been studied, most of them being more restrictive than our general definition. In particular, in [44] the authors consider wf-nets which satisfy the following condition, which we have not required: for each node n, there are paths from in to n and from n to out. We are going to consider a less restrictive requirement, namely that every transition has some dynamic postcondition. In that case, and considering some very reasonable requirements, dynamic soundness is decidable

even if shared resources can be consumed or created by instances. This reasonable requirement is the following: when a single instance is given arbitrarily many global resources, then it evolves properly. This is equivalent to just removing static places in the case of one instance.

We are now ready to define our subclass of rcwf-nets:

Definition 5.4.1 (Proper rcwf-net) *We say that a rcwf-net $N = (P, T, F, H)$ is a* proper *rcwf-net if the two following conditions hold:*

- *for each $t \in T$, $t^\bullet \cap P_D \neq \emptyset$,*
- *the production net N_p of N is weakly sound.*

Intuitively, the behavior of N_p represents the maximal behavior of each instance of N, meaning that it is not constrained by the presence (or absence) of resources. Hence, the behavior N_p is the behavior of each instance in N when providing an unbounded number of resources. In particular, if m is a reachable marking of a rcwf-net N, then the markings of N_p obtained by projecting m to each of the names in m are all reachable too.

In [44, 53] other classes more restricted than proper rcwf-nets are defined.[1] However, the previous conditions are enough for our decidability result, and indeed our requirement can be deduced from the conditions required in [44, 53].

Lemma 5.4.1 *The production net N_p of a proper rcwf-net N is bounded.*

Proof: Let us suppose that N_p is sound and unbounded (assuming the initial marking m_0^1). Then, there are markings of N_p, m_1, m_2, and m_1' such that $m_0^1 \rightarrow^* m_1 \rightarrow^* m_2 = m_1 + m_1'$ with m_1' non empty. Since N_p is sound, $m_1 \rightarrow out$, so that $m_2 = m_1 + m_1' \rightarrow^* out + m_1'$. Again, by soundness of N_p, it must be the case that $out + m_1' \rightarrow^* out$. Since $out^\bullet = \emptyset$, it must be the case that $m_1' \rightarrow^* \emptyset$, but this is not possible because N is proper (and, in particular, all the transitions of N_p have postconditions).

□

Actually, in the proof of decidability of dynamic soundness for proper rcwf-nets, we only need that the production net is bounded (and boundedness is decidable for P/T nets). By the previous result, we know that the production net of

[1] E.g., by demanding that there are paths from *in* to every node, and from every node to *out*.

a proper rcwf-net is bounded, but even if our rcwf-net is not proper, we can still check whether its production net is bounded, in which case our proof still holds.

We reduce dynamic soundness to the so-called home space problem in P/T nets, which is decidable. Hence, we first introduce this problem.

Definition 5.4.2 (Home space problem for P/T nets) *Given a P/T net N, a marking m_0 and a set $\mathcal{H}$ of markings of N, we say that $\mathcal{H}$ is a home space if for every reachable marking m, there is a marking $m' \in \mathcal{H}$ reachable from m.*

The problem of deciding whether a linear set of markings is a home space is decidable too [27, 34]. A linear set is a set of vectors that can be obtained as linear combinations of a given set of vectors. More precisely, and considering that markings can be seen as vectors, a marking m and a finite set of markings $\{m_1, ..., m_n\}$ define the linear set of markings $\mathcal{L} = \{m + \sum_{i=1}^{n} \lambda_i * m_i \mid \lambda_i \in \mathbb{N}\}$.

Let us explain intuitively how the construction works. It is similar to a construction used in [53]. Given a proper rcwf-net N, we know that N_p is bounded. Then, we can consider the state machine associated to the reachability graph of N_p. More precisely, if m is a reachable marking in N_p, then we will consider a place also denoted by m. A token in m stands for an instance of N in marking m. Notice that this is correct because all the markings reachable in N must be reachable in N_p (after projecting). So far, the construction is like in [53]. Moreover, the static places will be considered as places of the new net too, and will be pre/postconditions of the transitions we add, in the same way as they were in the original net.

Finally, we have to consider one more place *src* in order to set the initial number of instances that we are going to consider for the net. Let us denote by $\mathcal{R}(N)$ the set of markings reachable in a wf-net net N from m_{in}. Now we are ready to define the construction which will let us prove the decidability of dynamic soundness.

Definition 5.4.3 *Let $N = (P, T, F, H)$ be a proper rcwf-net and $m_0^s \in P_S^{\oplus}$. We define the P/T net $N^{tr} = (P^{tr}, T^{tr}, F^{tr})$ as follows:*

- $P^{tr} = P_S \cup \mathcal{R}(N_p) \cup \{src, ok\}$,
- $T^{tr} = \{(m_1, t, m_2) \in \mathcal{R}(N_p) \times T \times \mathcal{R}(N_p) \mid m_1 \xrightarrow{t} m_2 \text{ in } N_p\} \cup \{new, stop\}$,
- F^{tr} *is such that:*
  - $F^{tr}(m_1, (m_1, t, m_2)) = F^{tr}((m_1, t, m_2), m_2) = 1$,

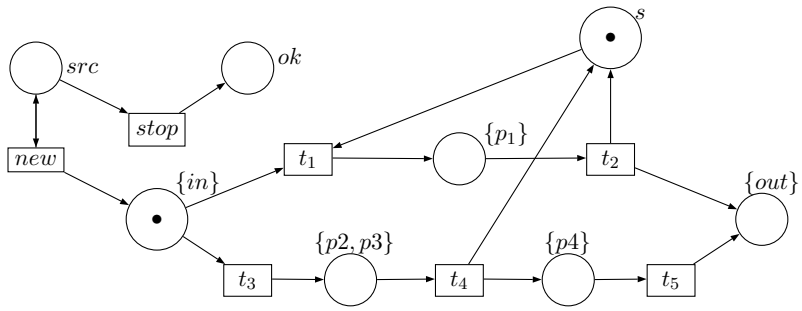


Figure 5.6: N^{tr} obtained by applying Def. 5.4.3 to N in Fig. 5.2 (omitting the arcs from *ok*)

- $F^{tr}(src, stop) = F^{tr}(stop, ok) = 1$,
- $F^{tr}(src, new) = F^{tr}(new, src) = F^{tr}(new, in) = 1$,
- $F^{tr}(ok, (m_1, t, m_2)) = F^{tr}((m_1, t, m_2), ok) = 1$,
- *If* $s \in P_S$ *and* $(m_1, t, m_2) \in T^{tr}$ *then* $F^{tr}((m_1, t, m_2), s) = H(t)(s, \epsilon)$ *and* $F^{tr}(s, (m_1, t, m_2)) = F(t)(s, \epsilon)$,
- $F^{tr}(x, y) = 0$, *otherwise.*

The initial marking of N^{tr} *is* m_0^{tr}, *given by* $m_0^{tr}(src) = 1$, $m_0^{tr}(m) = 0$ *for* $m \in \mathcal{R}(N_p)$ *and* $m_0^{tr}(s) = m_0^s(s)$ *for* $s \in P_S$.

Figure 5.6 shows the previous construction for the net in Fig. 5.2. Note that N^{tr} is finite because N_p is bounded, so that it can be effectively computed. Intuitively, N^{tr} creates by means of transition *new* several instances in its initial state, after which if fires *stop*, marking place *ok*, which is a precondition of the rest of the transitions, so that from then on they can be fired.[2] Each token in a place $m \in \mathcal{R}(N_p)$ of N^{tr} represents an instance of N, running concurrently with other instances and sharing the resources in the static places with them. Therefore, the net will simulate runs of as many instances of the original net as times the transition *new* has been fired. Let us define a correspondence between the markings of N and the markings of N^{tr}.

Definition 5.4.4 *Given a marking* m *of* N, *we define the marking* m^{tr} *of* N^{tr} *as follows:* $m^{tr}(src) = 0$, $m^{tr}(ok) = 1$, $m^{tr}(s) = m(\bullet, s)$ *for* $s \in P_S$, *and for* $m' \in \mathcal{R}(N_p)$ $m^{tr}(m') = |\{a \in Id(m) \mid m(a, p) = m'(p)\ \forall p \in P_D\}|$, *that is, the number of instances in marking* m' *in* m.

[2] Actually, the construction still works without place *ok*, though it simplifies the forthcoming explanations.

Notice that all the markings reachable in N^{tr} with *ok* marked are of the form m^{tr} for some marking m reachable in N. The following result is trivial by construction of N^{tr}.

Lemma 5.4.2 *$m_0^k \to^* m$ in N if and only if $m_0^{tr} \stackrel{new^k \cdot stop}{\longrightarrow} (m_0^k)^{tr} \to^* m^{tr}$. Moreover, all the computations in N^{tr} start by firing new $k \geq 0$ times, possibly followed by stop, in which case $(m_0^k)^{tr}$ is reached.*

Finally, we are ready to prove that this construction reduces the dynamic soundness problem for proper rcwf-nets to the home space problem for P/T nets. Given $p \in P$, we denote by e_p the marking given by $e_p(p) = 1$ and $e_p(q) = 0$ for $q \neq p$.

Proposition 5.4.3 *Let N be a proper rcwf-net. N is dynamically sound if and only if the linear set $\mathcal{L}$ generated by $\{out\} \cup \{e_s \mid s \in P_S\}$ is a home space for N^{tr}.*

Proof: We start by remarking that $\mathcal{L}$ contains markings with any number of tokens in *out* and in static places, and empty elsewhere. Notice also that each transition different from *new* and *stop* has exactly one precondition in $\mathcal{R}(N_p)$ and one postcondition in $\mathcal{R}(N_p)$. Therefore, after the firing of *stop*, the total number of tokens in places in $\mathcal{R}(N_p)$ remains constant. Therefore, if *new* is fired k times and a marking in $\mathcal{L}$ is reached, then necessarily this marking has k tokens in *out*. Finally, notice that $m \in \mathcal{M}_{out}^k$ iff $m^{tr} \in \mathcal{L}$ and it contains exactly k tokens in *out*.

Let us first suppose that N is not dynamically sound. Then, there is a $k > 0$ and a marking m reachable from m_0^k from which no marking in $\mathcal{M}_{out}^k$ is reachable. By Lemma 5.4.2, the marking m^{tr} is reachable after firing *new* k times. Then, from m^{tr} no marking in $\mathcal{L}$ can be reached. Indeed, if some marking m'^{tr} in $\mathcal{L}$ is reached from m^{tr} it has necessarily k tokens in *out* and again by Lemma 5.4.2, $m' \in \mathcal{M}_{out}^k$ is reached in N, contradicting our first hypothesis. Then, $\mathcal{L}$ is not a home space and we conclude this implication.

Reciprocally, let us assume that $\mathcal{L}$ is not a home space. Then, there is a reachable marking of N^{tr} from which no marking of $\mathcal{L}$ can be reached. Let us suppose that this marking is of the form m^{tr} (otherwise, we consider the marking obtained after firing *stop*, and no marking of $\mathcal{L}$ can be reached from it). Let us suppose that there are k tokens in places in $\mathcal{R}(N_p)$ in m^{tr}. Then, by Lemma 5.4.2 and the previous remarks (analogously to the previous case) no marking in $\mathcal{M}_{out}^k$ can be reached from m, so that N is not dynamically sound.

□

Finally, as the home space problem is decidable for linear sets of markings of P/T nets [27], we obtain the following result:

Corollary 5.4.4 *Dynamic soundness for proper rcwf-nets is decidable.*

Chapter 6

Resource Constrained Workflow Nets with Time and Prices

In this chapter we finally extend wf-nets and rcwf-nets with time and prices. More precisely, we are going to add storage and firing costs to them in a similar way as we did for ν-$PTdPN$. We are going to do it in two different ways. First, we define priced workflow nets and priced resource-constrained workflow nets, for which we suppose that time elapses only when transitions are fired, that is, the firing of transitions simulate the performance of activities that take some time, and no time can elapse between one activity is accomplished and another activity starts. Hence, both storage and firing costs are produced when firing transitions. On the other hand, we also define priced-timed resource-constrained workflow nets, which represent a model of costs as in ν-$PTdPN$, that is, basically, time may elapse at any time, so storage costs are produced when time elapses and firing costs are produced when transitions are fired.

For both models of costs, we will define and study several soundness problems. Essentially, a net is sound if we can always reach the final marking, without spending more than a given budget. First, we study the decidability of this problem for Priced workflow nets. However, since for rcwf-nets we must consider the behavior of an arbitrary number of instances, the definition of soundness for our new model is not so straightforward. We consider a parametric definition: For any run, we collect the prices of every instance in the run, so that safety and soundness are parametric in the way in which local prices are aggregated to obtain

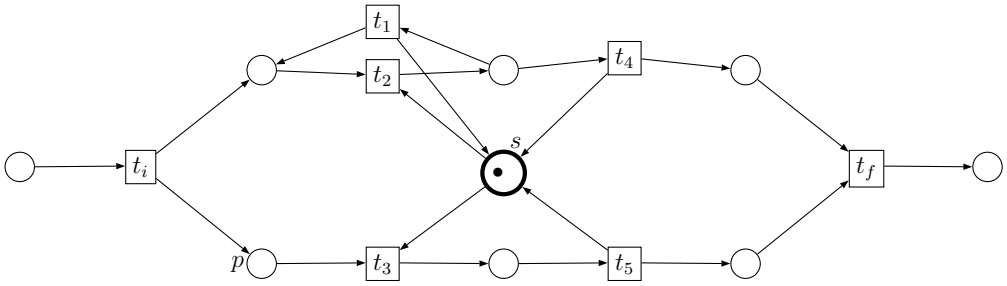


Figure 6.1: prcwf-net representing the practical exercise of the Chemistry degree

a global price. The definition is open to many different variants. We study several such variants: the maximum, the sum, the average and the discounted sum.

Before defining our models, let us introduce a motivating example. We consider the following real scenario, taken from the Chemistry Degree of the Complutense University, where every student of the second course must do a practical exercise which consists in synthesizing two different chemical components and comparing their properties. Each student has to perform several steps for which they need to use some devices. We can model the procedure that each student has to follow as a wf-net, and the interaction of all the students (with limited resources) as the rcwf-net depicted in Fig. 6.1. We model each step of the exercise as a transition, and tokens in places represent students who are ready to perform the next step. In most of the steps, the use of the corresponding device has a cost, which corresponds to a *firing costs*. Hence, each transition t will have a firing cost $FC(t)$ associated. However, these are not the only costs we need to take into account. When all the devices of a certain kind are being used at the same time, there may be students who need to wait to use these devices and the components they have made until this moment, may go wrong because of the delay if they do not keep them at some specific conditions, as a specific temperature, for example. Therefore, there are costs that depend on the number of students that are waiting to use the devices. These costs will correspond to *storage costs*, that is, costs produced when a transition is fired and there are some tokens in some specific places. Therefore, for each transition t and place p, there is a storage cost $SC(p,t)$ associated to them. In our example, performing steps two (transition t_2) and three (transition t_3) costs 1, so that $C(t_2) = C(t_3) = 1$. In the exercise, step two can possibly go wrong, in which case the students need to repeat it, which is possible thanks to t_1. The device used in those two steps is the same, and there is only one such device, which is modeled by the static place s with initially one

token. Moreover, the cost of a student waiting in p for the necessary device is 1, and therefore $S(p, t_2) = S(p, t_3) = 1$. Then, the price for the university of a student performing the whole exercise is the sum of the costs of using each device and the costs that come from other students waiting for the devices she is using. Then, the university could be interested in setting a bound for:

- The total amount of money spent in all students in a class;
- The money spent on each student;
- The average of the amounts spent in each student.

This corresponds to the study of priced soundness of the corresponding model for the different price predicates.

6.1 Priced resource-constrained workflow nets

We first define a priced extension of wf-nets, adding prices supposing that time elapses only when transitions are fired. Essentially, we add two functions to a wf-net, defining the price of the firing of each transition, and the cost of storing tokens when a transition is fired, respectively, following the cost model in [9].

6.1.1 Priced workflow-nets

Definition 6.1.1 (Priced workflow net) *A* priced workflow net *(pwf-net) with* price arity $k \geq 0$ *is a tuple* $N = \langle P, T, F, C, S \rangle$ *such that:*

- $\langle P, T, F \rangle$ *is a wf-net, called the underlying wf-net of* N*,*
- $C : T \to \mathbb{Z}^k$ *is a function assigning firing costs to transitions, and*
- $S : P \times T \to \mathbb{Z}^k$ *is a function assigning storage costs to pairs of places and transitions.*

Notice that costs may be negative. The behavior of a pwf-net is given by its underlying wf-net. In particular, adding prices to a wf-net does not change its behavior, as the costs are not a precondition for any transition. That is the main difference between resources and prices. Indeed, although firing costs could be seen as resources, storage costs cannot, because they depend not only on the transitions which are fired, but also on the number of tokens in the rest of the places when the transitions are fired. Let us define the price of a transition.

Definition 6.1.2 (Price of a run) *Let t be a transition of a pwf-net enabled at a marking m. We define $\mathcal{P}(t, m)$, the* price of the firing of t at m, *as*

$$\mathcal{P}(t, m) = C(t) + \sum_{p \in m - {}^\bullet t} S(p, t)$$

Then, the price of a run $r = m_1 \xrightarrow{t_1} m_2 \xrightarrow{t_2} m_3 \dots m_n \xrightarrow{t_n} m_{n+1}$ *of a pwf-net is* $\mathcal{P}(r) = \sum_{i=1}^{n} \mathcal{P}(t_i, m_i)$.

Notice that in the definition of $\mathcal{P}(t, m)$ the term $m - {}^\bullet t$ is a multiset, so that if a place p appears twice in it then we are adding $S(p, t)$ twice in turn. It can be seen that firing costs can be simulated by storage costs, though we prefer to keep both to follow the approach in [9]. However, storage costs cannot be simulated by firing costs, since the former are marking dependent, while the latter are not. Next, we define safety of a pwf-net with respect to prices.

Definition 6.1.3 (b-p-safety) *Given $b \in \mathbb{N}_\omega^k$, we say that a pwf-net is* b-p-safe *if for each run r reaching m_{out}, $\mathcal{P}(r) \leq b$.*

Therefore, a pwf-net is b-p-safe if all the runs that reach the final marking cost less than the given budget. Next, we define soundness for pwf-nets.

Definition 6.1.4 (b-p-soundness) *Given $b \in \mathbb{N}_\omega^k$, we say that a pwf-net is* b-p-sound *if from each marking m, reachable from m_{in} via some run r_1, we can reach m_{out} via some run r_2 such that $\mathcal{P}(r_1 \cdot r_2) \leq b$.*

Intuitively, for a pwf-net to be sound we need to be able to reach the final marking at any point with a price that does not exceed the budget $b \in \mathbb{N}_\omega^k$. It is easy to see that a pwf-net is b-p-sound iff it is weakly sound and b-p-safe. However, as we prove next, the latter is undecidable when dealing with negative costs.

Proposition 6.1.1 *b-p-safety is undecidable, even for priced workflow nets whose underlying workflow satisfies the path property.*

Proof: We reduce the cost-threshold-reachability problem for PPN with negative costs, and price arity 1, which is undecidable [9]. A PPN with arity 1 is a P/T net $\langle P, T, F \rangle$, endowed with a function $C : P \cup T \to \mathbb{Z}$ associating costs to transitions and places. Moreover, T is the disjoint union of T_0, the set of *instantaneous* transitions, and T_1, the set of *timed* transitions. The cost of firing $t \in T_0$ in any marking is just $C(t)$. The cost of $m \xrightarrow{t} m'$ with $t \in T_1$ is

$C(t) + \sum_{p \in P} C(p) \cdot m(p)$. The cost of a run is the sum of all the transitions costs in it. The cost-threshold-reachability problem consists in, given m_f and $b \in \mathbb{N}$, decide whether there is a run σ with $m_0 \xrightarrow{\sigma} m_f$ such that $C(\sigma) \leq b$. It is proved in [9] that this problem is undecidable.

Given a PPN $N = \langle P, T, F, C \rangle$, a final marking m_f and $b \in \mathbb{N}$, let us build a pwf-net N' as follows.

First, we add new places, p_i, p_0, *in* and *out*, and transitions t_i, t_0 and t_f. Transition t_0 sets the initial marking of N, and t_f has m_f as precondition and puts a token in *out*. Places p_i and p_0 and transition t_i are added in order to make N' satisfy the second and third conditions of the definition of workflow net. In order to satisfy the path property, at the beginning of each run we put a token in each place of the net, and remove them by firing t_i and t_0. Moreover, p_0 will be connected to each transition of the net and will be emptied in the final step, when t_f is fired. More precisely:

- Transition t_i takes a token from *in*, and puts a token in each place of N' except for p_0, *in* and *out*, connecting each place with *in*.
- We set each place of N' except for *in*, p_0 and *out* as a precondition of t_0, and p_0 and every place of m_0 as a postcondition of t_0, connecting each place except for *out* to p_0.
- We make p_0 be a precondition and postcondition of each $t \in T$, connecting each transition to p_0.
- Finally, we set p_0 and every place of m_f as a precondition of t_f, connecting each place and transition of the net with *out*.

The new places have no storage cost, t_0 has firing cost $b + 1$ and t_f and t_i have firing cost 0. For every $t \in T$ we take $-C(t)$ as the firing cost of t in N'. Moreover, if t is an instantaneous transition we set $S(p, t) = 0$ for every $p \in P$, and if it is a timed transition we take $S(p, t) = -C(p)$ for every $p \in P$.

By construction, if r is a run in N with cost c, then $t_i t_0 \cdot r$ is a run in N' with cost $b + 1 - c$. Let us see that there exists a run r of N such that $m_0 \xrightarrow{r} m_f$ with $C(r) \leq b$ iff N' is not 0-p-safe. Let r be a run such that $m_0 \xrightarrow{r} m \geq m_f$ and $C(r) \leq b$. Let us call $r' = t_i t_0 r t_f$ the corresponding run of N'. Then, $C(r') = 1 + b - C(r) \geq 1$. Therefore, N' is not 0-p-safe. Conversely, suppose that for every run r of N with $m_0 \xrightarrow{r} m_f$, $C(r) > b$. Then, for every run r' of N'

reaching m_{out}, necessarily of the form $t_i t_0 r t_f$, $C(r') = 1 + b - C(r) \leq 0$. Therefore, N' is b-p-sound.

□

Despite the previous result, we can prove that b-p-soundness is decidable.

Proposition 6.1.2 *b-p-soundness is decidable.*

Proof: Let N be a pwf-net. If the underlying wf-net N' of N is not weakly sound, then there is a reachable marking from which m_{out} is not reachable, and therefore N is not b-sound. Let us suppose that N' is weakly sound. Then, N' is bounded [94] and therefore we can build its reachability graph, which is finite. More precisely, we build the graph whose nodes are the reachable markings of N', and there is an arc connecting two markings m_1 and m_2 if and only if $m_1 \xrightarrow{t} m_2$ for some $t \in T$, labeled by $\mathcal{P}(t, m_1)$. In that way, the price of a run of N corresponds to the length of the corresponding path in the reachability graph. Then, since N' is weakly sound, N is not b-sound if and only if there is a run from m_{in} to m_{out} with a price which is not smaller than b, that is, if and only if there is a path from m_{in} to m_{out} in the reachability graph with a length c, and $i \in \{1, ..., k\}$, such that $c[i] > b[i]$. Therefore, in this case b-p-soundness can be reduced to computing the longest path from m_{in} to m_{out} in the finite reachability graph, which can be done polynomially in the number of states, even with negative labels [24].

□

To conclude this section, notice that if a pwf-net is b-p-safe (b-p-sound) then it is also b'-p-safe (b'-p-sound) for any $b' > b$, so that the set $\mathcal{B}(N) = \{b \in \mathbb{N}^k_\omega \mid N$ is b-p-safe (b-p-sound)$\}$ is an upward-closed set. In this situation, we can apply the Valk & Jantzen theorem:

Theorem 6.1.3 ([89]) *Let V be an upward-closed set. We can compute a finite basis of V if and only if for each $v \in \mathbb{N}^k_\omega$ we can decide whether $\downarrow v \cap V \neq \emptyset$.*

Therefore, we can compute a finite basis of the set $\mathcal{B}(N)$, i.e., the minimal budgets b for which the pwf-net is b-p-safe (b-p-sound), provided we can decide b-p-safety (b-p-soundness) for each $b \in \mathbb{N}^k_\omega$.

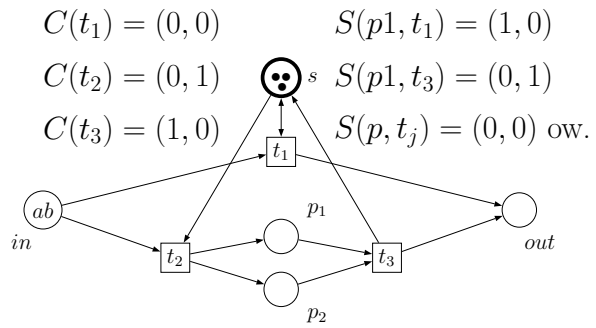


Figure 6.2: A priced resource-constrained workflow net

6.1.2 Priced resource-constrained workflow-nets

Now, we define the first model dealing with different instances and prices. Basically, we add to rcwf-nets two functions C and S, assigning the firing and storage costs [9].

Definition 6.1.5 (Priced rcwf-net) *A* priced rcwf-net *(prcwf-net) with* price arity $k \geq 0$ *is a tuple* $N = \langle P, T, F, H, C, S \rangle$ *such that:*

- $\langle P, T, F, H \rangle$ *is an rcwf-net, called the underlying rcwf-net of* N*,*
- $C : T \to \mathbb{Z}^k$ *and* $S : P \times T \to \mathbb{Z}^k$ *are functions specifying the firing and storage costs, respectively.*

The behavior of a priced rcwf-net is given by its underlying rcwf-net. In fact, prices are different from resources in that they do not constrain the behavior of the net. However, the runs of a priced rcwf-net have a price which is defined as follows.

Definition 6.1.6 (Price of an instance) *We define the* price of an instance $a \in Id(m_0)$ *in a run* $r = m_0 \xrightarrow{t_1(\sigma_1)} m_1 \xrightarrow{t_2(\sigma_2)} m_2 \dots m_{n-1} \xrightarrow{t_n(\sigma_n)} m_n$ *of a prcwf-net as*

$$\mathcal{P}(a, r) = \sum_{\substack{i=1 \\ \sigma_i(x)=a}}^{n} (C(t_i) + \sum_{p \in P} |m_{i-1}(p) - \sigma(F_t(p))| * S(p, t_i))$$

Intuitively, we are considering those transitions in r fired by a, and computing their price. In particular, we are assuming that when computing the price of the firing of a transition by an instance, the tokens belonging to other instances are accounted for. In other words, a pays a penalization for the storage of all tokens

when it fires a transition. We could have also decided that each instance only pays for its own tokens, thus being in a slightly different setting, but the techniques used in our results would also apply.

Since in rcwf-nets we are interested in the behavior of several concurrent instances, we collect their prices in the following definition.

Definition 6.1.7 (Price of a run) *Given a run r of a prcwf-net starting in m_0, we define the* price of r *as the multiset*

$$\mathcal{P}(r) = \{\mathcal{P}(a, r) \mid a \in Id(m_0)\} \in (\mathbb{Z}^k)^{\oplus}.$$

Let us illustrate this definition by an example.

Example 6.1.1 *Focus on Fig. 6.2, and let us consider the run r in which we first fire t_2 with instance b, then we fire t_1 with instance a and finally we fire transition t_3 with instance b. Then, we have that the prices of the instances are $\mathcal{P}(a, r) = (0, 1)$ and $\mathcal{P}(b, r) = (1, 1)$, so the price of the run is $\mathcal{P}(r) = \{(1, 0), (1, 1)\}$.*

Instead of fixing the condition to be satisfied by all the prices of each instance, we define a parametric version of priced safety and dynamic soundness. More precisely, those properties for prcwf-nets are parameterized with respect to a price-predicate.

Definition 6.1.8 (Price-predicate) *A* price-predicate ϕ *of arity $k \geq 0$ is a predicate over $\mathbb{N}_\omega^k \times (\mathbb{Z}^k)^{\oplus}$ such that if $b \leq b'$ and $A' \leq^{\oplus} A$ then $\phi(b, A) \rightarrow \phi(b', A')$ holds.*

Intuitively, b stands for the budget, and A stands for the price of a run. Notice that price-predicates are upward-closed in their first argument, but downward-closed in their second argument. Intuitively, if a price-predicate holds for given budget and costs, then it holds with a greater budget and less costs, as expected. From now on, for a price-predicate ϕ and $b \in \mathbb{N}_\omega^k$, we will denote by $\phi(b)$ the predicate over $(\mathbb{Z}^k)^{\oplus}$ that results of specializing ϕ with b. Moreover, when there is no confusion we will simply say that a run r satisfies a predicate when $\mathcal{P}(r)$ satisfies it.

Now, we define the concepts of priced safety (p-safety for short) and dynamic soundness with respect to a given price predicate.

Definition 6.1.9 (ϕ-p-safety) *Let $b \in \mathbb{N}_\omega^k$ and ϕ be a price-predicate. We say that the prcwf-net N is $\phi(b)$-*p-safe *for $m_0 \in P_S^\oplus$ if for each $j > 0$, every run of N starting in m_0^j satisfies $\phi(b)$.*

Hence, ϕ-p-safety is only related to the prices of the runs of the net, and not to its behavior. In order to encompass the requirements about the behavior of the net and the prices of the run, we define the (priced) dynamic soundness with respect to some price predicate.

Definition 6.1.10 (ϕ-dynamic soundness) *Let $b \in \mathbb{N}_\omega^k$ and ϕ be a price-predicate. We say that the prcwf-net N is $\phi(b)$-*dynamically sound *for $m_0 \in P_S^\oplus$ if for each $j > 0$ and for each marking m reachable from m_0^j by firing some r_1, we can reach a marking $m_f \in \mathcal{M}_{out}^j$ by firing some r_2 such that $r_1 \cdot r_2$ satisfies $\phi(b)$.*

Note that ordinary dynamic soundness is obtained by taking ϕ as the constantly true predicate. As we have mentioned before, prices are different from resources in that they do not constrain the behavior of the net. However, once we are interested in checking a priced-soundness problem, it is natural to consider the available "budget" as an extra resource. Indeed, this can be done but only for firing costs, which are local to transitions, but again this is not possible for storage costs. Let us see some simple facts about ϕ-p-safety and ϕ-dynamic soundness.

Proposition 6.1.4 *The following holds:*

1. *If $\phi_1 \to \phi_2$ holds, then $\phi_1(b)$-p-safety implies $\phi_2(b)$-p-safety, and $\phi_1(b)$-dynamic soundness implies $\phi_2(b)$-dynamic soundness.*

2. *For any ϕ, ϕ-dynamic soundness implies (unpriced) dynamic soundness.*

3. *In general, ϕ-dynamic soundness is undecidable for (non-proper) rcwf-nets.*

Proof: (1) is straightforward by Def. 6.1.9 and Def. 6.1.10. (2) follows from (1), considering that any ϕ entails the constantly true predicate. (3) follows from the undecidability of (unpriced) dynamic soundness for general (non proper) rcwf-nets proved in Chapter 5.

□

Therefore, ϕ-dynamic soundness is undecidable for some ϕ, though certainly not for all. As a (not very interesting) example, if ϕ is the constantly false price-predicate, no prcwf-net is ϕ-dynamically sound, so that it is trivially decidable. Now we factorize ϕ-dynamic soundness into unpriced dynamic soundness and p-safety. As we proved in the previous section, if we consider negative costs safety is undecidable even for priced wf-nets. Therefore, for now on we will focus on rcwf-nets with non-negative costs.

Proposition 6.1.5 *Let ϕ be a price-predicate and N a prcwf-net with non-negative costs. Then N is $\phi(b)$-dynamically sound if and only if it is dynamically sound and $\phi(b)$-p-safe.*

Proof: First notice that for any run r of N and any run r' extending r we have $\phi(b, \mathcal{P}(r \cdot r')) \to \phi(b, \mathcal{P}(r))$. Indeed, it is enough to consider that, because we are considering that costs are non-negative, $\mathcal{P}(r) \leq^{\oplus} \mathcal{P}(r \cdot r')$ holds and, by Def. 6.1.8, ϕ is downward closed in its second parameter. For the if-part, if N is dynamically sound and all its runs satisfy $\phi(b)$ then it is clearly $\phi(b)$-dynamically sound. Conversely, if it is $\phi(b)$-dynamically sound it is dynamically sound by Prop. 6.1.4. Assume by contradiction that there is a run r that does not satisfy $\phi(b)$. By the previous observation, no extension of r can satisfy $\phi(b)$, so that N is not $\phi(b)$-dynamically sound, thus reaching a contradiction.

□

Therefore, to decide ϕ-dynamic soundness we can consider those two properties separately. Though (unpriced) dynamic soundness is undecidable for rcwf-nets (without the path property), it is decidable for the subclass of proper rcwf-nets. Next, we study the decidability of $\phi(b)$-p-safety for various price-predicates, even if N is not proper.

As previously, notice that for any price-predicate ϕ, the set $\mathcal{B}_\phi(N) = \{b \in \mathbb{N}^k_\omega \mid N \text{ is } \phi(b)\text{-dynamically sound } (\phi(b)\text{-p-safe})\}$ is upward-closed because of the upward-closure in the first parameter of price-predicates. Therefore, and as we did for pwf-nets, we can apply the Valk & Jantzen result to compute the minimal budgets b for which N is $\phi(b)$-p-safe ($\phi(b)$-dynamically sound) whenever we can decide $\phi(b)$-p-safety ($\phi(b)$-dynamic soundness) for each $b \in \mathbb{N}^k_\omega$.

6.1.3 Selected price predicates

Now, we study some specific cases of these price predicates. In particular, we study the maximum, the sum, the average and the discounted sum.

$$S(p, t_1) = 2 \qquad S(q, t_j) = 0 \text{ otherwise.}$$
$$C(t_1) = C(t_2) = 0$$

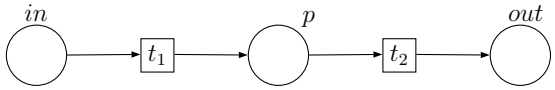


Figure 6.3: prcwf-net not $Sum(b)$ neither $Max(b)$-dynamically sound for any $b \in \mathbb{N}$

Sum and *Max*-dynamic soundness

Let us now study the two first of the concrete priced problems for prcwf-nets. When we consider several instances of a workflow net running concurrently, we may be interested in the overall accumulated price, or in the highest price that the execution of each instance may cost.

Definition 6.1.11 (*Sum* and *Max* price-predicates) *We define the price-predicates Sum and Max as:*

$$Sum(b, A) \iff \sum_{x \in A} x < b$$
$$Max(b, A) \iff x < b \text{ for all } x \in A$$

Sum and *Max* are indeed price-predicates because they satisfy the conditions in Def. 6.1.8. They are both upward closed in the first parameter and downward closed in the second. Let us remark that the cost model given by *Sum*, in which all the prices are accumulated, is the analogous to the cost models in [9, 2]. However, since we are here interested in the behavior of an arbitrary number of instances, a necessary condition for $Sum(b)$-p-safety is that all instances, except for a finite number of them, have a null price (for those components in b that are not ω).

Example 6.1.2 *Consider the prcwf-net N in Fig. 6.3, and a run of N with n instances, and in which t_2 is not fired until t_1 has been fired n times. The price of the i-th instance in any such run is $2 \cdot (i-1)$. Indeed, the first firing of t_1 costs nothing, because there are no tokens in p, but in the second one there is already a token in p, so that the second firing costs 2 (because $S(p, t_1) = 2$). In particular, the last instance of the net costs $2 \cdot (n-1)$. Therefore, the net is neither $Max(b)$-p-safe nor $Sum(b)$-p-safe for any $b \in \mathbb{N}$.*

Now, suppose that $S(p,t) = 0$ *for each place* p *and transition* t, $C(t_1) = 1$ *and* $C(t_2) = 0$. *Each instance costs exactly* 1, *so that it is* $Max(2)$*-p-safe. However, if we consider a run in which* n *instances have reached out, then the sum of the prices of all instances is* n, *and the net is not* $Sum(b)$*-p-safe for any* $b \in \mathbb{N}$.

Before tackling the decidability of $Max(b)$-p-safety and $Sum(b)$-p-safety, we need to introduce a model which will be useful in the proofs of decidability of the predicates. As previously, in order to model different instances running in the same net, we will use names, each name representing a different instance. We obtain $w\nu$-PNs (w stands for whole-place) as an extension of ν-PN which also allows whole-place operations and broadcasts, similar to Data Nets [62]. Data Nets extend P/T nets by considering a linearly ordered and dense domain of tokens, and in which whole place operations can be performed. Therefore, $w\nu$-PNs can be seen as an unordered version of Data nets [62] in which names can be created fresh. When a transition t of a $w\nu$-PN is fired, four operations are performed: the subtraction of several tokens of different colors, whole-place operations (affecting every color in the same way), the creation of new names and the addition of tokens.

As for ν-PN, let us consider a set Var of variables and $\Upsilon \subset Var$ a set of name creation variables. A $w\nu$-PN is a tuple $N = \langle P, T, F, G, H\rangle$ where P and T are finite disjoint sets of places and transitions, respectively; for each $t \in T$, $F_t : P \to (Var \backslash \Upsilon)^{\oplus}$ is its subtraction function, $G_t : P \times P \to \mathbb{N}$ is its whole-place operations matrix, and $H_t : P \to Var^{\oplus}$ is its addition function. Moreover, if $x \in H_t(p) \setminus \Upsilon$ then $x \in F_t(p')$ for some $p' \in P$.

Let Id be an infinite set of names. A marking is any $m : P \to Id^{\oplus}$. An a-token in p is an occurrence of $a \in m(p)$. $Id(m)$ is the set of names appearing in m, that is, $Id(m) = \bigcup_{p\in P} S(m(p))$. We denote by $Var(t) = \{x \in Var \mid \exists p \in P,\ x \in F_t(p) \cup H_t(p)\}$ and $Var(p) = \{x \in Var \mid \exists t \in T,\ x \in F_t(p) \cup H_t(p)\}$. A mode is a mapping $\sigma : Var(t) \to Id$ extended pointwise to $\sigma : Var(t)^{\oplus} \to Id^{\oplus}$. A transition t is enabled at a marking m with mode σ if for all $p \in P$, $\sigma(F_t(p)) \subseteq m(p)$ and for all $\nu \in \Upsilon$, $\sigma(\nu) \notin Id(m)$. Then, we say that t can be fired, reaching a new marking m', where for all $p \in P$, $m'(p) = \sum_{p'\in P}((m(p') - \sigma(F_t(p'))) * G_t(p', p)) + \sigma(H_t(p))$, which is denoted by $m \overset{t(\sigma)}{\to} m'$.

Example 6.1.3 *Let* $N = (\{p_1, p_2\}, \{t\}, F, G, H)$ *be a* $w\nu$*-PN, where:*

- $F_t(p_1) = \{x\}$, $F_t(p_2) = \emptyset$.

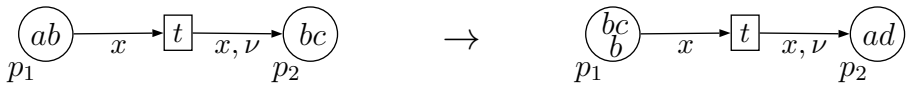


Figure 6.4: The firing of a $w\nu$-PN

- $H_t(p_1) = \emptyset$, $H_t(p_2) = \{x, \nu\}$.
- $G_t(p_1, p_1) = 1$, $G_t(p_1, p_2) = 0$, $G_t(p_2, p_1) = 1$, $G_t(p_2, p_2) = 0$.

This net is depicted in Fig 6.4. Note that although F_t and H_t are represented by arrows labeled by the corresponding variables, the effects of G_t are not depicted. Let m be the marking of N such that $m(p_1) = \{a, b\}$ and $m(p_2) = \{b, c\}$. Then, t can be fired at m with mode σ, where $\sigma(x) = a$ and $\sigma(\nu) = d$, reaching a new marking m', such that $m'(p_1) = \{b, b, c\}$ and $m'(p_2) = \{a, d\}$. Note that m' is obtained from m by the following steps:

- *Removing an a-token from the place p_1, due to the effect of F.*
- *Removing all tokens from p_2 and copying them to p_1, because of the effects of G. Indeed, as $G_t(p_1, p_1) = 1$, the tokens in p_1 remain unchanged in this step, and as $G_t(p_2, p_2) = 0$, the tokens in p_2 are all removed. Moreover, as $G_t(p_2, p_1) = 1$, the tokens in p_2 before this step are copied to p_1.*
- *Adding an a-token and a d-token to p_2, because of H.*

We write $m_1 \sqsubseteq m_2$ if there is a renaming m_1' of m_1 such that $m_1'(p) \subseteq m_2(p)$ for every $p \in P$. A marking m is coverable from an initial marking m_0 if we can reach m' from m_0 such that $m \sqsubseteq m'$.

A $w\nu$-PN could be considered as an unordered Data Net, except for the fact that $w\nu$-PNs can create fresh names. In [4] the authors extend Data Nets with fresh name creation and prove that coverability is still decidable by instantiating the framework of Well Structured Transition Systems [37].

Proposition 6.1.6 *Coverability is decidable for $w\nu$-PN.*

Now we prove decidability of *Max* and *Sum*-p-safety by reducing them to non-coverability problems in a $w\nu$-PN. Given a prcwf-net N and a budget $(b_1, \ldots, b_n)$, we build a $w\nu$-PN $\mathcal{C}(N)$, the *cost representation net* of N, by adding to N n new places, whose tokens represent the costs of each run. Then, the net will be safe

iff, for all $i \in n^+$, no marking with b_i tokens in the place representing the i^{th} component of the prices can be covered. We simulate firing costs by adding to N "normal arcs", without whole-place operations, but for the simulation of storage costs we need the whole-place capabilities of $w\nu$-PN.

Proposition 6.1.7 *Max-p-safety and Sum-p-safety are decidable for prcwf-nets, even without the path property. Max-dynamic soundness and Sum-dynamic soundness are decidable for proper prcwf-nets.*

Proof: By Prop. 6.1.5, it is enough to consider the price-safety problems. We reduce *Sum*-p-safety to (non)coverability for $w\nu$-PN. Then, we show how to adapt this reduction to the case of *Max*-p-safety. Let $N = \langle P, T, F, H, C, S\rangle$ be a prcwf-net with price arity k and $b \in \mathbb{N}_\omega^k$. We can assume that b has no ω-components, or we could safely remove the cost information of those components. We build the $w\nu$-PN $\mathcal{C}(N) = \langle P^c, T^c, F^c, G^c, H^c\rangle$ as follows:

- $P^c = P \cup \{c_1, ..., c_k\}$,
- $T^c = T \cup \{t_\nu\} \cup \{t_{c_1}, ..., t_{c_k}\}$.
- For each $t \in T$,
  - $F_t^c(p) = F_t(p)$ if $p \in P$, and $F_t^c(p) = \emptyset$, otherwise,
  - $H_t^c(p) = H_t(p)$ if $p \in P$, and $H_t^c(c_i) = C(t)[i] * \{x\}$, otherwise,
  - $G_t^c(p, p') = \begin{cases} S(p,t)[i] & \text{if } p \in P, \text{ and } p' = c_i, \\ 1 & \text{if } p = p', \\ 0 & \text{otherwise.} \end{cases}$
- For each $i \in \{1, ..., k\}$,
  - $F_{t_{c_i}}^c(c_i) = \{x, y\}$, and $F_{t_{c_i}}^c(p) = \emptyset$ otherwise,
  - $H_{t_{c_i}}^c(c_i) = \{x, x\}$, and $H_{t_{c_i}}^c(p) = \emptyset$ otherwise, and
  - $G_{t_{c_i}}^c$ is the identity matrix.

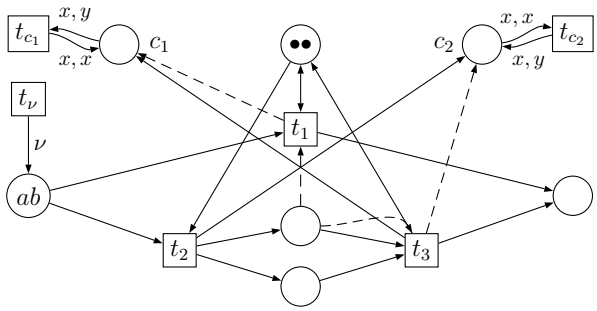


Figure 6.5: The costs representation $w\nu$-PN of the prcwf-net in Fig. 6.2

- $F_{t_\nu}(p) = \emptyset$ for any $p \in P^c$, $H_{t_\nu}(in) = \{\nu\}$ and returns the empty multiset elsewhere, and G_{t_ν} is the identity matrix.

Any run r of N can be simulated by a run of $\mathcal{C}(N)$, preceded by several firings of t_ν. Moreover, if r starts in m_0 and finishes in m (seen as a run of $\mathcal{C}(N)$), then by construction of $\mathcal{C}(N)$ it holds that the sum of the prices of the instances in r, is the vector formed by considering the number of tokens (maybe with different colors) in $c_1, ..., c_k$. In particular, when a transition in T is fired, we add the corresponding firing costs to places c_i by "normal arcs", that is, we have $H^c_t(c_i) = C(t)[i] * \{x\}$ (which denotes the multiset with $C(t)[i]$ xs). Moreover, we add storage costs by copying the necessary number of times the tokens in dynamic places to places c_i, that is, we have $G^c_t(p, c_i) = S(p, t)[i]$. Finally, as each transition t_{c_i} takes two tokens with different names from c_i, and puts them back, changing the name of one of them by the name of the other token, these transitions allow to reach each marking in which the sum of the prices of all instances of a run is represented by the tokens in the places c_i, all of them with the same name. Then, N is $Sum(b)$-p-unsafe if and only if there is $j \in \{1, ..., k\}$ such that the marking with $b[j]$ tokens of the same color in c_j and empty elsewhere is coverable, and we are done.

The previous construction with some modifications also yields decidability of $Max(b)$-p-safety. We add one more place *last* (which will always contain the name of the last instance that has fired a transition) and for each $i \in \{1, ..., k\}$, we add a new place d_i (where we will compute the costs).

When a transition $t \in T$ is fired, in $\mathcal{C}(N)$ we replace the name in *last* by the name of the current transition, and reset every place c_i (by setting $G_t(c_i, c_i) = 0$). Moreover, we change the effect of every t_{c_i}: they now take a token from c_i, and put a token of the name in *last* in the place d_i (see Fig. 6.6). Hence, each time an instance a fires a transition, the corresponding firing and storage costs are stored

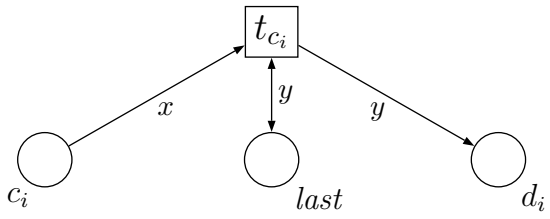


Figure 6.6: The mechanism added for *Max*

in c_i with different names, and can be transferred to d_i with name a by repeatedly firing t_{c_i}, if no other transition of the net is fired while the transference is being performed. When other transition is fired, c_i is emptied, and the token in *last* is changed, ensuring that we do not consider the same name to represent the costs of different instances in d_i.

Therefore, when a transition $t \in T$ is fired, it is possible to reach a marking in which the costs of firing t are added to every d_i (represented by the name of the instance that has fired t) by firing t followed by the firing of every t_{c_i} n_i times, provided t put n_i tokens in c_i. Notice that if another transition fires before t_{c_i} is fired exactly n_i times, then that run is lossy, in the sense that it is computing an underapproximation of its cost, but it is always possible to compute the exact cost. Therefore, N is $Max(b)$-p-unsafe if and only if there is $j \in \{1, ..., k\}$ such that the marking with $b[j]$ tokens of the same color in d_j and empty elsewhere is coverable.

□

Example 6.1.4 *Fig. 6.5 shows the costs representation net of the net N in Fig. 6.2. For a better readability, we have removed some of the labels of the arcs. As the prices in N are vectors of $\mathbb{N}^2$, we have added two places, c_1 and c_2, to store the costs; and two transitions t_{c_1} and t_{c_2}, which take two tokens of different colors of the corresponding places and put them back, with the same color. Moreover, we have added arcs that manage the addition of the cost of transitions. In particular, dashed arcs denote copy arcs, meaning that when the corresponding transition is fired, tokens are copied in the places indicated by the arrows (which is the effect of G in the proof of the previous result). Then, $Sum(b)$-p-safety is reduced to non-coverability problems: the prcwf-net is $Sum(1,1)$-p-safe iff neither m_1 (the marking with only one token carrying the same name in c_1) neither m_2 (the marking with only one token carrying the same name in c_2) are coverable.*

We remark that if we consider a cost model in which each instance only pays for its own tokens, as discussed after Def. 6.1.6, the previous proof can be adapted by considering a version of $w\nu$-PN with finer whole-place operations, which are still a subclass of the ones considered in [4], so that the result would still apply. Next we show that we can reduce b-p-soundness and b-p-safety problems for pwf-nets with non-negative costs, defined in Sect. 6.1.1, to *Max*-dynamic soundness for prcwf-nets. Therefore, if we consider non-negative costs, we can prove the decidability of b-p-soundness (already proved in Prop. 6.1.2 in the case with negative costs but considering that the path property holds), and decidability of b-p-safety.

Corollary 6.1.8 *b-p-safety and b-soundness are decidable for pwf-nets with non-negative costs, even without the path property.*

Proof: Let N be a pwf-net. To decide b-safety it is enough to build a prcwf-net N' by adding to N a single static place s, initially containing one token, two new places in' and out' (the new initial and final places), and two new transitions t_{in} and t_{out}. Transition t_{in} can move a name from in' to in whenever there is a token in s, that is, $F_{t_{in}}(in') = \{x\}$, $F_{t_{in}}(s) = \{\epsilon\}$ and $F_{t_{in}}$ is empty elsewhere, and $H_{t_{in}}(in) = \{x\}$ and empty elsewhere. Analogously, t_{out} can move a name from out to out', putting the black token back in s, that is, $F_{t_{out}}(out) = \{x\}$, and empty elsewhere, and $H_{t_{out}}(out') = \{x\}$, $H_{t_{out}}(s) = \{\epsilon\}$, and empty elsewhere. In this way, the concurrent executions of N' are actually sequential. Since there is no other way in which instances can synchronize with each other (because there are no more static places) the potential behavior of all instances coincide, and coincide in turn with the behavior of N. Finally, we take the cost of firing t_{in} and t_{out} as null, as well as the cost of storing tokens in in' and out' for any transition, and the cost of storing tokens in any place for t_{in} and t_{out}. More precisely, $C(t_{in}) = C(t_{out}) = \mathbf{0}$, $S(p, t_{in}) = S(p, t_{out}) = \mathbf{0}$ for any $p \in P$, and $S(in', t) = S(out', t) = \mathbf{0}$ for any $t \in T$. In this way, the cost of each instance is the cost of a run of N. Therefore, N is b-p-safe if and only if N' is $Max(b+1)$-p-safe. Since weak soundness is decidable for wf-nets [92], we conclude.

□

Av-dynamic soundness

Now we study the next of the concrete priced-soundness problems. Instead of demanding that the execution of each instance does not exceed a given budget

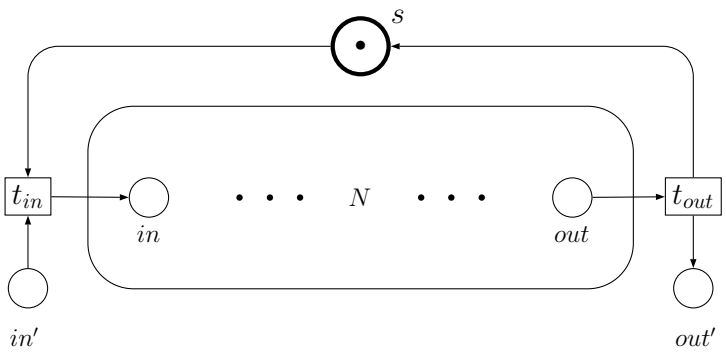


Figure 6.7: The construction of Cor. 6.1.8

(though the price of one instance depends on the others), we will consider an amortized, or average price.

Definition 6.1.12 (*Av* price-predicate) *We define the price-predicate Av as* $Av(b, A) \Leftrightarrow (\sum_{x \in A} x)/\,|A| < b$.

Therefore, N is $Av(b)$-p-safe if in average, the price of each instance does not exceed b, for any number of instances. Alternatively, we could have a slightly more general definition, in which we only considered situations in which the number of instances exceeds a given threshold $l > 0$. More precisely: $Av_l(b, A) \Leftrightarrow |A| \geq l \rightarrow (\sum_{x \in A} x)/\,|A| < b$. We will work with Av, though we claim that with fairly minor changes in our techniques we could also address the slightly more general price-predicate Av_l.

Example 6.1.5 *Consider the prcwf-net in Fig. 6.8. The cost of firing* t_1 *is twice the number of instances in place in when* t_1 *is fired. Therefore, the net is* $Av(2)$*-p-safe, though it is not* $Max(b)$*-p-safe for any* $b \in \mathbb{N}$.

Now suppose that we force t_2 *to be fired in the first place,* t_1 *in second place, and then* t_2 *as many times as possible, by adding some static conditions. Moreover, consider that the cost of firing* t_1 *is three times the number of instances in place out (instead of twice the number of instances in in) when* t_1 *is fired. Then, if we consider any run* r *with two or more instances, in which we fire* t_1 *and* t_2 *in the beginning, the sum of the prices of the instances of* r *is* 3*. Therefore, if we consider such a run with two instances, the average price is* $3 \div 2 > 1$*, and then the net is not* $Av(1)$*-p-safe. However, if we consider that the number of instances is greater than three, the average of the prices always remains under* 1*, and therefore the net is* $Av_l(1)$*-p-safe if we consider any threshold* $l \geq 4$.

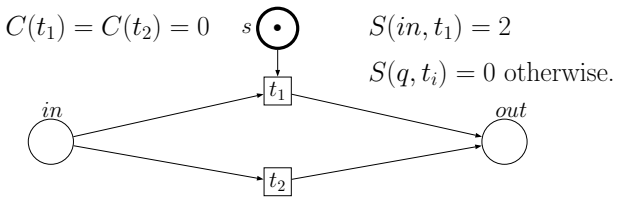


Figure 6.8: Av-p-safety does not imply Max-p-safety

We can reduce $Av(b)$-dynamic soundness of a prcwf-net N to (unpriced) dynamic soundness of an rcwf-net N^b. In order to ensure $Av(b)$-p-safety, the maximum budget we may spend in an execution with n instances is $(b * n) - 1$. Essentially, the idea of this construction is to add to N new places $s_1 \dots s_k$ in which tokens represent the remaining budget, and remove tokens from them when transitions are fired. Moreover, each transition will have $s_1 \dots s_k$ as preconditions, so that if the net has consumed all the budget, then it halts before reaching the final marking. Therefore, we add $b\,[i]$ tokens to s_i each time an instance starts its execution, for each i. The simulation is "lossy" because of how we manage storage costs, but it preserves dynamic soundness. The proof of the next proposition gives a detailed explanation of this construction.

Proposition 6.1.9 *Given a prcwf-net N and $b \in \mathbb{N}^k_\omega$, there is an rcwf-net N^b such that N is $Av(b)$-dynamically sound if and only if N^b is dynamically sound.*

Proof: Let k be the price arity of N. We start the construction of N^b by adding to N new static places $s_1, ..., s_k$ that are initially empty. These new places store the budget that can be consumed by instances. For that purpose, every instance adds $b[i]$ tokens to s_i when it starts. When a transition t is fired, we remove from s_i $C(t)[i]$ tokens to cope with firing costs. We will later explain how to cope with storage costs (notice that N^b is an rcwf-net, and in particular it does not have whole-place operations). Moreover, each transition has $s_1, ..., s_k$ as preconditions and postconditions. In particular, we add a final transition t_{out} that has $s_1, ..., s_k$ and *out* as preconditions, and $s_1, ..., s_k$ and a new place *out*$'$, which works as the final place of N^b, as postconditions. Therefore, the net will deadlock when some s_i is empty, meaning that it has used strictly more than the allowed budget. Then, if N is not $Av(b)$-dynamically sound, N^b halts before reaching the final marking for some execution, and therefore, it is not dynamically sound. Moreover, if N is $Av(b)$-dynamically sound, then N^b is dynamically sound, because each place s_i always contains tokens, and therefore the executions of N^b

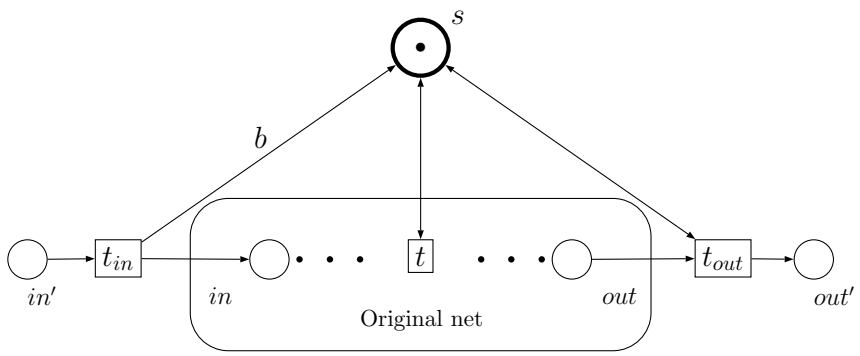


Figure 6.9: Construction for $Av(b)$-dynamic soundness

represent executions of N. Fig. 6.9 shows a schema of the reduction for price arity 1.

Now we address the simulation of storage costs. Fig. 6.10 depicts the following construction. We simulate them in a "lossy" way, meaning that if the firing of t in N costs v, in the simulation we will remove *at most* $v[i]$ tokens from s_i. To do that, for each place p of N we will add a new place p'. When a transition t is fired, for each place p we transfer tokens from p to p', one at a time (transition t_p in the figure), removing at each time $S(p,t)[i]$ tokens from s_i. We add the same mechanism for the transfer of tokens from p' to p and some static conditions to make sure that if we have started transferring tokens from p to p' because of the firing of a transition, we do not transfer tokens from p' to p because of the same firing, and the other way around. At any point, the transfer can stop (even if some tokens have not been transfered), which finishes the simulation of t. Since we now have two places representing each place p (p and p'), for each transition of N, we need to add several transitions in order to be able to take (or put) tokens from p, p' or both.

Having lossy computations of the cost of a run, if N exceeds the average budget for some execution and some number of instances, then N^b will have a deadlock when this execution is simulated correctly (meaning that all the tokens which have to be transfered are indeed transfered). Then, N^b is not dynamically sound. Conversely, if N is $Av(b)$-dynamically sound (and in particular no run of N exceeds the average budget), then N^b never consumes all the tokens in any s_i, and it behaves as N, so that it is dynamically sound.

□

Hence, we obtain the following decidability result.

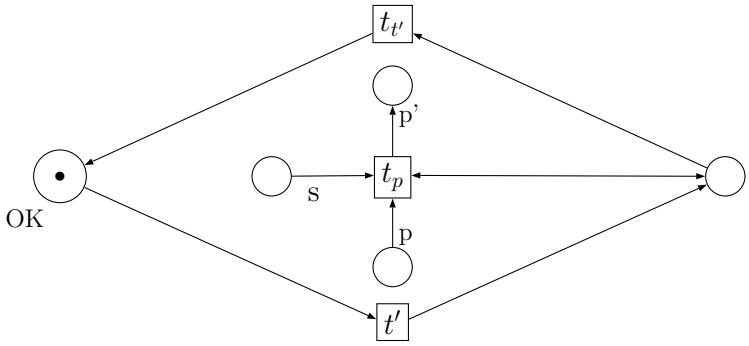


Figure 6.10: Schema of the managing of storage costs assuming $S(p,t) = 1$

Corollary 6.1.10 *Av-dynamic soundness is decidable for proper prcwf-nets.*

Ordered prices

So far, we have considered that instances are not ordered in any way, following directly the approaches in [44, 53]. Nevertheless, we could consider an order between the instances, and use it to compute the price of a run in such a way that the relative order between instances matter. A sensible way to do that is to assume a linear order between instances within a run given by the order in which they start their execution.

Definition 6.1.13 (Order between instances) *Let N be a prcwf-net, and a and b be two instances in a run r of N. We write $a <_r b$ if a is removed from in in r before b, and $a =_r b$ if neither a nor b have been removed from in in r. We write $a \leq_r b$ if $a =_r b$ or $a <_r b$.*

Then, the order $\leq_r$ is a total order over the set of instances in r. In this situation we can write $Id(r) = a_1 \leq_r \cdots \leq_r a_n$ to denote that $a_1, ..., a_n$ are all the instances in r, ordered as indicated.

Definition 6.1.14 (Ordered price of a run) *Given a run r of a prcwf-net with $Id(r) = a_1 \leq_r \cdots \leq_r a_n$ we define the* ordered price *of r as the word $\mathcal{P}_o(r) = \mathcal{P}(a_1, r)...\mathcal{P}(a_n, r) \in (\mathbb{N}^k)^*$.*

Notice that the previous definition is correct in the sense that whenever $a =_r b$ then we have $\mathcal{P}(a, r) = \mathcal{P}(b, r) = 0$. Moreover, the instances of a run are always ordered as $a_1 <_r \cdots <_r a_m < a_{m+1} =_r ... =_r a_n$.

With the notion of ordered price, we can consider price-predicates that depend on the order in which instances are fired. Therefore, ordered price-predicates are predicates over $\mathbb{N}_\omega^k \times (\mathbb{N}^k)^*$. We consider the order $\leq^*$ over $(\mathbb{N}^k)^*$ given by $w_1...w_n \leq^* w_1...w'_m$ iff $n \leq m$ and for each $0 < i \leq n$, $w_i < w'_i$. For instance, following [21], we can model situations in which costs in the future are less important than closer ones.

Definition 6.1.15 (*Ds*-price predicate) *Given $0 < \lambda < 1$, we define the* discounted-sum *price-predicate Ds_λ as*

$$Ds_\lambda(b, v_1...v_n) \Leftrightarrow \sum_{i=1}^{n} \lambda^i * v_i < b$$

Example 6.1.6 *Let us recall the run of the net N of Fig. 6.3 described in Ex. 6.1.2. We proved that the net is neither Max(b)-p-safe nor Sum(b)-p-safe for any $b \in \mathbb{N}$. Moreover, the average price of the run is $\sum_{i=1}^{n} 2(i-1)/n$, which equals $n-1$, so that it is not Av(b)-p-safe for any $b \in \mathbb{N}$. However, the discounted price of the run is $\sum_{i=1}^{n} 2(i-1)\lambda^i$, with $0 < \lambda < 1$. By using standard techniques, it can be seen that the limit of those sums is $b = 2\lambda^2/(1-\lambda)^2$. Moreover, for $\lambda = 1/c$ with $c > 1$ that formula simplifies to $2/(c-1)^2$. As it is easy to prove that the considered runs are the most expensive ones of N, it follows that it is $Ds_\lambda(b+1)$-p-safe for that $b \in \mathbb{N}$.*

Note that if we consider $\leq^*$, then Ds_λ is downward-closed in its second argument. Decidability of Ds_λ-p-safety remains open, but a weaker version of this problem, in which we only consider finitely many instances, is decidable.

Definition 6.1.16 (*Fds*-price predicate) *Given $0 < \lambda < 1$ and $l \in \mathbb{N}$, we define the finite-discounted-sum price-predicate*

$$Fds^l_\lambda(b, v_1...v_n) \Leftrightarrow \sum_{i=1}^{min\{n,l\}} \lambda^i * v_i < b$$

For this finite version of discounted-sum, p-safety is decidable.

Proposition 6.1.11 *Let $l \in \mathbb{N}$, $c \in \mathbb{N} \setminus \{0\}$ and $\lambda = 1/c$. Fds^l_λ-p-safety is decidable for prcwf-nets. Fds^l_λ-dynamic soundness is decidable for proper prcwf-nets.*

Proof: We reduce Fds^l_λ-p-safety to coverability for $w\nu$-PN. Basically, we build a new net, in which the first l instances are managed separately, in order to store their weighted prices in places as in the proof of Prop. 6.1.7. We consider $l+1$ copies of the net, one for each of the first l instances, and one for the rest of the instances, to give the proper weight to the prices that we store. Let $N = \langle P, T, F, H, C, S\rangle$ be a prcwf net. Let us build a new $w\nu$-PN $N' = \langle P', T', F', G', H'\rangle$ as follows:

For each dynamic place $p \in P$, we consider $p, p_1, \ldots, p_l$ in P', and for each $t \in T$, we take $t, t_1, \ldots, t_l$ in T'. If $p \neq in$ then, for each $i \in \{1, \ldots, l\}$, $F'_t(p) = F'_{t_i}(p_i) = F_t(p)$ and $H'_t(p) = H'_{t_i}(p_i) = H_t(p)$. For each static place $s \in P$, we consider $s \in P'$. Moreover, for each $t \in T$ and for each $i \in \{1, \ldots, l\}$, $F'_t(s) = F'_{t_i}(s) = F_t(s)$ and $H'_t(s) = H'_{t_i}(s) = H_t(s)$. Therefore N' has $l+1$ copies of the dynamic part of N, sharing the static part.

We manage separately the places $in, in_1, \ldots, in_l$, in order to make sure that the i^{th} copy of the dynamic part of N corresponds with the i^{th} instance that has started. In particular, when we remove a token from a place in_i, we set a token in in_{i+1}, to make possible the next instance to start. Given $t \in T$ such that $F_t(in) \neq \emptyset$ then, for each $i \in \{1, \ldots, l\}$, $F'_t(in_i) = F'_{t_i}(in) = F_t(in)$ and for each $i \in \{1, \ldots, l-1\}$, $H'_{t_i}(in_{i+1}) = \nu$. Finally, the last instances will be managed in the same last copy of the net, in which places do not have indexes, so $H'_{t_l}(in) = H'_t(in) = \nu$.

Now we have the structure for the different copies of the net, we add some places to store the weighted prices of the first l instances. Let n be the arity of b. Then, $pr_1, \ldots, pr_n \in P'$ will be places where we store the prices. To give a weight to each instance, we consider the following:

$\sum_{i=1}^{l} \lambda^i * v_i < b \Leftrightarrow \sum_{i=1}^{l} 1/c^i * v_i < b \Leftrightarrow c^l(\sum_{i=1}^{l} 1/c^i * v_i) < c^l * b \Leftrightarrow \sum_{i=1}^{l} c^{l-i} * v_i < c^l * b$.

Then, for each i, we will store the prices of the i^{th} instance with weight c_j^{l-i} as in the proof of Prop. 6.1.7, that is, we define G' and H' for places pr_i, considering G'_{t_i} and H'_{t_i} as G'_t and H'_t in Prop.6.1.7, multiplied by c^{l-i}. Therefore if we prove that for each run of N' and each place pr_j the total stored price in pr_j is less than $c^i * b_j$, the predicate $Fds^l_\lambda(b)$ will hold for the net N.

Finally, as in Prop. 6.1.7, we add transitions to make all the tokens in each pr_i be of the same name, and we just need to prove that no marking m with $m(pr_i) = \{a^{c^l * b_i}\}$ and $m(p) = \emptyset$ otherwise (for an arbitrary $a \in Id$) is coverable in N', to prove Fds-safety for λ, b and N.

□

6.1.4 Complexity

Of course, now that we have some positive decidability results, we would like to study the complexity of the previous problems. As a preliminary result, we study the complexity of the safety property for the defined priced predicates, in the case in which the path property does not necessarily hold. More precisely, we reduce coverability for ν-PN to each of the previous safety problems, and therefore, they are at least non primitive-recursive. For that purpose, we introduce the single-name coverability problem, which has the same complexity as coverability, and we see that this problem can be reduced to the problem of deciding ϕ-p-safety, for each of our predicates.

Definition 6.1.17 *We define the single-name-coverability problem as that of given a ν-PN N with initial marking m_0, and m_f a marking with a single identifier, deciding whether m_f can be covered in N.*

Single-name-coverability is a problem more restricted than coverability, which is decidable for ν-PN [80]. Next we prove that its complexity is the same as that of general coverability.

Proposition 6.1.12 *The single-name-coverability problem has non primitive recursive complexity.*

Proof: We reduce coverability, which has non primitive recursive complexity [80], to single-name-coverability. Let m_f be the final marking. It is enough to add a new transition that can be fired whenever m_f is covered, thus putting a (black) token in a new place. Thus, m_f can be covered in the original ν-PN iff the marking with a black token in the new place can be covered.

□

We prove that single-name-coverability for ν-PNs can be reduced to the some of the previous safety problems, and hence, these problems are non primitive recursive.

Proposition 6.1.13 *Single-name-coverability problem for ν-PNs can be reduced to each of the following problems for prcwf-nets without the path property:*

- $Max(1)$-*p-safety.*
- $Sum(1)$-*p-safety.*
- $Av(1)$-*p-safety.*
- $Ds(2)$-*p-safety.*

Proof: Let N be a ν-PN, and a marking m_f of N with a single name. In order to reduce the coverability of m_f to the previous problems, we first build a new rcwf-net N' which simulates the behavior of N, and then, we will add the prices in three different ways, building three different $prcwf$-nets, N_1, N_2 and N_3, in order to reduce coverability of m_f to $Max(1)$-p-safety, $Av(1)$-p-safety and $ds_\lambda(2)$-p-safety. In fact, in our construction the marking m_f of N can be covered if and only if the previous problems have a negative answer.

Let us first build the net N'. In the proof of undecidability of dynamic soundness in Ch. 5, given a ν-PN N, we show how to build a $rcwf$-net N' such that, considering every number of instances, it simulates all possible runs of N. The main idea of how we handle name creation, which is not permitted in rcwf-nets, is by considering the place in as a name storage. Then, when a transition with a ν is fired in N, in N' a name is taken from in, and put in the proper place. Therefore, as we consider runs of N' with any number of names in in, all possible runs of N are represented. Moreover, we add to N' a new transition t_f which has the marking m_f as precondition and a place p in which every name that is taken from in, is stored.

Finally, we assign the prices to N'. In fact, the only transition with storage or firing costs will be t_f. Then, a run will have a price greater than zero if and only if t_f is fired in r, so that m_f can be covered if and only if there is a run of N', with a price greater than zero. Now we define the three different prices, which will reduce coverability to $Max(1)$-p-safety, $Av(1)$-p-safety and $Ds(2)$-p-safety respectively.

- $Max(1)$-p-safety: $C(t_f) = 1$ and $S(q, t_f) = 0$ for each place q. If m_f is covered in N, there is a run r of N' in which t_f is fired, and then, the price of r is 1. Conversely, if m_f is not covered in N, t_f cannot be fired in any run of N', and the price of any run is 0.
- $Av(1)$-p-safety: $C(t_f) = 0$, $S(p, t_f) = 2$ and $S(q, t_f) = 0$ for each place $q \neq p$. If m_f is covered in N, there is a run r of N' in which t_f is fired, and then, the price of each instance in r is 2. Therefore, if n is the number

of instances which have started in r, the average sum of the price of r is $2 * n/n = 2 > 1$. The other implication is analogous to the previous case.

- $Ds(2)$-p-safety: $C(t_f) = 0$, $S(p, t_f) = 1/\lambda$ and $S(q, t_f) = 0$ for each place $q \neq p$. The proof is analogous to the previous one.

□

Hence, we obtain the following complexity result as a corollary.

Corollary 6.1.14 *$Max(1)$-p-safety, $Av(1)$-p-safety and $Ds(1)$-p-safety have non primitive-recursive complexity.*

6.1.5 Relating price predicates

In this subsection we study the relations between the previous price predicates. More precisely, for each pair of predicates ϕ and ψ, we will study whether $\phi(b)$-dynamic soundness entails $\psi(b')$-dynamic soundness for some b'. Moreover, we will try to set the relation between these two bounds.

First of all, let us focus on the prcwf-nets of Fig. 6.3 and Fig 6.8. As we showed in the previous examples, the first of these nets proves that a net may be $Ds_\lambda(b)$-dynamically sound for a certain b, but not Av-dynamically sound, Max-dynamically sound nor Sum-dynamically sound for any bound. Analogously, the second net is $Av(b)$-sound for some $b \in \mathbb{N}$, but not Ds_λ-dynamically sound, Max-dynamically sound nor Sum-dynamically sound for any bound.

The next propositions set the remaining relations:

Proposition 6.1.15 *Let N be a $Sum(b)$-dynamically sound prcwf-net for some $b \in \mathbb{N}$. Then, N is also $Max(b)$-dynamically sound, $Av(b)$-dynamically sound and $Ds_\lambda(b)$-dynamically sound for all λ.*

Proof: If we prove that $Sum(b) \rightarrow Max(b), Av(b), Ds_\lambda(b)$, then we are done (Prop.6.1.4). Let $A = \{x_1, \ldots, x_n\}$ be a set of prices which satisfies $Sum(b)$, that is, $\sum_{x \in A} x < b$. Then, since we are considering non-negative prices, for all $a \in A$, $a \leq \sum_{x \in A} x < b$, and therefore A satisfies $Max(b)$. Moreover, $\sum_{x \in A} x/n \leq \sum_{x \in A} x \leq b$ and if $0 < \lambda < 1$ then $\sum_{i=1}^{n} \lambda^i * x_i < \sum_{x \in A} x < b$. Then A satisfies $Av(b)$ and $Ds_\lambda(b)$ too.

□

	Sum	**Max**	**Ds**	**Av**
Sum	✓	✓	✓	✓
Max	× (Ex. 6.1.2)	✓	☑	✓
Ds	× (Fig. 6.3)	× (Fig. 6.3)	✓	× (Fig. 6.3)
Av	× (Fig. 6.8)	× (Fig. 6.8)	× (Fig. 6.8)	✓

Table 6.1: Relations between predicates

Proposition 6.1.16 *Let N be a $Max(b)$-dynamically sound prcwf-net for some $b \in \mathbb{N}$ and $0 < \lambda < 1$. Then, N is also $Av(b)$-dynamically sound and $Ds_\lambda(b')$-dynamically sound, where $b' = \lambda * b/(1-\lambda)$. In particular, N is $Ds_{1/2}(b)$-dynamically sound.*

Proof: Let us prove that $Max(b) \to Av(b)$ and $Max(b) \to Ds_\lambda(b')$. Let $A = \{x_1, \ldots, x_n\}$ be a set of prices which satisfies $Max(b)$, that is, for all $x \in A$ $x < b$. Let $m = max\{x_1, \ldots, x_n\}$. Then, $\sum_{x \in A} x/n \leq n * m/n = m < b$ and therefore A satisfies $Av(b)$. Moreover, if $0 < \lambda < 1$ then $\sum_{i=1}^{n} \lambda^i * x_i \leq \sum_{i=1}^{n} \lambda^i * m < \sum_{i=1}^{n} \lambda^i * b \leq \lambda * b/(1-\lambda)$, so A satisfies $Ds_\lambda(b')$.

□

Finally, Table 6.1 summarizes all the relations between the different predicates. A ✓ symbol in row ϕ_1 and column ϕ_2 means that $\phi_1(b)$-dynamic soundness implies $\phi_2(b)$-dynamic soundness; a ☑ symbol means that the implication holds possibly for a different b; a × means that the implication does not hold.

6.2 Priced-timed resource-constrained workflow nets

In the previous model we added prices to rcwf-nets supposing that time elapsed only while firing transitions. This extension could be useful to model systems in which no storage costs are produced between actions. However, sometimes we need to represent systems in which the waiting times between actions produce storage costs. Moreover, these delays could be restricted to values in concrete intervals of time. In order to model such systems, we define another extension of rcwf-nets by using the *ν-PTdPN* model which we defined in Chapter 4, that is, by adding clocks to tokens and time restrictions to the firing of transitions. Therefore, the storage costs will be produced in the delays instead of while firing transitions. Let us define the new model formally.

Definition 6.2.1 (Priced-timed rcwf-nets) *A priced-timed rcwf-net (ptrcwf-net for short) is a ν-PTdPN $N = \langle P, T, In, Out, Time, Cost \rangle$ such that:*

- *If we define* $F, H : T \to (P \times Var)^{\oplus}$ *such that for each* $t \in T$, $p \in P$ *and* $x \in Var$, $F(t)(p,x) = In_t(x)(p)$ *and* $H(t)(p,x) = Out_t(x)(p)$, *then* $\langle P, T, F, H\rangle$ *is an rcwf-net called the underlying rcwf-net of* N.
- *For each* $t \in T$, $Time_t(\epsilon) = ([0,\infty), [0,\infty))$.

Intuitively, a ptrcwf-net is a ν-*PTdPN* from which we get an rcwf-net by removing time constraints and prices. Moreover, the age of the resources is not constrained in the firings. The behavior of this net is given by the semantics of its ν-*PTdPN*. However, we compute the prices of the runs in a different way, in order to be able to apply the price predicates to them. Instead of putting together the prices of all instances, we will compute the prices of the runs in a similar way as done for prcwf-net, by considering the multiset of costs of the instances.

Definition 6.2.2 (Price of an instance) *We define the* price of an instance $a \in Id(m_0)$ *in a run* $r = M_0 \xrightarrow{\delta_0} M_0' \xrightarrow{t_1(\sigma_1)} M_1 \xrightarrow{\delta_1} M_1' \xrightarrow{t_2(\sigma_2)} M_2 \dots M_{n-1}' \xrightarrow{t_n(\sigma_n)} M_n$ *of a ptrcwf-net as*

$$\mathcal{P}(a,r) = \sum_{\substack{i=1 \\ \sigma_i(x)=a}}^{n} Cost(t_i) + \sum_{\substack{i=1 \\ p\in P}}^{n-1} \delta_i * Cost(p) * m_a^i(p),$$

where $a{:}(m_i^a, r)$ *is an instance in* M_i *for some* $r \in \mathbb{R}_{\geq 0}$.

Now, in order to study the decidability issues for this extension, we need to define the price of a run, the soundness problem and the safety problem for the different price predicates. Initially, we wanted to consider the same definitions as for prcwf-nets. However, we found several drawbacks which made these definitions not appropriate for ptrcwf-net. We now discuss these drawbacks and explain the reasons that led us to consider the following definitions.

The behavior of ptrcwf-nets is constrained by the intervals labelling their arcs. For example, focus on the net of Fig. 6.11. Note that if we start a run by elapsing more than 1 unit of time, the run gets stuck, and the desired final marking cannot be reached. This will happen whenever there is an interval with upper bound $b \neq \infty$ labelling the only outgoing arc of a place. Therefore, the standard concept of soundness is not appropriate in our new model, since delaying too much time always causes not being able to finish runs correctly. As we do not have mechanisms as urgent transitions to ensure that the system does not delay too much, we cannot avoid this situation. In fact, a necessary condition for soundness would

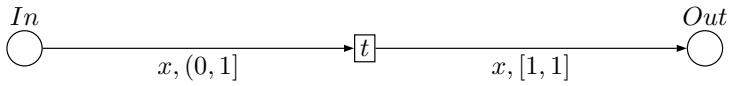


Figure 6.11: Problems with standard soundness for ptrcwf-nets.

be the existence of a path from every reachable place to *out* without "bounded" outgoing arcs from places, which is certainly too strong, because it effectively removes time constraints.

A solution to the previous problem could be considering a version of ptrcwf-net in which the elapsing of time is forbidden if some token becomes too old because of it. In this case, we could keep the previous definition of soundness. However, this option amounts to the addition of urgency to our model, which quickly leads to undecidability for most safety problems.

Summing up, we think that considering the soundness of the underlying rcwf-net instead is most suitable. Therefore, from now on, we only consider ptrcwf-nets with a proper underlying rcwf-net which is dynamically-sound (we require that the net is proper in order to ensure decidability of dynamic soundness of the underlying rcwf-net).

Although we have discarded considering a timed version of soundness, we can still check whether every run can always finish without consuming more than a given upper bound. Hence, we will focus on studying the decidability of ϕ-p-safety for the different price predicates. However, the definition of ϕ-p-safety for ptrcwf-net is also not appropriate for our new model, as happened with soundness. Focus again on the net of Fig. 6.11, which has a dynamically sound underlying rcwf-net, as we require. Suppose that the storage cost of place *in* is greater than zero, and consider the same run as before, in which an instance gets stuck in *in* because of delays. The price of this instance grows unboundedly, but the instance cannot properly complete.

In order to compute the price of a run, we will only consider the prices of the instances that have finished their task, that is, the names which have reached the place *out*. In that way, if a net is safe, we ensure that the "good" runs, in which the instances finish correctly, do not spend more than the given budget. Moreover, with this definition, timed restrictions can be seen in two ways: as a way to represent the time constraints of the systems, or a way to ensure that if we restrict the behavior of our system to satisfy these constraints, then we will

not exceed the given budget.

Definition 6.2.3 (Price of a run) *Given a run π of a ptrcwf-net starting in M_0 and finishing in M_f, we define the* price of π *as the multiset*

$$\mathcal{P}(\pi) = \{\mathcal{P}(a, r) \mid a \in Id(M_0), a{:}(\{out\}, r) \in M_f\} \in {\mathbb{R}_{\geq 0}}^{\oplus}.$$

Note that, since we consider ν-$PTdPN$s as a base for the definition of ptrcwf-nets, the price of a run is a multiset of real numbers instead of a multiset of tuples of natural numbers. Also notice that, as we are only considering nets with proper and dynamically sound underlying rcwf-net, and the time constraints only restrict the behaviors, if a token of an instance reaches the place *out* at a marking m, then there are no more tokens of this name at m.

Now, we define safety for ptrcwf-nets, which is analogous as ϕ-p-safety, considering the new definition of price of a run, that is, only taking into account the finished instances. As previously, given $k \in \mathbb{N}$ and $M_0 \in P_S^{\oplus}$, we consider M_0^k the initial marking which contains a $M_0(s)$ black tokens of age 0 in each static place $s \in P_S$, k pairwise different names in in with age 0, and d is empty for every $d \in P_D \setminus \{in\}$.

Definition 6.2.4 (ϕ-pt-safety) *Let ϕ be a price-predicate and $b \in \mathbb{N}_\omega$. We say that the ptrcwf-net N is* $\phi(b)$-pt-safe *for $M_0 \in P_S^{\oplus}$ if for each $j > 0$, every run of N starting in M_0^j, satisfies $\phi(b)$.*

Now, we study the decidability of ϕ-pt-safety for some of the price predicates defined in the previous section. We start by proving the decidability of Max-pt-safety (the definition of the predicate Max is the same as for the previous model). We do it by reducing this problem to the safety problem for ν-$PTdPN$. The main idea of the construction, depicted in Fig. 6.12 is to build a ν-$PTdPN$ with two copies of the dynamic places and transitions of the ptrcwf-net. Then, we select an instance which runs in one of the copies of the net. The rest of the instances run in the other copy. The firing and storage costs associated to the transitions and places of the copy of the net in which all the instances but one run are 0, and the costs of the other part of the net are like in the original ptrcwf-net. Hence, the price of the runs of the ν-$PTdPN$ will only take into account the selected instance. Then, asking for $Max(b)$-pt-safety in the original net is equivalent to asking for b-safety in the built ν-$PTdPN$ with the final marking which only marks the place *out* of the copy of the isolated instance with one token.

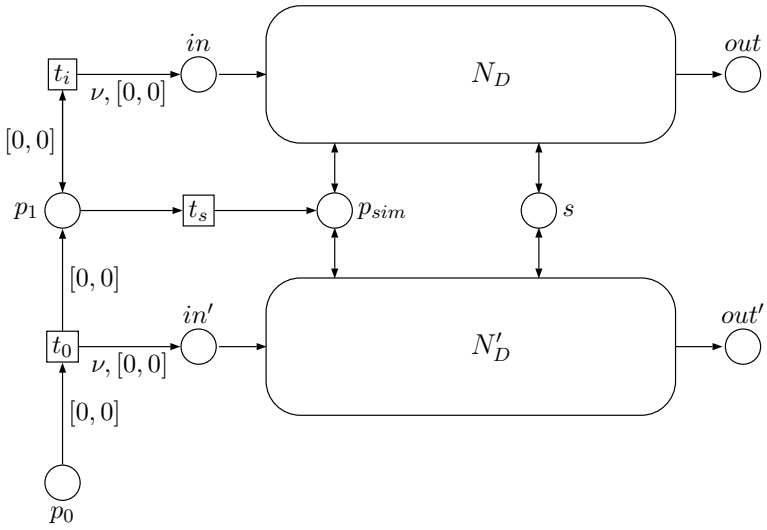


Figure 6.12: *Max* pt-safety is decidable.

Proposition 6.2.1 *Max-pt-safety is decidable for ptrcwf-nets with a proper underlying rcwf-net which is dynamically-sound.*

Proof: As mentioned before, we reduce this problem to the safety problem for ν-$PTdPN$. Let $N = \langle P, T, In, Out, Time, Cost\rangle$ be a ptrcwf-nets with a proper and dynamically-sound underlying rcwf-net, $b \in \mathbb{N}$ and $M \in P_S^{\oplus}$. Without lost of generality, we suppose that all the incoming arcs of place *out* are labelled by $[0,0]$, that is, the tokens set in *out* are of age 0 initially. Let us consider a new variable ϵ_2, and a new identifier $* \notin Id$. We build a ν-$PTdPN$ $N' = \langle P', T', In', Out', Time', Cost'\rangle$ as follows:

- $P' = P \cup \{p' \mid p \in P_D\} \cup \{p_0, p_1, p_{sim}\}$.
- $T' = T \cup \{T' \mid t \in T\} \cup \{t_0, t_i, t_s\}$.
- $In_{t_0}(\epsilon_2) = \{p_0\}$, $In_{t_i}(\epsilon_2) = \{p_1\}$ and $In_{t_s}(\epsilon_2) = \{p_1\}$. Moreover, for each $t \in T$:
  - $In'_t(x) = In_t(x)$, $In'_t(\epsilon) = In_t(\epsilon)$, $In'_t(\epsilon_2) = \{p_{sim}\}$.

  - For each $p_d \in P_D$, $In'_{t'}(x)(p'_d) = In_t(x)(p_d)$, for each $p_s \in P_S$, $In'_{t'}(\epsilon)(p_s) = In_t(\epsilon)(p_s)$, $In'_{t'}(\epsilon_2) = \{p_{sim}\}$ and $In'_{t'}(y)(p) = 0$ otherwise.

- Analogously, $Out_{t_0}(\epsilon_2) = \{p_1\}$, $Out_{t_0}(\nu) = \{in'\}$, $Out_{t_i}(\epsilon_2) = \{p_1\}$, $Out_{t_i}(\nu) = \{in\}$ and $Out_{t_s}(\epsilon_2) = \{p_s\}$. Moreover, for each $t \in T$, $x \in Var$:

  - $Out'_t(x) = Out_t(x)$, $Out'_t(\epsilon) = Out_t(\epsilon)$,$Out'_t(\epsilon_2) = \{p_{sim}\}$.
  - For each $p_d \in P_D$ $Out'_{t'}(x)(p'_d) = Out_t(x)(p_d)$, for each $p_s \in P_S$, $Out'_{t'}(\epsilon)(p_s) = Out_t(\epsilon)(p_s)$, $Out'_{t'}(\epsilon_2) = \{p_{sim}\}$ and $Out'_{t'}(y)(p) = 0$ otherwise.

- For each $t \in T$, $Time'_t(x) = Time'_{t'}(x) = Time_t(x)$. $Time_{t_0}(\epsilon_2) = Time_{t_i}(\epsilon_2) = Time_{t_s}(\epsilon_2) = Time_{t_0}(\nu) = Time_{t_i}(\nu) = ([0,0],[0,0])$. $Time'_t(y) = ([0,\infty),[0,\infty))$ otherwise.

- For each $p_d \in P_D$, $Cost'(p'_d) = Cost(p_d)$. Analogously, for each $t \in T$, $Cost'(t') = Cost(t)$. $Cost'(t) = 0$ and $Cost'(p) = 0$ otherwise.

Let us consider the initial marking M_0 with a token of name $*$ and age 0 in place p_0, $M(s)$ black tokens of age 0 in each static place $s \in P_S$ and empty elsewhere. Note that if some time elapses between the start of a run and the firing of t_s then t_s cannot be fired, and the simulation does not start. Otherwise, after the firing of t_0, a certain number n of firings of t_i and a firing of t_s, the simulation of a run of the workflow with $n+1$ instances of age 0 starts, where the n instances created by the firings of t_i run in a copy of the dynamic part of the net without prices, and the other instance, created by the firing of t_0, runs in a copy of the net for which the prices are the same as the prices in the original net. Note that the preconditions, postconditions and time constrains of the two copies of the dynamic part in the new net are the same as in the original net, and we start in a marking representing M^{n+1}. Hence, we simulate a run with $n+1$ instances. Let us suppose that the name of this first instance is a, and the marking reached after firing t_s is M_1. Then, we have that the price of a run $r = M_1 \xrightarrow{l_1} M_1 \xrightarrow{l_2} \ldots \xrightarrow{l_{k-1}} M_k$ of N' is

$$Cost(r) = \sum_{i=1}^{k-1} Cost(M_i \xrightarrow{l_i} M_{i+1}) =$$

$$\sum_{i=1,l_i\in\mathbb{R}_{\geq 0}}^{k-1} Cost(M_i \xrightarrow{l_i} M_{i+1}) + \sum_{i=1,\exists t\in T,l_i=t'}^{k-1} Cost(M_i \xrightarrow{l_i} M_{i+1}) =$$

$$\sum_{i=1,l_i\in\mathbb{R}_{\geq 0}}^{k-1} l_i * (\sum_{p\in P}\sum_{a:(m,r)\in M_i} m(p') * Cost(p')) + \sum_{i=1,\exists t\in T,l_i=t'}^{k-1} Cost(l_i) = \mathcal{P}(a,r'),$$

where r' is the run of N, in which each transition of the form t' for some $t \in T$ fired in r is replaced by the transition t. Note that the instance a can represent any instance in N, and as the underlying rcwf-net of N is proper and dynamically-sound, if a token of a reaches the place out' at a marking m, then there are no more tokens of this name at any other place of m. Therefore, N is $Max(b)$-pt-safe for initial static marking M_0 if and only if the price of every run of N' starting from M_0 and ending by a marking which marks out' is under b, that is, N' is b-safe for the pair of markings M_0, M_f, where M_f is the marking with only one token of age 0 in place out'.

□

The decidability of *Sum*-pt-safety and *Av*-pt-safety are still open for ptrcwf-nets. In order to tackle the discounted sum predicate, we need to make an alternative definition of the ordered price of a run for ptrcwf-nets. The reason is the same as the one for considering and alternative version of the price of a run: if we consider the prices of all the instances running in the net, without requiring them to have finished, then the discounted sum of the costs of the run may grow unboundedly just because of delayings. We consider the order between instances in Def. 6.1.13. However, for the ordered price of a run of a ptrcwf-net, we give an alternative definition which only consider the instances that have reached the place *out*. Formally:

Definition 6.2.5 (Ordered price of a run for ptrcwf-nets) *Given a run r of a ptrcwf-net with $Id(r) = a_1 \leq_r \cdots \leq_r a_n$, if $a_{f_1} \leq_r a_{f_2} \ldots \leq_r a_{f_k}$ are the instances that reach the place out in r, then we define the* ordered price *of r as the word $\mathcal{P}_o(r) = \mathcal{P}(a_{f_1}, r) \ldots \mathcal{P}(a_{f_k}, r) \in (\mathbb{R}_{\geq 0})^*$.*

With this definition of ordered price, we can consider the predicates Ds_λ and Fds_λ as in the previous section. Decidability of Ds_λ-p-safety remains open. However, we can tackle the decidability of *Fds*-pt-safety. Again, we perform a reduction to safety for ν-*PTdPN*, by merging the ideas of the previous proof and the

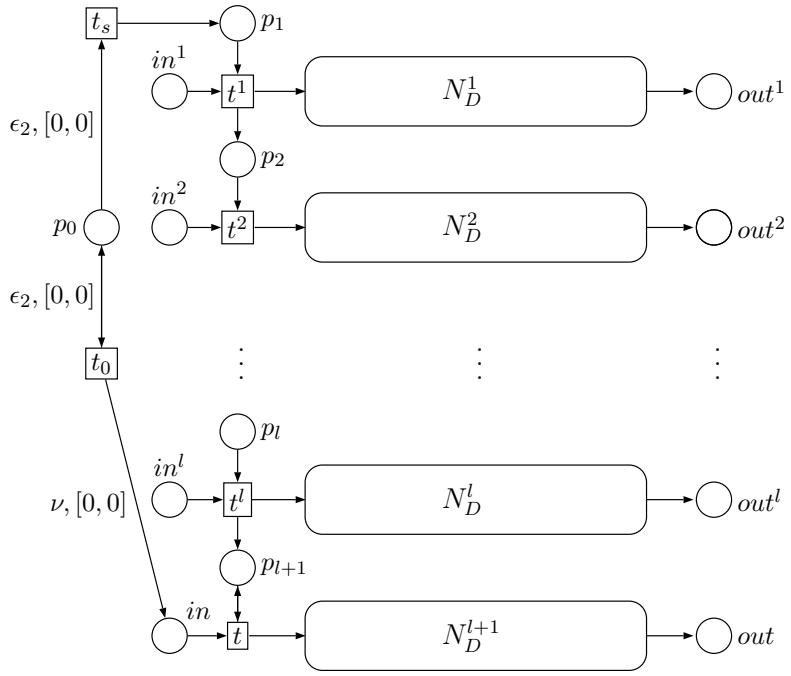


Figure 6.13: *Fds* pt-safety is decidable.

proof of decidability of *Fds*-p-safety (Prop. 6.1.11). Given N a ptrcwf-net and $l \in \mathbb{N}$, if we want to check Fds^l_λ-pt-safety for some λ, we consider $l+1$ different copies of the dynamic part of the net and an additional copy without the place *out*. A different instance will run in each of the first l copies, and the rest of the instances will run in the two remaining ones. More precisely, the instances which start before the l^{th} selected instance will run in the copy without place *out*, in order to avoid the runs in which these instances finish before the selected ones. Moreover, the costs of the i^{th} copy (where $i \leq l$) will be the costs of the original net, weighted by $(1/\lambda)^{l-i}$, in order to obtain the total cost proportional to the discounted sum. Of course, the costs of the two last copies will be 0. In that way, we will reduce Fds^l_λ-pt-safety to $(1/\lambda)^l * b$-safety of the new net we have built. Fig. 6.13 represents this construction (we have omitted some parts of the construction for simplicity).

Proposition 6.2.2 *Fds-pt-safety is decidable for ptrcwf-nets with a proper un-*

derlying rcwf-net which is dynamically-sound.

Proof: We reduce this problem to safety for ν-$PTdPN$. Let $N = \langle P, T, In, Out, Time, Cost\rangle$ be a ptrcwf-nets with a proper and dynamically-sound underlying rcwf-net, an initial static marking $M_0 \in P_S^{\oplus}$, $l \in \mathbb{N}$, $b \in \mathbb{N}$ and $0 < \lambda < 1$. Again, we consider a new variable $\epsilon_2 \notin Var$, and a new identifier $*$. We build a ν-$PTdPN$ $N' = \langle P', T', In', Out', Time', Cost'\rangle$ as follows:

- $P' = P\backslash\{out\} \cup \{p^1, \ldots, p^l, p^{l+1} \mid p \in P_D\} \cup \{p_0, p_1, \ldots, p_l, p_{l+1}\}$, that is, we consider the original places different from out, $l+1$ copies of the places of the dynamic part of the net and some accessory places.

- $T' = T\backslash\{t \mid out \in Out_t(x)\} \cup \{t^1, \ldots, t^{l+1} \mid t \in T\} \cup \{t_0, t_s\} \cup \{t_{res1}, \ldots, t_{resl}\} \cup \{t^{1'}, \ldots, t^{l'} \mid t \in T, in \in In_t(x)\}$, that is, we consider a copy of the transitions of the net which do not add any token to out, $l+1$ copies of the transitions of the net, l transitions $t_{res1}, \ldots, t_{resl}$, l additional copies of the initial transitions of the net and two more transitions t_0 and t_s.

- $In'_{t_0}(\epsilon_2) = \{p_0\}$ and $In'_{t_s}(\epsilon_2) = \{p_0\}$. For each $i \in l^+$, $In'_{t_{resi}}(x) = \{out^i\}$. For each $t \in T$:

  - Suppose $out \notin Out_t(x)$. Then, $In'_t(\epsilon) = In_t(\epsilon)$ and if $in \notin In_t(x)$ then $In'_t(x) = In_t(x)$. Otherwise, for each $i \in l^+$, $In'_{t^{i'}}(\epsilon_2) = \{p_i\}$ and $In'_{t^{i'}}(x) = \{in\}$.
  - For $i \in l^+$ and $q \in P_D$, $In'_{t^i}(x)(q^i) = In_t(x)(q)$, $In'_{t^i}(\epsilon) = In_t(\epsilon)$ and if $In_t(x) = \{in\}$ then $In'_{t^i}(\epsilon_2) = \{p^i\}$.
  - $In'_{t^{l+1}}(\epsilon) = In_t(\epsilon)$. If $q \in P_D$, $q \neq in$ then $In'_{t^{l+1}}(x)(q^{l+1}) = In_t(x)(q)$. $In'_{t^{l+1}}(x)(in) = In_t(x)(in)$. If $In_t(x) = \{in\}$ then $In'_{t^{l+1}}(\epsilon_2) = \{p^{l+1}\}$.

  and $In'_{t'}(y) = \emptyset$ otherwise.

- $Out'_{t_0}(\nu) = \{in\}$, $Out'_{t_0}(\epsilon_2) = \{p_0\}$ and $Out'_{t_s}(\epsilon_2) = \{p_1\}$. For each $i \in l^+$, $Out'_{t_{resi}}(x) = \{out^i\}$. For each $t \in T$:

  - Suppose $out \notin Out_t(x)$. Then $Out'_t(\epsilon) = Out_t(\epsilon)$. Moreover, if $in \notin In_t(x)$ then $Out'_t(x) = Out_t(x)$. Else, for each $i \in l^+$, $Out'_{t^{i'}}(x) = Out_t(x)$ and $Out'_{t^{i'}}(\epsilon_2) = \{p^i\}$.
  - For $i \in l^+$, $q \in P_D$, $Out'_{t^i}(x)(q^i) = Out_t(x)(q)$, $Out'_{t^i}(\epsilon) = Out_t(\epsilon)$. Moreover, if $In_t(x) = \{in\}$ and $i \in l^+$ then $Out'_{t^i}(\epsilon_2) = \{p^{i+1}\}$.

– $Out'_{t^{l+1}}(\epsilon) = Out_t(\epsilon)$. For each $q \in P_D$, $Out'_{t^{l+1}}(x)(q^{l+1}) = Out_t(x)(q)$. Moreover, if $In_t(x) = \{in\}$ then $Out'_{t^{l+1}}(\epsilon_2) = \{p^{l+1}\}$.

and $Out'_{t'}(y) = \emptyset$ otherwise.

- $Time'_{t_0}(\nu) = Time'_{t_0}(\epsilon_2) = Time'_{t_s}(\epsilon_2) = ([0,0],[0,0])$. For each $i \in l^+$, $Time'_{t_{resi}}(x) = ([0,\infty),[0,0])$. If $t \in T$ then $Time'_t(x) = Time'_{t^1}(x) = \ldots = Time'_{t^{l+1}}(x) = Time_t(x)$, $Time'_t(\epsilon) = Time'_{t^1}(\epsilon) = \ldots = Time'_{t^{l+1}}(\epsilon) = Time_t(\epsilon)$, if $in \in In_t(x)$ then $Time'_{t^{1'}}(x) = \ldots = Time'_{t^{l'}}(x) = Time_t(x)$ and $Time'_{t'}(y) = ([0,\infty),[0,\infty))$ otherwise.

- $p_d \in P_D$, and $i \in l^+$ $Cost'(p_d^i) = Cost(p_d) * (1/\lambda)^{l-i}$. Analogously, for each $t \in T$, $Cost'(t^i) = Cost(t) * (1/\lambda)^{l-i}$. $Cost'(t') = 0$ and $Cost'(q) = 0$ otherwise.

We consider the initial marking with an instance a_i of age 0 in each place of the form in^i, the static places marked as in $M_0 \in P_S^{\oplus}$, a token of name $*$ in place p_0 and empty elsewhere. Note that in the simulation, we can add any amount of tokens to place in, so we can simulate any amount of instances greater that l running in the net. Moreover, this instances can start before, after or between the instances represented in the l selected copies. Indeed, the instances that start before or between the selected copies are managed by the copy of the net without the place *out* (and therefore, we force that they do not finish), and the instances that start after the selected copies are managed in the copy $l+1$ of the net. The price of any run $r = M_1 \xrightarrow{l_1} M_1 \xrightarrow{l_2} \ldots \xrightarrow{l_{k-1}} M_k$ of N' is

$$Cost(r) = \sum_{i \in k-1^+} Cost(M_i \xrightarrow{l_i} M_{i+1}) =$$

$$\sum_{i \in k^+ | l_i \in \mathbb{R}_{\geq 0}} Cost(M_i \xrightarrow{l_i} M_{i+1}) + \sum_{i \in k^+ | \exists t \in T, j \in l^+, l_i = t^j} Cost(M_i \xrightarrow{l_i} M_{i+1}).$$

If $l_i \in \mathbb{R}_{\geq 0}$ then:

$$Cost(M_i \xrightarrow{l_i} M_{i+1}) = \sum_{p \in P} \sum_{j \in l^+} \sum_{a:(m,r) \in M_i} ((1/\lambda)^{l-j} * Cost(p) * l_i * m(p^j)),$$

and if there are $t \in T$, $j \in l^+$ with $l_i = t^j$, then

$$Cost(M_i \xrightarrow{l_i} M_{i+1}) = (1/\lambda)^{l-j} * Cost(t).$$

Therefore, the cost of r in N' is the sum of the cost of each instance a_j, weighted by $(1/\lambda)^{l-j}$. Moreover, note that if for $i \in l^+$, $c_i \in \mathbb{R}_{\geq 0}$, then:

$$\sum_{i=1}^{l} \lambda^i * c_i < b \Leftrightarrow \sum_{i=1}^{l} (1/\lambda)^i * c_i < b \Leftrightarrow$$

$$(1/\lambda)^l (\sum_{i=1}^{l} \lambda^i * c_i) < (1/\lambda)^l * b \Leftrightarrow \sum_{i=1}^{l} (1/\lambda)^{l-i} * c_i < (1/\lambda)^l * b.$$

Finally, note that each instance a_i in N' cannot start if the place p_i is not marked, and when it is removed from in^i a token is added to place p_{i+1}. Hence, we force that an instance a_i cannot start if the previous instance has not started yet. Therefore, we ensure that the order in which we weight the prices of the instances is the correct one.

Moreover, note that if an instance a_i has reached its corresponding final place out^i, then transition t_{resi} can be fired, setting the age of a_i to 0. Hence, if we prove that N' is safe for $b = (1/\lambda)^l$ and the final marking M_f which marks each place of the form out^i for some $i \in l^+$, with a token of a different instance a_i with age 0, then we are proving that the ordered price of every run of N in which l instances reach the place out, satisfies $Fds^l_\lambda(b)$. However, this would not ensure that N is $Fds^l_\lambda(b)$-pt-safe, because we are missing the runs in which less that l instances finish correctly, that may not satisfy the price predicate. Nevertheless, it is not difficult to fix this problem. We just have to consider safety for the l ν-$PTdPN$s which only consider the first $1, 2, \ldots, l$ copies of the dynamic part of the net in N', with the same temporal restrictions and costs as N'. Hence, if all of these nets are safe for $b = (1/\lambda)^l$ and the final marking which marks each place of the form out^i, for some $i \in l^+$, with a token of a different instance a_i, then N is $Fds^l_\lambda(b)$-pt-safe.

□

Unfortunately, the problems of *Sum*-pt-safety and Av-pt-safety are still open for ptrcwf-nets, even with a proper and dynamically-sound underlying rcwf-net. The finite version of these problems could be proved to be decidable by applying the same techniques as in the previous proofs, that is, by building the same number of copies of the dynamic part of the net as the number of instances that we need to consider for the price, and adding their prices accordingly to the price predicate we are considering. However, the finite versions of these problems are not very

interesting for *Sum* and *Av*, so that we will not discuss them further.

6.3 Priced rcwf-nets in practice

Studying the decidability of a problem is only the first step towards the implementation of a tool for its verification that works in practice. Despite the good decidability results we have achieved, the complexity of the problems we have studied is too high to directly implement algorithms for them that terminate in a reasonable time in all cases. Although in this thesis we focus on the theoretical decidability aspects of these problems, in this section we discuss some ideas about how we could tackle these problems in a more practical way in order to show that the models we have defined can still be useful and interesting in practice.

There are already plenty of tools, such as Woflan [96, 102], YAWL [91], Wolfgang (http://doku.telematik.uni-freiburg.de/wolfgang) or TAPAAL [52] which address the analysis of timed Petri nets, workflows, resource-constrained workflows and timed workflows. Now, we sketch how we could adapt the algorithms in these tools to give (partial) solutions to our problems.

Suppose we have an efficient algorithm to solve a certain problem for *TdPN* or timed workflows, as the algorithms in TAPAAL, for example. We can consider a bound b for the number of instances, that is, considering only runs with b or less instances. Then, if we want to solve the same problem for a *ν-lsPN* or a ptrcwf-net, we could adapt the previous algorithms in order to analyze our nets by building b different copies of the net, handling each instance in a different copy and simulating the creation of a new instance by the addition of a new token to a copy that has not been used before. In that way (but taking care of the particular issues of the problems and models we consider), we could solve the same problems without *significantly* increasing the complexity of the algorithms, assuming a bound for the number of instances. Moreover, as many of the algorithms consist in performing forward searches, in case we reach a certain marking which enables a transition that creates a new instance, but we have already used the b copies of the net, we could add a message to the result of the algorithm informing that we could have explored markings with more instances, and suggesting to run the algorithm providing a greater bound. Of course, if the net is bounded on the number of instances and we provide enough copies of the net, the algorithm would return a total solution for the problem. Next, we show an example of a possible application of the previous idea:

Example 6.3.1 *A timed workflow net is a TdPN such that if we remove the time constraints then we obtain a workflow net. The concept of (weak) soundness is defined analogously as for workflow nets: A workflow is sound if we can reach the final marking with a token in out from every marking M which is reachable from the initial marking with only a token in in. The algorithm in [64] studies the soundness of discrete-timed workflows and returns the minimum time that has to be spent to reach the final state. These algorithm is the following one:*

1 *INPUT: A timed workflow net* $N = \langle P, T, F, H \rangle$ *with* $in, out \in P$.

2 *OUTPUT: ''true'' together with the minimum execution time if* N *is sound; ''false'' otherwise.*

3 *BEGIN A marking M has an (initially emptyset) set of its parents* $M.parents$ *and a minimum execution time* $M.min$ *(initially* ∞*);*
$M_{in} := \{(in, 0)\}$; $Waiting := \{M_{in}\}$; $M_{in}.min = 0$; $Reached := Waiting$;
$Final := \emptyset$;

4 *while* $Waiting \neq \emptyset$ \textbf{do}
5 *Remove some marking* M *from* $Waiting$ *with the smallest* $M.min$.
6 *for each* M' *s.t.* $M \xrightarrow{1} M'$ *or* $M \xrightarrow{t} M'$ *for some* $t \in T$ *do*
7 $M'_c := cut(M')$; $M'_c.parents := M'_c.parents \cup \{M\}$;
8 *if* $M \xrightarrow{1} M'$ *then* $M'_c.min = MIN(M'c.min, M.min + 1)$;
9 *else* $M'_c.min = MIN(M'_c.min, M.min)$;
10 *if* $|M'_c(out)| \geq 1$ *then*
11 *if* M'_c *is a final marking then* $Final := Final \cup \{M'_c\}$;
12 *else return false;*
13 *else*
14 *if* $M'_c \notin Reached$ *then*
15 *if* M'_c *is a deadlock then return false;*
16 *if* $\exists M'' \in Reached$ *with* $cut(M'') \sqsubseteq cut(M'_c)$
then return false;
17 $Reached := Reached \cup \{M'_c\}$; $Waiting := Waiting \cup \{M'_c\}$;
18 $Waiting := Final$;
19 *while* $Waiting \neq \emptyset$ *do*
20 *Remove some marking* M *from* $Waiting$;

```
21        Waiting := Waiting ∪ (M.parents ∩ Reached);
22        Reached := Reached \ M.parents;
23    if Reached = ∅ then
24        time := ∞; for each M ∈ Final do time = MIN(time, M.min);
25        return true and time;
26    else
27        return false;
28 END
```

where, given a marking M, $cut(M)$ is the marking obtained by replacing the tokens in M older than max *by tokens of age* max $+1$.

Intuitively, the algorithm performs a forward search on the markings (lines from 4 to 17) abstracting the ages older than max *to* max $+1$, *and storing in $Waiting$ and Reached the discovered markings and the already explored markings, respectively. At the same time, the algorithm computes the shortest path between the initial marking and the reachable markings (lines 8 and 9). The algorithm terminates if it reaches a marking covering an already discovered marking (that is, if the workflow is not bounded). If we reach a final marking with a token in out which is not a final marking, the algorithm returns "false" (line 12). Otherwise, the marking is added to $Final$. If the marking we reach does not mark out and it is a deadlock then the algorithm returns "false" too (line 15). If the algorithm does not halt in the first phase, then it performs a backward search from the set $Final$, to check if there is a path from every reachable marking to a final marking (lines from 19 to 22). In that case, the algorithm returns "true" and the minimum execution time.*

A first step to adapt this algorithm for ptrcwf-nets with runs with a maximum number of instances b is to consider a new TdPN N_b with b copies of the dynamic part of the net, all sharing the static part. Moreover, we add two new places in and out, which work as the initial and final place of the workflow we want to build, respectively. Then, for each $i \in \{1, \ldots, b\}$ we add a transition, which removes a token from in and adds a token in the initial place of the i first copies of the net. Analogously, for each $i \in \{1, \ldots, b\}$, we add a transition which removes a token from the i first copies of the net and adds a token in out.

In that way, we have built a new timed workflow net without names. The algorithm cannot be applied to this net yet, mainly because of the storage costs, which do not need to be bounded in order to ensure soundness. However, a possible solution to this problem could be abstracting the value of the costs that may grow

unboundedly by ω.

Moreover, we could also adapt this algorithm in order to compute the maximum cost of a complete run, hence obtaining the safety or unsafety of the net for the sum. It would be enough to compute the maximum cost of reaching a marking instead of the minimum time spent in reaching it, that is, replacing $M'_c.min = MIN(M'c.min, M.min+1)$ *by* $M'_c.max = MAX(M'c.max, M.max + SC(M)$ *in line* 8*, where* $SC(M)$ *are the storage costs produced by a delay of* 1 *unit of time from* M*, and* $M'_c.min = MIN(M'_c.min, M.min)$ *by* $M'_c.max = MAX(M'_c.max, M.max + FC(t))$ *in line* 9*, where* $FC(t)$ *are the firing costs of the transition fired to reach* M'.

Another characteristic of some of the algorithms that are already used that we would need to adapt (as the previous example) is their managing of discrete time instead of real time. Since we have defined discrete models of regions which represent our (real timed) models in a good way to study safety properties, we could study these properties over regions instead of over the initial models. For example, in the case of ν-$PTdPN$, we have proved that in order to study the safety problem for them, we only need to focus on the continuous firings which are close to an integer. Hence, we could try to adapt the algorithms for discrete timed models, in which the steps are of size 1, to use them for ν-$aPTdPN$ (restricting the number of instances too).

Finally, the specific characteristics of Petri nets can be taken into account in order to reduce the number of states we need to visit or store. As explained in [104], several methods have been developed to improve the generation of state spaces of Petri nets. For example, partial order reduction techniques [90, 70, 41] consist in suppressing as many interleaved firings of concurrently enabled transitions as possible, the symmetry method [85, 86, 54] uses symmetry to suppress the consideration of a state if a symmetric state has been visited before, the sweep-line method [22, 59] assumes that there is some notion of progress and goes over the search space in order of increasing progress values, and cycle coverage [61] saves space by increasing the execution time, storing less states, having to compute the successors of forgotten states again. All of these ideas should be studied and applied to our new models when considering a more practical point of view.

Chapter 7

Conclusions

In this chapter we conclude this thesis by summarizing our results from a more global perspective. Moreover, we sketch the main future lines of work, and the possible applications of our mostly theoretical results. Finally, we give a list of publications related with this thesis, which validate this work among the scientific community.

7.1 Summary and contributions

The main goal of this thesis was to enrich the already existing extensions of Petri nets in order to define other extensions which could manage the representation and verification of very natural properties of real systems such as the management of time and costs, and the concurrency between different copies of the same process. We have defined several extensions of Petri nets dealing with these properties. Moreover, as we presented the different extensions, we have studied the decidability of safety properties for them, and we have compared their expressiveness with the expressiveness of other extensions which are already defined and well studied in the literature. Finally, in the last part of the thesis, we have focused on workflow nets, broadening the decidability results about soundness that already exist for resource-constrained workflow nets, and extending them with time and costs, making use of the previously defined extensions.

Fig. 7.1 summarizes the relations between the models that we have defined in this thesis. A solid arrow from some model to another represents that the second model extends the first one. The dashed arrow from ν-$PTdPN$ to ptrcwf represents that the second model has been built by restricting the first one. Finally, the dashed arrow from ν-$lsPN$ to ν-$PTdPN$ expresses that the second model has

been built by extending the first one with costs.

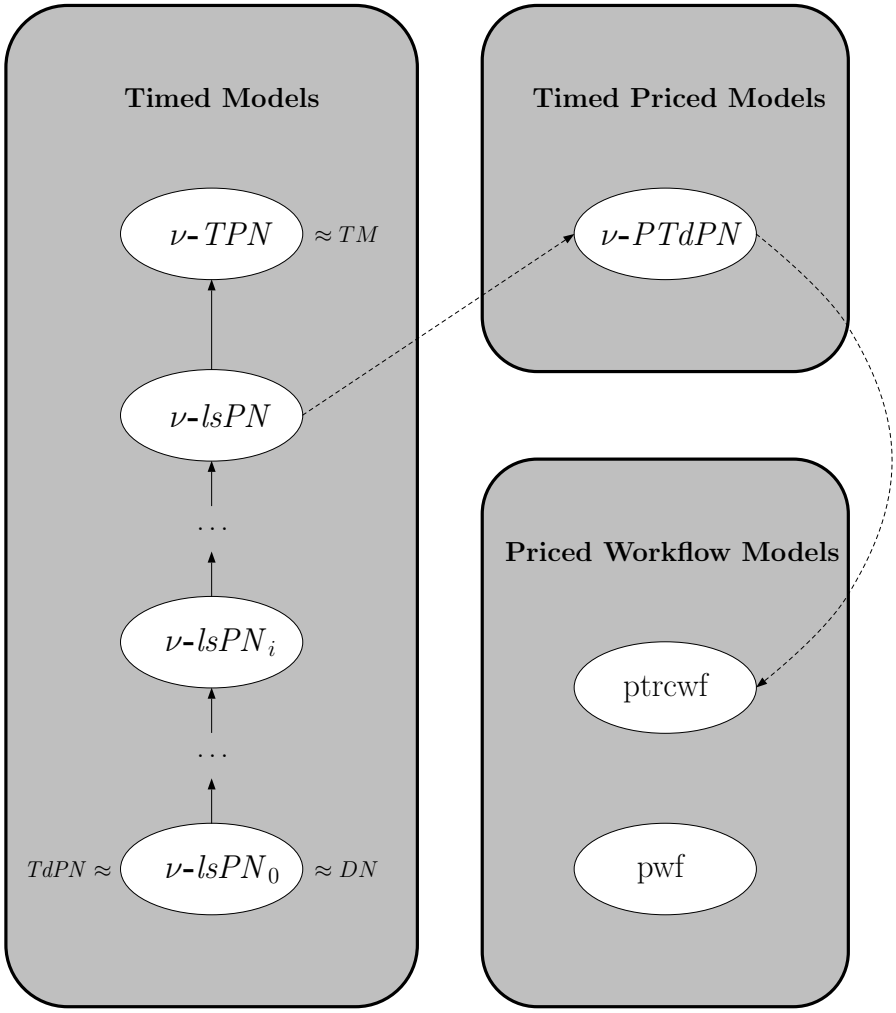


Figure 7.1: Overview of defined models

In Chapter 3 we have defined several extensions of Petri nets, based on the already existing models of ν-Petri nets and Timed Petri net. These extensions encompass the management of different instances running concurrently in the net, and the management of time. Hence, they are quite appropriate to model real systems in which the behavior is constrained by time and in which several processes are considered. We have studied the decidability of safety properties for the extensions we have defined. We have found that if we consider more than

one clock per instance in our model (ν-TPN), safety is undecidable. In fact, we have performed a reduction from a Turing-complete model with both names and different instances of the net, by representing the different instances by clocks. We claim that this technique is applicable to build reductions from different models in which we represent different instances (maybe ordered) by clocks or vice versa. Moreover, we prove that control-state reachability is undecidable even for ν-TPN with only two tokens per instance. Hence, from that point on, we have focused on the extension with one clock per instance (ν-$lsPN$), which is still able to represent significant real systems reliably (we give an example of a parametric version of Fischer's protocol). We have proved that control-state reachability for ν-$lsPN$ is decidable.

At that point, we became interested in the place of ν-$lsPN$ in the picture of expressiveness among other models for which safety is decidable (in fact, among WSTS). Models such as Data-nets, ν-Petri nets or Timed Petri nets are all more expressive than Petri nets. We have proved that ν-$lsPN$ are more expressive than all of these models, and still not Turing-complete. More precisely, we have proved that bounded ν-$lsPN$ are as expressive as TPN or Data nets, and the expressive power grows with the number of unbounded places we consider. In fact, ν-$lsPN$ is the most expressive WSTS extending Petri nets, up to our knowledge. Hence, the definition of this extension takes an additional importance as a theoretical model, in order to thin the line between WSTS and Turing machines, and exploring systems with levels of expressiveness not found before among well structured Petri net extensions.

Going back to the need of defining extensions to model the characteristics of real systems, in Chapter 4 we have defined a new model (ν-$PTdPN$) based on ν-$lsPN$ and the costs model of Priced-Timed Petri nets, which adds prices to the performance of actions and to the storage of resources in the runs. Again, this point of view is quite intuitive and corresponds to how costs are produced in many real systems, which need to pay not only for the use of machines, workers performing tasks or even resources spent while performing actions, but also for storing and conserving the materials and resources while, for example, other activities are performed. As previously, safety of these systems is an important concern. As the semantics of ν-$PTdPN$ is defined as for ν-$lsPN$, the safety problem which we consider for ν-$lsPN$ is decidable for our new model too. However, this time the safety properties that should be considered have also something to do with the costs of the executions, since in many cases, besides achieving the goal of the

systems without reaching pernicious states, these systems are required to stay on a given budget. That is the reason why we have defined a specific priced safety property for ν-*PTdPN*. Basically, a ν-*PTdPN* is price safe if each run which covers a desired marking does not cost more than a given budget. Again, we have proved that this problem is decidable. Hence, we successfully obtained the decidability of safety, where safety means avoiding both too expensive runs and runs reaching non desired states.

In the second part of the thesis we have focused on a concrete application of some of the models that we have defined (ν-*PTdPN*), or that already existed (ν-PN). We take advantage of the theoretical results about these models to study the decidability of soundness and safety problems for workflow nets. First, we exploit the capability of ν-PN to represent different processes running in the same net to express rcwf nets in terms of them. Moreover, we broaden the decidability results about soundness of rcwf nets in the literature by applying decidability results for ν-PN.

As workflows frequently represent business processes, it makes sense to consider prices over their runs, and to study priced safety properties over them. We take the same price model that we took for ν-*PTdPN*, by adding storage and firing costs to rcwf nets. We define two extensions which add prices to the runs of the rcwf nets in two ways: First, we consider that time only elapses while performing actions, and therefore we are not allowed to wait between the firing of different transitions. The second point of view considers that time elapses in between actions. For both extensions, we consider that the price of a run is the multiset of the prices produced by the different instances in the net. In that way, we can define the priced safety problem in a parametric way, considering different ways to add the prices of the different instances, computing a final combined price. We study the decidability of safety for different predicates: *Sum*, *Max*, *Av*, *Ds* and *Fds*, getting positive results for the cases of *Sum*, *Max*, *Av* and *Fds* in the case of the first extension, and *Max* and *Fds* in the case of the second extension.

7.2 Future lines of work

There are several ways in which we want to extend our work. First of all, there are still open decidability problems among the safety properties we have studied. Specifically, the decidability of *Ds*-safety for prcwf-nets and *Sum*, *Av* and *Ds*-safety for ptrcwf-nets remain open. Of course, as the definition of ptrcwf-nets

relies on restricted ν-*PTdPN*, and we have focused only on safety and priced safety problems for them, more theoretical work about the decidability of different properties for ν-*PTdPN*, such as liveness properties for certain restrictions of ν-*PTdPN*, could help us to achieve this goal.

Regarding complexity, since ν-*lsPN* are more expressive than *TdPN*, we know that the complexity of the control-state reachability problem is non-primitive recursive. More precisely, we can already obtain a lower bound at level $F_{\omega^{\omega^{\omega}}}$ [43] in the fast-growing hierarchy. Moreover, it is easy to reduce control-state reachability for ν-*lsPN* to priced safety for ν-*PTdPN*, and hence this problem is non-primitive recursive too. However, we would like to know if this lower bound is tight, though we expect it is not, due to the higher order types of the state space in ν-*lsPN*, and to obtain a finer-grained complexity analysis for both problems, as done in [43].

In this thesis we have focused on the study of the decidability of safety properties for the models we have defined. Nevertheless, it would be interesting to study other properties. For example, as our models deal with time, properties as the existence of Zeno behaviors [8] could be verified. Moreover, as Zeno behaviors are not present in most of the real systems, the non-existence of these behaviors could be required to define a significant restriction of our model. As some of the runs in the constructions of our proofs must exhibit this behavior, the models obtained by applying such a restriction could have different decidability properties from the ones we have found for our models. Other directions for further study could include the decidability of liveness properties, although negative results in the untimed case are discouraging [80], or the study of models with discrete time. To study the relation of our model with other models defined in literature would also be interesting. For instance, we could study the relation of ν-*lsPN* to the existing works that model GALS (globally asynchronous locally synchronous) systems using Petri nets [56]. In a different line, we have assumed that processes (or their identifiers) are not ordered in any way. It would be interesting to see whether our work scales in the case of ordered processes, which amounts to extend Data Nets [62] with time.

Finally, besides completing and widening our theoretical results, we consider that it is necessary to bring the models we have defined to a tool, in order to make them available for verifying real systems. We plan to implement the priced and timed models we have considered, maybe by extending the tool TAPAAL (http://www.tapaal.net/), which is a tool for the analysis and verification of bounded *TdPN* and workflows. As a first step, we are now extending the tool

to consider prices over *TdPN* and to add an algorithm dealing with price safety for workflow. In order to implement our models with names, we need to work on data structures and algorithms to represent and verify their bounded restrictions.

Appendix A

Effective $Pred$-basis of the transition relation $\twoheadrightarrow$ defined for regions of ν-$lsPN$

We are going to define a function $\overline{Pre}$ to compute the predecessors of a region. We split Pre into $Pre_\Delta(R) = \{R' \mid R' \overset{\Delta}{\twoheadrightarrow} R\}$ and $Pre_t(R) = \{R' \mid R' \overset{t}{\twoheadrightarrow} R\}$, and we define $\overline{Pre}_\Delta$ and $\overline{Pre}_t$ for each $t \in T$, so that $Pre_\Delta(\uparrow R) = \uparrow \overline{Pre}_\Delta(R)$ and $Pre_t(\uparrow R) = \uparrow \overline{Pre}_t(R)$. First, we define $\overline{Pre}_\Delta$, the function that computes the predecessors corresponding to time delays, using in turn $\overline{Pre}_\delta$ as an auxiliary function, which corresponds to small time delays. Then, $\overline{Pre}_\Delta$ will be defined as the reflexive and transitive closure of $\overline{Pre}_\delta$. Given $A = \{(a_1, r_1), \ldots, (a_n, r_n)\} \in (P^\oplus \times \mathbb{N})^\oplus$, with $r_1, \ldots, r_n > 0$, we define $A^{-1} = \{(a_1, r_1 - 1), \ldots, (a_n, r_n - 1)\}$.

Definition A.1 ($\overline{Pre}_\delta$) *Let $R = A_0 * A_1 * \ldots * A_n * A_\infty$. We define $\overline{Pre}_\delta(R)$ (and extend it pointwise) as*

$$\begin{cases} \{(A_1 + B_0^{-1}) * A_2 * ... * A_n * B_\infty, \\ \quad B_0^{-1} * A_1 * ... * A_n * B_\infty \mid A_\infty = B_0 + B_\infty\} \textit{ if } A_0 = \emptyset \\ \\ \{\emptyset * A_1 * A_2 * ... * A_n * A_0^{-1} * A_\infty\}, \textit{ if } A_0 \neq \emptyset \textit{ and } A_0^{-1} \textit{ is well defined} \\ \\ \emptyset, \textit{ otherwise} \end{cases}$$

Note that $\overline{Pre}_\delta$ is $\emptyset$ in case A_0^{-1} is not well defined, that is, if the age of some instance represented in A_0 is 0. That makes sense, since intuitively, an instance

of age 0 cannot be younger.

We obtain the definition of $\overline{Pre}_\Delta$ by considering the union of the regions obtained by applying $\overline{Pre}_\delta$ recursively to a region.

Definition A.2 ($\overline{Pre}_\Delta$) *We define:*

- $\overline{Pre}_\delta^{\,0}(\mathcal{R}) = \mathcal{R}$
- $\overline{Pre}_\delta^{\,i+1}(\mathcal{R}) = \overline{Pre}_\delta^{\,i}(\mathcal{R}) \cup \overline{Pre}_\delta(\overline{Pre}_\delta^{\,i}(\mathcal{R}))$ *and*
- $\overline{Pre}_\Delta(R) = \bigcup_{i\geq 0} \overline{Pre}_\delta^{\,i}(\{R\})$.

Example A.1 *Focus on the regions depicted in Fig. A.1, and let* $\max = 1$. *We are going to compute* $\overline{Pre}_\Delta(R_1)$. *If we apply the second case of Def. A.1, we obtain* $\overline{Pre}_\delta(R_1) = R_2$, *and hence* $\overline{Pre}_\delta^{\,1}(R_1) = \{R_1, R_2\}$. *Analogously, if we apply the first case of Def. A.1, we can consider* $B_0 = \emptyset$ *or* $B_0 = \{< \{q\}, 2 >\}$, *so we obtain* $\overline{Pre}_\delta(R_2) = \{R_2, R_3, R_4, R_5\}$, *and therefore* $\overline{Pre}_\delta^{\,2}(R_1) = \{R_1, R_2, R_3, R_4, R_5\}$. *If we continue calculating until* $\overline{Pre}_\delta$ *is the empty set for any new* R_i *in the* $\overline{Pre}_\delta^{\,j}(R_1)$ *we are considering, we obtain* $\overline{Pre}_\Delta(R_1) = \{R_1, R_2, R_3,\ R_4, R_5, R_6, R_7\}$.

Now we prove that $\overline{Pre}_\Delta$ is defined as needed, which intuitively means that, given a region R, $\overline{Pre}_\Delta$ is useful to compute $Pre_\Delta(\uparrow R)$ in the sense that $\overline{Pre}_\Delta(R)$ is finite and its upward closure is $Pre_\Delta(\uparrow R)$. First, we tackle the finiteness.

Lemma A.1 *Given a region* R, $\overline{Pre}_\Delta(R)$ *is finite.*

Proof: For any $R = A_0 * A_1 * ... * A_n * A_\infty$ we define $size(R) = (r, i, |A_\infty|) \in n^* \times n^*_\infty \times \mathbb{N}$, where $(r, i) = \min\{(r, i) \mid (m, r) \in A_i,\ i \in n^*\}$, where the pairs (r, i) are ordered lexicographically, and we also compare tuples $size(R)$ lexicographically. If $size(R) > (0, 0, 0)$ one of the following holds:

- $size(R) = (r, 0, s)$, with $r > 0$: then $\overline{Pre}_\delta(R) = \{R'\}$ with $R' = \emptyset * A_1 * A_2 * ... * A_n * A_0^{-1} * A_\infty$ and $size(R') = (r - 1, n, s)$.
- $size(R) = (r, i, s)$ with $0 < i \leq n$: then the ages in $A_0, ..., A_{i-1}$ are at least $r + 1$. The case $A_0 \neq \emptyset$ is analogous to the previous one: there is only one region R' in $\overline{Pre}_\delta(R)$, but now $size(R') = (r, i - 1, s)$. If $A_0 = \emptyset$ then any R' in $\overline{Pre}_\delta(R)$ is either of the form $(A_1 + B_0^{-1}) * A_2 * ... * A_n * B_\infty$ or $B_0^{-1} * A_1 * ... * A_n * B_\infty$, with $A_\infty = B_0 + B_\infty$, so that $size(R')$ is either $(r, i - 1, s')$ in the first case, or (r, i, s') in the second case. Notice also that in the second case, if $R \neq R'$ then $s' < s$.

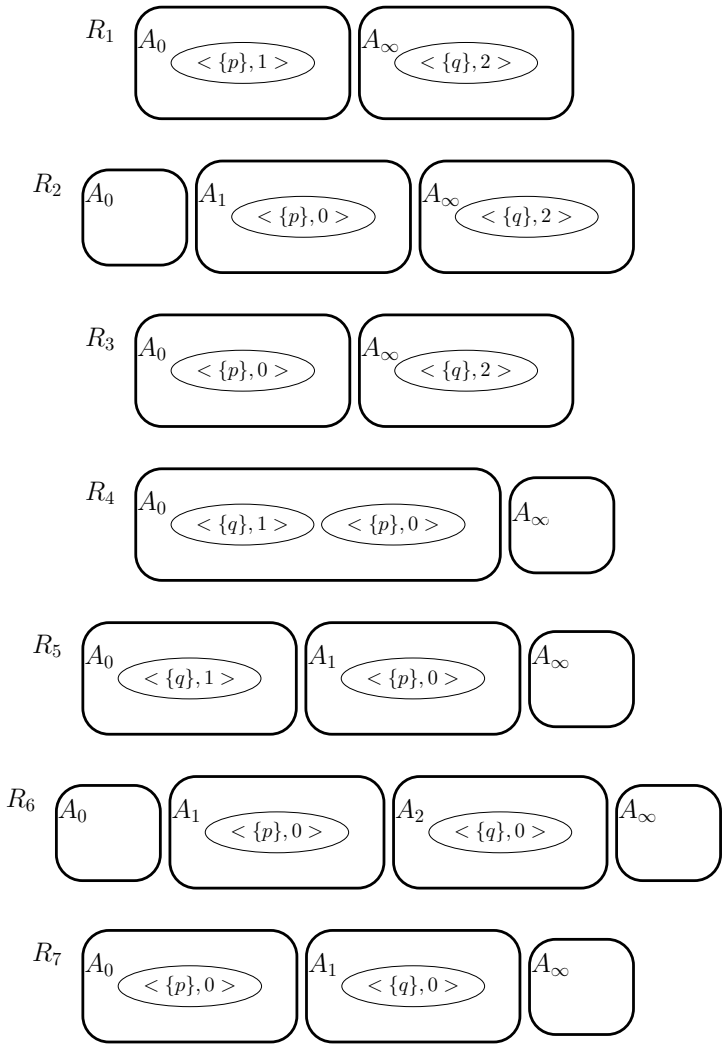


Figure A.1: $\overline{Pre}_\Delta(R_1) = \{R_1, R_2, R_3, R_4, R_5, R_6, R_7\}$

- If $size(R) = (\max + 1, \infty, s)$ then $R = \emptyset * A_\infty$ and every R' in $\overline{Pre}_\delta(R)$ is of the form $R' = B_0^{-1} * B_\infty$ with $A_\infty = B_0 + B_\infty$. Notice that if $B_0 = \emptyset$ then $R = R'$. Otherwise, $size(R') = (\max, 0, s')$.
- If $size(R) = (0, 0, s)$ then A_0^{-1} is undefined, and $\overline{Pre}_\delta(R) = \emptyset$.

Assume by contradiction that $\overline{Pre}_\Delta(R)$ is infinite. Then there is a sequence $(R_i)_{i \geq 0}$ of pairwise different regions such that $R_{i+1} \in \overline{Pre}_\delta(R_i)$. By the previous items notice that $size(R_{i+1}) < size(R_i)$, which is a contradiction because the

lexicographic order is well-founded in $n^* \times n^*_\infty \times \mathbb{N}$.

□

Let us now prove we can use $\overline{Pre}_\Delta(R)$ in order to compute $Pre_\Delta(\uparrow R)$.

Lemma A.2 *Given a region R, $\uparrow\overline{Pre}_\Delta(R) = Pre_\Delta(\uparrow R)$.*

Proof: Let us first prove by induction that $Pre_\Delta(\uparrow R) \subseteq \uparrow\overline{Pre}_\Delta(R)$, for which it is enough to see that $Pre_\delta(\uparrow R) \subseteq \uparrow\overline{Pre}_\Delta(R)$ (base case) and if $R_1 \in \uparrow\overline{Pre}_\Delta(R)$ then $Pre_\delta(R_1) \subseteq \uparrow\overline{Pre}_\Delta(R)$ (inductive step).

- Base case: Let $R = A_0 * A_1 * ... * A_n * A_\infty$, R' and R'' such that $R'' \overset{\delta}{\twoheadrightarrow} R'$ with $R \sqsubseteq R'$. We want to prove that $R' \in \uparrow\overline{Pre}_\Delta(R)$. Since $R \sqsubseteq R'$ we can write $R' = B_0 * u_0 * B_1 * ... * B_n * u_n * B_\infty$ with $A_i \leq^\oplus B_i$. We distinguish three cases:

  (i) If $A_0 \neq \emptyset$ then $B_0 \neq \emptyset$, in which case $R'' = \emptyset * u_0 * B_1 * ... * B_n * u_n * B_0^{-1} * B_\infty$, which is greater than $\emptyset * A_1 * ... * A_n * A_0^{-1} * A_\infty \in \overline{Pre}_\delta(R) \subseteq \overline{Pre}_\Delta(R)$.

  (ii) If $A_0 = \emptyset$ and $B_0 \neq \emptyset$ then $R'' = \emptyset * u_0 * B_1 * ... * B_n * u_n * B_0^{-1} * B_\infty \in \uparrow R \subseteq \uparrow\overline{Pre}_\Delta(R)$.

  (iii) Finally, if $A_0 = B_0 = \emptyset$ we distinguish two subcases. If $u_0 = \epsilon$ then $R'' = (B_1 + C_1) * u_1 * B_2 * ... * B_n * u_n * C_2$ with $C_1^{+1} + C_2 = B_\infty$, which is greater than $(A_1 + D_1) * A_2 * ... * A_n * D_2 \in \overline{Pre}_\delta(R)$ for some $D_1^{+1} + D_2 = A_\infty$. If $u_0 \neq \epsilon$ then $u_0 = B * u_0'$, in which case $R'' = (B + C_1) * u_0' * B_1 * ... * B_n * u_n * C_2$ with $C_1^{+1} + C_2 = B_\infty$, which is greater than $D_1 * A_1 * ... * A_n * D_2 \in \overline{Pre}_\delta(R)$ for some $D_1^{+1} + D_2 = A_\infty$.

- Inductive step: Now, suppose that $R_1 \in \uparrow\overline{Pre}_\Delta(R)$. Then, there is $R_2 \in \uparrow\overline{Pre}_\Delta(R)$ such that $R_2 \sqsubseteq R_1$. Hence, there is a region $R_3 \in \overline{Pre}_\Delta(R)$ such that $R_3 \sqsubseteq R_2 \sqsubseteq R_1$. Again, if $R_3 = A_0 * A_1 * ... * A_n * A_\infty$, we can write $R_1 = B_0 * u_0 * B_1 * ... * B_n * u_n * B_\infty$ with $A_i \leq^\oplus B_i$. Analogously as in the base case, we analyze the different subcases:

  (i) If $A_0 \neq \emptyset$ then $B_0 \neq \emptyset$. Therefore, $Pre_\delta(R_1)$ consist of the only region $R_1' = \emptyset * u_0 * B_1 * \ldots * B_n * u_n * B_0^{-1} * B_\infty$, in case B_0^{-1} is defined and no region otherwise. If $Pre_\delta(R_1) = \emptyset$, then $Pre_\delta(R_1) \in \uparrow\overline{Pre}_\Delta(R)$. Otherwise, $\overline{Pre}_\delta(R_3)$ consists of only one region $R_3' = A_1 * \ldots * A_n * A_0^{-1} * A_\infty$, which is in $\overline{Pre}_\Delta(R)$. Clearly, $R_3' \sqsubseteq R_1'$, so $Pre_\delta(R_1) = \{R_1'\} \subseteq \uparrow\overline{Pre}_\Delta(R)$.

(ii) Now, suppose $A_0 = \emptyset$ and $B_0 \neq \emptyset$. As in the previous case, $Pre_\delta(R_1)$ consist of the only instance $R'_1 = \emptyset * u_0 * B_1 * \ldots * B_n * u_n * B_0^{-1} * B_\infty$, which is clearly greater than $R_3 = \emptyset * A_1 * ... * A_n * A_\infty \in \overline{Pre}_\Delta(R)$. Therefore, $Pre_\delta(R_1) = \{R'_1\} \subseteq \uparrow\overline{Pre}_\Delta(R)$.

(iii) Finally, let $A_0 = B_0 = \emptyset$. We consider two different subcases:

a) If $u_0 = B_{00} * u'_0$, with $B_{00} \neq \emptyset$, then $Pre_\delta(R_1) = \{B_{00} + B_{\infty 1}^{-1} * u'_0 * B_1 * \ldots * B_n * u_n * B_{\infty 2}, B_{\infty 1}^{-1} * u_0 * B_1 * \ldots * B_n * u_n * B_{\infty 2} \mid B_\infty = B_{\infty 1} + B_{\infty 2}\}$. Consider $R'_1 = B_{00} + B_{\infty 1}^{-1} * u'_o * B_1 * \ldots * B_n * u_n * B_{\infty 2}^{-1} \in Pre_\delta(R_1)$ with $B_\infty = B_{\infty 1} + B_{\infty 2}$. As $A_\infty \leq^\oplus B_\infty$, we can split $A_\infty = A_{\infty 1} + A_{\infty 2}$ with $A_{\infty 1} \leq^\oplus B_{\infty 1}$ and $A_{\infty 2} \leq^\oplus B_{\infty 2}$, and then, if we define $R'_3 = A_{\infty 1}^{-1} * A_1 * \ldots * A_n * A_{\infty 2} \in \overline{Pre}_\Delta(R)$, we have that $R'_3 \sqsubseteq R'_1$. Analogously, if $R'_1 = B_{\infty 1}^{-1} * u_0 * B_1 * \ldots * B_n * u_n * B_{\infty 2}$ with $B_\infty = B_{\infty 1} + B_{\infty 2}$, we can split $A_\infty = A_{\infty 1} + A_{\infty 2}$ with $A_{\infty 1} \leq^\oplus B_{\infty 1}$ and $A_{\infty 2} \leq^\oplus B_{\infty 2}$. Again, we define $R'_3 = A_{\infty 1}^{-1} * A_1 * \ldots * A_n * A_{\infty 2} \in \overline{Pre}_\Delta(R)$, so $R'_3 \sqsubseteq R'_1$. Hence, $Pre_\delta(R_1) \subseteq \uparrow\overline{Pre}_\Delta(R)$.

b) If $u_0 = \emptyset$, then $R_1 = \emptyset * B_1 * ... * B_n * u_n * B_\infty$. Therefore, $Pre_\delta(R_1) = \{B_1 + B_{\infty 1}^{-1} * u_1 * B_2 * \ldots * B_n * B_{\infty 2}, B_{\infty 1}^{-1} * B_1 * u_1 * B_2 * \ldots * B_n * B_{\infty 2} \mid B_\infty = B_{\infty 1} + B_{\infty 2}\}$. Let us consider $R'_1 = B_1 + B_{\infty 1}^{-1} * u_1 * B_2 * \ldots * B_n * B_{\infty 2}$ and $R''_1 = B_{\infty 1}^{-1} * u_0 * B_1 * \ldots * B_n * u_n * B_{\infty 2}$, with $B_\infty = B_{\infty 1} + B_{\infty 2}$. As previously, we can split $A_\infty = A_{\infty 1} + A_{\infty 2}$ with $A_{\infty 1} \leq^\oplus B_{\infty 1}$ and $A_{\infty 2} \leq^\oplus B_{\infty 2}$. Therefore, if we define $R'_3 = A_{\infty_1}^{-1} + A_1 * A_2 * \ldots * A_n * A_{\infty 2} \in \overline{Pre}_\Delta(R)$ and $R''_3 = A_{\infty_1}^{-1} * A_1 * A_2 * \ldots * A_n * A_{\infty 2} \in \overline{Pre}_\Delta(R)$, then we have $R'_1 \sqsubseteq R'_3$ and $R''_1 \sqsubseteq R''_3$. Hence, $Pre_\delta(R_1) \subseteq \uparrow\overline{Pre}_\Delta(R)$.

Now we prove by induction that $\uparrow\overline{Pre}_\Delta(R) \subseteq Pre_\Delta(\uparrow R)$. Without loss of generality, we suppose that $\overline{Pre}_\delta(R) \neq \emptyset$ (otherwise, $\overline{Pre}_\Delta(R) = R$, and $\uparrow R \subseteq Pre_\Delta(\uparrow R)$ holds trivially). Again, we first prove the base case ($\uparrow\overline{Pre}_\delta(R) \subseteq Pre_\Delta(\uparrow R)$) and then we prove the inductive step (if $R_1 \in Pre_\Delta(\uparrow R)$ then $\uparrow\overline{Pre}_\delta(R_1) \subseteq Pre_\Delta(\uparrow R)$).

- Base Case: Let us suppose that $R = A_0 * A_1 * \ldots * A_n * A_\infty$. We analyze two different cases:

  (i) Suppose $A_0 \neq \emptyset$ and A_0^{-1} is defined. Then, $\overline{Pre}_\delta(R) = A_1 * A_2 * \ldots * A_n * A_0^{-1} * A_\infty$. Hence, each $R' \in \uparrow\overline{Pre}_\delta(R)$ is of the form $R' = B_1 * u_1 * B_2 * \ldots * B_n * u_n * B_0^{-1} * u_0 * B_\infty$, where for each $i \in n_\infty^*$, $A_i \leq^\oplus B_i$. Suppose $u_0 = U_0 * \ldots * U_k$. If we apply the definition of time elapsing for regions to

R' repeatedly, we obtain that $R' \overset{\Delta}{\twoheadrightarrow} R_1 = B_0 * U_0^{+1<} * \ldots * U_k^{+1<} * B_1 * u_1 * B_2 * \ldots * B_n * u_n * B_\infty + U_0^{=} + \ldots + U_k^{=}$, which is clearly greater than R. Therefore, $R' \in Pre_\Delta(\uparrow R)$, so $\uparrow\overline{Pre}_\delta(R) \subseteq Pre_\Delta(\uparrow R)$.

(ii) Now, suppose that $A_0 = \emptyset$. Then, $\overline{Pre}_\delta(R) = \{(A_1 + B_0^{-1}) * A_2 * ... * A_n * B_\infty, A_{\infty 1}^{-1} * A_1 * ... * A_n * A_{\infty 2} \mid A_\infty = A_{\infty 1} + A_{\infty 2}\}$. Let us consider $A_\infty = A_{\infty 1} + A_{\infty 2}$. We analyze both cases in $\overline{Pre}_\delta(R)$:

a) Suppose $R' = A_1 + A_{\infty 1}^{-1} * A_2 * ... * A_n * A_{\infty 2} \in \overline{Pre}_\delta(R)$. If $R' \sqsubseteq R''$, then R'' is of the form $R'' = B_1 + B_{\infty 1}^{-1} * u_1 * B_2 * u_2 * ... * B_n * u_n * B_{\infty 2}$, where $A_i \subseteq B_i$ for each $i \in n^* \cup \{\infty 1, \infty 2\}$. If we apply the definition of time elapsing for regions to such an $R'' \in\uparrow \overline{Pre}_\delta(R)$, then we obtain $R'' \overset{\delta}{\to} R_1 = \emptyset * (B_1 + B_{\infty 1}^{-1})^< * u_1 * B_2 * u_2 * ... * B_n * u_n * B_{\infty 2} + (B_1 + B_{\infty 1})^= = \emptyset * B_1^< + B_{\infty 1}^{-1<} * u_1 * B_2 * u_2 * ... * B_n * u_n * B_1^= + B_{\infty 2} + B_{\infty_1}^=$. Note that, since no pair in A_1 reaches the age max and $A_1 \leq^\oplus B_1$, $A_1 \leq^\oplus B_1^<$. Moreover, as the pairs in A_∞ has age max $+1$ and $A_\infty = A_{\infty 1} + A_{\infty 2} \leq^\oplus B_{\infty 1} + B_{\infty 2}$, $A_\infty \leq^\oplus B_{\infty 1}^= + B_{\infty 2}$ (the instances in $B_{\infty 1}^<$ are younger than max $+1$). Therefore, it is clear that R_1 is greater than R, which implies that $R'' \in Pre_\Delta(\uparrow R)$.

b) Let $R' = A_{\infty 1}^{-1} * A_1 * ... * A_n * A_{\infty 2}$. If $R' \sqsubseteq R''$, then R'' is of the form $R'' = B_{\infty 1}^{-1} * u_0 * B_1 * ... * B_n * u_n * B_{\infty 2}$, where $A_i \subseteq B_i$ for each $i \in n^* \cup \{\infty 1, \infty 2\}$. Let $u_0 = U_0 * \ldots * U_k$. Again, we apply the definition of time elapsing for regions to such an $R'' \in\uparrow\overline{Pre}_\delta(R)$, obtaining $R'' \overset{\delta}{\to} R_1 = \emptyset * B_{\infty 1}^{-1<} * u_0 * B_1 * ... * B_n * u_n * B_{\infty 2} + B_{\infty 1}^=$. Anagously as in the previous case, since all the pairs in A_∞ have age max $+1$ and $A_{\infty 1} + A_{\infty 2} \leq^\oplus B_{\infty 1} + B_{\infty 2}$, $A_\infty \leq^\oplus B_{\infty 2} + B_{\infty 1}^=$. Therefore, $R \sqsubseteq R_1$. Hence, $R'' \in Pre_\Delta(\uparrow R)$, so $\uparrow\overline{Pre}_\delta(R) \subseteq Pre_\Delta(\uparrow R)$.

- Inductive Step: Now we prove that if $R_1 \in Pre_\Delta(\uparrow R)$ then $\uparrow\overline{Pre}_\delta(R_1) \subseteq Pre_\Delta(\uparrow R)$ in two steps.

  Since we have proved compatibility for Δ in Lemma. 3.2.14, the fact that $R_1 \in Pre_\Delta(\uparrow R)$ then $\uparrow R_1 \subseteq Pre_\Delta(\uparrow R)$ is almost trivial: Consider $R_1' \in\uparrow R_1$. As $R_1 \in Pre_\Delta(\uparrow R)$ there is a region R_2 with $R_1 \overset{\Delta}{\to} R_2$ and $R \sqsubseteq R_2$. Therefore, due to the compatibility of Δ, there must be a region R_2' such that $R_1' \overset{\Delta}{\to} R_2'$ and $R \sqsubseteq R_2 \sqsubseteq R_2'$. Therefore, $R_1' \in Pre_\Delta(\uparrow R)$ and $\uparrow R_1 \subseteq Pre_\Delta(\uparrow R)$.

  Hence, it only remains to prove that if $R_1 \in Pre_\Delta(\uparrow R)$ then $\overline{Pre}_\delta(R_1) \subseteq Pre_\Delta(\uparrow R)$. Suppose $R_1 = A_0 * A_1 * \ldots * A_n * A_\infty \in Pre_\Delta(\uparrow R)$. We consider the different cases in the definition of $\overline{Pre}_\delta$:

(i) If $A_0 \neq \emptyset$ and A_0^{-1} is defined then $\overline{Pre}_\delta(R_1) = \{R_1' = A_1 * \ldots * A_n * A_0^{-1} * A_\infty\}$. Now, we consider the time elapsing $R_1' = A_1 * \ldots * A_n * A_0^{-1} * A_\infty \xrightarrow{\delta} A_0 * A_1 * \ldots * A_n = R_1$. As $R_1 \in Pre_\Delta(\uparrow R)$, we have that $R_1' \xrightarrow{\delta} R_1 \xrightarrow{\Delta} R'$, with $R \sqsubseteq R'$. Hence, $\overline{Pre}_\delta(R_1) \subseteq Pre_\Delta(\uparrow R)$.

(ii) If $A_0 \neq \emptyset$ and A_0^{-1} is not defined then $\overline{Pre}_\delta(R_1) = \emptyset$, and $\overline{Pre}_\delta(R_1) \subseteq Pre_\Delta(\uparrow R)$ clearly holds.

(iii) If $A_0 = \emptyset$ then $\overline{Pre}_\delta(R_1) = \{(A_1 + B_0^{-1}) * A_2 * ... * A_n * B_\infty, B_0^{-1} * A_1 * ... * A_n * B_\infty \mid A_\infty = B_0 + B_\infty\}$. Let $R_1' = (A_1 + B_0^{-1}) * A_2 * ... * A_n * B_\infty$ such that $A_\infty = B_0 + B_\infty$. Then, there is a time elapsing $R_1' \xrightarrow{\delta} \emptyset * (A_1 + B_0^{-1})^< * A_2 * \ldots * A_n * (B_\infty + (A_1 + B_0^{-1})^= = \emptyset * A_1 * A_2 * \ldots * (B_\infty + B_0) = R_1$. Now, consider $R_1' = B_0^{-1} * A_1 * ... * A_n * B_\infty \mid A_\infty = B_0 + B_\infty$ with $A_\infty = B_0 + B_\infty$. Again, we apply a time elapsing to R_1', so we obtain $R_1' \xrightarrow{\delta} \emptyset * A_1 * \ldots * A_n * (B_\infty + B_0) = R_1$. Since $R_1 \in Pre_\Delta(\uparrow R)$, in any of the previous cases, we have that $R_1' \xrightarrow{\delta} R_1 \xrightarrow{\Delta} R'$. Hence, $\overline{Pre}_\delta(R_1) \subseteq Pre_\Delta(\uparrow R)$.

□

Next, as previously done for the time delays, we define $\overline{Pre}_t$ for each discrete transition $t \in T$ of a *ν-lsPN* to prove that we are able to compute $Pre_t(\uparrow R)$. In order to build this definition, for each $t \in T$ and each region R, we first define a family $\mathcal{F}(t, R)$ of functions which will assign a part of the region taking some role in some firing to each variable in $Var(t)$.

Definition A.3 ($\mathcal{F}(t, R)$) *Let $t \in T$ and a region $R = A_0 * A_1 * \ldots * A_n * A_\infty$, with $A_i = \{(m_{i1}, r_{i1}), \ldots, (m_{ik_i}, r_{ik_i})\}$. Suppose that $l = |Var(t)|$ and $q = \max\{k_i \mid i \in n_\infty^*\}$. Moreover, we consider certain multiset $A_{n+1} = \{(m_{(n+1)1}, r_{(n+1)1}), \ldots, (m_{(n+1)l}, r_{(n+1)l})\}$ with $m_{(n+1)i} = \emptyset$ for each $i \in l^+$. A function $f : Var(t) \to (n+1)_\infty^* \times (q+1)^* \times (max+1)^* \times (n^* \cup \{\infty\}) \times l^*$ is in $\mathcal{F}(t, R)$ iff for all $x \in Var(t)$, $f(x) = (b_1, b_2, b_3, b_4, b_5)$ with:*

- *If $b_1 \in n^+$ and $b_2 \leq k_{b_1}$ then $r_{b_1 b_2} + 0.5 \in Time_t^2(x)$.*
- *If $b_1 = 0$ and $b_2 \leq k_0$ then $r_{b_1 b_2} \in Time_t^2(x)$.*
- *If $b_1 = \infty$ then $\max + 0.5 \in Time_t^2(x)$.*
- *If $x \in \Upsilon$ then $m_{b_1 b_2} \subseteq Out_t(x)$.*
- *If $b_4 \in n^+ \cup \{\infty\}$ then $b_3 + 0.5 \in Time_t^1(x)$.*

- *If* $b_4 = 0$ *then* $b_3 \in Time_t^1(x)$.
- *If* $b_4 = \infty$ *iff* $b_3 = max + 1$.
- *If* $y \neq x$ *and* $f(y) = (b'_1, b'_2, b'_3, b'_4, b'_5)$, *then* $(b_1, b_2) \neq (b'_1, b'_2)$.

Intuitively, the first two numbers that the previous functions assign to a variable x, correspond to the selection of the part of the region we assign to x to remove the effects of Out_t. Analogously, the two last components manage the effects of In_t. The third number assigns to each variable the natural number that correspond to the age of the instance in the predecessor.

Clearly, the family $\mathcal{F}(t, R)$ is finite. We define $\overline{Pre}_t(R)$ as the effects of computing the predecessors of R according to all the functions in $\mathcal{F}(t, R)$.

Definition A.4 ($\overline{Pre}_t$) *Let* $l = |Var(t)|$. *Given* $t \in T$, *suppose* $f \in \mathcal{F}(t, R)$ *and* $R = A_0 * A_1 * \ldots * A_n * A_\infty$, *with* $A_i = \{(m_{i1}, r_{i1}), ..., (m_{ik_i}, r_{ik_i})\}$. *Again, we consider certain multiset* $A_{n+1} = \{(m_{(n+1)1}, r_{(n+1)1}), \ldots, (m_{(n+1)l}, r_{(n+1)l})\}$ *with* $m_{(n+1)i} = \emptyset$ *for each* $i \in l^+$. *Then, we define* $\overline{Pre}_{ft}(R)$ *as follows:*

- *First, we define the* $\emptyset$*-expansion* $R'' = A'_{00} * A'_{01} * \ldots * A'_{0l} * A'_{10} * A'_{11} * ... * A'_{nl} * A'_{\infty 0}$, *where:*
  - $A'_{j0} = A_j - \{(m_{jk}, r_{jk}) \mid \exists x$ *with* $f(x) = (j, k, b_3, b_4, b_5)$ *for some* $b_3, b_4, b_5\}$
  - $A'_{\infty 0} = A_\infty - \{(m_{\infty k}, max{+}1) \mid \exists x$ *with* $f(x) = (\infty, k, b_3, b_4, b_5)$ *for some* $b_3, b_4, b_5\}$
  - $A'_{ij} = \emptyset$ *elsewhere.*
- *For each* $x \in Var(t)$, *if* $f(x) = (b_1, b_2, b_3, b_4, b_5)$, *then we define* $m'_x = (m_{b_1 b_2} \ominus Out_t(x)) + In_t(x)$ *and* $r'_x = b_3$, *where* $(m_1 \ominus m_2)(x) = \max(0, m_1(x) - m_2(x))$.
- *Finally,* $\overline{Pre}_{ft}(R)$ *is the* $\emptyset$*-contraction of* $B_{00} * B_{01} * \ldots * B_{0l} * B_{10} * B_{11} * \ldots * B_{nl} * B_{\infty 0}$, *where for each* $i \in n^*_\infty$ *and* $j \in l^*$, $B_{ij} = A'_{ij} + \{(m'_x, r'_x) \mid f(x) = (b_1, b_2, b_3, i, j)$ *for some* $b_1, b_2, b_3\}$.

Then, we define $\overline{Pre}_t(R) = \{\overline{Pre}_{ft}(R) \mid f \in \mathcal{F}(t, R)\}$.

Intuitively, in R'' we have removed the instances corresponding to the effects of Out_t, and added l empty multisets of instances between each A_i and A_{i+1} in order to be able to add pairs representing instances modified by In_t in the firing, with an age with a fractional part different to all the ones represented in R.

Lemma A.3 *Given a region R and a transition t of a ν-lsPN, the set $\overline{Pre}_t(R)$ is finite, computable and such that $Pre_t(\uparrow R) = \uparrow \overline{Pre}_t(R)$*

Proof: Clearly, $\overline{Pre}_t(R)$ as defined above is finite and computable, since the family $\mathcal{F}(t, R)$ is finite, and the steps in Def. A.4 can be performed in a univocal way.

First, we prove $\uparrow \overline{Pre}_t(R) \subseteq Pre_t(\uparrow R)$. Let $R' \in \uparrow \overline{Pre}_t(R)$, we are going to prove that $R' \in Pre_t(\uparrow R)$. Therefore, we need to prove that there is R''' with $R \sqsubseteq R'''$ such that $R' \xrightarrow{t} R'''$. Let us call $R = A_0 * A_1 * ... * A_{n_A} * A_\infty$ with $A_i = \{(m^A_{ij}, r^A_{ij}) \mid j \in |A_i|^+\}$ and $R' = E_0 * E_1 * ... * E_{n_E} * E_\infty$ with $E_i = \{(m^E_{ij}, r^E_{ij}) \mid j \in |E_i|^+\}$.

As $R' \in \uparrow \overline{Pre}_t(R)$, there is $R'' \in \overline{Pre}_t(R)$ such that $R'' \sqsubseteq R'$. Then, there is $f \in \mathcal{F}(t, R)$ such that $R'' = \overline{Pre}_{ft}(R)$. Suppose that $C_{00} * C_{01} * \ldots * C_{0l} * C_{10} * C_{11} * \ldots * C_{nl} * C_{\infty 0}$, is the $\emptyset$-expansion of R'' obtained in the third step of Def. A.4. For simplicity, we consider the firing from this $\emptyset$-expansion. Abusing notation, if $x \in Var(t)$ with $f(x) = (b_1, b_2, b_3, b_4, b_5)$, we call $(m^C_{b_4b_5x}, r^C_{b_4b_5x})$ the pair $(m^A_{b_1b_2} \ominus Out_t(x)) + In_t(x), b_3)$ we add in $C_{b_4b_5}$, that is, we call that pair in $C_{b_4b_5}$ after the variable. In order to fire t from R'', we define the mode $\tau = (\tau_1, \tau_2, \tau_3)$ such that, for each $x \in Var(t)$, if $f(t, R) = (b_1, b_2, b_3, b_4, b_5)$:

- $\tau_1(x) = (b_4b_5, x)$
- $\tau_2(x) = r_{b_4b_5}$
- $\tau_3(x) = b_10$

Now, we prove that t is enabled at R''. Indeed, for each $x \in Var(t)$, if $f(x) = (b_1, b_2, b_3, b_4, b_5)$, $\tau_1(x) = (b_4b_5, x)$, so:

- if $x \in \Upsilon$ then $m^A_{b_1b_2} \leq Out_t(x)$, so $m^C_{b_4b_5x} = (m^A_{b_1b_2} \ominus Out_t(x)) = \emptyset$,
- $m^C_{b_4b_5x} = (m^A_{b_1b_2} \ominus Out_t(x)) + In_t(x)$, and therefore $m^C_{b_4b_5x} \geq In_t(x)$
- $r^C_{b_4b_5x} = b_3$. If $b_4 = 0$ or $b_4 = \infty$ then $b_3 \in Time^1_t(x)$. Otherwise, $b_3 + 0.5 \in Time^1_t(x)$, because of the definition of $\mathcal{F}$.
- Therefore, if $i \in \{0, \infty\}$ then $match((m^C_{b_4b_5x}, r^C_{b_4b_5x}), (In_t(x), Time^1_t(x)))$, and $match((m^C_{b_4b_5x}, r^C_{b_4b_5x} + 0.5), (In_t(x), Time^1_t(x)))$ otherwise.

Therefore, t is enabled at R'', so it is enabled in R' too. Because of how we have defined τ, it is easy to see that $R'' \overset{t}{\twoheadrightarrow} \bar{R} \supseteq R$ with this mode. We just need to prove

that if $R'' \overset{t}{\twoheadrightarrow} \bar{R}$ then $\bar{R}$ is of the form $\bar{R} = A'_0 * a_0 * A'_1 * a_1 * ... * A'_{n_A} * a_{n_A} * A'_\infty$, where $A_i \leq^\oplus A'_i$ for each $i \in n_A^*{}_\infty$ and $a_j \in ((P^\oplus \times \mathbb{N})^\oplus)^\circledast$ for each $j \in n_A^*$. Suppose that $F_{00} * F_{001} * \ldots * F_{00l_t} * F_{01} * \ldots * F_{nl} * F_{nl1} * \ldots * F_{nll_t} * F_{\infty 0}$ is the $\emptyset$-expansion obtained when firing t from the $\emptyset$-expansion $C_{00} * C_{01} * \ldots * C_{0l} * C_{10} * C_{11} * \ldots * C_{nl} * C_{\infty 0}$ with the mode τ we have defined. Then, we can see that for each $i \in n_A^*{}_\infty$, $A_i \leq^\oplus F_{i0}0$. Indeed, $F_{i0}0 = B_{i0} + D_{i0}$ (with the notations in Def. 3.2.11, where $B_{i0} = C_{i0} - \{(m^C_{i0x}, r^C_{i0x}) \mid x \in Var(t)\}$ and $D_{i0} = \{(m'_{i'j}, r) \mid \exists x \in Var(t)$ with $\tau_1(x) = (i', j), \tau_2(x) = r, \tau_3(x) = i0\} = \{(m'_{i'j}, r_{i'}) \mid \exists x \in Var(t)$ with $i' = i'_1 i'_2, f(x) = (i, b_2, r, i'_1, i'_2)\}$ for some b_2, with $m'_{i'j} = (m_{i'j} - In_t(x)) + Out_t(x)$. Note that $A_i = (C_{i0} - \{(m'_x, r'_x) \mid f(x) = (b_1, b_2, r'_x, i, j)$ for some b_1, b_2 with $m_x = (m_{b_1 b_2} \ominus Out_t(x)) + In_t(x)\}) + \{(m_{ik}, r_{ik}) \mid \exists x$ with $f(x) = (i, k, b_3, b_4, b_5)$ for some $b_3, b_4, b_5\}$. Hence, $A_i \leq^\oplus B_{i0} + D_{i0} = F_{i0}0$. As $R'' \sqsubseteq R'$, and the transition system is monotonic, there is R''' such that $R \sqsubseteq R'''$, with $R' \overset{t}{\twoheadrightarrow} R'''$.

Now, we prove that $Pre_t(\uparrow R) \subseteq \uparrow \overline{Pre}_t(R)$. Let $R = A_0 * A_1 * ... * A_n * A_\infty$, R' and R'' such that $R'' \overset{t}{\twoheadrightarrow} R' \sqsupseteq R$ with mode τ. It is enough to see that there exist $f \in \mathcal{F}(t, R')$ such that $R'' \in \uparrow(\overline{Pre}_{ft}(R))$. In order to define f, we give some notations and renamings for these regions.

Suppose $R'' = F_0 * F_1 * \ldots * F_{n_F} * F_\infty$. We denote by $\mathcal{S}$ the set $\{F_i \mid F_i - \{(m_{ij}, r_{ij}) \mid \tau_1(x) = (i, j)\} = \emptyset\}$ and $l = max(\{|A_i| \mid i \in n_F^*\}) + |Var(t)|$. In order to set the notations, and without loss of generality, let us suppose that the $\emptyset$-expansion of R'' that we consider in the firing of t is $F_{00} * F_{01} * \ldots * F_{0(l+1)} * \ldots * F_{n_F 0} * \ldots * F_{n_F l} * F_\infty$, with $F_{i0} = F_i$ if $F_i \notin \mathcal{S}$, $F_{ij} = F_k$ if $F_k \in \mathcal{S}$ and F_k is the j^{th} set in $\mathcal{S}$ after F_i; and $F_{ij} = \emptyset$ otherwise. Hence, we consider that the mode τ in the firing is defined for these $\emptyset$-expansion (here we are abusing notation, since now the subscripts in the region are pairs of natural numbers instead of single numbers) . Intuitively, we have renamed R'' in order to define b_4 and b_5 of $f(x)$. That way, the F_ks which do not "disappear" in the firing of t because of the effects of In_t, have index $j0$ for some j. Consider the $\emptyset$-expansion of R' $E_{s_0} * E_{s_0 1} * \ldots * E_{s_{n_F}} * E_{s_{n_F 1}} * \ldots * E_\infty$ obtained when firing t from the previous $\emptyset$-expansion of R'', where $s_0, \ldots, s_{n_F}, \infty$ are the subindices in the multisets in R'' (which are in $(n^* \times l^*) \cup \{\infty\}$). Since $R \sqsubseteq R'$, we can rename it as $E_{k_0} * u_0 * E_{k_1} * ... * E_{k_n} * u_n * E_\infty$, and $k_i \in ((n^* \times l^*) \cup \{\infty\}) \times |Var(t)|^+$, where $A_i \leq^\oplus E_{k_i}$ for $i \in n_\infty^*$. Without loss of generality, let us consider the denotation

of the multisets A_i and E_{k_i}, for $i \in n^*_\infty$, such that for each $i \in n^*_\infty$, $j \in |A_i|^+$, (m_{ij}, r_{ij}) is a pair in A_i, (m_{k_ij}, r_{k_ij}) is a pair in E_i, with $m_{ij} \leq m^E_{k_ij}$ and $r_{ij} = r^E_{k_ij}$. For each $i \in n^*$, let us denote $u_i = E_{k_i1} * \ldots * E_{k_in_i}$. Moreover, let us define the function $rn : \{k_i \mid i \in n^*_\infty\} \cup \{(k_i, z) \mid i \in n^*_\infty, z \in n_i^+\} \to \{s_i \mid i \in {n_F}^*_\infty\} \cup \{s_ij \mid i \in n_F, j \in {|Var(t)|^+}^*_\infty\}$ such that $rn(\beta) = \alpha$ if the subindex α is renamed to β by the previous renaming. With these notations, let us define $f \in \mathcal{F}(R, t)$. Given $x \in Var(t)$, suppose that $\tau_1(x) = (i, j)$ and $\tau_3(x) = i'$, considering the renaming of the $\emptyset$-expansion of R'' we have defined. Then, we define $f(x) = (b_1, b_2, b_3, b_4, b_5)$ such that:

- If $(m^E_{k_{i'}j'}, r^E_{k'_ij'})$ is the pair in $E_{k_{i'}} \geq^\oplus A_{i'}$ that we add in the firing of t, associated to x, such that it is produced by the removing of the pair (m^F_{ij}, r^F_{ij}) in the firing from R'', that is, $m^E_{i'j'} = (m^F_{ij} - In_t(x)) + Out_t(x)$ and $r^E_{i'j'} = \tau_2(x)$, then $b_1 = i'$, $b_2 = j'$, $b_3 = r_{ij}$, $b_4 = i$ and $b_5 = j$.

- In case the pair we add in the firing of t, associated to x is in some multiset $E_{s_{i'}j'} \in u_h$, for some $h \in n^*$, or the pair is $(m^E_{k_{i'}j'}, r^E_{k'_ij'}) \in E_{k_{i'}}$ but $j' \geq |A_{i'}|$, then we define $b_1 = n + 1$. Moreover, if we consider that $x_1, \ldots, x_{nf}$ are the non free variables of t, and $x = x_{j''}$, then we define $b_2 = j''$. Again, if the pair $(m^E_{k_{i'}j'}, r^E_{k'_ij'}) \in E_{k_{i'}}$ is produced by the removing of the pair (m^F_{ij}, r^F_{ij}) in the firing from R'', that is, $m^E_{i'j'} = (m^F_{ij} - In_t(x)) + Out_t(x)$ and $r^E_{i'j'} = \tau_2(x)$, then $b_1 = i'$, $b_2 = j'$, $b_3 = r_{ij}$, $b_4 = i$ and $b_5 = j$, as previously.

Finally, we prove that if $R_f = B_{00} * B_{01} * \ldots * B_{20} * \ldots * B_{\infty 0}$ is the $\emptyset$-expansion obtained when calculating $\overline{Pre}_{ft}(R)$ and R'_f is its $\emptyset$-contraction, then $R'_f \sqsubseteq R''$. Given $R_p = G_0 * G_{i_1} * \ldots * G_{i_m} * G_\infty$ an $\emptyset$-expansion of a region we denote $Nemp(R_p) = \{j \mid G_{i_j} \neq \emptyset \text{ is a multiset in } R_p\}$. We define a monotone injection $\chi : Nemp(R_f) \to Nemp(R'')$ such that, for each $B_{ij} \in Nemp(R_f)$, $B_{ij} \leq^\oplus E_{\chi(ij)}$. Given $ij \in Nemp(R_f)$ we define $\chi(i, j) = rn(k_i)$ if $j = 0$ and $\chi(i, j) = k_ij$ otherwise. As rn is a (monotone and injective) renaming, it is clear that χ is an injection. Let us prove that for each $B_{ij} \in Nemp(R_f)$, $B_{ij} \leq^\oplus E_{\chi(ij)}$. We analyze two different cases.

- If $j = 0$ then $\chi(i, j) = rn(k_i)$. In this case, $B_{i0} = (A_i - \{(m_{ik}, r_{ik}) \mid \exists x \text{ with } f(x) = (i, k, b_3, b_4, b_5) \text{ for some } b_3, b_4, b_5\}) + \{(m'_x, r'_x) \mid f(x) = (b_1, b_2, b_3, i, 0) \text{ for some } b_1, b_2, b_3, m'_x = (m_{b_1b_2 \ominus Out_t(x)) + In_t(x), r'_x = b_3})\} \leq^\oplus (E_{k_i} - \{(m^E_{k_ik}, r^E_{k_ik}) \mid \exists x \text{ such that } (m^E_{k_ik}, r^E_{k_ik}) \text{ is the pair in } E_{k_i} \text{ that we add in the firing of } t, \text{ associated to } x\}) + \{(m^F_{ij}, r^F_{ij}) \mid \exists x \text{ such that } (m^F_{ij}, r^F_{ij}) \text{ is the pair removed in the firing of } t \text{ associated to } x\} = E_{rn(k_i)} = E_{\chi(ij)}$.

- Analogously, if $j \neq 0$ then we have $B_{ij} = \emptyset + \{(m'_x, r'_x) \mid f(x) = (b_1, b_2, b_3, i, 0)$ for some $b_1, b_2, b_3, m'_x = (m_{b_1 b_2 \ominus Out_t(x)) + In_t(x), r'_x = b_3})\} \leq^{\oplus} \{(m^F_{ij}, r^F_{ij}) \mid \exists x$ such that (m^F_{ij}, r^F_{ij}) is the pair removed in the firing of t associated to $x\}$ $= E_{rn(k_i j)} = E_{\chi(ij)}$

Hence, $R'' \sqsupseteq R'_f \in \overline{Pre}_{ft}(R)$, and therefore $Pre_t(\uparrow R) \subseteq \uparrow \overline{Pre}_t(R)$.

□

As $\twoheadrightarrow = \overset{\Delta}{\twoheadrightarrow} \cup \bigcup_{t \in T} \overset{t}{\twoheadrightarrow}$ and we have defined $\overline{Pre}_\Delta$ and $\overline{Pre}_t$ for each $t \in T$, which are finite and computable, such that $Pre_\Delta(\uparrow R) = \uparrow \overline{Pre}_\Delta(R)$ and $Pre_t(\uparrow R) = \uparrow \overline{Pre}_t(R)$, we obtain the following corollary.

Corollary A.4 *$\twoheadrightarrow$ has effective Pred-basis.*

Appendix B

Effective $Pred$-basis of the transition relation $\rightarrowtail$ defined for ν-$aPTdPN$

In order to prove that $\rightarrowtail$ has effective $Pred$-basis we handle the priced and timed transitions separately. Therefore, as done for ν-$lsPN$, we split Pre into $Pre_\Delta(R) = \{R' \mid \exists i \in 4^+ R' \overset{i}{\rightarrowtail} R\}$ and $Pre_t(R) = \{R' \mid R' \overset{t}{\rightarrowtail} R\}$, and we define $\overline{Pre}_\Delta$ and $\overline{Pre}_t$ for each $t \in T$, so that $Pre_\Delta(\uparrow R) = \uparrow \overline{Pre}_\Delta(R)$ and $Pre_t(\uparrow R) = \uparrow \overline{Pre}_t(R)$.

We begin by defining $\overline{Pre}_\Delta$, the function in charge of computing the predecessors corresponding to timed transitions of a region R. As there are four kinds of time transitions, we will define four auxiliary functions $\overline{Pre}_{\delta_1}$, $\overline{Pre}_{\delta_2}$, $\overline{Pre}_{\delta_3}$ and $\overline{Pre}_{\delta_4}$ to handle each case separately. Let us recall from the previous chapter that given $A = \{(a_1, r_1), \ldots, (a_n, r_n)\} \in (P^\oplus \times \mathbb{N})^\oplus$, with $r_1, \ldots, r_n > 0$, we define $A^{-1} = \{(a_1, r_1 - 1), \ldots, (a_n, r_n - 1)\}$.

Definition B.1 (Pre_Δ) *Suppose that $R = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * \ldots * A_m \rangle$ is a region of a ν-aPTdPN. Then, we define:*

- *If $A_0 = \emptyset$ then $\overline{Pre}_{\delta_1}(R) = \{\langle c, A_{-n} * \ldots * A_{-1}, A_1, A_2 * \ldots * A_m \rangle\}$ (else $\overline{Pre}_{\delta_1}$ is not defined),*
- *if A_0^{-1} is well defined then $\overline{Pre}_{\delta_2}(R) = \{\langle c, A_{-n} * \ldots * A_{-1} * A_0^{-1}, \emptyset, A_1 * \ldots * A_m \rangle\}$,*
- *$\overline{Pre}_{\delta_3}(R) = \{\langle c, A_{-n}^{-1} * \ldots * A_{-k-1}^{-1}, A_{-k}, A_{-k+1} * \ldots * A_{-1} * A_0^{-1} * A_1^{-1} * \ldots * A_m^{-1} \rangle \mid A_{-n}^{-1}, \ldots, A_{-k-1}^{-1}, A_0^{-1}, A_1^{-1}, \ldots, A_m^{-1}$ are well defined$\}$, where $c' =$*

$c - \sum_{i=-n}^{m} \sum_{(m',r)\in A_i} A_i(m',r) * \sum_{p\in P} m'(p) * Cost(p)$ *and*

- $\overline{Pre}_{\delta_4}(R) = \{\langle c, A_{-n}^{-1} * \ldots * A_{-k-1}^{-1} * A_{-k}^{-1}, \emptyset, A_{-k+1} * \ldots * A_{-1} * A_0^{-1} * A_1^{-1} * \ldots * A_m^{-1}\rangle \mid A_{-n}^{-1}, \ldots, A_{-k}^{-1}, A_0^{-1}, A_1^{-1}, \ldots, A_m^{-1}$ *are well defined*$\}$, *where* $c' = c - \sum_{i=-n}^{m} \sum_{(m',r)\in A_i} A_i(m',r) * \sum_{p\in P} m'(p) * Cost(p)$.

Then, we define $\overline{Pre}_{\Delta}(R) = \{R\} \cup \overline{Pre}_{\delta_1} \cup \overline{Pre}_{\delta_2}(R) \cup \overline{Pre}_{\delta_3}(R) \cup \overline{Pre}_{\delta_4}(R)$.

Note that, as in the previous chapter, we ensure we do not compute predecessors with ages lower than 0, which would not make sense, by requiring A_i^{-1} is well defined whenever we consider it for the predecessor we are computing. Now, let us prove that $Pre_{\Delta}(R)$ compute the predecessors corresponding to time delays in the proper way.

Lemma B.1 *Given an abstract region,* $\overline{Pre}_{\Delta}(R)$ *is finite and* $\uparrow\overline{Pre}_{\Delta}(R) = \uparrow Pre_{\Delta}(\uparrow R)$.

Proof: We first prove the finiteness of $\overline{Pre}_{\Delta}(R)$. The multisets obtained from applying $\overline{Pre}_{\delta_1}, \overline{Pre}_{\delta_2}, \overline{Pre}_{\delta_3}$ and $\overline{Pre}_{\delta_4}$ to any region R are finite. Indeed, the multisets we obtain when we apply $\overline{Pre}_{\delta_1}$ and $\overline{Pre}_{\delta_2}$ to a region have one element as many. Moreover, as the number of multisets A_i in a region are finite, there is a finite number of ways of decomposing the region, by considering different values of k as in the previous definition. Hence, the multisets we obtain when applying $\overline{Pre}_{\delta_3}$ and $\overline{Pre}_{\delta_4}$ are finite too. Hence, $\overline{Pre}_{\Delta}(R)$ is finite.

Now we prove that $\uparrow \overline{Pre}_{\Delta}(R) = \uparrow Pre_{\Delta}(\uparrow R)$. We first prove that $\uparrow\overline{Pre}_{\Delta}(R) \subseteq \uparrow Pre_{\Delta}(\uparrow R)$, and then we will prove the other containment.

Let us suppose that $R_1 \in \uparrow\overline{Pre}_{\Delta}(R)$, and $R = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * A_2 * \ldots * A_m\rangle$. Then, there is $R_1' \ll R_1$ and $i \in 4^+$ such that $R_1' \in \overline{Pre}_{\delta_i}(R)$. We analyze the four different cases separately, proving that in each of them $R_1 \in \uparrow Pre_{\Delta}(\uparrow R)$, that is, that there are $R_1'' \ll R_1$ and $R_2 \gg R$ such that $R_1'' \overset{i}{\rightarrowtail} R_2$.

- Suppose $R_1' \in \overline{Pre}_{\delta_1}(R)$. Then, $A_0 = \emptyset$ and $R_1' = \langle c, A_{-n} * \ldots * A_{-1}, A_1, A_2 * \ldots * A_m\rangle$. Hence, R_1 is of the form $\langle c, a_{-n-1} * A'_{-n} * a_{-n} \ldots * A'_{-1} * a_{-1}, A_1', a_1 * A_2' * \ldots * A_m' * a_m'\rangle$, where $c' \leq c$, $A_i' \leq^{\oplus} A_i$ for all $i \in \{-n, \ldots, m\}$ and $a_i \in ((P^{\oplus} \times \mathbb{N})^{\oplus})^{\circledast}$. Then, $R_1' \overset{1}{\rightarrowtail} \langle c, a_{-n-1} * A'_{-n} * a_{-n} \ldots * a_{-1}, \emptyset, A_1' * a_1 * A_2' * \ldots * A_m' * a_m'\rangle \gg R$.

- Now, let $R_1' \in \overline{Pre}_{\delta_2}(R)$. Then, A_0^{-1} is well defined, and $R_1' = \langle c, A_{-n} * \ldots * A_{-1} * A_0^{-1}, \emptyset, A_1 * A_2 * \ldots * A_m\rangle$. Therefore, R_1 is of the form $\langle c, a_{-n-1} * A'_{-n} *$

$a_{-n} \ldots * a_{-1} * {A'_0}^{-1} * a_0, A', a * A'_1 * a_1 * A'_2 * \ldots * A'_m * a'_m\rangle$, where $c' \leq c$, $A'_i \geq^{\oplus} A_i$ for all $i \in \{-n, \ldots, m\}$ and $a, a_i \in ((P^{\oplus} \times \mathbb{N})^{\oplus})^{\circledast}$. Moreover, $R_1 \gg \langle c, a_{-n-1} * A'_{-n} * a_{-n} \ldots * a_{-1} * {A'_0}^{-1}, \emptyset, a * A'_1 * a_1 * A'_2 * \ldots * A'_m * a'_m\rangle \overset{2}{\rightarrowtail} \langle c, a_{-n-1} * A'_{-n} * a_{-n} \ldots * a_{-1}, A'_0, a * A'_1 * a_1 * A'_2 * \ldots * A'_m * a'_m\rangle \gg R$.

- Let us suppose that $R'_1 \in \overline{Pre}_{\delta_3}(R)$. Then, $R'_1 = \langle c', A_{-n}^{-1} * \ldots * A_{-k-1}^{-1}, A_{-k}, A_{-k+1} * \ldots * A_{-1} * A_0^{-1} * A_1^{-1} * \ldots * A_m^{-1}\rangle$ for some $k \in n^+$, where $A_{-n}^{-1}, \ldots, A_{-k-1}^{-1}$ and $A_0^{-1}, A_1^{-1}, \ldots, A_m^{-1}$ are well defined and $c' = c - \sum_{i=-n}^{m} \sum_{(m',r)\in A_i} A_i(m', r) * \sum_{p\in P} m'(p) * Cost(p)$. Therefore, R_1 is of the form $\langle c'', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-k-1} * a_{-k-1}, A'_{-k}, a_{-k} * A'_{-k+1} * \ldots * a_{-2} * A'_{-1} * a_{-1} * A'_0 * a_0 * A'_1 * a_1 * \ldots * A'_m * a_m\rangle$, where $c'' \geq c'$, $a_i \in ((P^{\oplus} \times \mathbb{N})^{\oplus})^{\circledast}$ for each $i \in \{-n-1, \ldots, m\}$, for each $j \in \{-n, \ldots, -k-1, 0, \ldots, m\}$ $A_j^{-1} \leq^{\oplus} A'_j$ and for each $i \in \{-k \ldots -1\}$, $A_i \leq^{\oplus} A'_i$. Moreover, $R_1 \gg \langle c', A'_{-n} * \ldots * A'_{-k-1}, A'_{-k}, A'_{-k+1} * \ldots * A'_{-1} * A'_0 * A'_1 * \ldots * A'_m\rangle \overset{4}{\rightarrowtail} \langle c', {A'_{-n}}^{+1} * \ldots * {A'_{-k-1}}^{+1} * A'_{-k} * A'_{-k+1} * \ldots * A'_{-1}, {A'_0}^{+1}, {A'_1}^{+1} * \ldots * {A'_m}^{+1}\rangle \gg R$ (note that $c''' \geq c$ because in R_1 there are at least the same amount of tokens in each place as in R). In case $A'_0 = \emptyset$, the firing must be performed with $\overset{4}{\rightarrowtail}$ instead, but the rest of the proof would work.

- The last case is analogous to the previous one: let $R'_1 \in \overline{Pre}_{\delta_4}(R)$. Then, $R'_1 = \langle c', A_{-n}^{-1} * \ldots * A_{-k-1}^{-1} * A_{-k}^{-1}, \emptyset, A_{-k+1} * \ldots * A_{-1} * A_0^{-1} * A_1^{-1} * \ldots * A_m^{-1}\rangle$ for some $k \in n^+$, where $A_{-n}^{-1}, \ldots, A_{-k}^{-1}, A_0^{-1}, A_1^{-1}, \ldots, A_m^{-1}$ are well defined and $c' = c - \sum_{i=-n}^{m} \sum_{(m',r)\in A_i} A_i(m', r) * \sum_{p\in P} m(p) * Cost(p)$. Then, R_1 is of the form $R_1 = \langle c'', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-k-1} * a_{-k-1} * A'_{-k} * a_{-k}, A, a, A'_{-k+1} * \ldots * a_{m-1} * A'_m * a_m\rangle$, , where $c'' \geq c'$, $a, a_i \in ((P^{\oplus} \times \mathbb{N})^{\oplus})^{\circledast}$ for each $i \in \{-n-1, \ldots, m\}$, for each $j \in \{-n, \ldots, -k, 0, \ldots, m\}$ $A_j^{-1} \leq^{\oplus} A'_j$ and for each $i \in \{-k+1 \ldots -1\}$, $A_i \leq^{\oplus} A'_i$. Then, $R_1 \gg \langle c', A'_{-n} * \ldots * A'_{-k-1} * A'_{-k}, \emptyset, A'_{-k+1} * \ldots * A'_m\rangle \overset{4}{\rightarrowtail} \langle c''', {A'_{-n}}^{+1} * \ldots * {A'_{-k}}^{+1} * A'_{-k+1} * \ldots * A'_{-1}, {A'_0}^{+1}, {A'_1}^{+1} * \ldots * {A'_m}^{+1}\rangle \gg R$, because $c''' \geq c$, since in R_1 there are at least the same amount of tokens in each place as in R). Again, if $A'_0 = \emptyset$, the proof would work by performing the firing with $\overset{4}{\rightarrowtail}$ instead.

Therefore, $\uparrow\overline{Pre}_{\Delta}(R) \subseteq \ \uparrow Pre_{\Delta}(\ \uparrow R)$.

Conversely, suppose that $R_1 \in \ \uparrow Pre_{\Delta}(\ \uparrow R)$. Then, there are two regions R_2 and R'_1, and $i \in 4^+$ such that $R_1 \gg R_2 \overset{i}{\rightarrowtail} R'_1 \gg R$. We prove that in each of the possible cases, there is R'_2 with $R_2 \gg R'_2 \in \overline{Pre}_{\delta_i}(R)$. Suppose that $R_2 = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * A_2 * \ldots * A_m\rangle$.

- First, suppose $R_2 \xrightarrow{1} R_1'$. Then, $R_1' = \langle c, A_{-n} * \ldots * A_{-1}, \emptyset, A_0 * A_1 * A_2 * \ldots * A_m \rangle \gg R$. Therefore, R is of the form $\langle c', A'_{-n} * \ldots * A'_{-1}, \emptyset, A'_0 * A'_1 * A'_2 * \ldots * A'_m \rangle$, where $c \geq c'$ and for each $i \in \{-n \ldots m\}$, $A'_i \leq^{\oplus} A_i$ (if A'_i is $\emptyset$, then it is removed). Then, $R_2' = \langle c', A'_{-n} * \ldots * A'_{-1}, A'_0, A'_1 * A'_2 * \ldots * A'_m \rangle \in \overline{Pre}_{\delta_1}(R)$, and clearly $R_2 \gg R_2'$ (Note that if $A'_0 = \emptyset$, then $R_2 \gg R$, and we are done).

- Analogously, let $R_2 \xrightarrow{2} R_1'$. Now, $A_0 = \emptyset$ and $R_1' = \langle c, A_{-n} * \ldots * A_{-2}, A_{-1}^{+1}, A_1 * A_2 * \ldots * A_m \rangle \gg R$. Hence, R is of the form $\langle c', A'_{-n} * \ldots * A'_{-2}, {A'_{-1}}^{+1}, A'_1 * A'_2 * \ldots * A'_m \rangle$, where $c \geq c'$ and for each $i \in \{-n \ldots m\}$, $A'_i \leq^{\oplus} A_i$ (if A'_i is $\emptyset$, then it is removed). Then, $R_2' = \langle c', A'_{-n} * \ldots * A'_{-2} * A'_{-1}, \emptyset, A'_1 * A'_2 * \ldots * A'_m \rangle \in \overline{Pre}_{\delta_2}(R)$, and $R_2 \gg R_2'$ clearly holds. Again, in case $A'_{-1} = \emptyset$, then $R_2 \gg R$.

- Suppose that $R_2 \xrightarrow{3} R_1'$. Then, there is $k \in m^+$ such that $R_1' = \langle c', A_{-n}^{+1} * \ldots * A_{-1}^{+1} * A_0 * \ldots * A_k, \emptyset, A_{k+1}^{+1} * \ldots * A_m^{+1} \rangle$, with $c' = c + \sum_{i=-n}^{m} \sum_{(m',r) \in A_i} A_i(m', r) * \sum_{p \in P} m'(p) * Cost(p)$. Therefore, R is of the form $\langle c'', {A'_{-n}}^{+1} * \ldots * {A'_{-1}}^{+1} * A'_0 * \ldots * A'_k, \emptyset, {A'_{k+1}}^{+1} * \ldots * {A'_m}^{+1} \rangle$, with $c' \geq c''$ and for each $i \in \{-n \ldots m\}$, $A'_i \leq^{\oplus} A_i$ (if A'_i is $\emptyset$, then it is removed). Then, $R_2' = \langle c''', A'_{-n} * \ldots * A'_{-1}, A'_0, A'_1 * \ldots * A'_m \rangle \in \overline{Pre}_{\delta_3}(R)$ (or $R_2' \in \overline{Pre}_{\delta_4}(R)$ if $A'_0 = \emptyset$) and $R_2 \gg R_2'$.

- Finally, suppose that $R_2 \xrightarrow{4} R_1'$. Then, there is $k \in m^+$ such that $R_1' = \langle c', A_{-n}^{+1} * \ldots * A_{-1}^{+1} * A_0 * \ldots * A_k, A_{k+1}^{+1}, A_{k+2}^{+1} * \ldots * A_m^{+1} \rangle$, with $c' = c + \sum_{i=-n}^{m} \sum_{(m',r) \in A_i} A_i(m', r) * \sum_{p \in P} m'(p) * Cost(p)$. Therefore, R is of the form $\langle c'', {A'_{-n}}^{+1} * \ldots * {A'_{-1}}^{+1} * A'_0 * \ldots * A'_k, {A'_{k+1}}^{+1}, {A'_{k+2}}^{+1} * \ldots * {A'_m}^{+1} \rangle$, with $c' \geq c''$ and for each $i \in \{-n \ldots m\}$, $A'_i \leq^{\oplus} A_i$ (if A'_i is $\emptyset$, then it is removed). Then, $R_2' = \langle c''', A'_{-n} * \ldots * A'_{-1}, A'_0, A'_1 * \ldots * A'_m \rangle \in \overline{Pre}_{\delta_3}(R)$ (or $R_2' \in \overline{Pre}_{\delta_4}(R)$ if $A'_0 = \emptyset$) and $R_2 \gg R_2'$.

Hence, $\uparrow Pre_\Delta(\uparrow R) \subseteq \uparrow \overline{Pre}_\Delta(R)$, so $\uparrow Pre_\Delta(\uparrow R) = \uparrow \overline{Pre}_\Delta(R)$.

□

Now we focus on the discrete firings. We define $\overline{Pre}_t$ for each transition $t \in T$ to prove that we can compute $\uparrow Pre_t(\uparrow R)$ for any priced region R. Now, we are ready to define $\overline{Pre}_t$.

Definition B.2 ($\overline{Pre}_t$) *Let $t \in T$ be a transition of a ν-aPTdPN N, and $R = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * A_2 * \ldots * A_m \rangle$ a region of N. Then, $\langle c', B_{-n'} * \ldots * B_{-1}, B_0, B_1 * B_2 * \ldots * B_m \rangle \in \overline{Pre}_t(R)$ if and only if:*

- $c' = c - Cost(t)$.
- *There are* $O, O^\nu \in (P^\oplus \times (max+1)^*)^\oplus$ *and for each* $i \in \{-n, \ldots, -1, 1, \ldots, m\}$ *there is* A_i^{rest}*, such that there is a multiset* $A^\emptyset \in (P^\oplus \times (max+1)^*)^\oplus$ *, such that if* $(m, r) \in A^\emptyset$ *then* $m = \emptyset$*, with* $A^\emptyset + \biguplus_{i \in \{-n,\ldots,-1,1,\ldots,m\}} A_i = (\biguplus_{i \in \{-n,\ldots,-1,1,\ldots,m\}} A_i^{rest}) + O + O^\nu$.
- *There are* $A_0^O, A_0^\nu \in (P^\oplus \times (max+1)^*)^\oplus$ *and* A_0^{rest} *such that there are not pairs with empty marking in* A_0^{rest} *and* $A_0 + A = A_0^O + A_0^\nu + A_0^{rest}$*, where* A *is some multiset made of pairs with empty marking (as* $A^\emptyset$*).*
- *There is an injection* $h' : fVar(t) \to O^\nu + A_0^\nu$ *such that, if* $h'(\nu) = (m, r)$ *then* $m \subseteq Out_t(\nu)$ *and if* $h'(\nu)$ *is selected from* A_0^ν *then* $r \in Time_t^2(\nu)$ *and* $r + \epsilon \in Time_t^2(\nu)$ *otherwise.*
- *There is a bijection* $h : nfVar(t) \to O + A_0^O$ *a function* $g : Var(t) \to \{-n', \ldots, m'\}$ *and a monotone injection* $\phi : \{i \in \{-n+1, \ldots, m\} \mid A_i^{rest} \neq \emptyset\} \cup \{0\} \to \{i \in \{-n', \ldots, m'\} \mid\}$ *such that* $\phi(0) = 0$ *and:*
  - *For each* $(m, r) \in O$ *with* $h(x) = (m, r)$*,* $r + \epsilon \in Time_t^2(x)$ *and we call* $\gamma_x((m, r)) = (m', r')$*, where* $m' = (m \ominus Out_t(x)) + In_t(x)$ *(where* $a \ominus b = max\{0, a - b\}$*), and* $r' \in (max+1)^*$*, with* $r' \in Time_t^1(x)$ *if* $g(x) = 0$ *and* $r' + \epsilon \in Time_t^1(x)$ *otherwise.*
  - *Analogously, for each* $(m, r) \in A_0^O$ *with* $h(x) = (m, r)$*,* $r \in Time_t^2(x)$ *and we call* $\gamma_x((m, r)) = (m', r')$*, where* $m' = (m \ominus Out_t(x)) + In_t(x)$ *and* $r' \in (max+1)^*$*, with* $r' \in Time_t^1(x)$ *if* $g(x) = 0$ *and* $r' + \epsilon \in Time_t^1(x)$ *otherwise.*
  - *For each* $i \in \{-n', \ldots, m'\}$*,* $i \neq 0$*, if* $i = \phi(j)$ *for some* $j \in \{-n+1, \ldots, m\}$*, then* $B_i = A_j^{rest} + \biguplus_{x \in nfVar(t) \mid g(x) = i} \gamma_x(h(x))$ *and* $B_i = \biguplus_{x \in nfVar(t) \mid g(x) = i} \gamma_x(h(x))$ *otherwise.*

It only remains to prove that $\overline{Pre}_t$ computes the predecessors of the regions of ν-$aPTdPN$ correctly.

Lemma B.2 *Given an abstract region* R *and a transition* t*,* $\overline{Pre}_t(R)$ *is finite, computable and* $\uparrow \overline{Pre}_t(R) = \uparrow Pre_t(\uparrow R)$.

Proof: $\overline{Pre}_t(R)$ is clearly finite and computable, since there is a finite number of such $\emptyset$-expansions of the region R, and they are all finite, and therefore there

is a finite number of ways of choosing the multisets O, O^ν, A_0^O, A_0^ν and A_i^{rest}. Analogously, as the regions in $\overline{Pre}_t(R)$, the functions h, h', g and ϕ can be defined in a finite number of ways too. Moreover, note that for each pair (m, r) with $\gamma_x((m, r)) = (m', r')$, m' can only be $(m \ominus Out_t(x)) + In_t(x)$, and $r' \in (max + 1)^*$ and therefore, it is able to take a finite number of values. Hence, $\overline{Pre}_t(R)$ is finite.

Now we prove that $\uparrow\overline{Pre}_t(R) = \uparrow Pre_t(\uparrow R)$. Again, we prove the two containments separately. Let us first prove that $\uparrow\overline{Pre}_t(R) \subseteq \uparrow Pre_t(\uparrow R)$.

Suppose $R_1 \in \uparrow\overline{Pre}_t(R)$. Then, there is $R_1' \ll R_1$ such that $R_1' \in \overline{Pre}_t(R)$. If we prove that $R_1' \overset{t}{\rightarrowtail} R' \gg R$ then, by compatibility, $R_1 \overset{t}{\rightarrowtail} R'' \gg R$, and we are done. Suppose that $R = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * A_2 * \ldots * A_m \rangle$, $R_1' = \langle c', B_{-n'} * \ldots * B_{-1}, B_0, B_1 * B_2 * \ldots * B_m \rangle$ and $O, O^\nu, A_0^O, A_0^\nu, h, h', g, \phi, \gamma_x$ and A_i^{rest} for $i \in \{-n, \ldots, m\}$ and $x \in nfVar(t)$, are the multisets and functions of the previous definition for R_1'. Then, we prove that $\langle c', B_{-n'} * \ldots * B_{-1}, B_0, B_1 * B_2 * \ldots * B_m \rangle \overset{t}{\rightarrowtail} \langle c, A'_{-n} * \ldots * A'_{-1}, A'_0, A'_1 * A'_2 * \ldots * A'_m \rangle$, where for each $i \in \{-n, \ldots, m\}$, $A_i \leq^\oplus A_i'$. We prove the five conditions for the firing:

- By the first point of Def. B.2, $c' = c - Cost(t)$. Hence, $c = c' + Cost(t)$, as required.

- For each $i \in \{-n, \ldots, m\}$, if $\phi(j) = i$ then we define $B_i^{rest} = A_j^{rest}$ and $B_i^I = B_i - B_i^{rest}$. Otherwise, $B_i^{rest} = \emptyset$ and $B_i^I = B_i$. Clearly, we have that $B_i = B_i^{rest} + B_i^I$.

- For the firing, we consider the multisets O' and $A_0^{O'}$ as follows: if $(m, r) \in P^\oplus \times (max + 1)^*$, then $O'((m, r)) = |\{(m', r) \in O \mid \exists x \in nfVar, h(x) = (m', r), \gamma_x((m', r)) = (m'', r'')$ and $m'' = (m - In_t(x)) + Out_t(x)\}|$. Hence, in O' we consider the same number of pairs as in O, and for each pair $(m, r) \in O'$ there is a different pair $(m', r) \in O$, such that $m' \subseteq m$. Hence, $O \leq^\oplus O'$. We define $A_0^{O'}$ analogously. Moreover, we can consider a bijection $H : nfVar(t) \to O' + A_0^{O'}$ such that, for each $x \in nfVar(t)$, $H(x) = (m, r)$ with $(m, r) \geq h(x)$.

  Note that $\biguplus_{i \in n', \ldots, m'} B_i^I = \biguplus_{i \in n', \ldots, m'} \biguplus_{x \in nfVar(t) | g(x) = i} \gamma_x(h(x))$. Hence, we define $f : nfVar(t) \to \biguplus_{i \in n', \ldots, m'} \biguplus_{x \in nfVar(t) | g(x) = i} \gamma_x(h(x))$. Indeed, for each $x \in nfVar(t)$, with $h(x) = (m, r)$, we define $f(x) = \gamma_x((m, r))$. Then, if $h(x) = (m, r)$ and $f(x) = \gamma_x((m, r)) = (m', r')$:

  - If $f(x) = \gamma_x((m, r))$ is in B_0 (so $g(x) = 0$), then $m' = (m \ominus Out_t(x)) + In_t(x)$, so $In_t(x) \subseteq m'$, and $r' \in Time_t^1(x)$ by the fifth condition of

Def. B.2. Hence, $match(f(x), (In_t(x), Time_t^1(x)))$, as required. Analogously, if $f(x) = \gamma_x((m, r))$ is not in B_0, then $m' = (m \ominus Out_t(x)) + In_t(x)$, so $In_t(x) \subseteq m'$, and $r' + \epsilon \in Time_t^1(x)$, so $match(f(x) + \epsilon, (In_t(x), Time_t^1(x)))$.

- By the fifth point of Def. B.2, $m' = (m \ominus Out_t(x)) + In_t(x)$, and therefore $m \subseteq m'' = (m' - In_t(x)) + Out_t(x)$. Moreover, by the fourth condition, we have that if (m, r) (which in the firing will correspond to (m'', r)) is selected from A_0^O then $r \in Time_t^2(x)$ and if (m, r) is selected from O then $r + \epsilon \in Time_t^2(x)$.

- We consider the multisets $O^{\nu\prime}$ and $A_0^{\nu\prime}$ as follows: if $(m, r) \in P^{\oplus} \times (max + 1)^*$, then $O^{\nu\prime}((m, r)) = |\{(m', r) \in O \mid \exists \nu \in nfVar, h(\nu) = (m', r) \text{ and } m = Out_t(\nu)\}|$. Hence, there is a bijection between O^{ν} and $O^{\nu\prime}$ such that, to each pair $(m, r) \in O^{\nu}$, it corresponds a pair $(m', r) \in O^{\nu\prime}$ with $m \subseteq m'$. We define $A_0^{\nu\prime}$ analogously. Moreover, we consider the injection $H' : fVar(t) \to O^{\nu\prime} + A_0^{\nu\prime}$ such that, for each $\nu \in nfVar(t)$, $H'(\nu) = (Out_t(\nu), r) \geq h'(\nu) = (m, r)$, for some $m \subseteq Out_t(\nu)$. Then, if $H'(\nu) = (m, r)$, then we have:

  - $m = Out_t(\nu)$ and
  - by the fourth point of Def. B.2, if $H'(\nu)$ is selected from A_0^{ν} then $r \in Time_t^2(\nu)$ and $r + \epsilon \in Time_t^2(\nu)$ otherwise.

- We define the injection $\varphi : \{i \in \{-n', \ldots, m'\} \mid B_i^{rest} \neq \emptyset\} \to \{-n, \ldots, m\}$ such that $\phi(i) = j$ if and only if $\varphi(j) = \varphi(i)$. Note that $\big|\{i \mid B_i^{rest} \neq \emptyset\}\big| = \big|\{i \mid A_i^{rest} \neq \emptyset\}\big|$, so φ is completely defined this way. Moreover:

  - By the last point of the previous definition, if $i = \phi(j)$ for some $j \in \{-n', \ldots, m'\}$, then $B_i = A_j^{rest} + \biguplus_{x \in nfVar(t) \mid g(x) = i} \gamma_x(h(x))$. In this case, $j = \varphi(i)$ and $B_i^{rest} \subseteq A_{\varphi(i)}$.
  - By the third condition of Def. B.2, $A_0 + A = A_0^O + A_0^{\nu} + B_0^{rest}$, where A only have instances with empty marking. Hence, we can define $A_0' = A_0^{O'} + A_0^{\nu\prime} + B_0^{rest} \geq^{\oplus} A_0$.
  - Finally, by the second condition, we have that $A^{\emptyset} + \biguplus_{i \in \{-n, \ldots, -1, 1, \ldots, m\}} A_i = (\biguplus_{i \in \{-n, \ldots, -1, 1, \ldots, m\}} A_i^{rest}) + O + O^{\nu}$, with $A^{\emptyset}$ made of pairs with empty marking, which implies that $A^{\emptyset} + \biguplus_{i \in \{-n, \ldots, -1, 1, \ldots, m\}}$ $A_i = (\biguplus_{i \in \{-n', \ldots, -1, 1, \ldots, m'\}} B_i^{rest}) + O + O^{\nu} \leq^{\oplus} (\biguplus_{i \in \{-n', \ldots, -1, 1, \ldots, m'\}} B_i^{rest}) +$ $O' + O^{\nu\prime}$. Hence, for each $i \in \{-n, \ldots, m\}$, we can define A_i' such that

$B_i^{rest} \subseteq A'_{\phi(i)}$ and $\biguplus_{i\in\{-n,\ldots,-1,1,\ldots,m\}} A'_i = (\biguplus_{i\in\{-n',\ldots,-1,1,\ldots,m'\}} B_i^{rest}) + O' + O^{\nu\prime}$

Hence $\langle c', B_{-n'} * \ldots * B_{-1}, B_0, B_1 * B_2 * \ldots * B_m\rangle \overset{t}{\rightarrowtail} \langle c, A'_{-n} * \ldots * A'_{-1}, A'_0, A'_1 * A'_2 * \ldots * A'_m\rangle \gg \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * A_2 * \ldots * A_m\rangle$, as we wanted to prove.

Now we prove that $\uparrow Pre_t(\uparrow R) \subseteq \uparrow \overline{Pre}_t(R)$. Suppose that $R_1 \in \uparrow Pre_t(\uparrow R)$. Then, there is $R'_1 \ll R_1$ such that $R'_1 \overset{t}{\rightarrowtail} R_2 \gg R$. We prove that there is $R_3 \in \overline{Pre}_t(R)$ with $R'_1 \gg R_3$. Let $R = \langle c, A_{-n} * \ldots * A_{-1}, A_0, A_1 * A_2 * \ldots * A_m\rangle$, $R_2 = \langle c', a_{-n-1} * A'_{-n} * a_{-n} * \ldots * A'_{-1} * a_{-1}, A'_0, a_0 * A'_1 * a_1 \ldots * A'_m * a_m\rangle$, with $c' \geq c$ and for each $j \in \{-n-1, \ldots, m\}$, $A_j \leq^{\oplus} A'_j$ and $a_j \in ((P^{\oplus} \times (max+1)^*)^{\oplus})^{\circledast}$. Moreover, let $R'_1 = \langle c'', B_{-n'} * \ldots * B_{-1}, B_0, B_1 * \ldots * B_{m'}\rangle$. We are going to define $R_3 \in \overline{Pre}_t(R)$ with $R_3 = \langle c''', B'_{-n'} * \ldots * B'_{-1}, B'_0, B'_1 * \ldots * B'_{m'}\rangle$, with $c'' \geq c'''$ and for each $j \in \{-n', \ldots, m'\}$, $B_j \leq^{\oplus} B'_j$ (in fact, the R_3 we obtain may be an $\emptyset$-expansion of the required region).

Let us consider the functions f, h, h' and φ, the multisets O, A_0^O, O^ν, A_0^ν, and the decomposition $B_i = B_i^{rest} + B_i^I$ for each $i \in \{-n', \ldots, m'\}$ in the firing of t from R'_1. Then, we define $R_3 = \langle c''', B'_{-n'} * \ldots * B'_{-1}, B'_0, B'_1 * \ldots * B'_{m'}\rangle$ with:

- $c''' = c - Cost(t)$, as required in the first condition of Def. B.2. Moreover, $c' \geq c$, and $c'' = c' - Cost(t)$. Hence, $c''' \leq c''$.

- Let us consider some $A_i = \{(a_1, r_1), \ldots, (a_{k_i}, r_{k_i})\}$, with $i \in \{-n, \ldots, m\}$. As $A_i \subseteq A'_i$, there is an injection $\alpha_i : A_i \xrightarrow{A}{}'_i$ such that for each $(m, r) \in A_i$, $\alpha_i((m, r)) = (m', r)$, with $m \leq m'$. Then, if there is j with $\varphi(j) = i$, then for each pair $(m, r) \in P^{\oplus} \times (max+1)^*$ we define $A^{rest}(m, r) = |\{k \in k_i^+ \mid (m, r) = (m_k, r_k), \alpha_i((m_k, r_k))$ is one of the pairs in B_j^{rest} in the firing$\}|$. Moreover, if $i \neq 0$, we define $O'_i(m, r) = |\{k \in k_i^+ \mid (m, r) = (m_k, r_k), \alpha_i((m_k, r_k))$ is one of the pairs in O in the firing$\}|$ and $O^{\nu\prime}{}_i(m, r) = |\{k \in k_i^+ \mid (m, r) = (m_k, r_k), \alpha_i((m_k, r_k))$ is one of the pairs in O^ν in the firing$\}|$.

  Let us call O'' and $O^{\nu\prime\prime}$ the multisets of pairs in O and O^ν respectively, which are set to a multiset in some a_i , or have not been selected to build any lower pair in any O'_i or $O^{\nu\prime}{}_i$. Then, we define two multisets $O'_{m+1}, O^{\nu\prime}{}_{m+1} \in (P^{\oplus} \times (max+1)^*)^{\oplus}$ such that, $O'_{m+1}(m, r) = 0$ if $m \neq \emptyset$ and $O'_{m+1}(\emptyset, r) = \sum_{m\in P^{\oplus}} O''(m, r)$. Analogously, $O^{\nu\prime}{}_{m+1}(m, r) = 0$ if $m \neq \emptyset$ and $O^{\nu\prime}{}_{m+1}(\emptyset, r) = \sum_{m\in P^{\oplus}} O''(m, r)$.

By Def. 4.2.4, $\biguplus_{i\in\{-n,\ldots,-1,1,\ldots,m\}} A'_i + \biguplus_{i\in\{-n-1,\ldots,-1,1,\ldots,m\}} \biguplus_{A\in a_i} A = (\biguplus_{i\in\{-n',\ldots,-1,1,\ldots,m'\}} B_i^{rest}) + O + O^\nu$. Hence, for each $A_i \neq A_0$ in R, $A_i = A^{rest} + O'_i + O^{\nu\prime}$ and if we define $O' = \biguplus_{i\in\{-n,\ldots,-1,1,\ldots,m+1\}} O'_i$ and $O^{\nu\prime} = \biguplus_{i\in\{-n,\ldots,-1,1,\ldots,m+1\}} O_i^{\nu\prime}$, then $O'_{m+1} + O^{\nu\prime} + \biguplus_{i\in\{-n,\ldots,-1,1,\ldots,m\}} A_i = (\biguplus_{i\in\{-n,\ldots,-1,1,\ldots,m\}} A_i^{rest}) + O' + O^{\nu\prime}$, as required.

- Analogously, we define $A_0^{O''}$ and $A_0^{\nu\prime\prime}$ such that, for each $(m,r) \in (P^\oplus \times (m+1)^*)^\oplus$, $A_0^{O''}(m,r) = |\{k \in k_0^+ \mid (m,r) = (m_k,r_k), \alpha_0((m_k,r_k)) \text{ is one of the pairs in } A_0^O \text{ in the firing}\}|$ and $A_0^{\nu\prime\prime}(m,r) = |\{k \in k_0^+ \mid (m,r) = (m_k,r_k), \alpha_0((m_k,r_k)) \text{ is one of the pairs in } A_0^\nu \text{ in the firing}\}|$.

  Moreover, we call $A_{0\,1}^O$ and $A_{0\,1}^\nu$ the multisets of pairs in A_0^O and A_0^ν respectively, which we have not selected to build any lower pair in any $A_0^{O''}$ or $A_0^{\nu\prime\prime}$. Then, we define two multisets $A_0^{O'''}, A_0^{\nu\prime\prime\prime} \in (P^\oplus \times (max+1)^*)^\oplus$ such that, $A_0^{O'''}(m,r) = 0$ if $m \neq \emptyset$ and $A_0^{O'''}(\emptyset,r) = \sum_{m\in P^\oplus} A_{0\,1}^O(m,r)$. Analogously, $A_0^{\nu\prime\prime\prime}(m,r) = 0$ if $m \neq \emptyset$ and $A_0^{\nu\prime\prime\prime}(\emptyset,r) = \sum_{m\in P^\oplus} A_{0\,1}^\nu(m,r)$. We define $A_0^{O'} = A_0^{O''} + A_0^{O'''}$ and $A_0^{\nu\prime} = A_0^{\nu\prime\prime} + A_0^{\nu\prime\prime\prime}$.

  By Def. 4.2.4, $B_0 = A_0^{rest} + A_0^O + A_0^\nu$. Hence, $A_0 = A_0^{rest} + A_0^{O''} + A_0^{\nu\prime\prime}$. Moreover, $A_0 + A_0^{O'''} + A_0^{\nu\prime\prime\prime} = A_0^{rest} + A_0^{O'} + A_0^{\nu\prime}$, as required.

- Note that there is a bijection $\beta : O^\nu + A_0^\nu \to O^{\nu\prime} + A_0^{\nu\prime}$ such that if $\beta((m,r)) = (m',r')$, then $m' \subseteq m$ and $r = r'$. Moreover, β assigns elements of $O^{\nu\prime}$ to elements of O^ν and elements of $A_0^{\nu\prime}$ to elements of A_0^ν. Hence, we can define $H' : fVar(t) \to O^{\nu\prime} + A_0^{\nu\prime}$ with $H'(\nu) = \beta(h'(\nu))$, and by Def. 4.2.4, we have that if $H'(\nu) = (m,r)$ then $m \subseteq Out_t(\nu)$ and if $H'(\nu)$ is selected from $A_0^{\nu\prime}$ then $r \in Time_t^2(\nu)$ and $r + \epsilon \in Time_t^2(\nu)$ otherwise.

- In an analogous way, there is a bijection $\chi : O + A_0 \to O' + A_0^{O'}$ such that if $\chi((m,r)) = (m',r')$, then $m' \subseteq m$ and $r = r'$, which assigns elements of O' to elements of O and elements of $A_0^{O'}$ to elements of A_0^O. Then, we define $H : nfVar(t) \to O + A_0^O$ with $H(x) = \chi(h(x))$. We also define $g : Var(t) \to \{-n', \ldots, m'\}$ such that, for each $x \in Var(t)$, $g(x) = i$ if $f(x)$ is selected from A_i in the firing from R'_1. Moreover, we define ϕ : $\{i \in \{-n+1,\ldots,m\} \mid A_i^{rest} \neq \emptyset\} \cup \{0\} \to \{-n', \ldots, m'\}$ such that for each $i \in \{0\} \cup \{j \in \{-n+1,\ldots,m\} \mid A_j^{rest} \neq \emptyset\}$ if $i = \varphi(f)$, then $\phi(i) = j$ (it is well defined because of the definition and conditions over φ in Def. 4.2.4). We have:

  - For each $(m,r) \in O'$ with $H(x) = (m,r)$, we have that $h(x) = (m_1,r) \in$

O, for some $m_1 \supseteq m$. Hence, by the third point of Def. 4.2.4, we have $r + \epsilon \in Time_t^2(x)$. Let us call $\gamma_x((m,r)) = (m', r')$, where $m' = (m \ominus Out_t(x)) + In_t(x)$, and r' is such that there is m_2 with $f(x) = (m_2, r')$. Then, again by Def. 4.2.4, $r' \in (max+1)^*$, with $r' \in Time_t^1(x)$ if $g(x) = 0$ and $r' + \epsilon \in Time_t^1(x)$ otherwise. In this case, since $m_1 = (m_2 - In_t(x)) + Out_t(x) \supseteq m$, $m' = (m \ominus Out_t(x)) + In_t(x) \subseteq m_2$. Then, if (m_2, r') is in B_i, then we can set the corresponding pair (m', r') in B'_i.

- For each $(m,r) \in A_0^O$ with $H(x) = (m,r)$, now we have that $h(x) = (m_1, r) \in A_0^O$, for some $m_1 \supseteq m$. Therefore, by the third point of Def. 4.2.4, we have $r \in Time_t^2(x)$. Again, if we call $\gamma_x((m,r)) = (m', r')$, where $m' = (m \ominus Out_t(x)) + In_t(x)$, and r' is such that there is m_2 with $f(x) = (m_2, r')$. Then, $r' \in (max+1)^*$, with $r' \in Time_t^1(x)$ if $g(x) = 0$ and $r' + \epsilon \in Time_t^1(x)$ otherwise. Since $m_1 = (m_2 - In_t(x)) + Out_t(x) \supseteq m$, we have $m' = (m \ominus Out_t(x)) + In_t(x) \subseteq m_2$. Then, if (m_2, r') is in B_i, we can set the corresponding pair (m', r') in B'_i.
- Therefore, for each $i \in \{-n', \ldots, m'\}$, $i \neq 0$, if $i = \phi(j)$ for some $j \in \{-n+1, \ldots, m\}$, then we define $B'_i = A_j^{rest} + \biguplus_{x \in nfVar(t) | g(x) = i} \gamma_x(H(x))$. Note that $A_j^{rest} \leq^{\oplus} B_j^{rest}$ and $\biguplus_{x \in nfVar(t) | g(x) = i} \gamma_x(H(x)) \leq^{\oplus} B_i^I$, and hence $B'_i \leq^{\oplus} B_i$. Otherwise, we define $B'_i = \biguplus_{x \in nfVar(t) | g(x) = i} \gamma_x(h(x)) \leq^{\oplus} B_i^I = B_i$.

In fact, the obtained region $\langle c''', B'_{-n'} * \ldots * B'_{-1}, B'_0, B'_1 * \ldots * B'_{m'} \rangle$ may have some empty mutisets. If we discard these multisets, we obtain a region R_3 such that, as we have proved, $R'_1 \gg R_3 \in \overline{Pre}_t(R)$. Hence, $\uparrow Pre_t(\uparrow R) \subseteq \ \uparrow \overline{Pre}_t(R)$.

□

Therefore, as $\rightarrowtail = \{\overset{i}{\rightarrowtail} \mid i \in 4^+\} \cup \{\overset{t}{\rightarrowtail} \mid t \in T\}$, and we have defined $\overline{Pre}_\Delta$ and $\overline{Pre}_t$ for each $t \in T$, so that $Pre_\Delta(\uparrow R) = \uparrow \overline{Pre}_\Delta(R)$ and $Pre_t(\uparrow R) = \uparrow \overline{Pre}_t(R)$, we obtain the following corollary.

Corollary B.3 *$\rightarrowtail$ has effective Pred-basis.*

Bibliography

[1] P. Abdulla, J. Deneux, and P. Mahata. Multi-clock timed networks. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 345–354, July 2004.

[2] P. Abdulla and R. Mayr. Computing optimal coverability costs in priced timed Petri nets. In *26th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 399–408, June 2011.

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

[4] P. A. Abdulla, G. Delzanno, and L. V. Begin. A classification of the expressive power of well-structured transition systems. *Inf. Comput.*, 209(3):248–279, 2011.

[5] P. A. Abdulla, J. Deneux, P. Mahata, and A. Nylén. Forward reachability analysis of timed Petri nets. In Y. Lakhnech and S. Yovine, editors, *FORMATS/FTRTFT*, volume 3253 of *Lecture Notes in Computer Science*, pages 343–362. Springer, 2004.

[6] P. A. Abdulla and B. Jonsson. Verifying networks of timed processes (extended abstract). In B. Steffen, editor, *TACAS*, volume 1384 of *Lecture Notes in Computer Science*, pages 298–312. Springer, 1998.

[7] P. A. Abdulla and B. Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theoretical Computer Science*, 256(1-2):145–167, 2001.

[8] P. A. Abdulla, P. Mahata, and R. Mayr. Dense-timed Petri nets: Checking zenoness, token liveness and boundedness. *Logical Methods in Computer Science*, 3(1), 2007.

[9] P. A. Abdulla and R. Mayr. Minimal cost reachability/coverability in priced timed Petri nets. In de Alfaro [26], pages 348–363.

[10] P. A. Abdulla and A. Nylén. Timed Petri nets and bqos. In J. M. Colom and M. Koutny, editors, *ICATPN*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2001.

[11] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *INFORMATION AND COMPUTATION*, 104:2–34, 1993.

[12] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[13] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[14] T. Araki and T. Kasami. Some decision problems related to the reachability problem for Petri nets. *Theoretical Computer Science*, 3(1):85–104, 1976.

[15] G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *STTT*, 8(3):204–215, 2006.

[16] R. Bonnet, A. Finkel, S. Haddad, and F. Rosa-Velardo. Ordinal theory for expressiveness of well structured transition systems. *Inf. Comput.*, 2012.

[17] P. Bouyer, T. Brihaye, V. Bruyre, and J.-F. Raskin. On the optimal reachability problem of weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, 2007.

[18] M. Boyer and O. H. Roux. Comparison of the expressiveness of arc, place and transition time Petri nets. In *Proceedings of the 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency*, ICATPN'07, pages 63–82, Berlin, Heidelberg, 2007. Springer-Verlag.

[19] G. Cantor. Beitrge zur begrndung der transfiniten mengenlehre. *Mathematische Annalen*, 46(4):481–512, 1895.

[20] I. Cervesato. Typed msr: Syntax and examples. In V. Gorodetski, V. Skormin, and L. Popyack, editors, *Information Assurance in Computer Networks*, volume 2052 of *Lecture Notes in Computer Science*, pages 159–177. Springer Berlin Heidelberg, 2001.

[21] K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4), 2010.

[22] S. Christensen, L. M. Kristensen, and T. Mailund. A sweep-line method for state space exploration. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS 2001, pages 450–464, London, UK, UK, 2001. Springer-Verlag.

[23] G. Ciardo. Discrete-time markovian stochastic Petri nets, 1995.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2 edition, 2001.

[25] A. David, L. Jacobsen, M. Jacobsen, and J. Srba. A forward reachability algorithm for bounded timed-arc Petri nets. In F. Cassez, R. Huuck, G. Klein, and B. Schlich, editors, *Proceedings Seventh Conference on Systems Software Verification, SSV 2012, Sydney, Australia, 28-30 November 2012.*, volume 102 of *EPTCS*, pages 125–140, 2012.

[26] L. de Alfaro, editor. *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5504 of *Lecture Notes in Computer Science*. Springer, 2009.

[27] D. de Frutos-Escrig and C. Johnen. Decidability of home space property. Technical Report LRI-503, Univ. de Paris-Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique, 1989.

[28] D. H. J. de Jongh and R. Parikh. Well partial orderings and hierarchies. In *Indagationes Mathematicae*, volume 80, pages 195–207, 1977.

[29] G. Delzanno. An assertional language for the verification of systems parametric in several dimensions. *Electr. Notes Theor. Comput. Sci.*, 50(4):371–385, 2001.

[30] G. Delzanno. An overview of MSR(C): A CLP-based framework for the symbolic verification of parameterized concurrent systems. *Electronic Notes in Theoretical Computer Science*, 76:65 – 82, 2002. {WFLP} 2002, 11th International Workshop on Functional and (Constraint) Logic Programming, Selected Papers.

[31] G. Delzanno. Constraint-based automatic verification of abstract models of multithreaded programs. *TPLP*, 7(1-2):67–91, 2007.

[32] G. Delzanno and F. Rosa-Velardo. On the coverability and reachability languages of monotonic extensions of Petri nets. *Theor. Comput. Sci*, 467:12–29, 2013.

[33] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422, 1913.

[34] J. Esparza and M. Nielsen. Decidability issues for Petri nets - a survey, 1994.

[35] A. Finkel. A generalization of the procedure of Karp and Miller to well structured transition systems. In T. Ottmann, editor, *ICALP*, volume 267 of *Lecture Notes in Computer Science*, pages 499–508. Springer, 1987.

[36] A. Finkel. Reduction and covering of infinite reachability trees. *Inf. Comput.*, 89(2):144–179, Dec. 1990.

[37] A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.

[38] R. Floyd. Assigning meanings to programs. In T. Colburn, J. Fetzer, and T. Rankin, editors, *Program Verification*, volume 14 of *Studies in Cognitive Systems*, pages 65–81. Springer Netherlands, 1993.

[39] J. Gannon, P. McMullin, and R. Hamlet. Data abstraction, implementation, specification, and testing. *ACM Trans. Program. Lang. Syst.*, 3(3):211–223, July 1981.

[40] G. Geeraerts, J.-F. Raskin, and L. V. Begin. Well-structured languages. *Acta Inf.*, 44(3-4):249–288, 2007.

[41] P. Godefroid and P. Wolper. A partial approach to model checking. *Inf. Comput.*, 110(2):305–326, 1994.

[42] M. Hack. Petri net language. Technical report, Cambridge, MA, USA, 1976.

[43] S. Haddad, S. Schmitz, and P. Schnoebelen. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *LICS*, pages 355–364. IEEE, 2012.

[44] K. V. Hee, E. Serebrenik, N. Sidorova, and M. Voorhoeve. Soundness of resource-constrained workflow nets. In *In ICATPN*, pages 250–267, 2005.

[45] M. Hennessy. Acceptance trees. *J. ACM*, 32(4):896–928, Oct. 1985.

[46] F. C. Hennine. Fault detecting experiments for sequential circuits. In *Proceedings of the 1964 Proceedings of the Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, SWCT '64, pages 95–110, Washington, DC, USA, 1964. IEEE Computer Society.

[47] G. Higman. Ordering by Divisibility in Abstract Algebras. *Proc. London Math. Soc.*, s3-2(1):326–336, jan 1952.

[48] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, Oct. 1969.

[49] K. Jensen. Coloured Petri nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*, volume 254 of *Lecture Notes in Computer Science*, pages 248–299. Springer, 1986.

[50] K. Jensen. Coloured Petri nets: A high level language for system design and analysis. In Rozenberg [82], pages 342–416.

[51] K. Jensen, L. M. Kristensen, and L. Wells. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. *STTT*, 9(3-4):213–254, 2007.

[52] K. Y. J. Joakim Byg and J. Srba. In *Proceedings of 7th International Symposium on Automated Technology for Verification and Analysis (ATVA'09), volume 5799 of LNCS, Springer-Verlag*, pages 84–89, Netherlands.

[53] G. Juhs, I. Kazlov, and A. Juhsov. Instance deadlock: A mystery behind frozen programs. In J. Lilius and W. Penczek, editors, *Applications and Theory of Petri Nets*, volume 6128 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2010.

[54] T. Junttila. New canonical representative marking algorithms for place/transition-nets. In J. Cortadella and W. Reisig, editors, *Applications and Theory of Petri Nets 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 258–277. Springer Berlin Heidelberg, 2004.

[55] R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.

[56] J. Kleijn and M. Koutny. Localities in systems with a/sync communication. *Theoretical Computer Science*, 429(0):185 – 192, 2012.

[57] M. Knapik, W. Penczek, M. Szreter, and A. Pólrola. Bounded parametric verification for distributed time Petri nets with discrete-time semantics. *Fundam. Inf.*, 101(1-2):9–27, Jan. 2010.

[58] S. R. Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 267–281, New York, NY, USA, 1982. ACM.

[59] L. Kristensen and T. Mailund. A generalised sweep-line method for safety properties. In L.-H. Eriksson and P. Lindsay, editors, *FME 2002:Formal MethodsGetting IT Right*, volume 2391 of *Lecture Notes in Computer Science*, pages 549–567. Springer Berlin Heidelberg, 2002.

[60] J. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99(1):79–104, 1992.

[61] K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Efficient verification of real-time systems: compact data structure and state-space reduction. In *Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS '97), December 3-5, 1997, San Francisco, CA, USA*, pages 14–24. IEEE Computer Society, 1997.

[62] R. Lazic, T. Newcomb, J. Ouaknine, A. W. Roscoe, and J. Worrell. Nets with tokens which carry data. *Fundam. Inform.*, 88(3):251–274, 2008.

[63] J. Leroux. Vector addition system reachability problem: A short self-contained proof. *SIGPLAN Not.*, 46(1):307–316, Jan. 2011.

[64] J. A. Mateo, J. Srba, and M. Sø rensen. Soundness of timed-arc workflow nets. In G. Ciardo and E. Kindler, editors, *Application and Theory of Petri Nets and Concurrency*, volume 8489 of *Lecture Notes in Computer Science*, pages 51–70. Springer International Publishing, 2014.

[65] E. W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, pages 238–246, New York, NY, USA, 1981. ACM.

[66] J. Mccarthy. A basis for a mathematical theory of computation. In *Computer Programming and Formal Systems*, pages 33–70. North-Holland, 1963.

[67] M. L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

[68] E. P. Moore. Gedanken experiments on sequential machines. *Automata Studies. Princeton University Press*, 1956.

[69] R. D. Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34(1-2):83–133, 1984.

[70] D. Peled. All from one, one for all: on model checking using representatives. In C. Courcoubetis, editor, *Computer Aided Verification, 5th International Conference, CAV '93, Elounda, Greece, June 28 - July 1, 1993, Proceedings*, volume 697 of *Lecture Notes in Computer Science*, pages 409–423. Springer, 1993.

[71] J. L. Peterson. Petri nets. *ACM Comput. Surv.*, 9(3):223–252, Sept. 1977.

[72] C. A. Petri. *Kommunikation mit Automaten.* PhD thesis, Institut fur Instrumentelle Mathematik, Bonn, 1962.

[73] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978.

[74] C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets.* PhD thesis, Massachusetts Institute of Tenchnology, Cambridge., 1974.

[75] A. V. Ratzer, L. Wells, H. M. Lassen, M. Laursen, J. F. Qvortrup, M. S. Stissing, M. Westergaard, S. Christensen, and K. Jensen. CPN tools for editing, simulating, and analysing coloured Petri nets. In W. M. P. van der Aalst and E. Best, editors, *Applications and Theory of Petri Nets 2003, 24th International Conference, ICATPN 2003, Eindhoven, The Netherlands, June 23-27, 2003, Proceedings*, volume 2679 of *Lecture Notes in Computer Science*, pages 450–462. Springer, 2003.

[76] W. Reisig. *Primer in Petri Net Design.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1992.

[77] M. Riesz, M. Seckár, and G. Juhás. Petriflow: A Petri net based framework for modelling and control of workflow processes. In S. Donatelli, J. Kleijn, R. J. Machado, and J. M. Fernandes, editors, *Proceedings of the Workshops of the 31st International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (PETRI NETS 2010) and of the 10th International Conference on Application of Concurrency to System Design (ACSD 2010), Braga, Portugal, June, 2010*, volume 827 of *CEUR Workshop Proceedings*, pages 191–205. CEUR-WS.org, 2010.

[78] F. Rosa-Velardo. *Redes de Petri móviles para la especificación y verificación de propiedades de seguridad en sistemas ubicuos.* PhD thesis, Facultad de Ciencias Matemáticas, Universidad Complutense de Madrid, 2007.

[79] F. Rosa-Velardo and D. de Frutos-Escrig. Name creation vs. replication in Petri net systems. *Fundam. Inform.*, 88(3):329–356, 2008.

[80] F. Rosa-Velardo and D. de Frutos-Escrig. Decidability and complexity of Petri nets with unordered data. *Theoretical Computer Science*, 412(34):4439–4451, 2011.

[81] L. E. Rosier and H.-C. Yen. A multiparameter analysis of the boundedness problem for vector addition systems. *Journal of Computer and System Sciences*, 32(1):105–135, 1986.

[82] G. Rozenberg, editor. *Advances in Petri Nets 1990 [10th International Conference on Applications and Theory of Petri Nets, Bonn, Germany, June 1989, Proceedings]*, volume 483 of *Lecture Notes in Computer Science.* Springer, 1991.

[83] V. V. Ruiz, D. de Frutos Escrig, and F. C. Gomez. On non-decidability of reachability for timed-arc Petri nets. In *Proc. 8th. International Workshop on Petri Nets and Performance Models*, pages 188–196, 1999.

[84] D. Schmidt. Well-partial orderings and their maximal order types, 1979. Habilitationsscrift.

[85] K. Schmidt. How to calculate symmetries of Petri nets. *Acta Informatica*, pages 545–590, 1997.

[86] K. Schmidt. Integrating low level symmetries into reachability analysis. In S. Graf and M. Schwartzbach, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785 of *Lecture Notes in Computer Science*, pages 315–330. Springer Berlin Heidelberg, 2000.

[87] D. Scott. Toward a mathematical semantics for computer languages. Technical Report PRG06, OUCL, August 1971.

[88] J. Sifakis. Use of Petri nets for performance evaluation. *Acta Cybern.*, 4(2):185–202, 1979.

[89] R. Valk and M. Jantzen. The residue of vector sets with applications to decidability problems in Petri nets. *Acta Inf.*, 21:643–674, 1985.

[90] A. Valmari. Stubborn sets for reduced state space generation. In Rozenberg [82], pages 491–515.

[91] W. van der Aalst and A. ter Hofstede. Yawl: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[92] W. van der Aalst, K. van Hee, A. ter Hofstede, N. Sidorova, H. Verbeek, M. Voorhoeve, and M. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.

[93] W. M. P. van der Aalst. Three good reasons for using a Petri-net-based workflow management system, 1996.

[94] W. M. P. van der Aalst. Verification of workflow nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997,*

Proceedings, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer, 1997.

[95] W. M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[96] W. M. P. van der Aalst, D. Hauschildt, and H. M. W. Verbeek. A Petri-net-based tool to analyze workflows. In *Proceedings of Petri Nets in System Engineering (PNSE97)*, pages 78–90. University of Hamburg, 1997. FBI-HH-B-205/97.

[97] W. M. P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[98] K. van Hee, N. Sidorova, and M. Voorhoeve. Generalised soundness of workflow nets is decidable. In J. Cortadella and W. Reisig, editors, *Applications and Theory of Petri Nets 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 197–215. Springer Berlin Heidelberg, 2004.

[99] K. M. van Hee and N. Sidorova. The right timing: Reflections on the modeling and analysis of time. In J. M. Colom and J. Desel, editors, *Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*, volume 7927 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2013.

[100] K. M. van Hee, L. J. Somers, and M. Voorhoeve. Executable specifications for distributed information systems. pages 139–156, 1989.

[101] F. R. Velardo and D. de Frutos Escrig. Decidability problems in Petri nets with names and replication. *Fundamenta informaticae*, 105(3):291–317, 2010.

[102] H. M. W. Verbeek and W. M. P. van der Aalst. Woflan 2.0: A Petri-net-based workflow diagnosis tool. In *Application and Theory of Petri Nets 2000*, volume 1825, pages 475–484. Springer, Berlin, Verlag, 2000.

[103] A. Weiermann. A computation of the maximal order type of the term ordering on finite multisets. In K. Ambos-Spies, B. Löwe, and W. Merkle, editors, *CiE*, volume 5635 of *Lecture Notes in Computer Science*, pages 488–498. Springer, 2009.

[104] K. Wolf. Generating Petri net state spaces. In J. Kleijn and A. Yakovlev, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2007*, volume 4546 of *Lecture Notes in Computer Science*, pages 29–42. Springer Berlin Heidelberg, 2007.