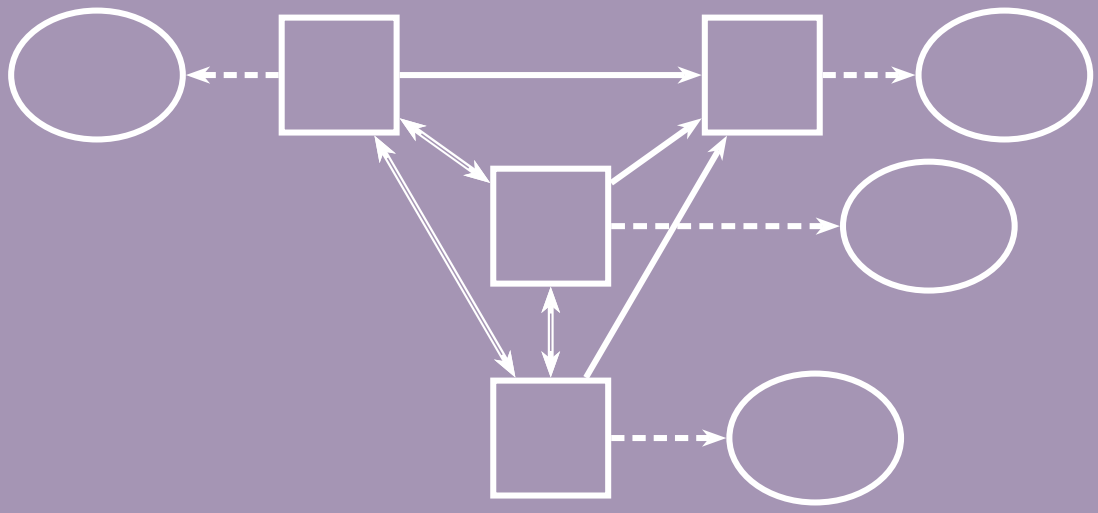
Compositional modelling using Petri nets with the analysis power of stochastic hybrid processes



Mariken Everdij

Compositional modelling using Petri nets with the analysis power of stochastic hybrid processes

The research for this thesis was funded in part by
the National Aerospace Laboratory NLR, Amsterdam
and Research Institute CTIT of the University of Twente

Graduation committee:

prof. dr. A. Bagchi	University of Twente	(promotor)
prof. dr. ir. B.R.H.M. Haverkort	University of Twente	(promotor)
prof. dr. ir. R. Boel	University of Gent	
dr. ir. H.A.P. Blom	National Aerospace Laboratory NLR	
dr. ir. R. Langerak	University of Twente	
dr. J.W. Polderman	University of Twente	
dr. ir. H.A. Reijers	Eindhoven University of Technology	
prof. dr. A.J. van der Schaft	University of Groningen	
prof. dr. A.A. Stoorvogel	University of Twente	

Title: Compositional modelling using Petri nets with the analysis power of stochastic hybrid processes
Author: M.H.C. Everdij
ISBN 978-90-365-3015-6

COMPOSITIONAL MODELLING USING PETRI NETS WITH THE ANALYSIS POWER OF STOCHASTIC HYBRID PROCESSES

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. H. Brinksma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 11 juni 2010 om 16.45 uur

door

Maria Hendrika Clara Everdij

geboren op 17 maart 1968

te Wageningen

Dit proefschrift is goedgekeurd door de promotoren:

prof. dr. A. Bagchi
prof. dr. ir. B.R.H.M. Haverkort

Contents

Chapter 1

Introduction

Motivation – safety assessment of large scale air transport operations

During the last three decades, the demand for air transport increased significantly. Statistics show that the number of commercial flights worldwide doubled from 18 million in 1980 to 38 million in 2007, [ITWM08]. It is generally expected that this trend will continue. However, the growth of air transport is bounded by limits to accommodate these numbers of flights, such as limits on the acceptable number of incidents and accidents that may occur, on the amount of noise and pollution, on the number of flight delays, on acceptable workload for air traffic controllers and pilots, and on the availability of suitable infrastructure.

In response to the growth trends, the air transport community has been continuously investigating means to create more capacity for the expected demand for air transport. In addition, even under the assumption that this demand does not increase, the occurrence of major accidents, such as the mid-air collision in 2002 between a Boeing-757 and a Tupolev-154 above Überlingen, Germany, and the subsequent media uproar, is a main driver to improve upon the ways in which air transport is managed and accommodated. New operational concepts are being developed, which involve the development of new procedures, modern technical systems and tools for pilots and controllers, new runways and taxiways, and the re-organisation of airspace structure.

One of the key questions during the development of such operational concept is: does the new concept indeed improve what it aims to improve? For example, is it indeed able to safely accommodate a doubling of air transport, does it indeed lead to acceptable workload for the air traffic controller, does it indeed lead to an acceptable number of aircraft accidents? Obviously, such questions need to be answered before the concept is actually introduced into practice, and before large investments are made to enable it.

Safety risk analysis of air transport operational concepts are a means towards addressing the safety-related questions above. Formally, risk is a product of *probability* (or frequency) and *consequences*. Probability is usually expressed in terms of a given exposure, e.g., the number

of events per aircraft flight hour, or per landing or departure. Consequences are often described in terms like catastrophic, major, minor. *Safety risk analysis* is a systematic approach for evaluating or assessing safety risk. It involves the identification of all perceivable safety-related situations, including their combinations and interactions, a predictive analysis of how and how often these situations occur, and a predictive analysis of the impact of these situations. In addition, the main contributors to risk are identified, so that they can be addressed at an early stage by the developers of the new operational concept in order to improve the situation. If several alternative operational concepts are evaluated in parallel, the analysis results can be used to drop prospectless ideas at an early stage, and to further improve the prospective ideas.

For any proposed operational concept, but particularly if the proposed operational concept is of a *large scale*, i.e., involving many elements, human operators, distributed systems, and complex interactions between all these elements, it is usually difficult to analyse the safety-related situations that may occur during its operation. The human mind, even the mind of an experienced safety expert, is simply not capable of having the overview of all combinations of safety-related situations, in order to assess their frequency and their consequences. The way out of this is to make a *model* of the operational concept, which covers the relevant elements and their interactions and combinations, and to analyse the concept based on the model.

Challenges in modelling of large scale air transport operations

The most popular existing risk modelling formalisms typically represent interactions between all entities involved by means of *linear relations*. Examples of these formalisms are fault trees and event trees, see, e.g., [EB08] for an overview and descriptions. The big advantage of these formalisms is that, once such models have been constructed, they are transparent and understandable to most experts, and as such, they are a great tool for risk communication purposes. A main disadvantage is that in case of complex operations, the interactions between entities are usually not linear and these formalisms fall short; the risk level due to the model will not represent the risk level of reality and even estimating the error made is very difficult. Typical non-linear properties of air transport operations are:

- Dynamics: Many processes are time-dependent and there is no fixed sequence of events. For example, the reaction time of an operator in response to an event may be longer due to the complexity of the situation, leading to other operators undertaking action first, though with another solution than the first operator would have taken.

- Multi-dimensional continuous processes: For example, the positions and velocities of multiple aircraft are continuous processes that have an impact on other processes such as collision detection and avoidance activities.

- Jumps: Air transport operations are influenced by discrete occurrences like technical failures or human interventions and decisions, which create discontinuities in otherwise continuous processes.
- Stochastics: Many of the processes and events are unpredictable or uncertain. Stochastics are present in different ways: in time, such as in sudden occurrences of events, and in state, such as uncertainties in observations of otherwise reasonably 'smooth' processes like the position of an aircraft.
- Complex interactions between distributed multiple agents: Air transport is a highly distributed safety critical operation. Each aircraft has its own crew, and each crew is communicating with and receives safety critical instructions from multiple human operators in different centres on the ground. All these agents interact, and common cause hazards may affect several agents as well as how they interact.

Since these non-linear properties cannot be captured well with the traditional linear approaches, an alternative modelling formalism is needed.

Stochastic hybrid processes to face the challenges

A *stochastic hybrid process* (SHP) is a generic name for a group of mathematical formalisms that capture the interaction of discrete and continuous dynamics and uncertainty. Here, the word *hybrid* refers to the notion that two different types of process (i.e., discrete and continuous) are combined, and the word *stochastic* refers to the uncertainties captured. Examples of stochastic hybrid processes are *piecewise deterministic Markov process* (PDP) [Dav84, Dav93], *switching diffusion process* (SDP) [GAM91], *stochastic hybrid system* (SHS) [HLS00], and *general stochastic hybrid process* (GSHP), [BBEP03, KB05a, BL06, Kry06]. Bujorianu *et al.* [BLGP03] and Krystul *et al.* [KB05a, Kry06] give comparative studies of these formalisms, which show that GSHP combine the features of the other approaches mentioned. This thesis focuses on the classes of PDP and GSHP. Since PDP are a special case of GSHP, the term GSHP is sometimes referred to as meaning "PDP and/or GSHP".

A GSHP is a stochastic hybrid process that, most of the time, follows the solution of a *stochastic differential equation*. At some times, however, the process may jump. Such jumps may be *spontaneous*, i.e., occurring at a random time, or *forced*, i.e., occurring when the process state hits the boundary of its state space. After the jump, the process follows the solution of a stochastic differential equation that may be different from the previous one, until the next jump occurs. For PDP, the stochastic differential equations are replaced by *ordinary differential equations*.

GSHP can represent most of the non-linear properties of air transport operations listed above, hence can be used to capture virtually all processes existing in air transport operations. In addition,

GSHP are supported by stochastic analysis instruments and have powerful mathematical properties, which guarantee a unique evaluation of the model and which allow speeding up this evaluation while keeping the model properties intact. The one property of air transport that cannot be easily addressed directly by means of GSHP is the last property in the list above, i.e., the complex interactions property. Using GSHP to construct a model of a complex air transport operation that is influenced by many factors such as human operators who communicate and make decisions, technical systems that interact, external influences, etc., is not easy. To support the modelling, and particularly the subsequent verification both by mathematical and by multiple operational domain experts, a supporting graphical modelling formalism is desired.

Petri nets to support the modelling of stochastic hybrid processes

For safety-critical operations in the nuclear and chemical industries, *Petri nets* have proven to be useful for the compositional specification of appropriate accident risk assessment models, and there is an abundance of literature available on their use, properties and applications, see, e.g., [RH10]. Therefore, Petri nets form an excellent candidate for providing graphical support to modelling GSHP. A Petri net is a graph of places (circles) and transitions (squares), connected by arcs (arrows). The places represent modes or conditions, the transitions represent mode switches, actions or events. In order to be able to capture the qualities of GSHP, a supporting Petri net class needs to have the same powerful mathematical properties as GSHP. More specifically, we need a Petri net class for which *equivalence* can be proven. Since such property does not hold for the existing Petri net classes, this thesis develops a new class, referred to as *stochastically and dynamically coloured Petri net*. This new class contains three Petri net extensions. The first is *dynamically coloured Petri net* (DCPN), which is shown to be equivalent to PDP. The second is *stochastically and dynamically coloured Petri net* (SDCPN), which is shown to be equivalent to GSHP. The third is *stochastically and dynamically coloured Petri net with interconnection mapping types* (SDCPN$^{\text{imt}}$), which is shown to be equivalent to both SDCPN and GSHP.

These developments extend the power-hierarchy of dependability models developed by Malhotra and Trivedi [MT94] and Muppala *et al.* [MFT00], see Figure 1.1. An arrow from a model to another model indicates that the second model has more modelling power[1] than the first model. At the bottom of this power-hierarchy are fault trees and the related reliability block diagrams. Towards the top, on the left-hand-side of the power hierarchy are Petri net models, with generalised stochastic Petri nets (GSPN) in the middle, and deterministic and stochastic Petri nets (DSPN) at the top. On the right-hand-side of this power hierarchy are continuous-time Markov chains in the middle and semi-Markov processes at the top. The developments of this thesis extend this power-

[1] In [MT94], *modelling power* is determined by the kinds of dependencies within subsystems that can be modelled and the kinds of dependability measures that can be computed.

hierarchy with DCPN, SDCPN and SDCPN^{imt} on the left-hand-side and PDP and GSHP on the right-hand-side.

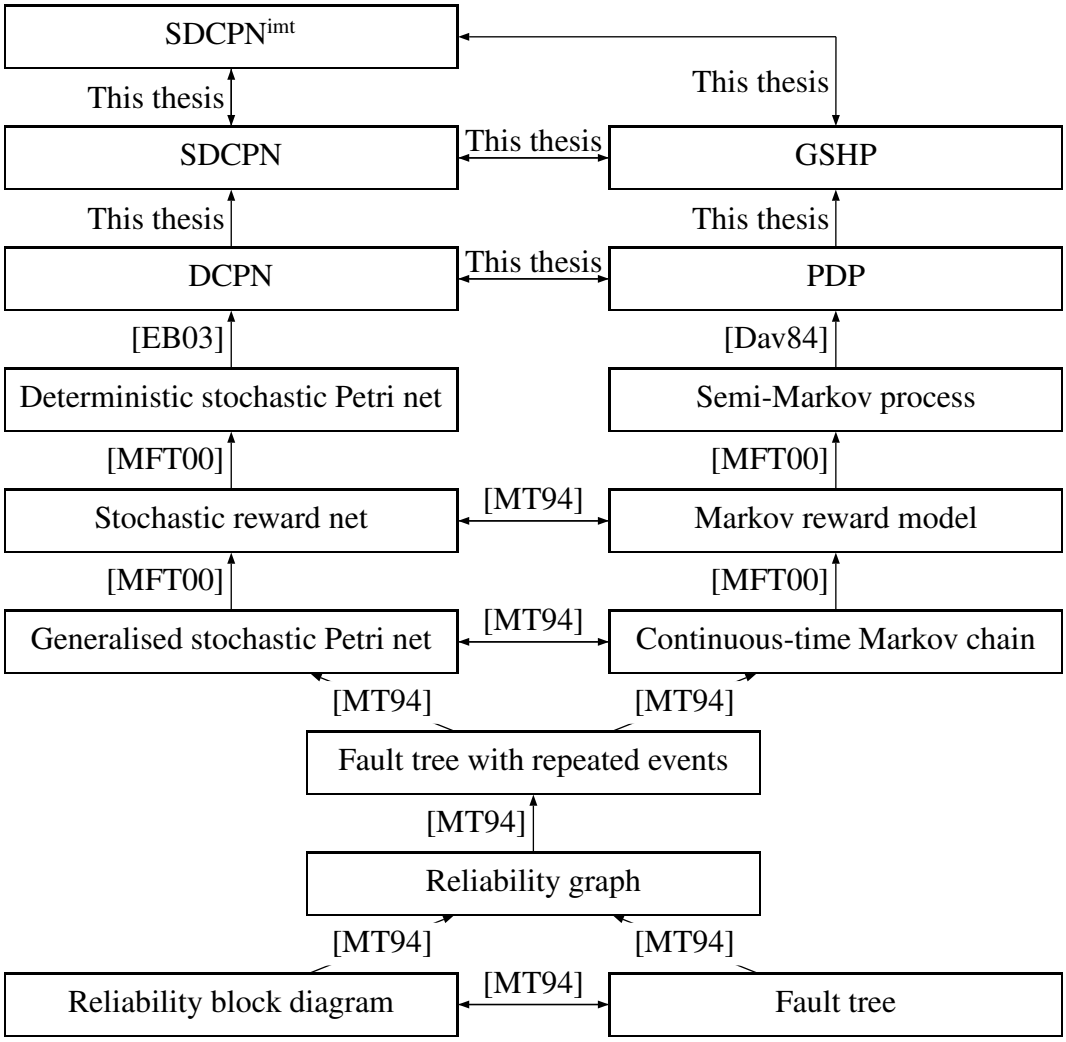


Figure 1.1 Power hierarchy among various model types. An arrow from a model to another model indicates that the second model has more modelling power than the first model.

Combining the strengths of the approaches developed

The classes of stochastic hybrid process and the classes of stochastically and dynamically coloured Petri net each have their own features and strengths. With the equivalence relations between the two types of formalisms proven in this thesis, the strengths of the two formalisms are combined. The compositional specification power of Petri nets is enhanced with the stochastic analysis power of stochastic hybrid processes and vice versa, see Figure 1.2. Due to the equivalence between SDCPN and GSHP, typical GSHP properties can be used to analyse the SDCPN, even without elaborating the particular transformation from SDCPN to GSHP for the application

considered. The complementary advantages of SDCPN and GSHP perspectives tend to even increase with the complexity of the considered operation.

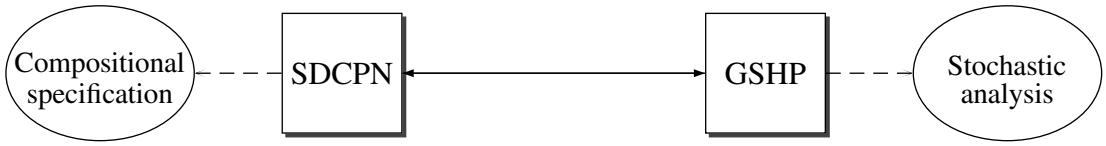


Figure 1.2 Relationship between SDCPN and GSHP, and their main capability support

Organisation of this thesis

The organisation of this thesis is as follows:

- Chapter 2 gives an overview of Petri net literature, which starts with a description of the most widely studied Petri net class, i.e., place/transition net, including analysis techniques for the evaluation of typical properties like boundedness. Subsequently, the chapter treats various extensions of Petri net classes from literature. These classes contain elements relevant for the development of DCPN, SDCPN and SDCPN$^{\text{imt}}$.

- Chapter 3 develops dynamically coloured Petri net (DCPN), and proves equivalence to piecewise deterministic Markov process (PDP) developed in [Dav93]. This chapter is based on (Everdij and Blom, 2005), [EB05].

- Chapter 4 develops stochastically and dynamically coloured Petri net (SDCPN), and proves equivalence with general stochastic hybrid process (GSHP), which is defined as solution of a *hybrid stochastic differential equation* on a hybrid state space (HSDE) developed in [Blo03, BBEP03]. In addition, it proves equivalence between SDCPN and a particular class of GSHP-related *automaton*, referred to as *general stochastic hybrid system* (GSHS), developed in [BL06]. This chapter is based on (Everdij and Blom, 2006, 2010b), [EB06, EB10b].

- Chapter 5 further increases the modelling power of SDCPN by extending the SDCPN definition to SDCPN$^{\text{imt}}$. The extension is by the inclusion of rules and notations that allow to develop a Petri net by a hierarchical approach that separates local modelling issues from compositional or interaction modelling issues, and that significantly reduces the graphical representation of the number of interconnections between local Petri nets. It is shown that the extension maintains the equivalence relations with GSHP. This chapter is based on (Everdij, Klompstra, Blom and Klein Obbink, 2006), [EKBK06].

- Chapter 6 provides several examples which apply DCPN, SDCPN and SDCPN$^{\text{imt}}$ to air transport operations, and describes tools for their analysis. The examples cover the SDCPN

formalism and their mapping to GSHP, the analysis of DCPN by making use of PDP properties, and the effectiveness of the SDCPN^{imt} approach.

- Chapter 7 draws conclusions. It explains the main result of this thesis, which is the development of three types of Petri net, DCPN, SDCPN and SDCPN^{imt}, with the analysis power of PDP and GSHP. With this, the compositional modelling power of Petri nets is combined with the analysis power of stochastic hybrid processes. This chapter is based on (Everdij and Blom, 2010a, 2010b), [EB10a, EB10b].

- Appendix A provides a brief overview of definitions and notations on stochastic processes adopted from literature.

Chapter 2

Petri nets literature

2.1 Introduction to Petri nets

A *Petri net* is a graphical and mathematical instrument to model discrete event systems. It consists of *places* (circles), *transitions* (squares), and *arcs* (arrows) that connect them. Ingoing arcs connect places with transitions, while outgoing arcs start at a transition and end at a place. If an arc is labelled with a number, it has a weight. The places may contain zero or more *tokens* (dots); the current discrete state of the Petri net (the *marking*) is given by the number of tokens in each place.

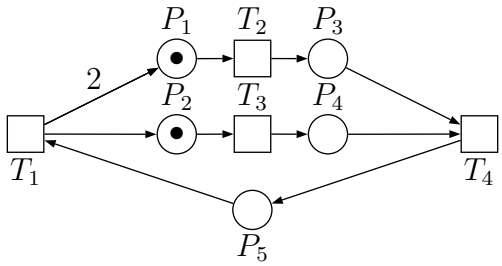


Figure 2.1 Example Petri net with five places, four transitions, and two tokens. The arc from transition T_1 to place P_1 has weight 2

Transitions may *fire*, i.e., remove tokens from their input places and produce tokens for their output places, thus modelling a (discrete) event. A transition is only allowed to fire if it is *enabled*, which is the case if there are enough tokens available in its input places (i.e., all the preconditions for the event are fulfilled). The arc weights indicate how many tokens are moved along that arc upon firing.

Petri nets were first[1] developed by Carl Adam Petri in 1962 in his dissertation [Pet62] (second edition: [Pet66]). These first nets were called *condition/event nets* (C/E nets). In this net model,

[1] Petri is reported to have originally invented them in 1939, at the age of 13, for the purpose of describing chemical processes, [RH10].

each place may contain at most one token; the place represents a Boolean condition, which is either true (there is a token) or false (there is no token), and transition events change the truth value of the conditions. Many researchers contributed to the development of new net models, basic concepts, and analysis methods, see, e.g., [CL99], [DA94], and [Mur89] for good overviews. Place/transition nets (P/T-nets), introduced around 1980, became the generally best known; they allow a place to contain several tokens. Petri nets have proven to be very useful in developing models for various practical applications. As [BSC+93] puts it, Petri nets have the following practical features for modelling:

- Graphical and equational representations, allowing comparative advantages for documentation and analytical studies.
- Natural expression of causal dependencies, conflicts, and concurrency.
- Simple, appealing and powerful synchronisation mechanism, making natural the construction of mutual exclusion constraints.
- Locality of states and actions, which allows the hierarchical and the modular construction of large net models.

The purpose of this chapter is to give an overview of Petri net literature, in order to illustrate how a variety of Petri net classes has been developed in the literature by incorporation of powerful features, to paint a picture of the origin of the Petri net types developed in this thesis, i.e., DCPN, SDCPN and SDCPN$^{\text{imt}}$, as a mixture of existing features and new ones, and to present techniques for the analysis of Petri nets that could be borrowed or adapted to the analysis of these new types. Since there is an abundance of material available, the chapter does not aim to be complete. The overview starts with P/T nets, which is the most widely studied class. Subsequently, it treats several particular Petri net classes beyond P/T nets: coloured nets (in which the tokens are distinguished by values), timed nets (in which the tokens are temporarily held at places or transitions before being fired), hybrid Petri nets (which combine discrete and continuous net elements), and classes that exploit the compositional specification of Petri nets.

Remark 2.1. *It is noted that many other Petri net classes exist beyond the ones mentioned in this chapter. Links and references to more classes, and to supporting software tools, can be found at the Petri net world website, [RH10]. Several attempts have been reported to develop a classification scheme in which all Petri net classes fit. A popular one outlines classes that can be derived from P/T nets, referred to as Restrictions, Extensions, Abbreviations, and Parametrisations of P/T nets, see [GV03] and [DA94]. A very extensive exercise to obtain a structured access to Petri nets is being undertaken by the DFG-Forschergruppe Petri Net Technology. This group developed the* Petri

Net Baukasten, *[WER+03], [Pad99], [BBD+99], which distinguishes an application developer view, an expert view, and a tool developer view. These views are related via the common base, i.e., a classification of Petri net techniques. The classification has a root that splits into twelve specialisation paths, which consider different options for the composition and behaviour of the possible Petri net elements.*

2.2 Place/transition nets

As an introduction to the Petri net formalism, this section describes the most widely studied class, i.e., place/transition net (P/T net). This covers a general definition of P/T net, and an explanation of their use in terms of properties that can be studied. For more detail and for references to supporting material, see, e.g., [CL99], [DA94], [Mur89], and [BSC+93].

2.2.1 Definitions

Definition 2.1 (P/T net graph)**.** *A* P/T net graph *is a weighted bipartite graph represented by the collection* $(\mathcal{P}, \mathcal{T}, A, w)$*, where*

- $\mathcal{P}$ *is the finite set of* places
- $\mathcal{T}$ *is the finite set of* transitions
- $\mathcal{A} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ *is the set of arcs*
- $w : \mathcal{A} \to \{1, 2, 3, \ldots\}$ *is the* weight function *on the arcs; default weight is* 1.

If the set $\mathcal{P}$ contains m places, these places are generally referred to as $P_1, \ldots, P_m$. If the set $\mathcal{T}$ contains n transitions, these transitions are generally referred to as $T_1, \ldots, T_n$. An arc from a place in $\mathcal{P}$ to transition T_j ($j \in \{1, \ldots, n\}$) is called an *incoming arc* of T_j. The set of all places with incoming arcs to transition T_j (*input places*) is denoted by $I(T_j)$. An arc from transition T_j to a place in $\mathcal{P}$ is called an *outgoing arc* of T_j. The set of all places with outgoing arcs from transition T_j (*output places*) is denoted by $O(T_j)$. If all arc weights are equal to 1, the P/T net is referred to as *ordinary Petri net*.

Definition 2.2 (Incidence matrix)**.** *For a P/T net graph with* m *places and* n *transitions, the* incidence matrix $E = [e_{ij}]$ *is an* $m \times n$ *matrix of integers given by*

$$e_{ij} = -w(P_i, T_j) + w(T_j, P_i)$$

where $w(P_i, T_j)$ is the weight of the arc from place P_i to transition T_j, $w(T_j, P_i)$ is the weight of the arc from transition T_j to place P_i, and where the weight is defined to be zero for arcs that are not in $\mathcal{A}$.

A pair comprised of a place P and a transition T is called a *self-loop* if P is both an input place and an output place of T. A P/T net is said to be *pure* if it contains no self-loop. Pure nets are completely characterised by their incidence matrix. If a net is not pure, the self-loops cannot be identified from the incidence matrix. A self-loop can be easily eliminated, e.g., by expanding the transition into a sequence: initial transition – intermediate place – final transition.

Definition 2.3 (Marking)**.** *A P/T net* marking M *defines a distribution of tokens among the places of a P/T net, i.e.:*

$$M = (M(P_1), M(P_2), \ldots, M(P_m))' \in \mathbb{N}^m$$

with m the number of places in $\mathcal{P}$, $M(P_i)$ equal to the number of tokens in place $P_i \in \mathcal{P}$, and $\mathbb{N} \triangleq \{0, 1, 2, \ldots\}$ the set of natural numbers.

Here, $(\cdot, \cdot)'$ denotes the column vector which is the transposed form of the row vector $(\cdot, \cdot)$.

Definition 2.4 (Marked P/T net, or P/T net)**.** *A* marked P/T net *is a collection $(\mathcal{P}, \mathcal{T}, \mathcal{A}, w, M_0)$, where $(\mathcal{P}, \mathcal{T}, \mathcal{A}, w)$ is a P/T net graph and M_0 is the initial marking of the P/T net.*

In other words, a marked P/T net is a P/T net graph with tokens. A marked P/T net can also be written as (N, M_0) where N is a P/T net graph $(\mathcal{P}, \mathcal{T}, A, w)$. In the literature, the word 'marked' in 'marked P/T net' is generally omitted.

Definition 2.5 (Enabled)**.** *A transition $T_j \in \mathcal{T}$ in a P/T net is* enabled *at a given marking if each input place has at least as many tokens as the weight of the arc joining it to the transition, i.e.:*

$$M(P_i) \geq w(P_i, T_j) \text{ for all } P_i \in I(T_j).$$

Definition 2.6 (Firing)**.** *A transition that is enabled can* fire*, i.e., remove and produce tokens. If T_j is enabled in marking M_{k-1}, the new marking after T_j fires is M_k where*

$$\begin{aligned} M_k(P_i) &= M_{k-1}(P_i) - w(P_i, T_j) + w(T_j, P_i),\ i = 1, \ldots, m \\ &= M_{k-1}(P_i) + e_{ij},\ i = 1, \ldots, m \end{aligned}$$

This means that when firing, a transition removes tokens from all its input places and produces tokens for all its output places. The number of tokens removed and produced is given by the weights of the arcs. An important remark concerning the firing rule of P/T nets is that enabled transitions are never forced to fire.

Definition 2.7 (State equation or fundamental equation)**.** *With M_k a column vector representing the marking at step k, E the incidence matrix, and u_k a vector noting which transition(s) fire(s) at step k, i.e., its jth component equals 1 if T_j fires and equals 0 otherwise, the* state equation *(also referred to as* fundamental equation*) is given as:*

$$M_k = M_{k-1} + E \cdot u_k$$

Note that the state equation can be used to take multiple steps directly. E.g., if $u_1, u_2, \ldots, u_k$ are vectors noting which transition(s) fire(s) at steps 1 through k, then the sum of the corresponding state equations yields:

$$M_k = M_0 + E \cdot \sum_{i=1}^{k} u_i$$

Also note that if a non-negative solution u exists for $M = M_0 + E \cdot u$, this does not imply that there exists a sequence of transitions so that M can be reached from M_0.

Definition 2.8 (Firing sequence or occurrence sequence)**.** *A sequence of firings will result in a sequence of markings. A* firing sequence *or* occurrence sequence *is denoted by $\sigma = M_0 T_{j_1} M_1 T_{j_2} M_2 \ldots T_{j_k} M_k$ or simply $\sigma = T_{j_1} T_{j_2} \ldots T_{j_k}$, if T_{j_r} fires at step r; $r = 1, \ldots, k$.*

Definition 2.9 (Reachable)**.** *A marking M is said to be* reachable *from M_0 if there exists a firing sequence σ that transforms M_0 to M. Notation: $M_0[\sigma\rangle M$.*

Definition 2.10 (Reachable set, language)**.** *Consider a P/T net (N, M_0). The* reachable set *$R(N, M_0)$ is the set of all markings reachable from M_0, i.e., $R(N, M_0) = \{M \mid M_0[\sigma\rangle M$ for some firing sequence $\sigma\}$. The* language *$L(N, M_0)$ is the set of all (finite length) firing sequences, including the zero-length (empty) sequence, i.e., $L(N, M_0) = \{\sigma \mid M_0[\sigma\rangle M$ for some reachable marking $M\}$.*

Definition 2.11 (Reachability graph)**.** *If $R(N, M_0)$, i.e., the set of all markings reachable from M_0, is finite, the* reachability graph *of the P/T net exists (is finite) and is defined by a graph with nodes equal to the elements of $R(N, M_0)$. In the graph there is an arrow between nodes M_i and M_j, labelled by transition T_k, if and only if $M_i[T_k\rangle M_j$.*

If $R(N, M_0)$ is not finite, the reachability graph would get infinitely large. Coverability graphs allow to obtain finite representations of infinite reachability graphs.

Definition 2.12 (Coverability graph)**.** *A* coverability graph *is a graph with nodes equal to a finite set of reachable markings (called the coverability set) that covers all markings of $R(N, M_0)$. Here, marking $\overline{M}$ covers marking $\underline{M}$ if $\overline{M}(P) \geq \underline{M}(P)$ for all places $P \in \mathcal{P}$. (And $\underline{M}$ is coverable if there exists a marking $\overline{M} \in R(N, M_0)$ such that $\overline{M}(P) \geq \underline{M}(P)$ for all places P.) In a coverability*

graph there is an arrow between nodes M_i and M_j, labelled by transition T_k if and only if T_k is firable from M_i and a marking covered by M_j is reached. A symbol ϖ is used in the nodes of the graph to represent 'any number of tokens' in a particular place.

Definition 2.13 (Place invariant, transition invariant)**.** *A* place invariant *or* P-invariant *is a solution to the equation $y' \cdot E = 0$, where E is the incidence matrix and y is a vector of integers. It characterises a set of places whose weighted sum of tokens remains constant at all reachable markings. A P-invariant is also defined by all integer vectors y such that for all reachable markings $M \in R(N, M_0)$, $y' \cdot M = y' \cdot M_0$ (use that $M = M_0 + E \cdot u$ and multiply from the left by y'). A linear combination of P-invariants is also a P-invariant. A* transition invariant *or* T-invariant *is a solution to the equation $E \cdot y = 0$, where E is the incidence matrix and y is a vector of non-negative integers. If each transition fires as many times as the value of the corresponding component of the vector y indicates, the original marking is restored. A linear combination of T-invariants is also a T-invariant.*

Example 2.1 (P/T net, incidence matrix, marking, state equation, firing sequence, coverability graph)**.** *Figure 2.1 on Page 9 shows a P/T net defined by $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5\}$; $\mathcal{T} = \{T_1, T_2, T_3, T_4\}$; $\mathcal{A} = \{(T_1, P_1), (T_1, T_2), (P_1, T_2), (P_2, T_3), (T_2, P_3), (T_3, P_4), (P_3, T_4), (P_4, T_4), (T_4, P_5), (P_5, T_1)\}$; $w(T_1, P_1) = 2$, and $w(A) = 1$ for all other $A \in \mathcal{A}$.*

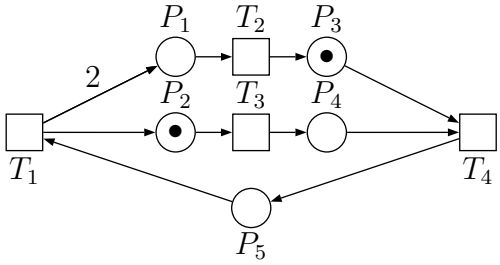
The incidence matrix corresponding to this P/T net is:

$$E = \begin{pmatrix} 2 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{pmatrix}$$

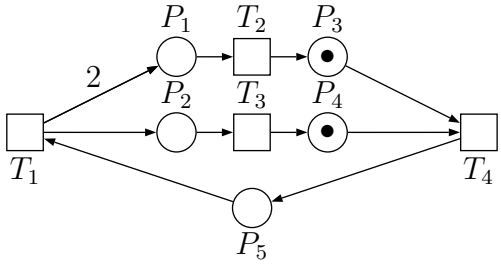
As one can see, each column in the incidence matrix corresponds with one transition, and with the marking modification if that transition is fired. For example, the second column means that if T_2 is fired, one token is removed from P_1 and one token is produced for P_3.

The current marking M_0 of the P/T net in Figure 2.1 is $(1, 1, 0, 0, 0)'$. Transitions T_2 and T_3 are both enabled, since they each have a token in their input place. The other transitions are not enabled. If transition T_2 fires (and T_3 does not), it removes its input token from P_1 and produces an output token for its output place P_3, making the new marking equal to $M_1 = (0, 1, 1, 0, 0)'$ (see Figure 2.2 (a)). In terms of the state equation, this can be represented by $M_1 = M_0 + E \cdot (0, 1, 0, 0)' = (0, 1, 1, 0, 0)'$. The firing sequence is $\sigma_1 = T_2$ (or $\sigma_1 = M_0 T_2 M_1$).

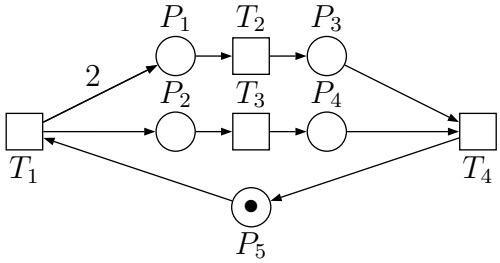
After this, only transition T_3 is enabled. If it fires, the marking is changed into $(0, 0, 1, 1, 0)'$ (Figure 2.2 (b)). Now, transition T_4 is enabled: it has two input places which both contain a token.



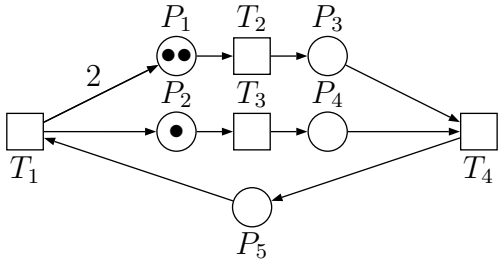
(a) Due to firing sequence $\sigma_1 = T_2$



(b) Due to firing sequence $\sigma_2 = T_2T_3$



(c) Due to firing sequence $\sigma_3 = T_2T_3T_4$



(d) Due to firing sequence $\sigma_4 = T_2T_3T_4T_1$

Figure 2.2 P/T net of Figure 2.1 in which subsequently T_2, T_3, T_4 and T_1 have fired

It removes both these tokens, and produces a token for its only output place P_5; the marking is $(0, 0, 0, 0, 1)$ (Figure 2.2 (c)). This makes transition T_1 enabled, which removes the token from P_5, produces one token for place P_2, and (since the weight of the arc from T_1 to P_1 equals 2) produces two tokens for place P_1. The new marking is $(2, 1, 0, 0, 0)'$ (Figure 2.2 (d)).

The resulting markings can also be found directly from the initial marking by using the state equation: For example, if $M_0 = (1, 1, 0, 0, 0)'$, then after all transitions have fired once, i.e., $\sum_{k=1}^{4} u_k = (1, 1, 1, 1)'$, the new marking equals:

$$M_4 = M_0 + E \cdot \sum_{k=1}^{4} u_k = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

This is the situation of Figure 2.2 (d), which is due to firing sequence $\sigma_4 = T_2T_3T_4T_1$. One may easily see that from this point onwards, the number of tokens in places P_1 and P_3 may continue to increase. This yields that the reachability graph is of infinite size. A coverability graph of the P/T net in Figure 2.1 is given in Figure 2.3.

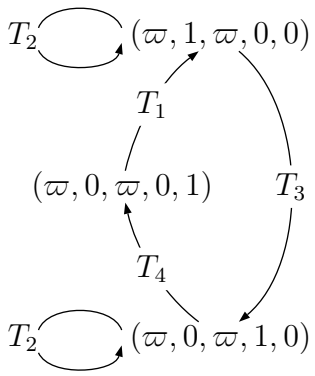


Figure 2.3 Coverability graph for the P/T net of Figure 2.1

2.2.2 Properties of P/T nets and their decidability

Once a P/T net has been constructed, one can analyse it in order to find an answer to questions like does its reachability graph exist?, or is marking M reachable? A term important in studying such Petri net properties is *decidability*, hence we explain that term first.

Decidability

A *decision problem* H is a set of questions, each of which has a yes or no answer. A solution to a decision problem H is an algorithm that determines the appropriate answer to every question $h \in H$. The term *decidability*[2], denotes whether one can determine the answer in a finite number of computational steps.

Definition 2.14 (Decidable, algorithm, effective)**.** *A yes-or-no question is* decidable *if there is an effective algorithm that is guaranteed to give an answer to the question in a finite amount of time. An* algorithm *is a finite list of well-defined instructions for accomplishing some task that, given an initial state, will terminate in a defined end-state. In [Sud97], an algorithm is called* effective *if it is:*

- *Complete: It produces an answer, either yes or no, to each question in the problem domain.*
- *Mechanistic: It consists of a finite sequence of instructions, each of which can be carried out without requiring insight, ingenuity, or guesswork.*
- *Deterministic: If presented with identical input, it always produces the same result.*

[2]Introduced by David Hilbert in 1928 at the Bologna International Congress, following up on his influential speech in 1900 at the Second International Congress of Mathematicians in Paris. [Wik10, Hilbert's problems].

A *Turing machine* is a theoretical computing machine developed by Alan Mathison Turing, [Tur36]. Following [Wik10, Turing machine] and [KD99] it consists of:

- A *tape* which is divided into cells, one next to the other. Each cell contains a symbol from a finite *tape alphabet*, which includes a special *blank* symbol. The tape represents the computer's memory and is assumed to be arbitrarily extendable to the left and to the right, i.e., the Turing machine is always supplied with as much tape (memory) as it needs for its computation.

- A *head* that can read and write symbols on the tape and move the tape left and right one (and only one) cell at a time.

- A *state register* that stores the current state of the Turing machine. The possible states are from a finite *state alphabet* and there is one special *start state*, start, with which the state register is initialized, and a *halt state*, halt, which, when current, makes the Turing machine stop its actions.

- An *action table* which is a finite number of instructions that, given the current state in the state register and given the symbol the head is reading on the tape, tells the machine to do the following in sequence: (i) write on the tape a symbol from the tape alphabet, (ii) move the head one step to the left or the right, (iii) adopt a new current state at the state register. More formally, the action table is a function $F : \Sigma \setminus \{\mathsf{halt}\} \times \Gamma \to \Gamma \times \{\mathsf{left}, \mathsf{right}\} \times \Sigma$, where Σ is the state alphabet and Γ is the tape alphabet.

Variations to this scheme have also been proposed. A Turing machine that is able to simulate any other Turing machine is called a *universal Turing machine*.

The *Church-Turing thesis*, see, e.g., [Wik10, Church-Turing thesis], first proposed by Alonzo Church in 1934 and reformalised in 1936 by Alan Turing, states that any 'calculation' that is possible can be performed by a Turing machine, provided that sufficient time and memory are available. This yields it is not possible to build a calculation device that can compute more functions than Turing machines can, and hence that all ordinary computers are equivalent to each other in terms of *theoretical* computational power (*practical* factors such as speed or memory capacity are disregarded). It is important to note that although it is widely accepted, the Church-Turing thesis cannot be mathematically proven; it is sometimes proposed as a physical law or as a definition.

A programming language[3] that is capable of emulating a universal Turing machine is called *Turing-complete* (or *Turing-equivalent* or *Turing-powerful*). Turing-completeness of a language is

[3] According to [Wik10, programming language], this is an artificial language that can be used to control the behaviour of a machine, particularly a computer.

shown by providing a mapping from Turing machines into the language[4]. *Rice's theorem*[5] [Ric53] states that *all non-trivial questions* about the behaviour or output of a Turing-complete language are *undecidable*[6]. This makes Turing machines a formal framework that can be used to construct solutions to decision problems.

Since P/T nets are *not* Turing-complete, the decidability of their properties was an open problem, and it remained an open problem for a long time. However, many researchers contributed to solving them, as will be shown below.

See [KD99] for a good overview of Turing machine issues.

Definition 2.15 (Reducible)**.** *A decision problem H is* reducible *to a decision problem H' if there is a Turing machine that takes any question $h_i \in H$ as input and produces an associated question $h'_i \in H'$ where the answer to h_i can be obtained from the answer to h'_i.*

Definition 2.16 (Equivalent)**.** *A decision problem H is* equivalent *to a decision problem H' if H is reducible to H' and vice versa.*

Properties of P/T nets

There is much literature available on properties of P/T nets, and their associated decidability. The remainder of this subsection briefly describes the properties most relevant to this thesis. References used are [Mur89], [EN94], and [Esp98], which also provide details on other properties, such as promptness, persistence, controllability, marking equivalence, and non-termination.

Boundedness A P/T net (N, M_0) is *bounded* if its set of reachable markings $R(N, M_0)$ is finite. In a bounded P/T net, each place can only have a finite number of tokens. A P/T net (N, M_0) is *k-bounded* if no reachable marking puts more than k tokens in any place, i.e., $M(P) \leq k$ for every place P and every marking $M \in R(N, M_0)$. A P/T net is *safe* if it is 1-bounded. A P/T net N is *structurally bounded* if it is bounded for any finite initial marking M_0.

Boundedness is decidable [KM69]. There are several ways to decide boundedness, e.g., with coverability graph (however, this is not the most efficient way [Mur89]):

1. A net (N, M_0) is bounded iff ϖ does not appear in any coverability graph node.

2. A net (N, M_0) is safe iff only 0's and 1's appear in coverability graph nodes.

[4]For an example of such mapping, see [Koo05, Section 4.6.2] or [She05, Page 161]

[5]After Henry Gordon Rice. See also [Wik10, Rice's theorem] for a proof.

[6]Formulated in another way: According to Rice's theorem, if C is a particular class of computable functions, and there exist f_1 and f_2 such that $f_1 \in C$ and $f_2 \notin C$, then the problem of deciding whether a particular programme computes a function in C is undecidable.

3. A net (N, M_0) is structurally bounded iff the system of linear inequations $y' \cdot E \leq 0$, with E the incidence matrix, has a positive solution, [EN94].

Algorithms to decide boundedness still require a lot of computational space, e.g., Lipton [Lip76] proved that deciding boundedness for P/T nets requires at least space $2^{c\sqrt{n}}$, where c is some constant and n is the size of the P/T net N. Rackoff [Rac78] proved that an upperbound for the space required is $2^{cn \log n}$. Here, the size of a P/T net is defined by Esparza [Esp98] as $n = O(|\mathcal{P}| \cdot |\mathcal{T}|)$, where $|\mathcal{P}|$ is the number of places and $|\mathcal{T}|$ is the number of transitions.

Conservativeness is a special case of structural boundedness. If $y = (y_1, \ldots, y_m)'$ is a vector, with y_i corresponding to a positive integer weight for place P_i, then a P/T net is said to be *conservative* with respect to y if $y' \cdot M =$ constant. A *strictly conservative* P/T net is conservative with respect to the weighting vector $(1, \ldots, 1)'$. A weighting vector for which the net is conservative is found by solving $y' \cdot E = 0$, with y positive.

Reachability A marking M is *reachable* if there exists a firing sequence σ that brings the initial marking M_0 to M, i.e., if $M_0[\sigma\rangle M$. The *reachability problem* of a P/T net is whether a given marking M is reachable from the initial marking M_0, i.e., whether $M \in R(N, M_0)$. Hack [Hac75] and Keller [Kel75] observed that many other problems are equivalent to the reachability problem, hence reachability became a central issue.

Reachability is decidable, [May81], [May84], [MM81], [Kos82]. If the P/T net is bounded, its reachability graph exists and a marking M is reachable iff there exists a node labelled M in the reachability graph. If the P/T net is not bounded, then one can use the coverability graph to find a sufficient condition for reachability [Mur89]: If a marking M is reachable from M_0 then there exists a node labelled $\overline{M}$ such that $M \leq \overline{M}$. However, because of the information lost by the use of the symbol ϖ, in general, the reachability problem cannot be solved by using the coverability graph alone. Murata [Mur89] gives a necessary condition for reachability, and a sufficient condition for non-reachability, both based on the incidence matrix.

Liveness A P/T net (N, M_0) is *live* if every transition can always occur again. There are different levels of liveness for a transition T:

Definition 2.17 (Liveness)**.**

- *T is $L0$-live (*dead*) if T can never be fired in any firing sequence in $L(N, M_0)$.*
- *T is $L1$-live (potentially firable) if there is a firing sequence in $L(N, M_0)$ in which T can be fired at least once.*
- *T is $L2$-live if, given any positive integer k, there is a firing sequence in $L(N, M_0)$ in which T can be fired at least k times.*

- *T is $L3$-live if there is a firing sequence in $L(N, M_0)$ in which T appears infinitely often.*
- *T is $L4$-live (*live*) if T is $L1$-live in $L(N, M)$ for every marking $M \in R(N, M_0)$.*

A P/T net is said to be Lk live if every transition in the net is Lk-live, $k = 0, 1, 2, 3, 4$. Murata [Mur89] notes that $L4$-liveness implies $L3$-liveness, $L3$-liveness implies $L2$-liveness, and $L2$-liveness implies $L1$-liveness. A P/T net is called *deadlock-free* if from any reachable marking at least one transition can always occur.

The liveness problem is recursively equivalent with the reachability problem [Hac75], [AK77] and thus decidable.

Local properties Local properties of a system or operation can be modelled with a P/T net by using only a few places or transitions, isolated from the rest of the P/T net. Below, an overview is given of some of these properties, with a graphical illustration in Figure 2.4.

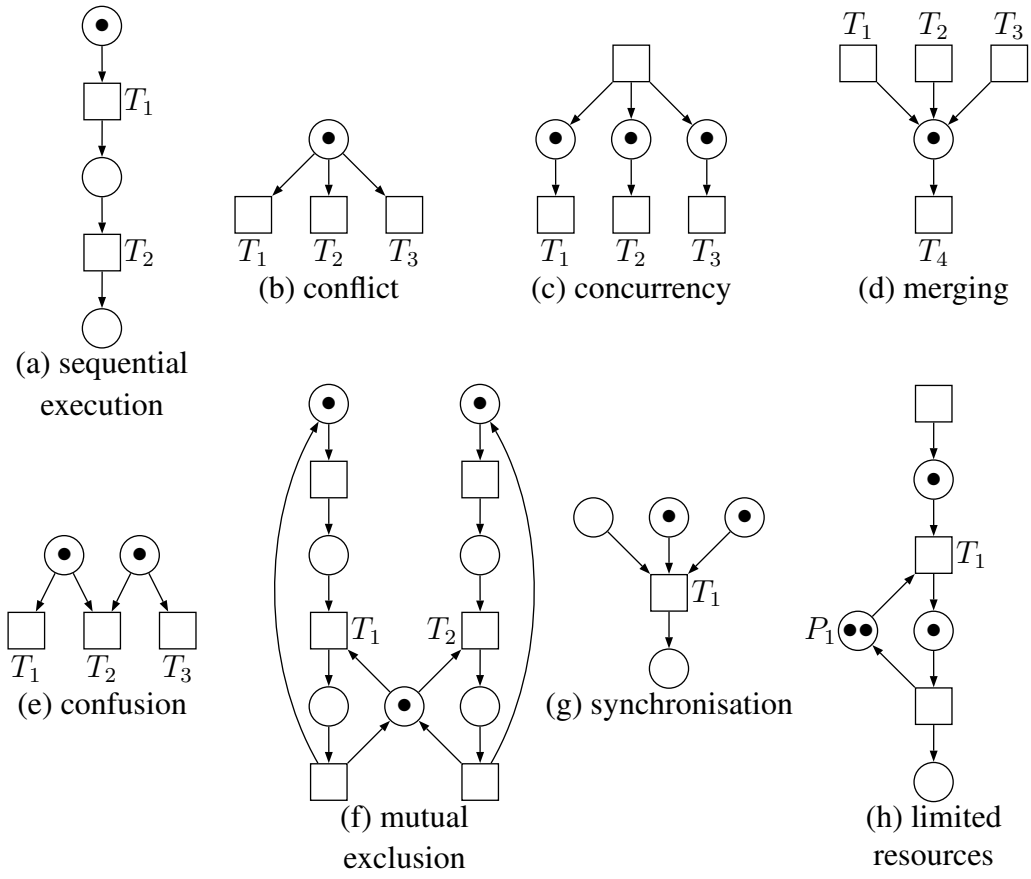


Figure 2.4 Local P/T properties, from [VN92]

Sequential execution. In *sequential execution*, an event can only take place after the occurrence of a specified other event. This can be modelled as in Figure 2.4 (a), where transition T_2 can fire

only after the firing of transition T_1. Also, this P/T net shows the causal relationship among activities.

Conflict. A *conflict* between events occurs for example if only one of these events can occur at a time and a choice has to be made. This can be modelled as in Figure 2.4 (b), where transitions T_1, T_2 and T_3 are in conflict. All are enabled but the firing of any leads to the disabling of the other transitions.

Concurrency or parallellism. Besides events occurring sequentially, also events occurring *concurrently* (in parallel) may exist. This can be modelled as in Figure 2.4 (c), where transitions T_1, T_2 and T_3 are concurrent (are enabled at the same time). A necessary condition for transitions to be concurrent is the existence of a forking transition that deposits a token in two or more output places.

Merging. If parts from several streams arrive for service at the same machine, these streams have to *merge*. The resulting situation can be depicted as in Figure 2.4 (d).

Confusion. *Confusion* is a situation where concurrency and conflicts both exist, as in Figure 2.4 (e).

Mutual exclusion. In Figure 2.4 (f), the firing of transition T_1 prevents the firing of transition T_2 and vice versa.

Synchronisation. Sometimes parts in a system have to wait for other appropriate parts or for information to arrive. The *synchronization* of activities can be captured by a transition of the type shown in Figure 2.4 (g). Transition T_1 will be enabled only when a token arrives into the input place currently without token.

Limited resources. Situations of limited resources can be modelled as in Figure 2.4 (h), where transition T_1 can only fire if there are resources available in place P_1.

Decidability of several of these local properties has been studied in, e.g., [Frö04].

2.3 Coloured Petri nets

The remainder of this chapter gives an overview of relevant Petri net classes (other than P/T nets) found in the literature. The aim is to paint a picture of the origin of the Petri net class developed in this thesis, i.e., SDCPN$^{\text{imt}}$, as a mixture of existing features and new ones, and to identify techniques for the analysis of Petri nets that could be borrowed or adapted to the analysis of SDCPN$^{\text{imt}}$. The main focus here is on classes that can be considered strict extensions of P/T nets, since this thesis

focuses on Petri nets powerful enough to model complex air transport operations. The current section makes one exception for a class equivalent to P/T net, i.e., coloured Petri net, since many extensions were derived from it.

Although, since their introduction, P/T nets were used and studied widely, in many occasions it turned out that they were too low-level to manage more complex practical applications. Therefore, different researchers started to develop their own Petri net classes. Most of these early developments were designed for specific applications, and most analysis methods useful for one Petri net class could not be used for another class. This triggered the development of *Predicate/transition net* (PrT net) [GL81], [Gen86], which were constructed without any particular application in mind. They can be related to P/T nets in a formal way hence allow generalisation of the basic concepts and analysis methods. To overcome a few remaining technical problems in the generalisation of analysis methods of place invariants and transition invariants, coloured Petri nets (CP81-nets) were developed around 1980 by Kurt Jensen in his PhD work and first published in [Jen81]. The main idea was directly inspired by PrT nets. Later, the advantages of PrT nets and CP81-nets were combined, and the result is nowadays known as *coloured Petri net* (CP87-net or CPN).

The main feature of CPN is that tokens are no longer the indistinguishable black dots like they are in P/T nets, but are distinguished by a colour or assigned value. The transitions and arcs observe these colours and consider them in their firing. A primary advantage is that this may significantly reduce the size of the Petri net, since multiple subgraphs that are of equal or similar structure can now be folded into one subgraph containing multiple coloured tokens, where each colour refers to an original subgraph.

The class of coloured Petri net is explained below:

Coloured Petri net (CPN). The tokens are coloured, i.e., they have a value that is an element of a particular place-dependent *colour type*. The arcs are labelled by *arc expressions*, which are similar to the arc weights of P/T nets, but are extended to the use of (weighted) colours[7]. A transition is enabled if there are enough tokens (both in number and in colour) in its input places to satisfy the arc expressions, and in addition these tokens satisfy a transition-dependent Boolean *guard*. An enabled transition removes the input tokens that satisfy these criteria, and produces output tokens that have colours according to the arc expressions on its output arcs. The formal definition of CPN, see [Jen90], makes effective use of *multisets*, which are sets in which the elements are distinguishable[8].

Remark 2.2. *Note that different variants of coloured Petri net are presented in literature, which are still referred to as CPN. For example, [Zen85] and [YLB95] do not use the transition guard*

[7] An example arc expression would be: 'two tokens of colour a and 3 tokens of colour b'.

[8] In this formalism, for example, an arc expression 'two tokens of colour a and 3 tokens of colour b' is denoted by $2a + 3b$.

(although [YLB95] does add it later as an additional feature of extended coloured Petri net), and [Haa02] only allows transition enablings in one colour, rather than a binding of multiple colours.

2.4 Timed Petri nets

The concept of time was intentionally avoided in the original work of C.A. Petri, because of the effect that timing may have on the behaviour of nets. For example, time constraints may prevent certain transitions from firing, so that the behaviour of the net is not anymore defined by its structure alone. However, there are also situations to be modelled in which time plays an important role. A *timed Petri net* allows an operation to be described whose functioning is time dependent. This would allow to measure additional properties such as durations of states or activities. The pioneering works in the area of timed Petri nets were performed by Merlin and Farber [MF76], and by Noe and Nutt [NN73]. Timing can be specified in several ways:

Deterministic. The associated time durations are predicted exactly. This has been investigated by, e.g., [Ram73], [Sif77] and [RH80]. The analysis of deterministically timed Petri nets is however tractable only in the case of special classes such as marked graphs.

Stochastic. Time durations are associated with a random variable. This concept was first investigated independently from one another by Natkin [Nat80] and Molloy [Mol81] and this was the origin for the emergence of stochastic Petri nets and their extensions as a principal performance modelling tool.

Known to a lesser extent are two variants: *Non-deterministic*, studied by, e.g., [AHR00], which assumes constraints on the time delays (e.g., 'it takes less than 15 minutes to perform this action'), usually by means of an interval, but no further assumptions. And *possibilistic*, studied by, e.g., [KL00], which exploits fuzzy logic to represent imprecise durations.

Time can be associated with transitions, places, tokens, arcs, or combinations:

Transition timed Petri nets. There are two possibilities associating time to transitions: *delay time* (time that must occur between enabling and firing) or *firing time* (time associated to the firing). A token may be reserved for the delay or firing of a transition or it can be non-reserved. Used in, e.g., [Ram73].

Place timed Petri nets. Once a token has been added to a place, it will not contribute to enabling any transition before the waiting time associated with that place has elapsed. Introduced in [CR83].

Token timed Petri nets. The enabling of a transition depends on the *time stamps* of the tokens. Such time stamp may be interpreted as the age of a particular token, i.e., how much time elapsed since it was produced. Used in, e.g., [FM94].

Arc timed Petri nets. Delays are assigned to arcs. The delay is interpreted as a period of time that must elapse until a token will arrive from a place to a transition or vice versa. Used in, e.g., [Han93].

Usually, in contrast with most non-timed Petri nets, functioning at maximal speed is considered. This means that a transition is fired as soon as it is enabled, except possibly if this transition is in conflict with another, [DA94]. For an overview of issues related to timed Petri nets, see, e.g., [AHR00].

Notable examples of specific timed Petri net classes are:

Stochastic Petri nets (SPN). First studied by [Mol81], [FN84] and [ABB$^+$85]. The transition firing durations are associated with random variables. The reachability graph of an SPN is identical to the one of the underlying P/T net, hence all results available for the structural analysis of P/T nets can be applied to SPN. If the stochastic durations follow exponential distributions, see, e.g., [Ajm89], the marking of the stochastic Petri net is a homogeneous *Markovian process*, and the reachability graph of the SPN is equivalent to a homogeneous (continuous-time) *Markov chain*.

Generalised stochastic Petri nets (GSPN). This is a widely studied class, see, e.g., [ACB84], [ABC$^+$91], [ABD98], [Bal01] and [Haa02]. Each transition is either timed (firing with a particular exponentially distributed delay) or immediate (firing without delay), and each is assigned a priority level, where timed transitions have the lowest priority level. Weights of immediate transitions determine who will fire in case two or more transitions with the same priority level are simultaneously enabled. Due to this structure, the probability that any two transitions fire at the same time is zero. Another feature of GSPN is that some of its arcs are *inhibitor arcs*[9], i.e., the transition connected to such arc can only be enabled if the input place connected to the arc does *not* have a token. The reachability set of a GSPN is identical to the one of the underlying P/T net with inhibitor arcs and priorities. Therefore some of the structural properties valid for the basic underlying P/T net, such as place invariants, are retained by the GSPN. Usually, the GSPN reachability graph distinguishes the *tangible* markings (in which only timed transitions are enabled) from the *vanishing* markings (in which an immediate transition is enabled).

Decidability of GSPN properties has also been studied. However, the use of the inhibitor arcs make GSPN to be Turing-complete (see Page 17), see [Age74]. It was proven in [Cia87] that

[9] Introduced by [AF73] and sometimes referred to as *test arcs* or *zero test arcs*.

GSPN are also Turing-complete if the set of inhibitors is empty. It was proven in [ACB84] that GSPNs are *equivalent to continuous-time Markov chains* (see Figure 1.1), and this allows studying their properties despite the Turing-completeness.

Deterministic and stochastic Petri nets (DSPN). Developed by [AC87]. DSPN are an extension of GSPN (see Figure 1.1) that allows firing delays of timed transitions to be either constant or exponentially distributed random variables. Under the condition that in each reachable marking only one deterministic transition is enabled, analysis of DSPN can be by means of its embedded Markov chain, see [BSC+93] and [CL93].

Coloured stochastic Petri nets (CSPN). In [Zen85], *coloured stochastic Petri net* are defined as a class that uses elements from both coloured Petri net and stochastic Petri net, see also [Haa02]. The transition firing rate may be dependent on the colour fired and on the current marking.

2.5 Hybrid Petri nets

In a *hybrid Petri net*, continuous and discrete aspects are combined in an integrated way. The discrete aspect is usually similar to the 'usual' Petri net types; the continuous part can generally be traced back to one of the two following base forms:

Fluid tokens. Tokens are not discrete 'bullets' but are more like fluids residing in the places: a place can contain a real-valued, non-negative, amount of token.

Coloured tokens. The tokens have a value (or colour) that is from a 'continuous set', e.g., is a vector of real numbers.

The discrete and continuous aspects can be mixed or combined in many different ways. Reference [Giu06] maintains a collected list of references in the field of hybrid Petri nets, grouped on the basis of the models used. Reference [AKZ98] gives a brief overview of hybrid control systems, including hybrid Petri net classes.

Notable examples of specific hybrid Petri net classes based on *fluid tokens* are:

Continuous Petri nets (ContPN). Proposed by [DA87]. The marking of a place is a non-negative real and firing is carried out like a continuous flow. A continuous Petri net may either be autonomous (no time involved) or with firing speeds associated with the transitions. In the latter case, a transition can be *strongly enabled* (i.e., its input places are not empty and it can fire at maximum speed) or *weakly enabled* (i.e., the input places that are empty are currently being fed by other transitions). Two main variations are *constant speed continuous Petri*

net, [Dav97], in which weakly enabled transitions cannot fire until they are strongly enabled, and *variable speed continuous Petri net*, in which a weakly enabled transition is fired at the lower speed of the other transition that is feeding the input place. Decidability questions for continuous Petri nets have been studied in, e.g., [SR02].

Hybrid Petri nets (HPN). Proposed by [LAD91] as an extension of continuous Petri net. HPN have discrete places that contain an integer number of tokens and continuous places that may contain a real amount of tokens. The arcs have positive weights. A state equation for the marking of the net can be determined, which uses the number of times each discrete transition fires and the instantaneous firing speeds associated with continuous transitions. In *hybrid timed Petri net* as defined in [TTV06], discrete transitions, when enabled, fire after a (deterministically determined) delay; several analysis problems for hybrid timed Petri nets, like P-invariants, are studied.

Fluid stochastic Petri nets (FSPN). Proposed in [TK93]. FSPN move fluid tokens between continuous places and discrete tokens between discrete places. There are immediate transitions and timed transitions. The enabling of either type is controlled only by tokens in the *discrete* input places, and the firing of tokens from and to discrete places is as for ordinary Petri nets. An enabled timed transition removes fluid tokens from its continuous input places after an exponential delay time, and at a rate which is dependent on the connecting arcs and on the current marking of *all* discrete places in the net. A partial differential equation can be determined which specifies the change of fluid tokens on all continuous places, see also [GSB99]. The reachability graph associated with the discrete Petri net part is equivalent to a continuous-time Markov chain.

Differential Petri nets. Described in [DK96] and [DK98]. These have discrete places (with a non-negative integer marking), differential places (with a real-valued marking), discrete transitions and differential transitions. The marking of a differential place can also be negative, which allows 'negative amounts' of fluid token in a place. Weights of arcs connected to differential places are real numbers which may also be negative. A discrete transition is enabled if each input place has a non-reserved marking satisfying the input arc weights; it then reserves the enabling input tokens and fires after a transition-dependent constant delay. A differential transition is enabled if each *discrete* input place has a marking that satisfies the input arc weights; its firing yields a change of marking in the *differential* places equal to the speed of the transition, times the weight of the corresponding arc. This speed may be a constant, a linear combination, or a non-linear function of the markings connected to the transition, and may also be negative. Effectively, this scheme can represent any form of discrete approximation of an ordinary differential equation. Reference [DK98]

also discusses the evolution graph and some properties of differential Petri nets, like liveness.

Notable examples of specific hybrid Petri net classes based on *coloured tokens* are:

Extended coloured Petri nets (ECPN). Introduced in [YLB95]. The token colours are real-valued vectors following difference equations. The token colour is updated in an external loop around its residence place by an additional updating transition.

High-level hybrid Petri nets (HLHPN). Introduced in [GU96], [GU98]. An HLHPN combines hybrid Petri net with coloured Petri net. Discrete places have a marking that is a subset of the natural numbers, continuous places have a marking that is a real-valued vector. A discrete transition is enabled if the tokens in the input discrete places satisfy the input arc expressions and if each input continuous place contains a token of a particular value. It then fires after it has remained enabled during a transition-specific time delay. A continuous transition is enabled if the tokens in its input discrete places satisfy the arc expressions; the marking of continuous places does not affect its enabling. An enabled continuous transition fires continuously with a particular velocity, and the marking of its output continuous places is changed according to a differential equation which may be dependent on the current marking and on an external continuous input.

Hybrid high-level Petri nets (HyNets). Introduced in [Wie96a] and [Wie96b]. HyNets are an integration of coloured Petri nets, differential algebraic equations and object-oriented concepts. There is only one class of places, but there are discrete transitions and continuous transitions, continuous undirected arcs and discrete directed arcs. The set of discrete arcs is divided into ordinary arcs, enabling arcs (tokens should be present in input place of transition, but are not removed when transition fires) and inhibitor arcs. Places are labelled by a place type (Boolean, real, user defined, etc.) and a capacity (a positive integer or ∞). A discrete transition that has input tokens that satisfy the arc expressions, fires after a random delay, provided the output places have enough free capacity. During this delay, the input tokens are not reserved and may be consumed by other discrete transitions. A continuous transition fires without delay and continuously changes bound token colours according to its firing action. The firing action may be an algebraic equation (assigning a value to a token) or a differential equation (which changes values). The tokens stay in their places during firing, until the activation condition is no longer fulfilled or a discrete transition steels them away. Rules are available that decide how to proceed in case of conflicts between enablings or firings.

Particle Petri net. Introduced by [LT05]. Patricle Petri nets are composed of a numerical part and a symbolic part. The numerical part is similar to differential Petri net: token colours are solutions to differential equations associated with places. The symbolic part is a *possibilistic*

Petri net, in which a token in a place is associated with a 'possibility value', say, μ, denoting the 'possibility' that the token is really in this place. If this place is input to a transition that has a possibility of firing equal to λ, then after pseudo-firing of this transition, the output place of the transition has a token with possibility $\lambda\mu$.

2.6 Compositional specification

The development and verification of a complete model for very complex operations is generally a difficult task. To tackle the development problem it can be split up into sub-problems, after which the complete model is composed of the submodels. The different submodels can be verified separately, after which the results are translated to the complete model. The task of how to decompose the development problem in order to handle its complexity can be referred to as *compositional specification*; in [VJMV04], it is referred to as *modularity*. Generally, it can be addressed in two ways, although combinations are also possible:

Vertically. The operation is first modelled at a high level, after which the details are addressed by 'zooming in'; this is sometimes referred to as *unfolding*. The other way around (zooming out, *folding*) is also possible. This hierarchical modelling is generally seen as a set of representations of a complex system made at different levels of detail.

Horizontally. The operation is divided into parts, which are modelled individually, and next coupled at the same level. Horizontal decomposition is generally used if an operation consists of multiple 'agents' that each act individually but interact at certain instances.

One of the issues studied in this area is the conservation of properties, such as boundedness, by composition of Petri net models.

Notable examples of specific Petri net classes that address the compositionality issue of Petri nets are:

Hierarchical coloured Petri nets (HCPN). Described in [HJS90] and [Jen90]. An HCPN consists of a finite set of *pages*, where each page is a coloured Petri net (CPN). These individual CPNs are related in different ways, known as the hierarchy constructs, see, e.g., [NH04]:

Substitution node. This is a place or a transition that is related to a more complex CPN, called subpage, which gives a more precise and detailed description of the activity represented by the substitution node. Input socket nodes and output socket nodes in the substitution node communicate with port nodes in the subpage.

Invocation node. In contrast to substitution nodes, the invocation nodes are not substituted by their subpage, but their occurrence triggers the creation of a new instance of the subpage. These subpage instances are executed concurrently with the other page instances in the model, until a specified exit condition is reached. The invocation hierarchy is allowed to contain circular (i.e., recursive) dependencies.

Fusion set. Fusion allows to conceptually fold a set of nodes into a single node without graphically having to represent them as a single object.

Synchronous interpreted Petri net (SIPN). Proposed in [FAP97]. An SIPN is a C/E net (see Page 9) with guarded transitions and synchronous transitions firing, and with inhibitor arcs and enabling arcs. It combines horizontal and vertical hierarchy constructs. The vertical hierarchy is modelled by *macroplaces* or *macrotransitions*, which can be exploded (or imploded) to form (or hide) a complete sub-Petri net. The horizontal hierarchy is by *enabling arcs* which model synchronisation and by *inhibitor arcs* which model priority.

Modular multilabelled nets (M-nets). Developed in [BFF+95]. M-nets are PrT nets (see Page 22) that allow vertical unfolding of places and transitions, as well as horizontal composition of large Petri nets from smaller components. Unfolding associates a more elementary M-net to every M-net, as well as a marking of the former to every marking of the latter. Horizontal operations, such as copying of places, multiplying sets of places, adding and removing places, unions of nets, synchronisation, and restriction, are defined and are shown to be consistent with the vertical unfoldings in the sense that the unfolding of a composite net is equal to the composition of the unfoldings of its components.

In [FG97], the definition of M-nets is extended with non-singleton type (to handle identifier sets), transition refinement, and a relabelling function, to cover issues like recursion, global variables, and different types of parameter passing. The extended M-net is referred to as $B(PN)^2$ (*basic Petri net programming notation*).

Petri net components. A *Petri net component*, as introduced by [Kin97], is a Petri net equipped with *input places* and *output places*, which represent the interface of the component to other components. Each component is surrounded by a box, the input places and output places are located at the border of the surrounding box and are labelled by an arrowhead. Input places have an arrowhead that points into the box, the environment may put tokens in them, and the component itself may remove tokens from them. Output places have an arrowhead pointing in the opposite direction, the environment may remove tokens from them and the component may put tokens in them.

Higher-level Petri nets as developed in [JE02], also use components, where a component has

parameters, and can be instantiated and connected to the environment through ports. *Input ports* may be connected to places and *output ports* to transitions. To be able to connect components, places in a Petri net are also granted ports and they become *container places*. If a transition produces a token, that token is not put onto the container place itself but rather on its input port.

Modular construction. Reference [FKK97] constructs an overall Petri net from Petri net *modules* by using several module coupling mechanisms. The base is generalised stochastic Petri net (GSPN) with enabling arcs. The model is built in an incremental manner. First, one component is described by one Petri net module. At each following step, a new component is added and the Petri net model is updated by taking into account assumptions, interactions and influences of the new component on the already integrated components. Guidelines for the modular construction are:

Modules. Each module describes the behaviour of one component. It consists of internal places and transitions coupled by arcs. Each module contains only one token, and in order to maintain this basic rule, each internal transition has a single internal input place. In addition, inhibitor arcs or enabling arcs are not used within a module, module folding is recommended, and immediate internal transitions are avoided.

Module coupling mechanisms. The interactions between components are implemented by three basic module coupling mechanisms: *marking tests* (couplings of modules by inhibitor arcs or enabling arcs, which do not change the marking of the places involved in the test), *common transitions* (which are shared by several modules and which produce a token back into a module if they have removed one), and *interconnection blocks* (which are built of one in-place and a set of (immediate) out-transitions, and which connect one or several initialising modules to one or several target modules).

Block decomposition. Blocks are decomposed into more elementary blocks in order to ensure reusability.

2.7 Concluding remarks

This chapter gave an overview of Petri net literature, to illustrate how a variety of Petri net classes has been developed in the literature by incorporation of powerful features, to paint a picture of the origin of the Petri net classes developed in this thesis, as a mixture of existing features and new ones, and to present techniques that could be borrowed or adapted to their analysis.

The class of P/T nets appears to be the most widely studied class, and a lot of research results are available on their properties and their decidability. Several of these properties, like reachability and

boundedness, will be particularly relevant when modelling large-scale stochastic hybrid systems, since they are related to aspects of manageability of the models.

The Petri net classes beyond P/T nets provide several features to increase the modelling power of Petri nets:

- Coloured nets (in which the tokens are distinguished by values) are a mechanism to reduce the number of places in a Petri net graph; similar subgraphs can be folded into one graph by distinguishing the tokens by colours.

- Timed nets (in which the tokens are temporarily held at places or transitions before being fired) are a very useful extension when modelling systems in which the time element is important. Of particular interest are models equivalent to continuous-time Markov chains, since for these, many analysis instruments are available. An advantage of using Petri nets for modelling rather than Markov chains directly, is that Petri nets allow the use of multiple tokens and allow transitions to have multiple input and output places, so that a large state space may be modelled with only a few places.

- Hybrid Petri nets (which combine discrete and continuous net elements) significantly increase the modelling power towards systems which combine both discrete and continuous elements. Hybrid Petri nets show great potential as modelling formalism for air transport operations, since these operations typically combine discrete elements (e.g., aircraft modes of operation) and continuous process elements (e.g., aircraft position and velocity). In addition, as is noted by [DFGS07], the existence of continuous parameters allows to speed up optimisation and to perform sensitivity analysis.

- And finally, classes that exploit the compositional specification of Petri nets provide features that start modelling locally and next compose large scale nets from local nets. For the modelling of complex air transport operations, the use of compositional specification constructs is essential. These allow to break up the model development task into subtasks, thus increasing manageability. It may even be possible to have different submodels constructed (and verified) by different model developers, if a variety of expertise is called for.

The classes of stochastically and dynamically coloured Petri net (DCPN, SDCPN and SDCPN$^{\text{imt}}$), developed in Chapters 3, 4 and 5, combine these features in such a way that the modelling power is as rich as possible, with the additional feature that equivalence relations with classes of stochastic hybrid process (i.e., PDP and GSHP) are proven to exist. In fact, during the early development stages of DCPN, a literature search was made to see if an existing hybrid Petri net class could be found that satisfied the selection criteria identified. Unfortunately, although several hybrid Petri net

classes could be linked to continuous-time Markov processes, the link with PDP was not proven, and DCPN was developed as a new addition to the class of hybrid Petri net.

Remark 2.3. *It is noted that many of the Petri net classes described in this chapter have also been used in the air transport operations applications area on their own. For example,*

- *In [Obe06] and [OÜRS07], coloured Petri nets (CPN) are used to model and analyse the planning process for airplane arrivals in air traffic control.*
- *Kanoun and others, e.g., [KO00], [FKK98], use generalised stochastic Petri net (GSPN) for evaluation of service degradation of CAUTRA (coordination automatique du trafic aérien), an automated computing system for air traffic control in France. Emphasis is on interaction between hardware and software components and on modular construction.*
- *Lesire and Tessier [LT05] use particle Petri nets in a simulation of the Toulouse airport approach procedure.*
- *Villani and others [VJMV04] use* differential predicate transition net *for the modelling of an aircraft landing system.*

Chapter 3

Dynamically coloured Petri nets

3.1 Introduction

A *piecewise deterministic Markov process* (PDP)[1], as developed by Mark Davis in [Dav84] and [Dav93], is a stochastic hybrid process that, most of the time, follows the solution of an *ordinary differential equation*; this solution is referred to as the *flow*. At some times, however, the process may jump. Such jumps may be *spontaneous*, i.e., occurring at a random time, or *forced*, i.e., occurring when the flow hits the boundary of a predefined area. The value of the PDP right after the jump is determined by a particular PDP transition measure, and from this new value onwards, the process again follows the solution of an ordinary differential equation that may be different from the previous one. Between jumps, the PDP is deterministic, which explains the 'piecewise deterministic' part of the name PDP, and at all times, the process value in the future is not dependent on the history but only on the present state, which explains the 'Markov' part. PDP is also a hybrid process, since its state space is a combination of continuous values, i.e., the flow, and discrete ones, i.e., the discrete set of differential equations to choose from. PDP is (according to [Dav84]) the most general class of continuous-time Markov processes that include both discrete and continuous processes (except diffusion). It has a unique solution and strong Markov properties. In [Dav93] the stochastic analysis power is explained by the derivation of the PDP extended generator, the strong Markov property and the 'right process' property.

A *dynamically coloured Petri net* (DCPN) is a stochastic hybrid Petri net in which each token has a value (a *colour*) that is a Euclidean vector. More precisely, the value of a token is an element of $\mathbb{R}^n$, where $\mathbb{R}$ is the set of real numbers, and n is a natural number that is determined by the place in which the token resides. The value (colour) of a token may change through time (*dynamically*) while the token is residing in its place, which explains the 'dynamically coloured' part of the name. This value follows the solution of a place-dependent *ordinary differential equation*. There are three

[1] We adopt the abbreviation used by Davis, i.e., PDP rather than PDMP.

types of arcs: *ordinary arcs* (which are as for other Petri net classes), *inhibitor arcs* (a transition can only be enabled if the input place connected to the inhibitor arc does not contain a token) and *enabling arcs* (tokens are not removed from the connected input place if the transition fires). There are three types of transitions: *Immediate transitions* may fire as soon as they have a sufficient number of tokens in their input places; for *delay transitions*, the firing can only take place after a transition-dependent random delay has passed; and finally, for *guard transitions*, the combination of colours of the input tokens needs to reach a threshold value before the transition can fire. The firing of immediate transitions has priority over the firing of delay or guard transitions. The number of tokens fired (zero or one token is fired per outgoing arc), and their colours are determined by a transition-dependent probabilistic measure named *firing measure*, which uses the colours of the input tokens as input. The marking of a DCPN at a particular time is determined by the colours of all tokens residing in the DCPN at that time, and the places in which they reside, in a particular uniquely defined order. The modelling and specification power is revealed in, e.g., [BKKB04] and numerous other studies, in which DCPN have been used successfully to model large scale air transport operations involving multiple humans, technical systems, procedures, etc.

DCPN combine different features of already existing Petri net classes, such as those described in Chapter 2. The unique feature is the existence of equivalence relations with PDP. The existence of such equivalence relations allows combining the modelling and specification power of Petri nets with the stochastic analysis power of PDP.

This chapter is organised as follows: Section 3.2 provides a few preliminaries and definitions on stochastic processes that are necessary to properly understand the theorems proven in this chapter. Section 3.3 defines DCPN. Section 3.4 describes PDP. Section 3.5 shows that for each arbitrary PDP we can construct an equivalent DCPN process. Section 3.6 shows that for each arbitrary DCPN we can construct an equivalent PDP. Section 3.7 discusses the conditions under which the equivalence relations hold true. Finally, Section 3.8 draws conclusions. For illustrative examples we refer to Chapter 6.

3.2 Preliminaries

This thesis assumes that the reader is familiar with the basics and notational conventions of stochastic processes. For convenience, a brief introduction is given in Appendix A. This section provides a few additional preliminaries and definitions on stochastic processes that are necessary to properly understand the theorems proven in this chapter and Chapter 4.

Consider a probability space $(\Omega, \Im, \mathbb{P})$, where Ω is the sample space, $\Im$ is a σ-algebra on Ω, and $\mathbb{P}$ is a probability measure. Also consider an index set $\mathbb{T}$ and a measurable space $(E, \mathcal{B}(E))$, where $\mathcal{B}(E)$ is the Borel σ-algebra on E, i.e., the smallest σ-algebra that contains all open subsets of E.

Throughout this thesis, we take $\mathbb{T} = \mathbb{R}^+ = [0, \infty)$, i.e., the positive time line.

A *stochastic process* with index set $\mathbb{T}$ and state space $(E, \mathcal{B}(E))$ defined on a probability space $(\Omega, \Im, \mathbb{P})$ is a function $X : \mathbb{T} \times \Omega \to E$ such that for each $t \in \mathbb{T}$, $X(t, \cdot) : \Omega \to E$ is an E-valued random variable, i.e., $\{\omega \mid X(t, \omega) \in B\} \in \Im$ for every $B \in \mathcal{B}(E)$. Generally, $X(t, \omega)$ is denoted as $X_t(\omega)$ and $X(t, \cdot)$ is denoted as $X(t)$ or as X_t. The function $X(\cdot, \omega)$ (also denoted $\{X_t(\omega); t \in \mathbb{T}\}$) is called the *sample path* of the process at ω. A stochastic process may also consist of more than one component, e.g., $\{X_t^1, X_t^2\}$.

For $0 \leq t_1 < t_2 < \cdots < t_k$, let $\mu_{t_1,\ldots,t_k}$ be the probability measure on $\mathcal{B}(E) \times \cdots \times \mathcal{B}(E)$ defined by $\mu_{t_1,\ldots,t_k}(B) = \mathbb{P}\{(X(t_1), \ldots, X(t_k)) \in B\}$, with $B \in \mathcal{B}(E) \times \cdots \times \mathcal{B}(E)$. The probability measures $\{\mu_{t_1,\ldots,t_k} \mid k \geq 1, 0 \leq t_1 < t_2 < \cdots < t_k\}$ are called the *finite-dimensional distributions* of X.

- A stochastic process Y is said to be a *version* of X (and X is a version of Y) if X and Y have the same finite-dimensional distributions.
- A stochastic process Y is said to be a *modification* of X if X and Y are defined on the same probability space and for each $t \geq 0$, $\mathbb{P}\{X(t) = Y(t)\} = 1$.
- Two stochastic processes X and Y are said to be *indistinguishable* if there exists $N \in \Im$ such that $\mathbb{P}\{N\} = 0$ and $X(\cdot, \omega) = Y(\cdot, \omega)$ for all $\omega \notin N$.

If two processes are indistinguishable, then they are also modifications. If two processes are modifications, they are also versions. The term *versions* is often replaced by *identically distributed* or *probabilistically equivalent*. The term *indistinguishable* is often replaced by *pathwise equivalent*.

This chapter will prove that each arbitrary PDP can be mapped to a DCPN process which is *pathwise equivalent* to the original PDP, by which we mean that the PDP and this DCPN process are *indistinguishable*. In addition, the chapter will prove that each arbitrary DCPN can be mapped to a PDP which is *probabilistically equivalent* to the original DCPN process, by which we mean that the DCPN process and this PDP are *versions*.

Both PDP and DCPN will be defined by means of an *execution* of a *stochastic hybrid system*, where,

- A *stochastic hybrid system* is a collection of objects that specify a hybrid state space, a continuous flow mechanism and a hybrid jump mechanism.
- An *execution* of the stochastic hybrid system defines a sample path $X(\cdot, \omega)$ of a stochastic hybrid process for any arbitrary $\omega \in \Omega$. This means, for arbitrary $\omega \in \Omega$, it defines an initiation $X(0, \omega)$, an increasing sequence of random variables $\tau_i : \Omega \to [0, \infty]$ that characterise the times of hybrid jumps, the continuous flow $\{X(t, \omega); t \in (\tau_{i-1}, \tau_i)\}$ between jumps, and the hybrid state at each jump $X(\tau_i, \omega)$. This definition is in line with the definition provided in [LJSE99] for execution of hybrid automata.

Throughout this chapter and the next one, we need to characterise random vectors according to given probability measures. The procedure is explained for two special cases and one general case below:

- Suppose X is a random variable $X : \Omega \to \mathbb{R}$ with a corresponding *probability distribution function* $F_X : \mathbb{R} \to [0,1]$, i.e., $F_X(x) \triangleq \mathbb{P}\{\omega \in \Omega \mid X(\omega) \leq x\}$ (or $F_X(x) = \mathbb{P}\{X \leq x\}$ for short). Then $X(\omega)$ is characterised in terms of F_X as follows: Consider a random variable $U : \Omega \to [0,1]$ which has a uniform distribution on the unit interval $[0,1]$, i.e., for all $u \in [0,1]$, $\mathbb{P}\{U \leq u\} = u$. Then $X(\omega) = F_X^{qf}(U(\omega))$, where $F_X^{qf} : [0,1] \to \mathbb{R}$ is defined by $F_X^{qf}(u) \triangleq \inf\{x \mid F_X(x) \geq u\}$. This equality is due to $\mathbb{P}\{U \leq u\} = u$ and therefore $\mathbb{P}\{F_X(x) \geq U\} = \mathbb{P}\{U \leq F_X(x)\} = F_X(x) = \mathbb{P}\{X \leq x\}$. In statistics, F_X^{qf} is referred to as *quantile function*, see, e.g., [Mad02].
- Similarly, if X is a random variable with a corresponding *survivor function* $\Gamma_X(x) = 1 - F_X(x)$, then $X(\omega) = \Gamma_X^{qf}(U(\omega))$, where Γ_X^{qf} is defined by $\Gamma_X^{qf}(u) \triangleq \inf\{x \mid \Gamma_X(x) \leq u\}$. This is due to $U \sim U[0,1]$ then $1 - U \sim U[0,1]$.
- More generally, if X is a random vector with a corresponding *probability measure* Ψ on a Borel subset of E, then there exists a measurable function $\Psi^{qf} : [0,1] \to E$ such that $\mu_L\{u \mid \Psi^{qf}(u) \in A\} = \Psi\{X \in A\}$, where μ_L is the Lebesgue measure. With this, $X(\omega) = \Psi^{qf}(U(\omega))$. The existence of this Ψ^{qf} is proven in [Dav93, Corollary 23.4].

Notice that each of the characterisations above is in terms of uniform $U[0,1]$ random variables.

3.3 Dynamically coloured Petri nets

This section presents a definition of *dynamically coloured Petri net* (DCPN). Where possible, the notation introduced by Jensen [Jen92] for coloured Petri net is used.

Definition 3.1 (Dynamically coloured Petri net)**.** *A DCPN is a collection of elements* $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$*, together with a DCPN execution prescription which makes use of a sequence* $\{U_i; i = 0, 1, \ldots\}$ *of independent uniform* $U[0,1]$ *random variables and five rules R0–R4 that solve enabling conflicts.*

The execution of a DCPN defines the sample path of a stochastic process which is a hybrid random vector formed by the collection of colours of all tokens, and by the places in which these tokens reside. Informally, this is explained as follows: A DCPN graph is formed by the places $\mathcal{P}$, the transitions $\mathcal{T}$ and the arcs $\mathcal{A}$, which are connected to one another as specified by the node function $\mathcal{N}$. The initial marking $\mathcal{I}$ puts an initial set of tokens in some or all of the DCPN places, and gives these tokens a value (a colour). The state space of the colour of a token in place $P \in \mathcal{P}$ is $\mathcal{C}(P) \in \mathcal{S}$. From the initial time onwards, the colour C_t^P of a token in place P at time t equals the

trajectory solution of an ordinary differential equation formed by the drift coefficient $\mathcal{V}_P \in \mathcal{V}$, i.e., $dC_t^P = \mathcal{V}_P(C_t^P)dt$. A transition $T \in \mathcal{T}$ is enabled if it has tokens in each of its input places, and if a second condition holds true, which is defined by the DCPN elements $\mathcal{G}$ and $\mathcal{D}$ and which uses the colours of the input tokens. An enabled transition removes the input tokens, and produces coloured output tokens defined according to $\mathcal{F}_T \in \mathcal{F}$. Each new token follows the trajectory solution of the ordinary differential equation connected to the place in which it then resides, until it is removed by a transition that is enabled.

The formal DCPN definition provided below is organised as follows:

- Section 3.3.1 defines the DCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$).
- Section 3.3.2 explains the DCPN execution.
- Section 3.3.3 explains how the DCPN execution defines a unique stochastic process.

3.3.1 DCPN elements

The DCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) are defined as follows:

- $\mathcal{P}$ is a finite set of places.
- $\mathcal{T}$ is a finite set of transitions, such that $\mathcal{T} \cap \mathcal{P} = \emptyset$. The set $\mathcal{T}$ consists of 1) a set $\mathcal{T}_G$ of guard transitions, 2) a set $\mathcal{T}_D$ of delay transitions and 3) a set $\mathcal{T}_I$ of immediate transitions, with $\mathcal{T} = \mathcal{T}_G \cup \mathcal{T}_D \cup \mathcal{T}_I$, and $\mathcal{T}_G \cap \mathcal{T}_D = \mathcal{T}_D \cap \mathcal{T}_I = \mathcal{T}_I \cap \mathcal{T}_G = \emptyset$.
- $\mathcal{A}$ is a finite set of arcs such that $\mathcal{A} \cap \mathcal{P} = \mathcal{A} \cap \mathcal{T} = \emptyset$. The set $\mathcal{A}$ consists of 1) a set $\mathcal{A}_O$ of ordinary arcs, 2) a set $\mathcal{A}_E$ of enabling arcs and 3) a set $\mathcal{A}_I$ of inhibitor arcs, with $\mathcal{A} = \mathcal{A}_O \cup \mathcal{A}_E \cup \mathcal{A}_I$, and $\mathcal{A}_O \cap \mathcal{A}_E = \mathcal{A}_E \cap \mathcal{A}_I = \mathcal{A}_I \cap \mathcal{A}_O = \emptyset$.
- $\mathcal{N} : \mathcal{A} \to \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$ is a node function which maps each arc $A \in \mathcal{A}$ to a pair of ordered nodes $\mathcal{N}(A)$, where a node is a place or a transition[2]. The place of $\mathcal{N}(A)$ is denoted by $P(A)$, the transition of $\mathcal{N}(A)$ is denoted by $T(A)$, such that for all $A \in \mathcal{A}_E \cup \mathcal{A}_I$: $\mathcal{N}(A) = (P(A), T(A))$ and for all $A \in \mathcal{A}_O$: either $\mathcal{N}(A) = (P(A), T(A))$ or $\mathcal{N}(A) = (T(A), P(A))$. Further notation:
  - $A(T) = \{A \in \mathcal{A} \mid T(A) = T\}$ is the set of arcs connected to transition T,
    $A_{in}(T) = \{A \in A(T) \mid \mathcal{N}(A) = (P(A), T)\}$ is the set of input arcs of T,
    $A_{out}(T) = \{A \in A(T) \mid \mathcal{N}(A) = (T, P(A))\}$ is the set of output arcs of T,
    $A_{in,O}(T) = A_{in}(T) \cap \mathcal{A}_O$ is the set of ordinary input arcs of T,
    $A_{in,OE}(T) = A_{in}(T) \cap \{\mathcal{A}_E \cup \mathcal{A}_O\}$ is the set of input arcs of T that are either ordinary or enabling, and

[2]Note that the DCPN arcs have no arc weights like P/T nets have, but this node function definition leaves the freedom to define multiple arcs between the same pair of transition and place or place and transition (except if an inhibitor arc is involved).

  - $P(\mathcal{A}^{\subset}) = \{P(A); A \in \mathcal{A}^{\subset}\}$ is the multi-set of places connected to the subset of arcs $\mathcal{A}^{\subset} \subset \mathcal{A}$. If there are $A_i \in \mathcal{A}^{\subset}$ and $A_j \in \mathcal{A}^{\subset}$ for which $P(A_i) = P(A_j)$, then this place $P(A_i) = P(A_j)$ occurs in the set $P(\mathcal{A}^{\subset})$ multiple times.

  Finally, $\{A_i \in \mathcal{A}_I \mid \exists A \in \mathcal{A}, A \neq A_i : \mathcal{N}(A) = \mathcal{N}(A_i)\} = \emptyset$, i.e., if an inhibitor arc points from a place P to a transition T, there is no other arc from P to T.

- $\mathcal{S} \subset \{\mathbb{R}^0, \mathbb{R}^1, \mathbb{R}^2, \ldots\}$ is a finite set of colour types, with $\mathbb{R}^0 \triangleq \emptyset$.

- $\mathcal{C} : \mathcal{P} \to \mathcal{S}$ is a colour type function which maps each place $P \in \mathcal{P}$ to a specific colour type in $\mathcal{S}$. Each token in P is to have a colour in $\mathcal{C}(P)$. Since $\mathcal{C}(P) \in \{\mathbb{R}^0, \mathbb{R}^1, \mathbb{R}^2, \ldots\}$, there exists a function $n : \mathcal{P} \to \mathbb{N}$ such that $\mathcal{C}(P) = \mathbb{R}^{n(P)}$. If $\mathcal{C}(P) = \mathbb{R}^0 \triangleq \emptyset$ then a token in P has no colour. Further notation: if $P(\mathcal{A}^{\subset})$ contains more than one place, e.g., $P(\mathcal{A}^{\subset}) = \{P_{i_1}, \ldots, P_{i_k}\}$, then $\mathcal{C}(P(\mathcal{A}^{\subset}))$ is defined by $\mathcal{C}(P_{i_1}) \times \cdots \times \mathcal{C}(P_{i_k})$, where $\times$ denotes Cartesian product.

- $\mathcal{I} : \mathbb{N}^{|\mathcal{P}|} \times \mathcal{C}(\mathcal{P})^{\mathbb{N}} \to [0, 1]$ is a probability measure, which defines the initial marking of the net: for each place it defines a number ≥ 0 of tokens initially in it and it defines their initial colours. Here, $\mathbb{N}^{|\mathcal{P}|}$ denotes the space of $|\mathcal{P}|$-dimensional non-negative finite integer vectors $\mathbb{N}^{|\mathcal{P}|} \triangleq \{(m_1, \ldots, m_{|\mathcal{P}|}); m_i \in \mathbb{N}, m_i < \infty, i = 1, \ldots, |\mathcal{P}|\}$ and $\mathcal{C}(\mathcal{P})^{\mathbb{N}}$ denotes a set of Euclidean spaces defined by $\mathcal{C}(\mathcal{P})^{\mathbb{N}} \triangleq \{\mathcal{C}(P_1)^{m_1} \times \cdots \times \mathcal{C}(P_{|\mathcal{P}|})^{m_{|\mathcal{P}|}}; m_i \in \mathbb{N}, m_i < \infty, i = 1, \ldots, |\mathcal{P}|\}$, where $\mathcal{C}(P_i)^{m_i} \triangleq \mathbb{R}^{m_i n(P_i)}$ for all $i = 1, \ldots, |\mathcal{P}|$, where $\mathcal{P}$ is denoted $\mathcal{P} = \{P_1, \ldots P_{|\mathcal{P}|}\}$.

  It is assumed that all tokens in a place are distinguishable by a unique identification tag which translates to a unique ordering/listing of tokens per place.

- $\mathcal{V} = \{\mathcal{V}_P; P \in \mathcal{P}, \mathcal{C}(P) \neq \mathbb{R}^0\}$ is a set of token colour functions. For each place $P \in \mathcal{P}$ for which $\mathcal{C}(P) \neq \mathbb{R}^0$, it contains a measurable mapping $\mathcal{V}_P : \mathcal{C}(P) \to \mathcal{C}(P)$ that defines the drift coefficient of a differential equation for the colour of a token in place P. It is assumed that $\mathcal{V}_P$ satisfies conditions that ensure a unique solution of the differential equation $dC_t = \mathcal{V}_P(C_t)dt$.[3]

- $\mathcal{G} = \{\mathcal{G}_T; T \in \mathcal{T}_G\}$ is a set of transition guards. For each $T \in \mathcal{T}_G$, it contains a transition guard $\mathcal{G}_T$, which is an open subset in $\mathcal{C}(P(A_{in,OE}(T)))$ with boundary $\partial\mathcal{G}_T$. If $\mathcal{C}(P(A_{in,OE}(T))) = \mathbb{R}^0$ then $\partial\mathcal{G}_T = \emptyset$.[4] There is no requirement that $\mathcal{G}_T$ be connected.

[3]Note that in earlier DCPN definitions, e.g., [EB05], [EB06], it was assumed that $\mathcal{V}_P$ satisfies local Lipschitz condition. This condition has now been relaxed to existence and uniqueness of solution(s) of the related differential equation(s).

[4]In earlier DCPN definitions, the transition guard was defined as a Boolean function that evaluated to True if the boundary of an open subset was hit by the input token colours. Without losing generality, the transition guard is now defined to be the open subset itself.

- $\mathcal{D} = \{\mathcal{D}_T; T \in \mathcal{T}_D\}$ is a set of transition delay rates. For each $T \in \mathcal{T}_D$, it contains a locally integrable transition delay rate $\mathcal{D}_T : \mathcal{C}(P(A_{in,OE}(T))) \to \mathbb{R}^+$. If $\mathcal{C}(P(A_{in,OE}(T))) = \mathbb{R}^0$ then $\mathcal{D}_T$ is a constant function. [5]

- $\mathcal{F} = \{\mathcal{F}_T; T \in \mathcal{T}\}$ is a set of firing measures. For each $T \in \mathcal{T}$, it contains a firing measure $\mathcal{F}_T : (\{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))) \times \mathcal{C}(P(A_{in,OE}(T))) \to [0,1]$, which generates the number and values of the tokens produced when transition T fires, given the value of the vector $\in \mathcal{C}(P(A_{in,OE}(T)))$ that collects all input tokens.

  Here, $\{0,1\}^{|A_{out}(T)|} \triangleq \{(e_1, \ldots, e_{|A_{out}(T)|}); e_i \in \{0,1\}, i = 1, \ldots, |A_{out}(T)|\}$, and if $P(A_{out}(T)) = \{P_{O_1}, \ldots, P_{O_{|A_{out}(T)|}}\}$ then $\{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T))) \triangleq \{(e^T, a^T); e^T = (e_1, \ldots, e_{|A_{out}(T)|}), e_i \in \{0,1\}, a^T \in \mathbb{R}^{n_{out}(T)}, n_{out}(T) = \sum_{i=1}^{|A_{out}(T)|} e_i \cdot n(P_{O_i})\}$. Hence, for $i = 1, \ldots, |A_{out}(T)|$, if $e_i = 1$ then the ith output place P_{O_i} of T gets a token with a value in $\mathbb{R}^{n(P_{O_i})}$ and if $e_i = 0$ then the ith output place of T does not get a token. The vector e^T then denotes which output places get a token and the vector a^T collects all colours of tokens produced.

  For each fixed $H \subset \{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))$, $\mathcal{F}_T(H; \cdot)$ is measurable. For any $c \in \mathcal{C}(P(A_{in,OE}(T)))$, $\mathcal{F}_T(\cdot; c)$ is a probability measure.

For the places, transitions and arcs, there is a graphical notation as in Figure 3.1.

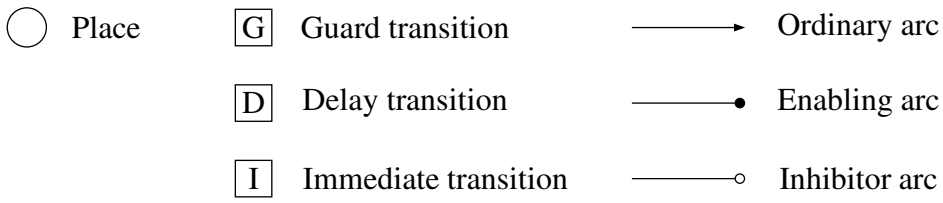


Figure 3.1 Graphical notation for places, transitions and arcs in a DCPN

3.3.2 DCPN execution

The execution of a DCPN provides a series of increasing stopping times $0 = \tau_0 < \tau_1 < \tau_2 < \cdots$, with for $t \in (\tau_i, \tau_{i+1})$ a fixed number of tokens per place and per token a colour which is the solution of an ordinary differential equation. The execution uses a countable sequence $\{U_i; i = 0, 1, \ldots\}$ of independent random variables each having a uniform $U[0,1]$ distribution, and five rules R0–R4 that solve enabling conflicts.

[5] In earlier DCPN definitions, the transition delay was defined as a probability distribution function that made use of an integrable transition delay rate. Without losing generality, the transition delay is now defined to be the delay rate itself.

Initiation The probability measure $\mathcal{I}$ characterises an initial marking at $\tau_0 = 0$, i.e., it gives each place $P \in \mathcal{P}$ zero or more tokens and gives each token in P a colour in $\mathcal{C}(P)$, i.e., a Euclidean-valued vector. Define, following Section 3.2, the quantile function of $\mathcal{I}$ by a measurable function $\mathcal{I}^{qf} : [0,1] \to \mathbb{N}^{|\mathcal{P}|} \times \mathcal{C}(\mathcal{P})^{\mathbb{N}}$ such that $\mu_L\{u \mid \mathcal{I}^{qf}(u) \in H\} = \mathcal{I}(H)$, for H Borel measurable and μ_L the Lebesgue measure. Then the initial marking is a random hybrid vector characterised by $(M_0, C_0) = \mathcal{I}^{qf}(U_0)$. Here, M_0 is a $|\mathcal{P}|$-dimensional vector of non-negative integers, the ith component $M_{i,0}$ of which denotes the number of tokens initially in place P_i, $i = 1, \ldots, |\mathcal{P}|$, and C_0 is a $\sum_{i=1}^{|\mathcal{P}|} M_{i,0} n(P_i)$-dimensional Euclidean-valued random vector which provides the colours of the initial tokens (tokens in place P_i have a colour in $\mathcal{C}(P_i) = \mathbb{R}^{n(P_i)}$). If $M_{1,0} \geq 1$ then the first $n(P_1)$ components of C_0 are assigned to the first token in P_1. If $M_{1,0} \geq 2$ then the next $n(P_1)$ components of C_0 are assigned to the second token in P_1, etc., until all tokens in P_1 have their assigned colour. The following components of C_0 are assigned to tokens in places $P_2, \ldots, P_{|\mathcal{P}|}$ according to the same recipe. If $\mathcal{C}(P) = \mathbb{R}^0$ (i.e., $n(P) = 0$) then the tokens in P get no colour.

Token colour evolution For each token in each place $P \in \mathcal{P}$ for which $\mathcal{C}(P) \neq \mathbb{R}^0$: if the colour of this token is C_τ^P at time $t = \tau$ and if this token is still in this place at $t > \tau$, then the colour C_t^P of this token follows the unique solution of the ordinary differential equation $dC_t^P = \mathcal{V}_P(C_t^P)dt$ with initial condition C_τ^P, i.e., $C_t^P = C_\tau^P + \int_\tau^t \mathcal{V}_P(C_s^P)ds$. Each token in a place for which $\mathcal{C}(P) = \mathbb{R}^0$ remains without colour.

Transition enabling A transition T is *pre-enabled* if it has at least one token per incoming ordinary and enabling arc in each of its input places and has no token in places to which it is connected by an inhibitor arc. For each transition T that is pre-enabled at τ, consider one token per ordinary and enabling arc in its input places and write $C_t^T \in \mathcal{C}(P(A_{in,OE}(T)))$, $t \geq \tau$, as the column vector containing the colours of these tokens; C_t^T evolves through time according to its corresponding token colour functions of the places in $P(A_{in,OE}(T))$. If this vector is not unique (for example, if one input place contains several tokens per arc), all possible such vectors are executed in parallel. Hence, a transition can be pre-enabled by multiple combinations of input tokens in parallel.

A transition T is *enabled* if two requirements hold true. The first requirement is that the transition is pre-enabled. The second requirement is as follows:

- If T is an immediate transition, i.e., $T \in \mathcal{T}_I$, there is no second requirement: the transition is enabled when it is pre-enabled.

- If T is a guard transition, i.e., $T \in \mathcal{T}_G$, the second requirement holds true when $C_t^T \in \partial\mathcal{G}_T$.

- If T is a delay transition, i.e., $T \in \mathcal{T}_D$, the second requirement holds true at $t = \tau + \sigma^T$, where σ^T is according to a probability distribution function $D_T(t-\tau) = 1 - \exp(-\int_\tau^t \mathcal{D}_T(C_s^T)ds)$.

Here, the characterisation of σ^T, in terms of uniform random variables, is as follows: Suppose $\mathcal{T}_\tau^D$ denotes the multiset of delay transitions that are pre-enabled at τ and that are still pre-enabled at $t > \tau$, where a transition is included in the multiset multiple times if it is pre-enabled by multiple vectors of input tokens in parallel. Without loss of generality, denote $\mathcal{T}_\tau^D = \{T_1, \ldots, T_{|\mathcal{T}_\tau^D|}\}$. Then for $T_i \in \mathcal{T}_\tau^D$, we find $\sigma^{T_i} = D_{T_i}^{qf}(U_j)$, where $D_{T_i}^{qf}$ is defined by $D_{T_i}^{qf}(u) = \inf\{t - \tau \mid \exp(-\int_\tau^t \mathcal{D}_{T_i}(C_s^{T_i})ds) \leq u\}$, with $\inf\{\ \} = +\infty$. Here, U_j runs through $|\mathcal{T}_\tau^D|$ uniform random variables, i.e., the number of pre-enabled delay transitions, such that each pre-enabled delay transition uses one uniform random variable (per vector of input tokens) to determine its time of enabling.[6]

In the case of competing enablings, the following rules apply:

R0 The firing of an immediate transition has priority over the firing of a guard or a delay transition.

R1 If one transition becomes enabled by two or more sets of input tokens at exactly the same time, and the firing of any one set will not disable one or more other sets, then it will fire these sets of tokens independently, at the same time.

R2 If one transition becomes enabled by two or more sets of input tokens at exactly the same time, and the firing of any one set disables one or more other sets, then the set that is fired is selected randomly, with the same probability for each set.

R3 If two or more transitions become enabled at exactly the same time and the firing of any one transition will not disable the other transitions, then they will fire at the same time.

R4 If two or more transitions become enabled at exactly the same time and the firing of any one transition disables some other transitions, then each combination of transitions that can fire independently without leaving enabled transitions gets the same probability of firing.

By these rules and their combinations, if a transition is enabled in a particular set of tokens, then it is either fired or it is disabled (in this set of tokens) by the firing of another transition. An example of application of rule R1+R2+R4 is given in Figure 3.2, and the explanation below it.

[6]Note that an equivalent derivation is $D_{T_i}^{qf}(u) = \inf\{t - \tau \mid \exp(-\int_\tau^t \mathcal{D}_{T_i}(C_s^{T_i})ds) \leq u\} = \inf\{t - \tau \mid \int_\tau^t \mathcal{D}_{T_i}(C_s^{T_i})ds \geq -\ln u\}$. Also note that if $U \sim U[0,1]$, i.e., uniform on $[0,1]$, then $-\ln U \sim \text{Exp}\{1\}$, i.e., exponential with intensity 1. This equivalent formulation is not pursued further since commonly, the way to generate exponential random variables is by means of uniform random variables and taking the natural logarithm transformation above, hence we are back at using uniform variables.

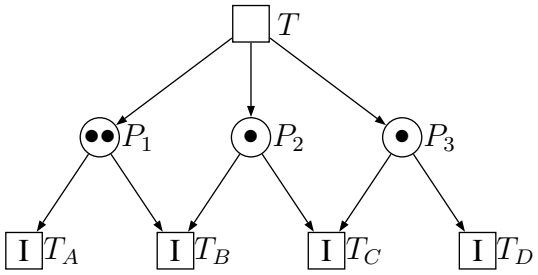


Figure 3.2 Example application of a combination of Rules R1, R2 and R4

In Figure 3.2, transitions T_A, T_B, T_C and T_D are all enabled, but they can only simultaneously fire in the sets $T_A + T_A + T_C$, $T_A + T_A + T_D$, $T_A + T_B + T_D$ or $T_A + T_B + T_D$. Here, the last set is listed twice: In one combination, the first token in P_1 is fired by T_A and the second by T_B; in the other combination, the first token in P_1 is fired by T_B and the second by T_A. This order may be relevant if the tokens in P_1 have different colours. Each of these four sets gets the same probability of occurrence, e.g., the probability that T_A fires twice in parallel with itself (by application of Rule R1), and together with T_C, is equal to $\frac{1}{4}$. The firing of only, e.g., T_D would leave enabled transitions behind, hence is not considered as an optional 'set'.

Transition firing If T is enabled, suppose this occurs at time τ and in a particular vector of token colours C_τ^T, it removes one token per arc in $A_{in,O}(T)$ corresponding with C_τ^T from each of its input places (i.e., tokens are not removed along enabling arcs, meaning that tokens in $P(A_{in,OE}(T))\backslash P(A_{in,O}(T))$ are not removed). Next, T produces zero or one token along each output arc: If (e_τ^T, a_τ^T) is a hybrid random vector with probability measure $\mathcal{F}_T(\cdot; C_\tau^T)$, then vector $e_\tau^T \in \{0,1\}^{|A_{out}(T)|}$ is an $|A_{out}(T)|$-dimensional vector of zeros and ones, where the ith vector element corresponds with the ith outgoing arc of transition T. An output place gets a token iff it is connected to an arc that corresponds with a vector element 1. Moreover, a_τ^T specifies the colours of the produced tokens, i.e., if the first 1 in e_τ^T corresponds with an arc from T to P_j, then the first $n(P_j)$ elements in vector a_τ^T are assigned to the token produced in output place P_j. The remaining elements in a_τ^T are assigned to other tokens in the same way.

Suppose $\mathcal{T}_\tau^F$ denotes the multiset of transitions that will fire at τ, where a transition is included in the multiset multiple times if it fires multiple vectors of input tokens in parallel (see Rule R1 above). Without loss of generality, denote $\mathcal{T}_\tau^F = \{T_1, \ldots, T_{|\mathcal{T}_\tau^F|}\}$. Then for each $T_i \in \mathcal{T}_\tau^F$, the random hybrid vector from $\mathcal{F}_{T_i}(\cdot; C_\tau^{T_i})$ is characterised by defining the quantile function of $\mathcal{F}_{T_i}(\cdot; C_\tau^{T_i})$ as a measurable function $\mathcal{F}_{T_i}^{qf} : [0,1] \times \mathcal{C}(P(A_{in,OE}(T_i))) \to \{0,1\}^{|A_{out}(T_i)|} \times \mathcal{C}(P(A_{out}(T_i)))$ such that $\mu_L\{u \mid \mathcal{F}_{T_i}^{qf}(u,c) \in H\} = \mathcal{F}_{T_i}(H; c)$ for H in the Borel set of $\{0,1\}^{|A_{out}(T_i)|} \times \mathcal{C}(P(A_{out}(T_i)))$ and μ_L is the Lebesgue measure. Then $(e_\tau^{T_i}, a_\tau^{T_i}) = \mathcal{F}_{T_i}^{qf}(U_j, C_\tau^{T_i})$. Here, U_j runs through $|\mathcal{T}_\tau^F|$ uniform random variables, such that each firing transition uses one uniform random variable (per vector of input tokens) to determine its output tokens.

Execution from τ_0 onwards Using the above principles, the execution from τ_0 onwards is as follows: At time τ_0, the initial marking produces an initial set of coloured tokens, according to the description above (at Initiation). Next, at $t = \tau_0$, zero or more transitions are pre-enabled (if this number is zero, the DCPN is *dead*, see Definition 2.17). If these include immediate transitions or enabled guard transitions (e.g., if the initial marking starts on the guard boundary), then these are fired without delay, but with use of rules R0–R4 (see at Transition enabling and at Transition firing above). If after this, still immediate transitions are enabled, then these are also fired, and so forth, until no more immediate or guard transitions are enabled. Each of the transitions that fire at τ_0 uses their firing measure and one uniform random variable (per firing) to determine the number and colours of their output tokens. If $\mathcal{T}^F_{\tau_0}$ is the multiset of immediate or guard transitions that fire at τ_0 (this multiset is possibly empty), then $|\mathcal{T}^F_{\tau_0}|$ uniform random variables are used by their respective firing functions, i.e., U_j, for $j = 1, \ldots, |\mathcal{T}^F_{\tau_0}|$.

Next, for $t > \tau_0$, the colours of the tokens in places for which $\mathcal{C}(P) \neq \mathbb{R}^0$ start evolving according to their token colour functions (see at Token colour evolution, above), until the next transition is enabled, say at $t = \tau_1$ (see at Transition enabling, above). Any pre-enabled guard transition T is enabled if its vector of input tokens satisfies $C^T_t \in \partial\mathcal{G}_T$. For $t > \tau_0$, any pre-enabled delay transition becomes enabled according to a time corresponding to its delay function. This time uses a random variable $U_{|\mathcal{T}^F_{\tau_0}|+j}$, for $j = 1, \ldots, |\mathcal{T}^D_{\tau_0}|$, where $|\mathcal{T}^D_{\tau_0}|$ is the number of delay transitions that are pre-enabled at τ_0. If at $t = \tau_1$, one or more transitions are enabled, they are fired and their firing measures are used to produce new tokens (see at Transition firing, above). If after this, at $t = \tau_1$, immediate or guard transitions are enabled, then these are fired without delay, as above for the situation at $t = \tau_0$, until no more transitions are enabled at $t = \tau_1$. Next, for $t > \tau_1$, the execution continues in the same way as described above for $t > \tau_0$. Here, if a pre-enabled delay transition was already pre-enabled at $t < \tau_1$ and still is pre-enabled at $t \geq \tau_1$ in the same vector of input tokens, then no new uniform variable is used to generate its time of enabling. If τ_2 denotes the time after τ_1 at which a next enabling occurs, the firing measures of the enabled transitions are used at τ_2, with help of a number of uniform random variables, to determine the tokens fired and their colours, and so forth.

In order to keep track of the identity of individual tokens, the tokens in a place are ordered according to the time at which they entered the place, or, if several tokens are produced for one place at the same time, according to the order within the set of arcs $\mathcal{A} = \{A_1, \ldots, A_{|\mathcal{A}|}\}$ along which these tokens were produced (the firing measure produces zero or one token along each output arc). If due to rule R1, a transition fires two or more tokens along one arc at the same time, their assigned order is according to the colours they have (smallest colour first). If under these conditions, two tokens have exactly the same colour, they are indistinguishable and the marking (defined next) will not be dependent on their order.

3.3.3 DCPN stochastic process

The marking of the DCPN is given by the numbers of tokens in the places and the associated colour values of these tokens. Due to the uniquely defined order of the tokens, the marking is unique except possibly when one or more transitions fire (particularly, immediate transitions fire without delay hence a sequence of immediate transitions firing will generate a sequence of markings at the same time instant). If a probability space $(\Omega, \Im, \mathbb{P})$ is given, the DCPN marking at each time instant can be mapped to a unique DCPN stochastic process $\{M_t, C_t\}$ as follows:

For any $t \geq \tau_{k-1}$, $k = 1, 2, \ldots$, let a token distribution be characterised by the vector $M'_t = (M'_{1,t}, \ldots, M'_{|\mathcal{P}|,t})$, where $M'_{i,t} \in \mathbb{N}$ denotes the number of tokens in place P_i at time t and $1, \ldots, |\mathcal{P}|$ refers to a unique ordering of places adopted for DCPN. At times $t \in (\tau_{k-1}, \tau_k)$ when no transition fires, the token distribution is unique and the DCPN discrete process state M_t is defined to be equal to M'_t. The associated colours of these tokens are uniquely gathered in a column vector C_t which first contains all colours of tokens in place P_1, next (i.e., below it) all colours of tokens in place P_2, etc, until place $P_{|\mathcal{P}|}$, where $1, \ldots, |\mathcal{P}|$ refers to a unique ordering of places adopted for DCPN. Within a place the colours of the tokens are ordered according to the unique ordering of tokens within their place defined for DCPN (see under DCPN execution above).

If at time $t = \tau_k$ ($k \geq 0$) one or more transitions fire, then the set of applicable token distributions is collected in $\widetilde{M}_{\tau_k} = \{M'_{\tau_k}; M'_{\tau_k}$ is a token distribution at time $\tau_k\}$, and the DCPN discrete process state at time τ_k is defined by $M_{\tau_k} = \{M'_{\tau_k}; M'_{\tau_k} \in \widetilde{M}_{\tau_k}$ and no transitions are enabled in $M'_{\tau_k}\}$; in words, M_{τ_k} is the token distribution that occurs after all transitions that fire at time τ_k have been fired. The associated colours of these tokens are uniquely gathered in a column vector C_{τ_k} in the same way as described above. This construction ensures that the process $\{M_t, C_t\}$ has limits from the left and is continuous from the right, i.e., it satisfies the càdlàg property. If at a time t when one or more transitions fire, the process $\{M_t\}$ jumps to the same value again, and only C_t makes a jump, then the càdlàg property for $\{C_t\}$ (hence for $\{M_t, C_t\}$) is still maintained due to the timing construction of $\{M_t\}$ above and the direct coupling of $\{C_t\}$ with $\{M_t\}$.

3.4 Piecewise deterministic Markov processes

This section presents, following [Dav93], a definition of *piecewise deterministic Markov process* (PDP). Consider a probability space $(\Omega^H, \Im^H, \mathbb{P})$, where Ω^H is the Hilbert cube, $\Im^H$ is the product σ-algebra, and $\mathbb{P}$ is the product probability measure. Here, the *Hilbert cube* Ω^H is defined by $\Omega^H = \prod_{i=0}^{\infty} V_i$, with V_i a copy of the unit interval $V = [0, 1]$. This provides the canonical space for a countable sequence of independent random variables $U_0(\omega), U_1(\omega), U_2(\omega), \ldots$, each having uniform $[0, 1]$ distribution, defined by $U_i(\omega) = \omega_i$ for elements $\omega = (\omega_0, \omega_1, \omega_2, \ldots) \in \Omega^H$.

Definition 3.2 (Piecewise deterministic Markov process)**.** *A PDP* $\{\theta_t, X_t\}$ *is defined by an execution on probability space* $(\Omega^H, \Im^H, \mathbb{P})$ *of a collection (*$\mathbf{K}$*,* d*,* E*,* f*,* θ_0*,* X_0*,* λ*,* Q*), under four standard conditions P1–P4.*

The elements in the collection ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q) are referred to as the *PDP elements*. An execution[7] of the PDP elements defines a PDP $\{\theta_t, X_t\}$; more precisely: it defines a sample path $\{(\theta_t(\omega), X_t(\omega)); t \in \mathbb{T}\}$ for arbitrary $\omega \in \Omega^H$. Informally, this is explained as follows: The PDP is a stochastic process that consists of two components: a discrete valued component $\{\theta_t\}$, which takes values in a countable set $\mathbf{K}$, and a continuous valued component $\{X_t\}$, which takes values in a Euclidean state space defined by d and E. The initial state is (θ_0, X_0), and from this state onwards, the PDP is defined by an ordinary differential equation with drift coefficient f, i.e., $d\theta_t = 0$ and $dX_t = f(\theta_t, X_t)dt$. A jump in the PDP state occurs spontaneously with rate λ, or forced, when $\{X_t\}$ hits the boundary of its state space. The PDP transition measure Q defines the value of the PDP state after the jump.

The formal explanation provided below is organised as follows:

- Section 3.4.1 defines the PDP elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q).
- Section 3.4.2 explains the execution of the PDP elements which defines the PDP $\{\theta_t, X_t\}$.
- Section 3.4.3 presents the PDP conditions P1–P4 and, following [Dav93], the PDP main properties.

3.4.1 PDP elements

The PDP elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q) are defined as follows:

- $\mathbf{K}$ is a countable set.
- $d : \mathbf{K} \to \mathbb{N}$ is a function that maps each $\theta \in \mathbf{K}$ to a natural number.
- $E = \{\{\theta\} \times E_\theta; \theta \in \mathbf{K}\}$ is the hybrid state space, where for each $\theta \in \mathbf{K}$, E_θ is an open subset of $\mathbb{R}^{d(\theta)}$. The boundary of E is $\partial E \triangleq \{\{\theta\} \times \partial E_\theta; \theta \in \mathbf{K}\}$, in which ∂E_θ is the boundary of E_θ.
- For all $\theta \in \mathbf{K}$, $f(\theta, \cdot) : \mathbb{R}^{d(\theta)} \to \mathbb{R}^{d(\theta)}$ is a vector field.
- $\theta_0 \in \mathbf{K}$ and $X_0 \in E_{\theta_0}$ are initial values.
- $\lambda : E \to \mathbb{R}^+$ is a jump rate function, i.e., the intensity of jumps.

[7] Note that [Dav93] uses the term sample path of a stochastic process, not the term execution.

- $Q : \mathcal{B}(E) \times (E \cup \partial E) \to [0, 1]$ is a PDP transition measure, where $\mathcal{B}(E)$ is the Borel σ-algebra on E. For any $(\theta, x) \in E \cup \partial E$, $Q(\cdot; \theta, x)$ is a probability measure.

3.4.2 PDP execution

This section describes the execution of the PDP elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q) which defines a piecewise deterministic Markov process $\{\theta_t, X_t\}$, i.e., a sample path $\{\theta_t(\omega), X_t(\omega); t \in \mathbb{T}\}$ for any arbitrary $\omega \in \Omega^H$. Recall that the Hilbert cube Ω^H provides a sequence $U_0(\omega), U_1(\omega), \ldots$ of independent uniform random variables defined by $U_i(\omega) = \omega_i$, for $\omega = (\omega_0, \omega_1, \ldots)$. This sequence is used to uniquely characterise a sample path of the PDP for any given $\omega \in \Omega^H$.

Initiation The execution starts with an initiation, say at time $\tau_0 = 0$. At this time, the PDP initial state is (θ_0, X_0), with $\theta_0(\omega) \in \mathbf{K}$ and $X_0(\omega) \in E_{\theta_0}$.

Process from initial time until first jump For $t \geq \tau_0$, the process $\{\theta_t\}$ keeps a constant value θ_0, i.e., $\theta_t(\omega) = \theta_0(\omega)$. The process $\{X_t\}$ is determined by the ordinary differential equation $dX_t = f(\theta_0, X_t)dt$ with initial value $X_0(\omega) \in E_{\theta_0}$, i.e., $X_t(\omega) = X_0(\omega) + \int_{\tau_0}^{t} f(\theta_0(\omega), X_s(\omega))ds =: \phi_{\theta_0, X_0, t-\tau_0}(\omega)$. Therefore, as long as no jumps occur, the PDP state at $t \geq \tau_0$ is given by $(\theta_t(\omega), X_t(\omega)) = (\theta_0(\omega), \phi_{\theta_0, X_0, t-\tau_0}(\omega))$. Here, $\phi_{\theta_\tau, X_\tau, t-\tau}(\omega))$ is referred to as the *flow*.

Generation of time of first jump At some random moment $\tau_1 > \tau_0$ the PDP state value may jump. This moment is the minimum of:

1. The time at which a forced jump occurs, i.e., when $\{X_t\}$ hits the boundary ∂E_{θ_0} of E_{θ_0}.

2. The time at which a spontaneous jump occurs with jump rate (intensity) $\lambda(\theta_t, X_t)$.

The *survivor function* for the time until the first jump after τ_0 is then given by $\Gamma_{\theta_0, X_0, t-\tau_0}$, where

$$\Gamma_{\theta, x, t-\tau} \triangleq \begin{cases} 0 & \text{if } t - \tau \geq t_*(\theta, x) \\ \exp\left(-\int_{\tau}^{t} \lambda(\theta, \phi_{\theta, x, s-\tau})ds\right) & \text{otherwise} \end{cases} \tag{3.4.1}$$

where $t_*(\theta, x) \triangleq \inf\{t - \tau > 0 \mid \phi_{\theta, x, t-\tau} \in \partial E_\theta\}$, with $\inf \emptyset = +\infty$. This means, $t_*(\theta_0, X_0)$ denotes the time until $\{X_t\}$ first hits the boundary ∂E_{θ_0}.

The time τ_1 of the first jump is given by $\tau_1(\omega) = \tau_0 + \sigma_1(\omega)$, where σ_1 is a random variable characterised by $\Gamma_{\theta_0, X_0, t-\tau_0}$: Define $\Gamma^{qf}(u, \theta, x, \tau) = \inf\{t - \tau \mid \Gamma_{\theta, x, t-\tau} \leq u\}$, with $\inf \emptyset = +\infty$, then $\sigma_1(\omega) = \Gamma^{qf}(U_1(\omega), \theta_0(\omega), X_0(\omega), \tau_0)$.

Process value right after the first jump The value $(\theta_{\tau_1}(\omega), X_{\tau_1}(\omega))$ of the PDP state right after the jump is a random hybrid vector ζ_1 characterised by PDP transition measure $Q(\cdot; \theta_{\tau_1-}, X_{\tau_1-}) = Q(\cdot; \theta_0, \phi_{\theta_0, X_0, \tau_1-\tau_0})$. Define the measurable function $Q^{qf} : [0,1] \times (E \cup \partial E) \to E$ such that $\mu_L\{u \mid Q^{qf}(u, \theta, x) \in B\} = Q(B; \theta, x)$ for B Borel measurable and μ_L the Lebesgue measure. Then $(\theta_{\tau_1}(\omega), X_{\tau_1}(\omega)) := \zeta_1(\omega) = Q^{qf}(U_2(\omega), \theta_0(\omega), \phi_{\theta_0, X_0, \tau_1-\tau_0}(\omega))$.

Process from time right after first jump onwards From time τ_1 onwards, the execution of the PDP is according to the same recipe: Let for $k = 2$, $(\theta_{\tau_{k-1}}(\omega), X_{\tau_{k-1}}(\omega))$ be the PDP state right after the $k-1$st jump, then for $k = 2, 3, \ldots$:

A random variable σ_k is characterised by survivor function $\Gamma_{\theta_{\tau_{k-1}}, X_{\tau_{k-1}}, t-\tau_{k-1}}$, i.e., $\sigma_k(\omega) = \Gamma^{qf}(U_{2k-1}(\omega), \theta_{\tau_{k-1}}(\omega), X_{\tau_{k-1}}(\omega), \tau_{k-1}(\omega))$. Then the time $\tau_k(\omega)$ of the kth jump is $\tau_k(\omega) = \tau_{k-1}(\omega) + \sigma_k(\omega)$. The sample path up to the kth jump is given by $(\theta_t(\omega), X_t(\omega))$, with for $\tau_{k-1}(\omega) \leq t < \tau_k(\omega) \leq \infty$, $\theta_t(\omega) = \theta_{\tau_{k-1}}(\omega)$ and $X_t(\omega) = X_{\tau_{k-1}}(\omega) + \int_{\tau_{k-1}}^{t} f(\theta_{\tau_{k-1}}(\omega), X_s(\omega))ds = \phi_{\theta_{\tau_{k-1}}, X_{\tau_{k-1}}, t-\tau_{k-1}}(\omega)$.

Next, a random hybrid vector $\zeta_k(\omega)$ is characterised by PDP transition measure Q, i.e., $\zeta_k(\omega) = Q^{qf}(U_{2k}(\omega), \theta_{\tau_{k-1}}(\omega), \phi_{\theta_{\tau_{k-1}}, X_{\tau_{k-1}}, \tau_k-\tau_{k-1}}(\omega))$. Then, if $\tau_k(\omega) < \infty$, the process state at the time $\tau_k(\omega)$ of the kth jump is given by $(\theta_{\tau_k}(\omega), X_{\tau_k}(\omega)) := \zeta_k(\omega)$.

3.4.3 PDP conditions

In [Dav93, Section 24.8], Davis presents the standard conditions under which the PDP above is uniquely defined. These conditions are formulated as P1–P4 below.

P1 For all $\theta \in \mathbf{K}$, $f(\theta, \cdot)$ is locally Lipschitz continuous, i.e., for any compact $D \subset \mathbb{R}^{d(\theta)}$, there exists a constant $L_D(\theta)$ such that $|f(\theta, x) - f(\theta, y)| \leq L_D(\theta)|x - y|$, for all $x, y \in D$. This ensures that for each initial state (θ_τ, X_τ) at initial time τ there exists a unique solution $X_t = \phi_{\theta_\tau, X_\tau, t-\tau}$ to the ordinary differential equation $dX_t = f(\theta_\tau, X_t)dt$. In addition, it is assumed that the flow $\phi_{\theta_\tau, X_\tau, t-\tau}$ never explodes, i.e., $t_\infty(\theta_\tau, X_\tau) = \infty$ whenever $t_*(\theta_\tau, X_\tau) = \infty$. Here,

- $t_\infty(\theta, x)$ denotes the explosion time of the flow $\phi_{\theta,x,t-\tau}$, i.e., $|\phi_{\theta,x,t-\tau}| \to \infty$ as $t \uparrow t_\infty(\theta, x)$,
- $t_*(\theta, x)$ denotes the time until $\phi_{\theta,x,t-\tau}$ hits the boundary of E_θ, i.e., $t_*(\theta, x) \triangleq \inf\{t - \tau > 0 \mid \phi_{\theta,x,t-\tau} \in \partial E_\theta\}$.

P2 $\lambda : E \to \mathbb{R}^+$ is a measurable function such that for all $(\theta, x) \in E$, there is $\epsilon(\theta, x) > 0$ such that $t \to \lambda(\theta, \phi_{\theta,x,t-\tau})$ is integrable on $[0, \epsilon(\theta, x))$.

P3 For each fixed $B \in \mathcal{B}(E)$, $Q(B;\cdot,\cdot)$ is measurable and $Q(\theta,x;\theta,x) = 0$[8] for all $(\theta,x) \in E$.

P4 If $N_t = \sum_k \mathbf{1}_{\{t \geq \tau_k\}}$, where $\mathbf{1}$ is an indicator function and $\tau_0, \tau_1, \ldots$ are the times at which a jump occurs, then it is assumed that for every starting point $(\theta,x) \in E$ and for all $t \in \mathbb{R}^+$, $\mathbb{E}N_t < \infty$, where $\mathbb{E}$ denotes expectation. This means, there will be a finite number of jumps in finite time.

Davis [Dav93] shows that under these standard conditions P1–P4:

- A PDP is a time-homogeneous strong Markov process, [Dav93, Theorem 25.5].
- $t_*(\cdot)$ is a Borel-measurable function, [Dav93, Lemma 27.1].
- A PDP is a Borel right process, [Dav93, Theorem 27.8].

In [Dav93, Theorem 26.14], Davis characterises the extended generator of the PDP, i.e., the operator $\mathcal{A}$ defined by $\frac{\partial}{\partial t}\mathbb{E}\{\varphi\} = \mathbb{E}\{\mathcal{A}\varphi\}$, as well as its domain.

Remark 3.1. *An important question is how to verify whether conditions P1–P4 are satisfied. For conditions P1–P3, this is relatively straightforward, except for the no-explosions condition in P1. Following [Øk02, Theorem 5.2.1], a sufficient condition for no explosions is that for all θ there exists $K(\theta)$ such that for all $x \in E_\theta$, $|f(\theta,x)| \leq K(\theta)(1+|x|)$.*

For condition P4, [Dav93, Page 60] gives sufficient conditions: (1) λ is bounded, and either (2a) or (2b) below are satisfied: (2a) $\forall(\theta,x) \in E$, $t_*(\theta,x) = \infty$, i.e., there are no 'active' boundaries; (2b) For some $\chi > 0$, $Q(A_\chi;\theta,x) = 1$ for all $(\theta,x) \in \partial E$, where $A_\chi = \{(\theta,x) \in E \mid t_*(\theta,x) \geq \chi\}$, where, as above, $t_*(\theta,x) \triangleq \inf\{t-\tau > 0 \mid \phi_{\theta,x,t-\tau} \in \partial E_\theta\}$. Notice that (2b) means that the process always jumps to a state from which it takes some time $\chi > 0$ to reach the boundary again.

3.5 Piecewise deterministic Markov processes into dynamically coloured Petri nets

This section shows that under a few conditions, each piecewise deterministic Markov process (PDP) can be represented by a dynamically coloured Petri net (DCPN), by providing a one-to-one mapping[9] from PDP into the set of DCPN processes.

Theorem 3.1 (PDP into DCPN)**.** *Consider an arbitrary PDP which is defined by an execution of its elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q) on the Hilbert cube $(\Omega^H, \Im^H, \mathbb{P})$. Suppose $\mathbf{K}$ is finite, and*

[8] Due to this condition it is possible to recognise jumps of the process.

[9] A *one-to-one mapping* is defined as a mapping $\varphi : A \to B$ such that if $\varphi(a_1) = \varphi(a_2)$ then $a_1 = a_2$, or, alternatively, if $a_1 \neq a_2$ then $\varphi(a_1) \neq \varphi(a_2)$.

that for each θ and initial value $x_0 = x$, the ordinary differential equation $dx_t = f(\theta, x_t)dt$ has a unique solution. Then the elements of this PDP can be mapped one-to-one to a DCPN ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) satisfying R0–R4. If the resulting DCPN is executed on the Hilbert cube $(\Omega^H, \Im^H, \mathbb{P})$ then the resulting DCPN process and the original PDP are pathwise equivalent.

Proof. For the proof we consider an arbitrary PDP $\{\theta_t, X_t\}$ which is defined by the execution of the PDP elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q). Next, we prove Theorem 3.1 by the following steps:

- (Construction of DCPN elements.) First, we assume that $\mathbf{K}$ is finite and that for each θ and initial value x_0, $dx_t = f(\theta, x_t)dt$ has a unique solution. We specify a one-to-one mapping that characterises DCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) in terms of PDP elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q). The thus constructed DCPN is referred to as DCPN$^{\text{PDP}}$.
- (DCPN$^{\text{PDP}}$ execution and verification of R0–R4.) The execution of the constructed DCPN$^{\text{PDP}}$ elements provides the DCPN$^{\text{PDP}}$ stochastic process, for which we verify that DCPN rules R0–R4 hold true.
- (Pathwise equivalence.) Finally, we show that if the original PDP elements and the constructed DCPN$^{\text{PDP}}$ are both executed on the Hilbert cube $(\Omega^H, \Im^H, \mathbb{P})$, then their resulting stochastic processes are pathwise equivalent.

3.5.1 Construction of DCPN$^{\text{PDP}}$ elements

We provide a mapping that characterises DCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) in terms of PDP elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q).

$\mathcal{P} = \{P_\theta; \theta \in \mathbf{K}\}$. Hence, for each $\theta \in \mathbf{K}$, there is one place P_θ. Since $\mathbf{K}$ is assumed finite, the set of places is finite as well, which satisfies DCPN conditions.

$\mathcal{T} = \mathcal{T}_G \cup \mathcal{T}_D \cup \mathcal{T}_I$, with $\mathcal{T}_I = \emptyset$, $\mathcal{T}_G = \{T^G_\theta; \theta \in \mathbf{K}\}$, $\mathcal{T}_D = \{T^D_\theta; \theta \in \mathbf{K}\}$. Hence, for each $\theta \in \mathbf{K}$ there is one guard transition T^G_θ and one delay transition T^D_θ.

$\mathcal{A} = \mathcal{A}_O \cup \mathcal{A}_E \cup \mathcal{A}_I$, with $|\mathcal{A}_I| = 0$, $|\mathcal{A}_E| = 0$, and $|\mathcal{A}_O| = 2|\mathbf{K}| + 2|\mathbf{K}|^2$. Hence, there are no inhibitor arcs or enabling arcs in this DCPN$^{\text{PDP}}$ constructed, and the number of ordinary arcs is $2|\mathbf{K}| + 2|\mathbf{K}|^2$.

$\mathcal{N}$**:** The node function maps each arc in $\mathcal{A}$ ($= \mathcal{A}_O$) to one pair of nodes. These connected pairs of nodes are: $\{(P_\theta, T^G_\theta); \theta \in \mathbf{K}\} \cup \{(P_\theta, T^D_\theta); \theta \in \mathbf{K}\} \cup \{(T^G_\theta, P_\vartheta); \theta, \vartheta \in \mathbf{K}\} \cup \{(T^D_\theta, P_\vartheta); \theta, \vartheta \in \mathbf{K}\}$. Hence, each place P_θ ($\theta \in \mathbf{K}$) has two outgoing arcs: one to guard transition T^G_θ and one to delay transition T^D_θ. Each transition has $|\mathbf{K}|$ outgoing arcs: one arc to each place in $\mathcal{P}$.

$\mathcal{S} = \{\mathbb{R}^{d(\theta)}; \theta \in \mathbf{K}\}$.

$\mathcal{C}$**:** For all $\theta \in \mathbf{K}$, $\mathcal{C}(P_\theta) = \mathbb{R}^{d(\theta)}$.

$\mathcal{I}$**:** $\mathcal{I}(M^{\theta_0}, X_0) = 1$, where M^θ is the $|\mathcal{P}|$-dimensional vector that has a one at the element corresponding to place P_θ and zeros elsewhere. Hence, place P_{θ_0} initially gets one token with colour X_0 while all other places initially get zero tokens.

$\mathcal{V}$**:** For all $\theta \in \mathbf{K}$, $\mathcal{V}_{P_\theta}(\cdot) = f(\theta, \cdot)$. Since it was assumed that for each θ and initial value x_0, $dx_t = f(\theta, x_t)dt$ has a unique solution, the constructed $\mathcal{V}_{P_\theta}$ satisfies DCPN conditions.

$\mathcal{G}$**:** For all $\theta \in \mathbf{K}$, $\mathcal{G}_{T_\theta^G} = E_\theta$.

$\mathcal{D}$**:** For all $\theta \in \mathbf{K}$, $\mathcal{D}_{T_\theta^D}(\cdot) = \lambda(\theta, \cdot)$.

$\mathcal{F}$**:** For a given transition, let e' be the vector of length $|\mathbf{K}|$ containing a one at the component corresponding with the arc to place $P_{\vartheta'}$ and zeros elsewhere. Then for all $\theta \in \mathbf{K}$, $\mathcal{F}_{T_\theta^G}(e', x'; x) = \mathcal{F}_{T_\theta^D}(e', x'; x) = Q(\vartheta', x'; \theta, x)$, for all $x \in E_\theta \cup \partial E_\theta$, $\vartheta' \in \mathbf{K}$ and $x' \in E_{\vartheta'}$. Here, for $T \in \{T_\theta^G, T_\theta^D\}$, $\mathcal{F}_T(e', x'; x)$ denotes the probability that transition T produces one token for place $P_{\vartheta'}$ with colour x', if it removes a token from P_θ which has colour x.

Figure 3.3 shows part of the graphical representation of the constructed DCPN$^{\text{PDP}}$, in which for simplicity all transitions except two have been omitted.

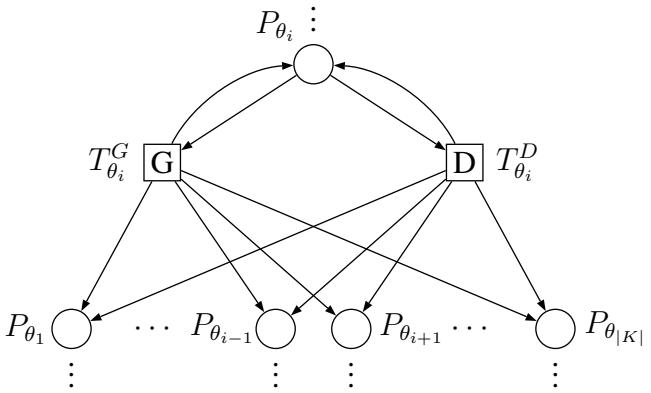


Figure 3.3 Part of graphical representation of a dynamically coloured Petri net representing a piecewise deterministic Markov process

Remark 3.2. *Notice that the mapping constructed above is one-to-one, i.e., if we have two different PDPs, e.g., PDP$_1$ and PDP$_2$, then the resulting DCPN^{PDP_1} and DCPN^{PDP_2} are also different. Here, we define PDP$_1$ and PDP$_2$ to be different, if at least one of their elements are different, or if they are defined on a different probability space. A similar definition holds for DCPN^{PDP_1} and DCPN^{PDP_2} being different.*

3.5.2 DCPN$^{\text{PDP}}$ execution

Following Section 3.3.3, the DCPN$^{\text{PDP}}$ execution uses a sequence $\{U_i; i = 0, 1, \ldots\}$ of independent uniform $U[0, 1]$ random variables.

Initiation The probability measure $\mathcal{I}$ constructed in Section 3.5.1 generates the initial marking at time $\tau_0 = 0$, which is characterised by $(M_0, C_0) = \mathcal{I}^{qf}(U_0)$, where $\mathcal{I}^{qf} : [0, 1] \to \mathbb{N}^{\mathcal{P}} \times \mathcal{C}(\mathcal{P})^{\mathbb{N}}$ is the quantile function of $\mathcal{I}$. By construction of $\mathcal{I}$, with probability one, $(M_0, C_0) = (M^{\theta_0}, X_0)$, where $M_0 = M^{\theta_0}$ is a unit vector of length $|\mathcal{P}| = |\mathbf{K}|$ which has a one at the element corresponding with place P_{θ_0}. Since with this, there is only one token, the entire colour vector C_0 is assigned to this single token in place P_{θ_0}, with $C_0 = X_0 \in \mathcal{C}(P_{\theta_0}) = \mathbb{R}^{d(\theta_0)}$.

Token colour evolution From the initiation $C_0^{P_{\theta_0}} = C_0$ onwards, the colour $C_t^{P_{\theta_0}}$ of the token in place P_{θ_0} evolves according to the corresponding token colour function, i.e., $dC_t^{P_{\theta_0}} = \mathcal{V}_{P_{\theta_0}}(C_t^{P_{\theta_0}})dt$, which has unique solution $C_t^{P_{\theta_0}} = C_0^{P_{\theta_0}} + \int_{\tau_0}^{t} \mathcal{V}_{P_{\theta_0}}(C_s^{P_{\theta_0}})ds$.

Transition enabling By construction, two transitions are pre-enabled, i.e., one guard transition $T_{\theta_0}^G$ and one delay transition $T_{\theta_0}^D$. These transitions compete for the token, which resides in their common input place P_{θ_0}. $T_{\theta_0}^G$ is enabled when $C_t^{P_{\theta_0}} \in \partial\mathcal{G}_{T_{\theta_0}^G}$; $T_{\theta_0}^D$ is enabled when $t = \tau_0 + \sigma_1^{T_{\theta_0}^D}$, where $\sigma_1^{T_{\theta_0}^D} = D_{T_{\theta_0}^D}^{qf}(U_1)$, in which $D_{T_{\theta_0}^D}^{qf}$ denotes the quantile function of $D_{T_{\theta_0}^D}(t - \tau_0) = 1 - \exp(-\int_{\tau_0}^{t} \mathcal{D}_{T_{\theta_0}^D}(C_s^{P_{\theta_0}})ds)$.

Transition firing The first transition that is enabled removes the token from P_{θ_0}. This happens at time τ_1, which is the minimum of $\inf\{t \mid C_t^{P_{\theta_0}} \in \partial\mathcal{G}_{T_{\theta_0}^G}\}$ and $\tau_0 + \sigma_1^{T_{\theta_0}^D}$. The firing measure ($\mathcal{F}_{T_\theta^G}$ or $\mathcal{F}_{T_\theta^D}$) of the enabled transition is used to determine the colour and place of a new token at time τ_1. It uses the colour of the input token, which is $C_{\tau_1}^{P_{\theta_0}}$, to generate a unit vector M_{τ_1} that denotes which output place gets a token, and a vector C_{τ_1} that denotes the colour of this token. If transition T fires ($T = T_{\theta_0}^G$ or $T = T_{\theta_0}^D$), then $(M_{\tau_1}, C_{\tau_1}) = \mathcal{F}_T^{qf}(U_2, C_{\tau_1}^{P_{\theta_0}})$. By construction, only one token is produced, and the possible places for this single token are $\{P_\theta; \theta \in \mathbf{K}\}$.

Execution after first transition firing After this, the execution is in the same way from the new state (M_{τ_1}, C_{τ_1}) on. Because at all times each transition firing removes one token and produces one token, the number of tokens does not change for $t > 0$. Hence, for $t \geq 0$ there is one token and the possible places for this single token are $\{P_\theta; \theta \in \mathbf{K}\}$.

Between times when a transition fires, i.e., during intervals (τ_{k-1}, τ_k), the DCPN$^{\text{PDP}}$ marking (M_t, C_t) is piecewise continuous, with M_t a constant unit vector and C_t continuous and equal to

the colour of the single token in the DCPN$^{\text{PDP}}$. At times τ_k ($k = 1, 2, \ldots$), the marking jumps. The time of jump is determined by one pre-enabled guard transition and one pre-enabled delay transition, which compete for their common input token. The delay transition uses one uniform random variable U_{2k-1} to determine when it is enabled. The marking (M_{τ_k}, C_{τ_k}) at τ_k equals the token distribution and token colour right after the jump. This marking is determined by the firing measure corresponding with the transition that is enabled. This firing measure uses one uniform random variable U_{2k} to characterise (M_{τ_k}, C_{τ_k}).

Verification of R0–R4 Next, we verify if DCPN rules R0–R4 hold true in the construction above. Since there are no immediate transitions in the constructed DCPN$^{\text{PDP}}$ instantiation, rule R0 holds true. Since there is only one token in the constructed DCPN$^{\text{PDP}}$ instantiation, R1–R3 also hold true. Rule R4 is in effect when for particular θ, transitions T_θ^G and T_θ^D become enabled at exactly the same time. Since $\mathcal{D}_{T_\theta^D}$ is integrable, the probability that this occurs is zero, yielding that R4 holds with probability one. However, if this event should occur, then due to $\mathcal{F}_{T_\theta^G} = \mathcal{F}_{T_\theta^D}$, the application of rule R4 has no effect on the path of the DCPN process.

3.5.3 Pathwise equivalence

Finally, we show that if the DCPN$^{\text{PDP}}$ is executed on the Hilbert cube $(\Omega^H, \Im^H, \mathbb{P})$, then the DCPN$^{\text{PDP}}$ stochastic process is pathwise equivalent to the original PDP. This is done by showing:

- Equivalence of initial states
- Equivalence of continuous evolution
- Equivalence of time of jumps
- Equivalence of size of jumps

Effectively, the execution on the Hilbert cube means that for a given $\omega = (\omega_0, \omega_1, \ldots) \in \Omega^H$, we can use that the sequence of uniform random variables $\{U_i; i = 0, 1, \ldots\}$ used to execute the DCPN$^{\text{PDP}}$, is provided by $U_i = U_i(\omega) = \omega_i$ ($i = 0, 1, 2 \ldots$).

Equivalence of initial states The initial marking of the DCPN$^{\text{PDP}}$ is mapped to the PDP initial state, i.e., $\mathcal{I}(M^\theta, X_0) = 1$, where M^θ is the $|\mathcal{P}|$-dimensional vector that has a one at the element corresponding to place P_θ and zeros elsewhere. If $\mathcal{I}^{qf}$ denotes the quantile function of $\mathcal{I}$, then the random variable $(M_0(\omega), C_0(\omega)) = \mathcal{I}^{qf}(U_0(\omega)) = (M^{\theta_0}(\omega), X_0(\omega))$ is equivalent to the random variable $(\theta_0(\omega), X_0(\omega))$, i.e., the initial states are indistinguishable.

Equivalence of continuous evolution The continuous part of the DCPN$^{\text{PDP}}$ stochastic process equals the vector that collects all token colours. Since there is only one token in the constructed DCPN$^{\text{PDP}}$ at all times, this vector equals the colour of this single token. Until the first jump, this colour follows the ordinary differential equation $dC_t^{P_{\theta_0}}(\omega) = \mathcal{V}_{P_{\theta_0}}(C_t^{P_{\theta_0}}(\omega))dt$ which has unique solution $C_t^{P_{\theta_0}}(\omega) = C_0^{P_{\theta_0}}(\omega) + \int_{\tau_0}^t \mathcal{V}_{P_{\theta_0}}(C_s^{P_{\theta_0}}(\omega))ds$. In the original PDP, the continuous process follows ordinary differential equation $dX_t(\omega) = f(\theta_0(\omega), X_t(\omega))dt$. Due to equivalence of initial states $M^{\theta_0} \equiv \theta_0$ and $C_0^{P_{\theta_0}} = C_0 = X_0$ and equivalence of drift coefficients $\mathcal{V}_{P_{\theta_0}}(\cdot) = f(\theta_0, \cdot)$, this PDP continuous state has the same solution $X_t(\omega) = X_0(\omega) + \int_{\tau_0}^t f(\theta_0(\omega), X_s(\omega))ds = \phi_{\theta_0, X_0, t-\tau_0}(\omega)$, i.e., for $t \geq \tau_0$ and for arbitrary $\omega \in \Omega^H$, $X_t(\omega) = C_t^{P_{\theta_0}}(\omega)$.

Equivalence of time of jumps The survivor function of the initial token in the DCPN$^{\text{PDP}}$ is the complementary probability distribution function for the time until it is removed from its place by an enabled transition. For each arbitrary place in which the initial token may reside, two transitions are pre-enabled: a guard transition and a delay transition. If $t_*^G(\cdot)$ denotes the time until the guard transition is enabled, i.e. $t_*^G(M^{\theta_0}, C_0) \triangleq \inf\{t - \tau_0 > 0 \mid C_t^{P_{\theta_0}} \in \partial\mathcal{G}_{T_{\theta_0}^G}\}$, then the survivor function of the initial token is:

$$\Upsilon_{P_{\theta_0}, C_0, t-\tau_0} \triangleq \begin{cases} 0 & \text{if } t - \tau_0 \geq t_*^G(M^{\theta_0}, C_0) \\ \exp\left(-\int_{\tau_0}^t \mathcal{D}_{T_{\theta_0}^D}(C_s^{P_{\theta_0}})ds\right) & \text{otherwise} \end{cases} \tag{3.5.2}$$

For PDP, the survivor function is given by Equation (3.4.1).

Due to $X_t(\omega) = C_t^{P_{\theta_0}}(\omega)$ for $t \geq \tau_0$, and due to the mapping $E_{\theta_0} = \mathcal{G}_{T_{\theta_0}^G}$, the PDP equivalent boundary hit event is $\phi_{\theta_0, X_0, t-\tau_0} \in \partial E_{\theta_0}$, where $X_0 = C_0$ and $\theta_0 \equiv M^{\theta_0}$. Due to $C_t^{P_{\theta_0}} = \phi_{\theta_0, X_0, t-\tau_0}$, we find that $t_*(\theta_0, X_0) = t_*^G(M^{\theta_0}, C_0)$. Due to mapping $\mathcal{D}_{T_{\theta_0}^D}(\cdot) = \lambda(\theta_0, \cdot)$, the PDP equivalent event is when a spontaneous jump is according to jump rate $\lambda(\theta_0, \cdot)$. Hence, the survivor functions of the PDP and the DCPN$^{\text{PDP}}$ are the same. In particular, $\Upsilon^{qf}(u, P_{\theta_0}, C_0, \tau_0) = \Gamma^{qf}(u, \theta_0, X_0, \tau_0)$, hence since the same uniform random variable $U_1(\omega) = \omega_1$ is used to evaluate both survivor functions, then the time of firing of the first DCPN$^{\text{PDP}}$ transition is equal to the time of the first jump of the original PDP: $\tau_1(\omega) = \tau_0 + \sigma_1(\omega)$, with $\sigma_1(\omega) = \Upsilon^{qf}(U_1(\omega), P_{\theta_0}, C_0, \tau_0) = \Gamma^{qf}(U_1(\omega), \theta_0, X_0, \tau_0)$.

Equivalence of size of jumps This leaves showing equivalence between the size of jumps. For the DCPN$^{\text{PDP}}$, this is determined by the firing measure of the enabled transition, which uses a uniform random variable $U_2(\omega)$. Due to the mapping $\mathcal{F}_{T_\theta^G}(e', x'; x) = \mathcal{F}_{T_\theta^D}(e', x'; x) = Q(\vartheta', x'; \theta, x)$, the PDP equivalent event is that if a random vector $\zeta_1(\omega)$ is characterised from $Q(\cdot; \theta_0, \phi_{\theta_0, X_0, \tau_1 - \tau_0})$ using the same uniform random variable $U_2(\omega)$, then $\zeta_1(\omega) = (\theta_j(\omega), X_{\tau_1}(\omega))$ if $\mathcal{F}$ generates a token for place P_{θ_j} with colour $C_{\tau_1}(\omega) = X_{\tau_1}(\omega)$. This means, the DCPN$^{\text{PDP}}$

state after the jump $(M^{\theta_j}(\omega), C_{\tau_1}(\omega))$ and the PDP state after the jump $(\theta_j(\omega), X_{\tau_1}(\omega))$ are indistinguishable.

From τ_1 onwards, the pathwise equivalence of the PDP and DCPN$^{\text{PDP}}$ processes is shown in the same way. Since the PDP and the DCPN$^{\text{PDP}}$ are both defined on the Hilbert cube, from stopping time τ_{k-1} to stopping time τ_k both processes use the same independent realisations of the random variables $U_{2k-1}(\omega), U_{2k}(\omega), \ldots$, each having uniform $[0,1]$ distribution to generate all random variables in both the PDP process and the DCPN$^{\text{PDP}}$ process. Hence, from stopping time to stopping time, the PDP and the associated DCPN$^{\text{PDP}}$ process have pathwise equivalent (i.e., indistinguishable) paths and pathwise equivalent stopping times. Due to the unique definition of the DCPN$^{\text{PDP}}$ stochastic process at times when transitions fire, the DCPN$^{\text{PDP}}$ state at stopping times is also equivalent to the PDP state at the stopping times and both processes are càdlàg.

This completes the proof of Theorem 3.1. □

3.6 Dynamically coloured Petri nets into piecewise deterministic Markov processes

This section shows that under a few conditions, each dynamically coloured Petri net (DCPN) can be represented by a piecewise deterministic Markov process (PDP), by providing an into-mapping[10] from DCPN into the set of PDPs.

Theorem 3.2 (DCPN into PDP)**.** *Consider an arbitrary DCPN* $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ *satisfying R0–R4. If the initial marking is deterministic*[11]*, if the initial marking does not enable a transition, if none of the transition firings enable a guard transition, and if the number of tokens remains finite for* $t \to \infty$*, then the elements of this DCPN can be mapped into the elements* $(\mathbf{K}, d, E, f, \theta_0, X_0, \lambda, Q)$ *of a PDP. If the original DCPN is executed on a probability space, then the PDP defined by the execution of the resulting PDP elements is probabilistically equivalent to the stochastic process defined by the execution of the original DCPN. If in addition, conditions D1–D3 below are satisfied, then PDP conditions P1–P4 hold true for the resulting PDP.*

D1 *For all* $P \in \mathcal{P}$ *for which* $\mathcal{C}(P) \neq \mathbb{R}^0$*,* $\mathcal{V}_P$ *is locally Lipschitz continuous. Moreover, there are* $\mathbb{P}$*-almost surely no explosions, i.e., if* t' *denotes the time at which any component of a token colour equals* $+\infty$ *or* $-\infty$*, and* t'' *denotes the time until the first guard transition is enabled, then* $t' \to \infty$ *whenever* $t'' \to \infty$*.*

[10] An *into-mapping* is defined as a mapping $\varphi : A \to B$ such that $R_\varphi \subset B$, where R_φ is the range of φ, i.e., $R_\varphi = \{b \in B \mid \exists a : \varphi(a) = b\}$.

[11] See also remark 3.4 on page 58.

D2 *After a transition firing (or after a sequence of firings that occur at the same time instant) at least one place must contain a different number of tokens, or the colour of at least one token must have jumped ($\mathbb{P}$-almost surely).*

D3 *In a finite time interval, each transition is expected to fire a finite number of times.*

Proof. Consider an arbitrary DCPN $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ that satisfies rules R0–R4. Next, we prove Theorem 3.2 by the following steps:

- (Construction of PDP elements.) First, we assume that the initial marking is deterministic, that the initial marking does not enable a transition, that none of the transition firings enable a guard transition, and that the number of tokens remains finite for $t \to \infty$. We characterise PDP elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q) in terms of DCPN elements. The thus constructed PDP elements are referred to as PDP$^{\text{DCPN}}$ elements.

- (Probabilistic equivalence.) The execution of the PDP$^{\text{DCPN}}$ elements defines a sample path of the PDP$^{\text{DCPN}}$. We show that the PDP$^{\text{DCPN}}$ is probabilistically equivalent to the stochastic process defined by the execution of the original DCPN.

- (Verification of P1–P4.) Finally, we show that if additionally, conditions D1–D3 are satisfied, then PDP conditions P1–P4 hold true in the constructed PDP$^{\text{DCPN}}$.

With this mapping, the constructed PDP$^{\text{DCPN}}$ discrete state θ_t will be a vector of length $|\mathcal{P}|$ that counts the numbers of tokens in each place at time t. The PDP$^{\text{DCPN}}$ continuous state X_t will be formed by a vector that contains the colours of all tokens in the DCPN at time t. This X_t evolves through time according to the respective token colour functions; the PDP$^{\text{DCPN}}$ jumps correspond with DCPN transitions firing, which may change the distribution of tokens among places (θ_t) and the colours of the tokens (X_t).

Note that Section 3.7 discusses all conditions for Theorem 3.2.

3.6.1 Construction of PDP$^{\text{DCPN}}$ elements

We assume that the initial marking is deterministic, that the initial marking does not enable a transition, that none of the transition firings enable a guard transition,and that the number of tokens remains finite for $t \to \infty$. We provide an into-mapping that characterises PDP elements ($\mathbf{K}$, d, E, f, θ_0, X_0, λ, Q) in terms of DCPN elements $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$.

K: The discrete state space $\mathbf{K}$ is constructed from the reachability graph (RG) of the DCPN graph, which is constructed from DCPN elements $\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{I}$. The nodes in the RG represent all possible distributions of tokens among places; they are written as vectors

$m = (m_1, \ldots, m_{|\mathcal{P}|})$, where m_i denotes the number of tokens in place P_i, $i = 1, \ldots, |\mathcal{P}|$, where these places are uniquely ordered. Under condition that the number of tokens remains finite, this graph is finite. It is constructed as follows:

The first nodes are found from the initial marking $\mathcal{I}$: Each token distribution m for which $\mathcal{I}(m, \cdot) > 0$, is represented by a node in the RG. Since $\mathcal{I}$ provides finite numbers of tokens, the number of initial nodes in the RG is finite. Under the condition that the initial marking is deterministic, the number of initial nodes in the RG is equal to one. From then on, the RG is constructed as follows: For each existing RG node, it is determined which transitions are pre-enabled in this token distribution, and which combinations of transitions may fire, using rules R0–R4. If it is possible to move in one jump from token distribution m to, say, either one of distributions $m^1, \ldots, m^k$, then arrows are drawn from m to (new or already existing) nodes $m^1, \ldots, m^k$. Each of $m^1, \ldots, m^k$ is treated in the same way. Each arrow is labelled by the transition(s) fired at the jump. If m^j can be reached from m^i by the simultaneous firing of several transitions, e.g., T_1 and T_2, then the label lists the transitions coupled by $+$ signs, e.g., $T_1 + T_2$. If a node m^j can be directly reached from m^i by different (sets of) transitions firing, then multiple arrows are drawn from m^i to m^j, each labelled by another (set) of transition(s). Multiple arrows are also drawn if m^j can be directly reached from m^i by firing of one transition, but by different sets of tokens, for example in case this transition has multiple input tokens per incoming arc in its input places. In this case, the multiple arrows each get this transition as label.

The nodes in the resulting reachability graph *excluding* the nodes from which an immediate transition is enabled, form the discrete state space $\mathbf{K}$. The nodes from which an immediate transition is enabled are called *vanishing nodes*. Since the number of places in the DCPN is finite and the number of tokens per place and the number of nodes in the RG are finite, $\mathbf{K}$ is a countable (even finite) set, which satisfies the PDP conditions.

Since each non-vanishing node $m \in$ RG corresponds uniquely with one $\theta \in \mathbf{K}$, sometimes, by abuse of notation, we refer to θ as a token distribution.

Note that the RG can also be used to determine which transitions are pre-enabled in a particular token distribution m: These are the transitions on the labels of all arrows leaving m in the RG. Here, the set of pre-enabled transitions is referred to as a multiset since the set of labels may contain one transition multiple times.

d**:** For each $\theta \in \mathbf{K}$, corresponding with node $m = (m_1, \ldots, m_{|\mathcal{P}|})$ in the RG, $d(\theta) = \sum_{i=1}^{|\mathcal{P}|} m_i n(P_i)$, where $n(P_i)$ is defined through $\mathcal{C}(P_i) = \mathbb{R}^{n(P_i)}$.

E**:** For each $\theta \in \mathbf{K}$, E_θ is an open subset of $\mathbb{R}^{d(\theta)}$. Due to the characterisation of d in terms of DCPN elements above, this $\mathbb{R}^{d(\theta)}$ is uniquely characterised. The open subset E_θ is

constructed from transition guards as follows: If under token distribution θ, no guard transitions are pre-enabled, then $E_\theta = \mathbb{R}^{d(\theta)}$. If under token distribution θ, one or more guard transitions are pre-enabled, then $E_\theta = \mathbb{R}^{d(\theta)} \setminus \partial E_\theta$, where ∂E_θ is constructed as follows:

Without loss of generality, suppose that under token distribution θ, the multi-set of pre-enabled guard transitions is $T_1, \ldots, T_k$. This set may contain one transition multiple times, if such transition evaluates multiple input token vectors in parallel. Suppose $\{P_{i_1}, \ldots, P_{i_{r_i}}\} = P(A_{in,OE}(T_i))$ are the input places of T_i that are connected to T_i by means of ordinary or enabling arcs. This set may contain one place multiple times if such place is connected to T_i by multiple arcs (input arcs of T_i). Define $d_i = \sum_{j=1}^{r_i} n(P_{i_j})$, then $\partial E_\theta = \partial \mathcal{G}'_{T_1} \cup \ldots \cup \partial \mathcal{G}'_{T_k}$, where $\mathcal{G}'_{T_i} = [[\mathcal{G}_{T_i} \times \mathbb{R}^{d(\theta)-d_i}]] \in \mathbb{R}^{d(\theta)}$. Here $[[\cdot]]$ denotes a special ordering of all vector elements: Vector elements are ordered according to the unique ordering of places and to the unique ordering of tokens within their place defined for DCPN (see Section 3.3). Finally, $E = \{\{\theta\} \times E_\theta; \theta \in \mathbf{K}\}$.

f: For each $\theta \in \mathbb{M}$ and $x \in E_\theta$, $f(\theta, x) = \text{Col}_{i=1}^{|\mathcal{P}|}\left\{\text{Col}_{j=1}^{m_i}\{\mathcal{V}_{P_i}(c_{ij})\}\right\}$, where $x = \text{Col}_{i=1}^{|\mathcal{P}|}\left\{\text{Col}_{j=1}^{m_i}\{c_{ij}\}\right\}$ and θ corresponds to $(m_1, \ldots, m_{|\mathcal{P}|})$. Here $\text{Col}_{j=1}^{0}\{\cdot\} \triangleq \emptyset$ and if $\mathcal{C}(P_i) = \mathbb{R}^0$ for particular P_i, then $\mathcal{V}_{P_i} = \emptyset$.

θ_0, X_0: $\theta_0 = M_0$ and $X_0 = C_0$, where due to the condition that the DCPN initial marking is deterministic, (M_0, C_0) is according to $\mathcal{I}(M_0, C_0) = 1$. Due to the assumption that in the initial marking, no immediate or guard transition is enabled, we find that the constructed θ_0 and X_0 are uniquely defined and that $\theta_0 \in \mathbf{K}$ and $X_0 \in E_{\theta_0}$.

λ: For each $\theta \in \mathbf{K}$ and $x \in E_\theta$, $\lambda(\theta, x) = \sum_{n=1}^{k} \mathcal{D}_{T_n}(c^{T_n})$, where, without loss of generality, $T_1, \ldots, T_k$ refers to the multi-set of transitions in $\mathcal{T}_D$ that, under token distribution θ, are pre-enabled and c^{T_n} are the respective elements of x that are used to pre-enable these transitions. This set $T_1, \ldots, T_k$ may contain one transition multiple times, if multiple input token vectors are evaluated in parallel. If the set of pre-enabled delay transitions is empty in θ, then $\lambda(\theta, \cdot) = 0$.

Q: For each $\theta \in \mathbf{K}$, $x \in E_\theta$, $\theta' \in \mathbf{K}$ and $x' \in E_{\theta'}$, $Q(\theta', x'; \theta, x)$ is characterised by the reachability graph, the sets $\mathcal{D}$, $\mathcal{G}$ and $\mathcal{F}$ and the rules R0–R4. The reachability graph is used to determine which transitions are pre-enabled in token distribution θ; the sets $\mathcal{D}$ and $\mathcal{G}$ and the rules R0–R4 are used to determine which pre-enabled transitions will actually fire from state (θ, x); and finally, set $\mathcal{F}$ is used to determine the probability of (θ', x') being the state after the jump, given the state (θ, x) before the jump and the set of transitions that will fire in the jump. Because of its length, the characterisation of Q is moved to an appendix, Section 3.9.

This shows that all PDP elements can be characterised uniquely in terms of DCPN elements.

Remark 3.3. *Notice that the mapping constructed above is an into-mapping; more specifically, it is* not *one-to-one. To see this, notice that the existence of immediate transitions is not visible in PDP: We can take a DCPN$_1$, and construct a DCPN$_2$ which is a copy of DCPN$_1$ with the addition of a sequence of immediate transitions which is a self-loop, i.e., the sequence brings the state right after a spontaneous or forced jump back to where it was right after this spontaneous or forced jump. Such sequence will not be visible in the mapped PDP^{DCPN_2} hence we will conclude that PDP^{DCPN_1} = PDP^{DCPN_2} while DCPN$_1 \neq$ DCPN$_2$.*

Remark 3.4. *In Theorem 3.2 we assumed that the DCPN initial marking is deterministic. This was done to be in line with PDP, which considers one initial state (θ_0, X_0) only. Alternatively, we could have assumed that the PDP initial state is according to a given probability measure μ_{θ_0,X_0}, and could have provided a mapping of μ_{θ_0,X_0} in terms of the DCPN initial marking $\mathcal{I}$, i.e., $\mu_{\theta_0,X_0}(\theta_0, X_0) = \mathcal{I}(\theta_0, X_0)$. Since this is not according to the PDP definition in [Dav93], this assumption is not adopted here.*

3.6.2 Probabilistic equivalence

We show that the execution of the constructed PDP$^{\text{DCPN}}$ delivers a stochastic process which is probabilistically equivalent to the process defined by the original DCPN executed on a probability space. This is done by showing

- Equivalence of initial states
- Equivalence of continuous evolution
- Equivalence of times of jumps
- Equivalence of size of jumps

Equivalence of initial states The initial state is (θ_0, X_0) which is formed by the initial marking of the DCPN. If $\mathcal{I}^{qf}$ denotes the quantile function of $\mathcal{I}$ then $(\theta_0, X_0) = (M_0, C_0) = \mathcal{I}^{qf}(U_0)$, with U_0 a uniform random variable from $U[0, 1]$. This means, the initial states are probabilistically equivalent.

Equivalence of continuous evolution From $t \geq \tau_0$, the PDP$^{\text{DCPN}}$ continuous state X_t is according to the ordinary differential equation $dX_t = f(\theta_0, X_t)dt$ until the first jump occurs, where f is formed by the token colour functions $\mathcal{V}_P$ corresponding to the places in which the DCPN tokens reside. This equation has a unique solution $X_t = X_0 + \int_{\tau_0}^{t} f(\theta_0, X_s)ds = \phi_{\theta_0,X_0,t-\tau_0}$.

For $t > \tau_0$, up until the first jump, the PDP$^{\text{DCPN}}$ continuous state is pathwise equivalent to the DCPN continuous state: At times t when no jump occurs, the PDP$^{\text{DCPN}}$ evolves according to f and

the DCPN process evolves according to $\mathcal{V} = \{\mathcal{V}_P; P \in \mathcal{P}\}$. Through the mapping between f and $\mathcal{V}$ developed above, these evolutions provide pathwise equivalent processes, i.e., for all $t > \tau_0$, until the first jump, $X_t(\omega) = C_t(\omega)$ for arbitrary ω.

Equivalence of time of jumps The PDP$^{\text{DCPN}}$ survivor function is characterised by the complementary probability distribution function for the time until the first transition is enabled. If $T_1^G, \ldots, T_k^G$ are the pre-enabled guard transitions and $T_1^D, \ldots, T_\ell^D$ are the pre-enabled delay transitions (where $k = 0$ or $\ell = 0$ may denote these sets to be empty) then ∂E_{θ_0} is constructed from the transition guards of $T_1^G, \ldots, T_k^G$ and $\lambda(\theta_0, \phi_{\theta_0, X_0, t-\tau_0})$ is constructed from the transition delays of $T_1^D, \ldots, T_\ell^D$ and the survivor function is defined by

$$\Gamma_{\theta_0, X_0, t-\tau_0} \triangleq \begin{cases} 0 & \text{if } t - \tau_0 \geq t_*(\theta_0, X_0) \\ \exp\left(-\int_{\tau_0}^t \lambda(\theta_0, \phi_{\theta_0, X_0, s-\tau_0}) ds\right) & \text{otherwise} \end{cases} \tag{3.6.3}$$

where $t_*(\theta_0, X_0) \triangleq \inf\{t - \tau_0 > 0 \mid \phi_{\theta_0, X_0, t-\tau_0} \in \partial E_{\theta_0}\}$.

In DCPN, the forced jumps are represented by guard transitions; in PDP, the forced jumps are represented by continuous state space boundary hits. Due to the into-mapping between the boundary of the PDP$^{\text{DCPN}}$ state space ∂E_θ and the boundaries of the transition guards of the guard transitions $\{\partial \mathcal{G}_T; T \in \mathcal{T}_G\}$ and due to equivalence of continuous states $X_t = C_t$, the PDP$^{\text{DCPN}}$ forced jumps and the DCPN forced jumps occur at the same time. The PDP$^{\text{DCPN}}$ spontaneous jumps are according to a survivor function that uses a rate λ. The DCPN spontaneous jumps are according to the delay transition rates $\{\mathcal{D}_T; T \in \mathcal{T}_D\}$. Due to the into-mapping between λ and $\{\mathcal{D}_T; T \in \mathcal{T}_D\}$, the time of spontaneous jump is according to the same rate for both PDP$^{\text{DCPN}}$ and DCPN. However, in this case there is no equivalence to indistinguishability. The reason is that the PDP$^{\text{DCPN}}$ uses precisely one uniform random variable to generate the time of spontaneous jump, whereas the delay transitions use one uniform random variable each (one per pre-enabled delay transition). For this reason (except under particular circumstances), equivalence is in probability only.

Equivalence of size of jumps The PDP$^{\text{DCPN}}$ state after the jump is characterised by Q, i.e., $(\theta_{\tau_1}, X_{\tau_1}) = Q^{qf}(U_2, \theta_0, \phi_{\theta_0, X_0, \tau_1 - \tau_0})$, where Q is constructed in terms of DCPN elements. At times when a jump occurs, the PDP$^{\text{DCPN}}$ makes a jump according to Q, while the DCPN process makes a jump according to $\mathcal{F}$. Through the mapping between Q and $\mathcal{F}$, these jumps provide equivalent processes. However, equivalence is in probability only, since the PDP$^{\text{DCPN}}$ uses precisely one uniform random variable to determine the PDP$^{\text{DCPN}}$ state value after the jump, whereas the DCPN may use several uniform random variables: one for each transition that fires at the jump.

Concluding, the sample paths of the PDP$^{\text{DCPN}}$ and the DCPN are equivalent in probability, but

in general not up to indistinguishability. The explanation is that the initial states, the continuous evolution mechanism and the jump mechanism are equivalent, but the mapping above does not include a one-to-one use of $U[0,1]$ random variables: The PDP$^{\text{DCPN}}$ uses two $U[0,1]$ variables for each jump: one variable to generate the time of jump, and one variable to generate the size of jump. The DCPN uses $U[0,1]$ variables to determine the time of firing of delay transitions, and to determine the tokens produced (number and colour) at the firing of any transition. The number of variables is not always equal to two for each jump:

- In DCPN, multiple delay transitions may be pre-enabled simultaneously. They each use one $U[0,1]$ random variable to determine their time of enabling. However, (with probability one) only one of them fires. If the firing of one delay transition disables the other pre-enabled delay transitions, then the $U[0,1]$ random variables of the disabled delay transitions are lost. In the corresponding PDP$^{\text{DCPN}}$ situation, only one $U[0,1]$ variable is used.
- It may happen that a guard transition fires before a delay transition does, in which case a jump only uses one $U[0,1]$ sample, i.e., to determine the tokens produced at the firing of the guard transition. If the delay transitions are not disabled by the firing of the guard transition, then the $U[0,1]$ they used to determine their time of enabling are maintained for the next jump.
- A sequence of immediate transitions firing at the same time instant (which counts as one PDP$^{\text{DCPN}}$ jump) will use a sequence of $U[0,1]$ random variables, i.e., to determine the tokens produced by the firing of each immediate transition.

3.6.3 Verification of P1–P4

Finally, we show that if additionally, D1–D3 are satisfied, then PDP conditions P1–P4 hold true in the constructed PDP$^{\text{DCPN}}$.

P1: This condition (local Lipschitz continuity and no explosions) follows from condition D1: since under D1, for all $P \in \mathcal{P}$ for which $\mathcal{C}(P) \neq \mathbb{R}^0$, $\mathcal{V}_P$ is locally Lipschitz continuous, this also holds for $f(\theta, x) = \text{Col}_{i=1}^{|\mathcal{P}|}\left\{\text{Col}_{j=1}^{m_i}\{\mathcal{V}_{P_i}(c_{ij})\}\right\}$. Since under D1, $\mathcal{V}_P$ does not produce explosions, this also holds for $f(\theta, x)$.

P2: This condition (λ is integrable) follows from the fact that $\mathcal{D}_T$ is integrable for all $T \in \mathcal{T}_D$: since $\mathcal{D}_T$ is integrable, this also holds for $\lambda(\theta, x) = \sum_{n=1}^{k} \mathcal{D}_{T_n}(c^{T_n})$.

P3: This condition (Q measurable and $Q(\theta, x; \theta, x) = 0$) follows from the assumption that $\mathcal{F}$ is measurable and from condition D2: since $\mathcal{F}$ is measurable, this also holds for Q. Since under D2, after a transition firing at least one place must contain a different number of tokens, or the colour of at least one token must have jumped ($\mathbb{P}$-almost surely), we find $Q(\theta, x; \theta, x) = 0$.

P4: This condition ($\mathbb{E}N_t < \infty$) follows from condition D3: since each transition is expected to fire a finite number of times, we find that the number of jumps is finite, hence $\mathbb{E}N_t < \infty$.

This completes the proof of Theorem 3.2. □

3.7 Discussion of conditions of Theorem 3.2

This section discusses the conditions for Theorem 3.2 (DCPN into PDP).

3.7.1 Discussion on finite number of tokens

The first set of conditions for Theorem 3.2 is that the initial marking is deterministic, that the initial marking does not enable a transition, that none of the transition firings enable a guard transition, and that for $t \to \infty$ the number of tokens remains finite. Here, the conditions on the initial marking can be easily verified. The condition on the immediate enabling of guard transitions is discussed in Section 3.7.4. We discuss the condition on finite number of tokens next.

This condition has been introduced in Theorem 3.2 in order to guarantee that the reachability graph (RG) of the DCPN exists and is finite, i.e., the DCPN is bounded (see Subsection 2.2.2). The RG consists of nodes $m = (m_1, \ldots m_{|\mathcal{P}|})$, where m_i denotes the number of tokens in place P_i. Two nodes m and m' are connected by an arrow if it is possible to jump from m to m' by the firing of a transition or by the firing of multiple transitions in parallel at the same time. The names of these transitions are written as labels on these arrows. The RG and its nodes are used in the construction of PDP$^{\text{DCPN}}$ elements $\mathbf{K}$, d and f, and indirectly also in the construction of E, λ and Q. A finite RG makes this construction easier.

A sufficient condition for a DCPN to be bounded is if its initial marking contains a finite number of tokens (note this is satisfied due to definition of $\mathcal{I}$), and each transition, when firing, produces a number of tokens equal to the number of tokens removed by the firing. Note that for most DCPN applications that we made in practice, these sufficient conditions are satisfied (see also Chapter 5) hence there has been no real practical reason to explore options where any of m_i could grow to infinity.

From a theoretical point of view, however, we note that there are options to relax the condition of finite number of tokens. Below, we give a discussion of how this could be done, but we leave the corresponding update of Theorem 3.2 to further research.

If the restriction of bounded number of tokens is removed, then in the proof of Theorem 3.2, we get a reachability graph (RG) which has an infinite, though still countable number of nodes. Such infinite RG can be represented by a *coverability graph* (see Definition 2.12). A symbol ϖ is used in the nodes of the coverability graph to represent 'any number of tokens' in a particular

place. Since the number of places and the number of transitions in the DCPN are finite, there is a finite permutation of labelled arrows in the coverability graph, hence, with making use of the 'any number of tokens' symbol, the number of nodes in the coverability graph is finite. With this, the effects on the proof of Theorem 3.2 are the following:

$\mathbf{K}$: If the RG is infinite, the number of elements in $\mathbf{K}$ will be infinite as well. For PDP, countability is sufficient, hence an infinite $\mathbf{K}$ still satisfies PDP conditions.

d: This element was constructed as follows: for each $\theta \in \mathbf{K}$, corresponding with node $m = (m_1, \ldots, m_{|\mathcal{P}|})$ in the RG, $d(\theta) = \sum_{i=1}^{|\mathcal{P}|} m_i n(P_i)$, where $n(P_i)$ is defined through $\mathcal{C}(P_i) = \mathbb{R}^{n(P_i)}$. This yields that if some of the m_i could grow to infinity (in the coverability graph this is denoted by the ϖ symbol), then $d(\theta)$ could become infinite as well. Since d represents the number of elements in vector X_t, an infinite d is an undesired situation. However, note that as long as $m_i < \infty$ if $n(P_i) > 0$ and $m_i = \infty$ (or $m_i = \varpi$) only if $n(P_i) = 0$, then $d(\theta) < \infty$.

f: This element was constructed as follows: For each $\theta \in \mathbb{M}$ and $x \in E_\theta$, $f(\theta, x) = \text{Col}_{i=1}^{|\mathcal{P}|} \left\{ \text{Col}_{j=1}^{m_i} \{ \mathcal{V}_{P_i}(c_{ij}) \} \right\}$, where $x = \text{Col}_{i=1}^{|\mathcal{P}|} \left\{ \text{Col}_{j=1}^{m_i} \{ c_{ij} \} \right\}$ and θ corresponds to $(m_1, \ldots, m_{|\mathcal{P}|})$. Here $\text{Col}_{j=1}^{0} \{\cdot\} \triangleq \emptyset$ and if $\mathcal{C}(P_i) = \mathbb{R}^0$ for particular P_i, then $\mathcal{V}_{P_i} = \emptyset$. We find again that as long as $m_i < \infty$ if $n(P_i) > 0$ and $m_i = \infty$ (or $m_i = \varpi$) only if $n(P_i) = 0$, then f is well defined.

E, λ, Q: The reachability graph is used in the construction of these components in order to determine which transitions are pre-enabled in a particular token distribution. This can also be handled by the coverability graph.

Note that the above effects pose no restrictions on the use of the ϖ symbol in vanishing nodes.

Concluding: theoretically, the condition on finite number of tokens might be relaxed to the following: "for $t \to \infty$, the number of *coloured* tokens remains finite, i.e., tokens in places for which $\mathcal{C}(P) \neq \mathbb{R}^0$".

3.7.2 Discussion on Condition D1 (local Lipschitz and no explosions)

Both the local Lipschitz condition and the no-explosions condition are posed on PDP, hence it is natural to pose them for DCPN as well. The local Lipschitz condition holds true if for any compact $D \subset \mathcal{C}(P)$ there exists a constant L_P such that $|\mathcal{V}_P(b) - \mathcal{V}_P(a)| \leq L_P |b - a|$ for all $a, b \in D$. The no-explosions condition holds true if a linear growth condition holds true for $\mathcal{V}_P$. For example, following [Øk02, Theorem 5.2.1], there are no explosions if $|\mathcal{V}_P(c)| \leq K_P(1 + |c|)$, $c \in \mathcal{C}(P)$ for some constant K_P. For DCPN, such linear growth conditions need to be satisfied by $\mathcal{V}_P$ for all $P \in \mathcal{P}$.

3.7.3 Discussion on Condition D2 (recognisable jumps)

Condition D2 was introduced to ensure that, once the DCPN is mapped to a PDP$^{\text{DCPN}}$, PDP condition $Q(\theta, x; \theta, x) = 0$ holds true, which ensures that all jumps are recognisable. Condition D2 holds if the reachability graph of the DCPN does not contain self-loops, where a self-loop is a non-vanishing node in the graph with an arrow directly back to itself, or a non-vanishing node in the graph with a sequence of vanishing nodes, connected by arrows in one direction, and with an arrow back to the first non-vanishing node. If the reachability graph does contain a self-loop, then the firing measure(s) of the transition(s) that is (are) on the label of the self-loop arrow need to be inspected. If the self-loop contains one transition and its firing measure satisfies $\mathcal{F}_T(\cdot, c; c) = 0$ then D2 holds true. If $\mathcal{F}_T(\cdot, c; c) > 0$ for any reachable c then D2 does not hold true. If a self-loop contains a sequence of transitions, D2 holds true if the consecutive use of their firing measures cannot produce a token of a colour equal to that of the token removed by the first transition in the sequence.

3.7.4 Discussion on Condition D3 (finite number of firings)

Under condition D3, in a finite time interval, each transition is expected to fire a finite number of times. This condition on DCPN has been introduced in Theorem 3.2, in order to guarantee that when the DCPN is mapped to a PDP, condition P4 is satisfied for the resulting PDP$^{\text{DCPN}}$, i.e., the resulting PDP$^{\text{DCPN}}$ makes a finite number of jumps in finite time. More formally: with $N_t = \sum_k \mathbf{1}_{\{t \geq \tau_k\}}$, for every starting point $(\theta, x) \in E$ and for all $t \in \mathbb{R}^+$, $\mathbb{E}N_t < \infty$.

As indicated in Remark 3.1, [Dav93, Page 60] gives sufficient conditions under which P4 is satisfied for PDP. Below, we translate these to sufficient conditions under which D3 is satisfied for DCPN. First we recall the sufficient conditions for P4: P4 is satisfied if (1) below is satisfied, as well as (2a) or (2b):

(1) λ is bounded.

(2a) Define $t_*(\theta, x) \triangleq \inf\{t - \tau > 0 \mid \phi_{\theta,x,t-\tau} \in \partial E_\theta\}$, i.e., $t_*(\theta, x)$ is the time until the flow hits the boundary of its state space, if the flow starts in (θ, x) at time τ. Then $\forall(\theta, x) \in E$, $t_*(\theta, x) = \infty$.

(2b) For some $\chi > 0$ and with $t_*(\theta, x)$ as above, define $A_\chi = \{(\theta, x) \in E \mid t_*(\theta, x) \geq \chi\}$. Then $Q(A_\chi; \theta, x) = 1$ for all $(\theta, x) \in \partial E$.

Sufficient condition (1) ensures that there will not be an infinite number of spontaneous jumps in finite time; (2a) ensures that there are no forced jumps; (2b) ensures that the process always jumps to a state from which it takes some time $\chi > 0$ to reach the boundary.

Before we can translate this to sufficient conditions for D3, note that

- An immediate transition in a DCPN can be regarded as a delay transition with an infinite jump rate (intensity): $\mathcal{D}_T = \infty$.
- One way to put some 'distance' between forced jumps in a DCPN, is if after a guard transition has fired, only delay transitions are pre-enabled.
- The previous idea can be extended by noting that a finite number of jumps is still guaranteed if there are at most two forced jumps before a delay transition is fired, or three . . ..

Using these notes, we are now prepared to translate (1)-(2a)-(2b) above: Condition D3 is satisfied if (1') below is satisfied, as well as (2a') or (2b') or (2c'):

(1') For all $T \in \mathcal{T}_D$, $\mathcal{D}_T$ is bounded. In addition, all possible firing sequences (see Definition 2.8) of immediate transitions are finite.

(2a') $\mathcal{T}_G = \emptyset$. Alternatively, $\mathcal{T}_G \neq \emptyset$ but for all $T \in \mathcal{T}_G$, $\partial\mathcal{G}_T$ is never reached by its input token colours.

(2b') Define $\mathcal{E}_T = \{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))$, define $t_*^{DCPN}(c,T)$ as the time until the next guard transition in the DCPN is enabled after transition T has fired with c as vector of input token colours, and for some $\chi > 0$ define $A_\chi = \{(e',x') \in \mathcal{E}_T \mid t_*^{DCPN}(c,T) \geq \chi\}$. Then we have: for all $T \in \mathcal{T}_G$ and for some $\chi > 0$: $\mathcal{F}_T(A_\chi; c) = 1$ for all $c \in \partial\mathcal{G}_T$.

(2c') In the DCPN all possible firing sequences (see Definition 2.8) of guard transitions have finite length.

Sufficient condition (1') can be verified by inspection of all $\mathcal{D}_T$ ($T \in \mathcal{T}_D$), and by inspection of the reachability graph to see if there are only finite firing sequences of immediate transitions.

Sufficient condition (2a') can be verified by verifying if $\mathcal{T}_G = \emptyset$, and if $\mathcal{T}_G \neq \emptyset$ to verify that no guard transition becomes enabled. Sufficient conditions for a guard transition to not become enabled are if such transition does not become pre-enabled, i.e., it cannot be reached from any reachable marking (the transition is *dead*, see Definition 2.17), or if the boundary of its transition guard $\partial\mathcal{G}_T$ is of a size or shape such that it is never reached by the transition's input tokens.

Sufficient condition (2b') can be verified by inspection of all firing measures of the DCPN and comparing their output with the guards of all guard transitions.

Sufficient condition (2c') can be verified by inspection of the reachability graph to see if all firing sequences of guard transitions have finite length.

Finally, we discuss the condition posed in Theorem 3.2 that none of the transition firings enable a guard transition. This condition has been introduced to make sure that the process state does not jump to the boundary of its state space, but jumps into an open subset. This condition is covered by condition (2b') above.

It is also noted that a guard transition that is *always* immediately enabled when it is pre-enabled, is in effect an immediate transition, and is better modelled as such.

3.8 Concluding remarks

Piecewise deterministic Markov processes (PDPs) can be used to describe virtually all continuous-time stochastic processes and to make the power of stochastic analysis available. However, for complex practical problems with many interactions and sub-behaviours it is often difficult to directly develop a PDP model, and have it verified both by mathematical and by multiple operational domain experts. This chapter has introduced a novel Petri net, which is named dynamically coloured Petri net (DCPN) and has shown that under some mild conditions, any DCPN can be mapped into the elements of a PDP, such that the resulting PDP and the DCPN process are probabilistically equivalent. Moreover, it is shown that the elements of any PDP with a finite discrete state domain can be mapped into a DCPN, such that the resulting DCPN process and the PDP are pathwise equivalent. In [Van04, BLB05] such relation between elements is referred to as *bisimilarity*: The PDP elements and the DCPN are bisimilar.

The key result of this chapter is that this is the first time that proof of the existence of equivalence between PDPs and Petri nets has been established. This significantly extends the modelling power hierarchy of [MT94], [MFT00] in terms of Petri nets and Markov processes, see Figure 1.1. Due to the equivalence relations, PDP theoretical results like stochastic analysis, stability and control theory, also apply to DCPN stochastic processes. The mapping of DCPN to PDP implies that any specific DCPN stochastic process can be analysed as if it is a PDP, often without the need to first apply the transformation into a PDP. Because of this, for accident risk modelling in air transport operations, in [BBB$^+$01] DCPNs are adopted for their specification power and for their PDP inherited stochastic analysis power. Here, DCPN are used as a basis for a Monte Carlo simulation, and the PDP-inherited stochastic analysis properties are used to make the simulations and analysis more efficient.[12]

It is also noted that, driven by practical applications to air transport operations, the DCPN modelling power is enriched by Petri net features beyond those that were actually required to prove equivalence to PDP, such as:

- Inhibitor arcs — these allow particular model structures to be made more compact and support modelling of priority constructs.
- Enabling arcs — these allow different submodels to be connected without tokens disappearing unwanted, and they support modelling of synchronisation constructs.

[12] For an example that illustrates this, we refer to Section 6.3.

- Immediate transitions — these allow decomposing certain complex Petri net transitions into simpler graphical structures.

- Multiple tokens to exist in the Petri net, rather than just one — this supports compositional specification and synchronisation constructs and considerably reduces the number of places in the Petri net graph.

- Option to include tokens that get no colour — this avoids having to introduce dummy colours where colours are not required to capture part of the operation.

For DCPN subclasses without one or more of these additional features, Theorems 3.1 and 3.2 still hold true, and in fact their proof will be simpler.

As a final concluding remark, we note that there is related research on equivalence relationships between PDP and *stochastic hybrid automata*. Here, a stochastic hybrid automaton is defined as a collection of elements containing a countable set of discrete variables, a mapping of each of the discrete variables into Euclidean set, a set of stochastic differential equations, a family of stochastic kernals for the size of jumps, and a transition rate function for spontaneous jumps, e.g., [Buj05]. Through a series of studies, [SV05a, SV05b, Str05] developed a powerful compositional specification approach for automaton of PDP type. The formalism was named *communicating piecewise deterministic Markov process* (CPDP). Under certain conditions (similar to those in Theorem 3.2 of DCPN into PDP), the evolution of the state of a CPDP can be modelled as a PDP. An extended CPDP framework has been developed called value-passing CPDP. This framework provides richer interaction possibilities, where components can communicate information about their continuous states to each other.

3.9 Appendix: Characterisation of Q in terms of DCPN elements

This appendix characterises the PDP transition measure Q in terms of DCPN elements, as part of the proof of Theorem 3.2.

For each $\theta \in \mathbf{K}$, $x \in E_\theta$, $\theta' \in \mathbf{K}$ and $A \subset E_{\theta'}$, the value of $Q(\theta', A; \theta, x)$ equals the probability that if a jump occurs, and if the value of the PDP just prior to the jump is (θ, x), then the value of the PDP just after the jump is in (θ', A) $(= \{\theta'\} \times A)$. Probability $Q(\theta', A; \theta, x)$ is characterised in terms of the DCPN by the reachability graph (RG), elements $\mathcal{D}$, $\mathcal{G}$ and Rules R0–R4 and the set $\mathcal{F}$, as below. This is done in four steps:

1. Determine which transitions are pre-enabled in (θ, x).

2. For each pre-enabled transition, determine the probability with which it is enabled in (θ, x).

3. For each pre-enabled transition, determine whether its firing can possibly lead to discrete state θ'.

4. Use the results of the previous two steps and the set of firing measures to characterise Q.

Step 1: Determine which transitions are pre-enabled in (θ, x).

Consider all arrows in the RG leaving node θ. These arrows are labelled by names of transitions which are pre-enabled in θ, for example T_1 (if T_1 is pre-enabled in θ), $T_1 + T_2$ (if T_1 and T_2 are both pre-enabled and there is a non-zero probability that they fire simultaneously), etc. Therefore the arrows leaving θ may be characterised by these labels. Denote the multi-set of arrows, characterised by these labels, by $\mathcal{H}_\theta$. This set is a multi-set since there may exist several arrows with the same label (e.g., if one transition is pre-enabled by different sets of input tokens). We use notation $H \in \mathcal{H}_\theta$ for an element H of $\mathcal{H}_\theta$ (e.g., $H = T_1$ represents an arrow with T_1 as label), and notation $T \in H$ for a transition T in label H (e.g., as in $H = T + T_1$). Multi-set $\mathcal{H}_\theta$ cannot contain immediate transitions since $\theta \in \mathbf{K}$.

Step 2: For each pre-enabled transition, determine the probability with which it is enabled in (θ, x).

Given that a jump occurs in (θ, x), the set of transitions that will actually fire in (θ, x) is not empty, and is given by one of the labels in $\mathcal{H}_\theta$. In the following, we determine, for each $H \in \mathcal{H}_\theta$, the probability $p_H(\theta, x)$ that all transitions in label H will fire.

- Denote the vector of input colours of transition T in a particular label by c_T^x. For a transition in a label this vector is uniquely determined since we consider transitions with multiple vectors of input colours separately in the multi-set $\mathcal{H}_\theta$.

- Consider the multi-set $\mathcal{H}_\theta^G = \{H \in \mathcal{H}_\theta \mid \forall T \in H : T \in \mathcal{T}_G \text{ and } c_T^x \in \partial\mathcal{G}_T\}$.

- If $\mathcal{H}_\theta^G \neq \emptyset$ then this set contains with probability one all transitions that are enabled in (θ, x). Rules R1–R4 are used (R0 is not applicable) to determine for each $H \in \mathcal{H}_\theta^G$ the probability with which the transitions in label H will actually fire:

  - Rules R1 and R3 are used as follows: Determine set $\mathcal{H}_\theta^{R1R3} = \{H \in \mathcal{H}_\theta^G \mid \exists H' \in \mathcal{H}_\theta^G, H \subsetneqq H'\}$. This set thus exists of those labels that form a real subset of other labels. Then for all $H \in \mathcal{H}_\theta^{R1R3}$, $p_H(\theta, x) = 0$.

  - Rules R2 and R4 are used as follows: If the multi-set $\mathcal{H}_\theta^G - \mathcal{H}_\theta^{R1R3}$ contains m elements, then each of these labels gets a probability $p_H(\theta, x) = 1/m$.
  - For all $H \in \mathcal{H}_\theta - \mathcal{H}_\theta^G + \mathcal{H}_\theta^{R1R3}$, $p_H(\theta, x) = 0$.

- If $\mathcal{H}_\theta^G = \emptyset$ then only Delay transitions can be enabled in (θ, x). Consider the multi-set $\mathcal{H}_\theta^D = \{H \in \mathcal{H}_\theta \mid \forall T \in H : T \in \mathcal{T}_D\}$. Each $H \in \mathcal{H}_\theta^D$ consists of one delay transition, with $p_H(\theta, x) = \frac{\mathcal{D}_H(c_H^x)}{\Sigma_{T \in \mathcal{H}_\theta^D} \mathcal{D}_T(c_T^x)}$. For all $H \in \mathcal{H}_\theta - \mathcal{H}_\theta^D$, $p_H(\theta, x) = 0$.

Step 3: For each pre-enabled transition, determine whether its firing can possibly lead to discrete state θ'.

In the RG, consider nodes θ and θ' and delete all other nodes that are elements of $\mathbf{K}$, including the arrows attached to them. Also, delete all arrows from θ for which $p_H(\theta, x) = 0$ and delete all nodes and arrows that are not part of a directed path from θ to θ'. The residue is named $\text{RG}_{\theta\theta'}$. Then, if θ and θ' are not connected in $\text{RG}_{\theta\theta'}$ by at least one path, a jump from θ to θ' is not possible.

Step 4: Use the results of the previous two steps and the set of firing measures to characterise Q.

From the previous step we have $Q(\theta', A; \theta, x) = 0$ if θ and θ' are not connected in $\text{RG}_{\theta\theta'}$ by at least one directed path. If θ and θ' are connected then in $\text{RG}_{\theta\theta'}$ one or more paths from θ to θ' can be identified. Each such path may consist of only one arrow, or of sequences of directed arrows that pass nodes that enable immediate transitions. All arrows are labelled by names of transitions, therefore the paths between θ and θ' may be characterised by the labels on these arrows, i.e., by the transitions that consecutively fire in the jump from θ to θ'. Denote the multi-set of paths, characterised by these labels, by $\mathcal{Z}_{\theta\theta'}$. Examples of elements of $\mathcal{Z}_{\theta\theta'}$ are T_1 (if T_1 is pre-enabled in θ and its firing leads to θ'), $T_1 + T_2$ (if there is a non-zero probability that T_1 and T_2 will fire at exactly the same time, and their combined firing leads to θ'), $T_4 \circ T_3$ (if T_3 is pre-enabled in θ, its firing leads to the immediate transition T_4 being enabled, and the firing of T_4 leads to θ'), etc.

Next, we factorise Q by conditioning on the path $Z \in \mathcal{Z}_{\theta\theta'}$ along which the jump is made. Under the condition that a jump occurs:

$$Q(\theta', A; \theta, x) = \sum_{Z \in \mathcal{Z}_{\theta\theta'}} p_{\theta', x' \mid \theta, x, Z}(\theta', A \mid \theta, x, Z) \times p_{Z \mid \theta, x}(Z \mid \theta, x),$$

where $p_{\theta', x' \mid \theta, x, Z}(\theta', A \mid \theta, x, Z)$ denotes the conditional probability that the DCPN state immediately after the jump is in (θ', A), given that the DCPN state just prior to the jump equals (θ, x), given that the set of transitions Z fires to establish the jump. Moreover, $p_{Z \mid \theta, x}(Z \mid \theta, x)$ denotes the

conditional probability that the set of transitions Z fires, given that the DCPN state immediately prior to the jump equals (θ, x).

In the remainder of this appendix, first $p_{Z|\theta,x}(Z \mid \theta, x)$ is characterised for each $Z \in \mathcal{Z}_{\theta\theta'}$. Next, $p_{\theta',x'|\theta,x,Z}(\theta', A \mid \theta, x, Z)$ is characterised for each $Z \in \mathcal{Z}_{\theta\theta'}$.

Characterisation of $p_{Z|\theta,x}(Z \mid \theta, x)$ for each $Z \in \mathcal{Z}_{\theta\theta'}$

Each path represented by Z starts with an arrow that leaves node θ. Denote the label on this arrow by $Z \cap \mathcal{H}_\theta$. This label consists of one or more guard transitions or of one delay transition. From Step 2, $p_H(\theta, x)$, with $H = Z \cap \mathcal{H}_\theta$, is the probability that all transitions in this label will fire. If the arrow that leaves node θ ends at θ', then $Z = Z \cap \mathcal{H}_\theta$ and $p_{Z|\theta,x}(Z \mid \theta, x)$ is determined by $p_{Z|\theta,x}(Z \mid \theta, x) = p_Z(\theta, x) = p_{Z\cap\mathcal{H}_\theta}(\theta, x)$.

If the arrow that leaves node θ does not also end at θ', then Z equals a series of labels that also include immediate transitions. The probability $p_{Z\cap\mathcal{H}_\theta}(\theta, x)$ is then equally divided among those paths Z that have the same initial arrow. This yields:

$$p_{Z|\theta,x}(Z \mid \theta, x) = \frac{p_{Z\cap\mathcal{H}_\theta}(\theta, x)}{|\{Z' \in \mathcal{Z}_{\theta\theta'} \mid Z' \cap \mathcal{H}_\theta = Z \cap \mathcal{H}_\theta\}|}$$

where $|\{\cdot\}|$ is the number of elements in $\{\cdot\}$, and where $Z' \cap \mathcal{H}_\theta = Z \cap \mathcal{H}_\theta$ is to be interpreted in the multi-set sense, i.e., the initial arrows should be equal, not only should their labels be the same.

With this, $p_{Z|\theta,x}(Z \mid \theta, x)$ is uniquely characterised.

Characterisation of $p_{\theta',x'|\theta,x,Z}(\theta', A \mid \theta, x, Z)$ for each $Z \in \mathcal{Z}_{\theta\theta'}$

For probability $p_{\theta',x'|\theta,x,Z}(\theta', A \mid \theta, x, Z)$, first notice that both (θ, x) and (θ', x') represent states of the complete DCPN, while the firing of Z changes the DCPN only locally. This yields that in general, several tokens stay where they are when the DCPN jumps from θ to θ' while the set Z of transitions fires.

- $p_{\theta',x'|\theta,x,Z}(\theta', A \mid \theta, x, Z) = 0$ if for all $x' \in A$, the components of x and x' that correspond with tokens not moving to another place when transitions Z fire, are unequal.

In all other cases:

- Assume Z consists of one transition T that, given θ and x, is enabled and will fire. Define again c_T^x as the vector containing the colours of the input tokens of T; c_T^x may not be unique. For each c_T^x that can be identified, a random hybrid vector from $\mathcal{F}_T(\cdot, \cdot; c_T^x)$ provides a vector e' that holds a one for each output arc along which a token is produced and a zero for each output arc along which no token is produced, and it provides a vector c' containing the colours

of the tokens produced. These elements together define the size of the jump of the DCPN state. This gives:

$$p_{\theta',x'|\theta,x,Z}(\theta', A \mid \theta, x, Z) = \sum_{c_T^x} \sum_{e'} \int_{c'} \mathcal{F}_T(e', c'; c_T^x) \times \mathbf{1}_{\{\theta', A; e', c', c_T^x\}} dc',$$

where $\mathbf{1}_{\{\theta', A; e', c', c_T^x\}}$ is the indicator function for the event that if tokens corresponding with c_T^x are removed by T and tokens corresponding with (e', c') are produced, then the resulting DCPN state is in (θ', A).

- If Z consists of several transitions $T_1, \ldots, T_k$ that, given θ and x, will all fire at the same time, then the firing measure $\mathcal{F}_T$ in the equation above is replaced by a product of firing measures for transitions $T_1, \ldots, T_k$:

$$p_{\theta',x'|\theta,x,Z}(\theta', A \mid \theta, x, Z) = \sum_{c_{T_1}^x, \ldots, c_{T_k}^x} \sum_{e'_1, \ldots, e'_k} \int_{c'_1, \ldots, c'_k} \mathcal{F}_{T_1}(e'_1, c'_1; c_{T_1}^x) \times \cdots \times$$

$$\times \mathcal{F}_{T_k}(e'_k, c'_k; c_{T_k}^x) \times \mathbf{1}_{\{\theta', A; e'_1, c'_1, c_{T_1}^x, \ldots, e'_k, c'_k, c_{T_k}^x\}} dc'_1 \ldots dc'_k,$$

where $\mathbf{1}_{\{\theta', A; e'_1, c'_1, c_{T_1}^x, \ldots, e'_k, c'_k, c_{T_k}^x\}}$ denotes indicator function for the event that the combined removal of $c_{T_1}^x$ through $c_{T_k}^x$ by transitions T_1 through T_k, respectively, and the combined production of (e'_1, c'_1) through (e'_k, c'_k) by transitions T_1 through T_k, respectively, leads to a DCPN state in (θ', A).

- If Z is of the form $Z = T_j \circ T_k$, with T_j an immediate transition, then the result is:

$$p_{\theta',x'|\theta,x,Z}(\theta', A \mid \theta, x, Z) = \sum_{c_{T_k}^x} \sum_{e'_j, e'_k} \int_{c'_j, c_j, c'_k} \mathcal{F}_{T_j}(e'_j, c'_j; c_j) \times \mathcal{F}_{T_k}(e'_k, c'_k; c_{T_k}^x) \times$$

$$\times \mathbf{1}_{\{\theta', A; e'_j, c'_j, e'_k, c'_k, c_T^x\}} dc'_j dc_j dc'_k,$$

where $\mathbf{1}_{\{\theta', A; e'_j, c'_j, e'_k, c'_k, c_T^x\}}$ denotes indicator function for the event that the removal of $c_{T_k}^x$ and the production of (e'_k, c'_k) by transition T_k leads to T_j having a vector of colours of input tokens c_j and the subsequent removal of c_j and the production of (e'_j, c'_j) by transition T_j leads to a DCPN state in (θ', A).

- In cases like $Z = T_m \circ T_j \circ T_k$, with T_j and T_m immediate transitions, the firing measures of this sequence of transitions are multiplied in a similar way as above.

With this, PDP transition measure Q of the constructed PDP is uniquely characterised in terms of DCPN elements.

Chapter 4

Stochastically and dynamically coloured Petri nets

4.1 Introduction

Although PDPs form a very general class of continuous-time Markov processes which include both discrete and continuous processes, PDPs do not include diffusion. Diffusion exists in air transport operations for example in the form of stochastic variations around position and velocity of an aircraft, due to, e.g., weather, navigation or surveillance uncertainties, or engine power fluctuations. In this chapter, we extend PDP to GSHP (general stochastic hybrid process) by inclusion of diffusion. With this extension, between jumps, the process $\{X_t\}$ follows the solution of a θ_t-dependent *stochastic* (rather than ordinary) differential equation. GSHP defines a powerful and useful class of processes that have support in stochastic analysis.

GSHP can be defined through various related formalisms. In the main part of this chapter, we adopt the one developed in [Blo03] and [BBEP03], in which GSHP is defined as the solution of a *hybrid stochastic differential equation (HSDE) on a hybrid state space*. The development of this formalism, in a version without forced jumps, started in [Blo90] and was published in [Blo03]. Forced jumps were added in [BBEP03]. Related formalisms were also studied in, e.g., [Kry06], and an overview of various related HSDE versions is given in [KBB07]. In [BL06], GSHP are alternatively defined as the execution of a stochastic hybrid automaton named *general stochastic hybrid system* (GSHS).

The aim of the current chapter is to

- introduce an extension of DCPN, referred to as *stochastically and dynamically coloured Petri net* (SDCPN), which covers diffusion;
- explain HSDE and its HSDE solution process (i.e., the GSHP).

- show that there exist equivalence relations between SDCPN-processes and HSDE-processes.
- show that there also exist equivalence relations between SDCPN-processes and GSHS-processes.

The existence of such equivalence relations allows combining the specification power of Petri nets with the stochastic analysis power of GSHP. In addition, it extends the power hierarchy of Figure 1.1 with GSHP and with GSHP-related Petri nets.

GSHP are a generalisation of PDP by the inclusion of a Wiener process in the continuous evolution. Intuitively, a GSHP is a copy of PDP with replacement of the ordinary differential equations by stochastic ones. Formally, however, this intuitive idea poses some challenges, particularly regarding the timing of jumps. As a result, the proof of equivalence between SDCPN-processes and GSHP is not a straightforward extension of the material in Chapter 3.

This chapter is organised as follows: Section 4.2 provides a few definitions and propositions on stochastic processes that are necessary to properly understand the extensions of this chapter. Section 4.3 defines SDCPN. Section 4.4 describes HSDE. Section 4.5 shows that for each arbitrary HSDE we can construct an equivalent SDCPN. Section 4.6 shows that for each arbitrary SDCPN we can construct an equivalent HSDE. Section 4.7 discusses the conditions under which the equivalence relations hold true. Section 4.8 describes GSHS and how it is different from HSDE, shows that for each arbitrary GSHS we can construct an equivalent SDCPN, and that for each arbitrary SDCPN we can construct an equivalent GSHS. Finally, Section 4.9 draws conclusions.

4.2 Preliminaries

This section provides a few definitions and propositions on stochastic processes that are necessary to properly understand the extensions of the current chapter. More specifically, the section defines Poisson random measure and explains how to generate its points.

Consider a complete stochastic basis $(\Omega, \Im, \{\Im_t\}, \mathbb{P}, \mathbb{T})$, with probability space $(\Omega, \Im, \mathbb{P})$, right continuous filtration $\{\Im_t\}$, and time index $\mathbb{T} = \mathbb{R}_+ = [0, \infty)$.

A *Poisson random measure* $p_P(dt, dz)$ on $\mathbb{R}^+ \times \mathbf{Z}$ is an extended Poisson random measure (see Appendix A), the intensity measure ν of which satisfies $\nu(\{t\} \times \mathbf{Z}) = 0$, for all t. Here, $\nu(A) = \mathbb{E}\{p_P(A)\}$, where $\mathbb{E}\{\cdot\}$ denotes expectation. Poisson random measure satisfies the memoryless property, i.e., for every A, the variable $p_P(A)$ is independent of the σ-algebra $\Im_t$. A Poisson random measure is said to be *homogeneous* if its intensity measure is of the form $\nu(dt, dz) = dt \cdot \tilde{\mu}(dz)$.

Intuitively, a Poisson random measure $p_P(dt, dz)$ on $\mathbb{R}^+ \times \mathbf{Z}$ with intensity measure $\nu(dt, dz) = \mathbb{E}\{p_P(dt, dz)\}$ defines an increasing sequence of arrival times τ_n at exponentially distributed intervals (Poisson process), which each generate a "mark" z_n. The Poisson process property comes

from the memoryless property. The Poisson random measure $p_P(dt, dz)$ assigns unit mass to $(\{\tau_n\}, \{z_n\})$ if there is an arrival at time τ_n of mark z_n. This yields:

$$p_P(dt, dz) = \sum_{n\geq 1} \mathbf{1}_{\{\tau_n \in [t, t+dt)\}} \cdot \mathbf{1}_{\{z_n \in [z, z+dz)\}}$$

In the special case where the Poisson random measure is homogeneous and the mark space $\mathbf{Z}$ is bounded, e.g., $\mathbf{Z} = [0, C]$, we can generate points in a straightforward way: Let N_t be a standard Poisson process with intensity λ. We denote by τ_n, $n = 1, 2, \ldots$ the jump times of N_t. Let $\{z_n\}$, $n \geq 1$ be a sequence of i.i.d. random variables with uniform distribution on $\mathbf{Z}$, independent of N_t, with $\mu_L(\mathbf{Z}) = C < \infty$, where μ_L is the Lebesgue measure. In this special case we can represent the Poisson random measure $p_P(dt, dz)$ with intensity $dt \cdot dz$ as a counting measure associated to the marked point process $\{\tau_n, z_n\}$, $n \geq 1$. Now, we can use that

$$\begin{aligned} p_P((0, t], \mathbf{Z}) &= \sum_{n\geq 1} \mathbf{1}_{\{\tau_n \in (0,t]\}} \cdot \mathbf{1}_{\{z_n \in \mathbf{Z}\}} \\ &= \sum_{n\geq 1} \mathbf{1}_{\{\tau_n \in (0,t]\}}, \end{aligned}$$

hence $p_P((0, t], \mathbf{Z})$ is simply the number of points τ_n generated by N_t during $(0, t]$. Since N_t is a Poisson process with intensity λ, we get $\mathbb{E}\{p_P((0, t], \mathbf{Z})\} = \lambda \cdot t$. On the other hand, the intensity of $p_P(dt, dz)$ was given as $dt \cdot dz$, therefore, $\mathbb{E}\{p_P((0, t], \mathbf{Z})\} = t \cdot C$. We find that $\lambda = C$, i.e., the intensity of the Poisson arrivals τ_n is equal to the 'size' of the mark space. We see that if $\mathbf{Z}$ is bounded and if p_P is homogeneous, then $p_P(dt, dz)$ can be generated just by sampling random variables $(\{\tau_n\}, \{z_n\})$, $n = 1, 2, \ldots$, with $\tau_n - \tau_{n-1} \sim \text{Exp}(C)$ (exponential with intensity $1/C$) and $z_n \sim U[\mathbf{Z}]$ (uniform on $\mathbf{Z}$). Note that if $\mathbf{Z}$ is not bounded, then $p_P(dt, dz)$ is still well defined, but the above way of working would generate infinitely many samples.

Next, we take $\mathbf{Z} = [0, C]$ and consider the Poisson random measure on a subarea of the mark space, e.g., $p_P(dt, [0, \Lambda])$, with $\Lambda \leq C$ and we write

$$\begin{aligned} p_P(dt, [0, \Lambda]) &= \sum_{n\geq 1} \mathbf{1}_{\{\tau_n \in [t, t+dt)\}} \cdot \mathbf{1}_{\{z_n \in [0,\Lambda)\}} \\ &= \sum_{n\geq 1} \mathbf{1}_{\{\tau_n \in [t, t+dt)\}} \cdot \mathbf{1}_{\{z_n \in \mathbf{Z}\}} \cdot \mathbf{1}_{\{z_n \in [0,\Lambda)\}} \end{aligned}$$

The Poisson random measure generates random variables $(\{\tau_n\}, \{z_n\})$, $n = 1, 2, \ldots$, with $\tau_n - \tau_{n-1} \sim \text{Exp}(C)$ and $z_n \sim U[0, C]$, however, those points $(\{\tau_n\}, \{z_n\})$ for which $z_n > \Lambda$ lead to $\mathbf{1}_{\{z_n \in [0,\Lambda]\}} = 0$ hence are rejected. It can be shown that the marks that are not rejected are uniformly distributed on $[0, \Lambda]$ and have exponential interarrival times with intensity $1/\Lambda$. This is proven by

the following two propositions:

Proposition 4.1. *Consider $c \in [0, \infty)$, $\lambda \in [0, c]$, and a sequence of random samples $U_1, U_2, \ldots$ from $U[0, c]$. Define $r \geq 1$ such that $U_1 > \lambda, \ldots, U_{r-1} > \lambda$ and $U_r \leq \lambda$, and take $V = U_r$. Then random variable V has a uniform distribution on $[0, \lambda]$.*

Proof. For $v \in [0, \lambda]$, $\mathbb{P}\{V \leq v\} = \mathbb{P}\{U \leq v \mid U \leq \lambda\} = \dfrac{\mathbb{P}\{U \leq v\}}{\mathbb{P}\{U \leq \lambda\}} = \dfrac{v}{\lambda}$. □

Proposition 4.2. *Consider $c \in [0, \infty)$ and $\lambda \in [0, c]$. Then the following two random variables T and Y have the same probability distribution: $T \sim$ Exp(λ) and $Y \sim$ Exp$(c;$ accept with $\lambda/c)$. Here, Exp(λ) denotes an exponential distribution with intensity $1/\lambda$ and Exp$(c;$ accept with $\lambda/c)$ denotes a distribution where a sequence $Y_1, Y_2, \ldots$ of samples from Exp(c) are consecutively either accepted (with probability λ/c) or rejected (with probability $1 - \lambda/c$), and where*

$$Y = \sum_{n=1}^{q} Y_n, \text{ with } q = \arg\min_i \{Y_i \text{ is accepted}\}$$

(i.e., $Y_1, \ldots, Y_{q-1}$ have been rejected and Y_q is the first sample that is accepted).

Proof. Due to $T \sim \text{Exp}(\lambda)$, we have $\mathbb{P}\{T > t\} = e^{-\lambda t}$.

Suppose we have a sequence of independent exponentially distributed variables $Y_1, Y_2, \ldots$, with $Y_i \sim \text{Exp}(c)$ for all i. Then $\mathbb{P}\{Y > t\} = \mathbb{P}\{Y_1 > t\} + \mathbb{P}\{Y_1 \leq t, Y_1 \text{ is rejected}, Y_1 + Y_2 > t\} + \mathbb{P}\{Y_1 + Y_2 \leq t, Y_1 \text{ and } Y_2 \text{ are rejected}, Y_1 + Y_2 + Y_3 > t\} + \mathbb{P}\{Y_1 + Y_2 + Y_3 \leq t, Y_1, \ldots, Y_3 \text{ are rejected}, Y_1 + Y_2 + Y_3 + Y_4 > t\} + \cdots$. Now,

$$\mathbb{P}\{Y_1 > t\} = e^{-ct}, \text{ and}$$

$$\begin{aligned}
&\mathbb{P}\{Y_1 \leq t, Y_1 \text{ is rejected}, Y_1 + Y_2 > t\} = \\
&\quad = \frac{c - \lambda}{c} \int_0^t \int_{t-y_1}^{\infty} c e^{-cy_1} \cdot c e^{-cy_2} dy_2 dy_1 \\
&\quad = (c - \lambda) \int_0^t e^{-cy_1} \cdot \left[-e^{-cy_2}\right]_{t-y_1}^{\infty} dy_1 \\
&\quad = (c - \lambda) \int_0^t e^{-cy_1} \cdot e^{-c(t-y_1)} dy_1 \\
&\quad = (c - \lambda) e^{-ct} \int_0^t dy_1 \\
&\quad = (c - \lambda) e^{-ct} t, \text{ and}
\end{aligned}$$

$$
\begin{aligned}
&\mathbb{P}\{Y_1+Y_2\le t, Y_1 \text{ and } Y_2 \text{ are rejected}, Y_1+Y_2+Y_3>t\} = \\
&\quad = \frac{(c-\lambda)^2}{c^2}\int_0^t\int_0^{t-y_1}\int_{t-y_1-y_2}^{\infty} ce^{-cy_1}\cdot ce^{-cy_2}\cdot ce^{-cy_3}dy_3dy_2dy_1 \\
&\quad = (c-\lambda)^2\int_0^t\int_0^{t-y_1} e^{-c(y_1+y_2)}\cdot\left[-e^{-cy_3}\right]_{t-y_1-y_2}^{\infty}dy_2dy_1 \\
&\quad = (c-\lambda)^2\int_0^t\int_0^{t-y_1} e^{-c(y_1+y_2)}\cdot e^{-c(t-y_1-y_2)}dy_2dy_1 \\
&\quad = (c-\lambda)^2\int_0^t\int_0^{t-y_1} e^{-ct}dy_2dy_1 \\
&\quad = (c-\lambda)^2e^{-ct}\int_0^t(t-y_1)dy_1 \\
&\quad = (c-\lambda)^2e^{-ct}\left[-\frac{1}{2}(t-y_1)^2\right]_0^t \\
&\quad = (c-\lambda)^2e^{-ct}\frac{1}{2}t^2,
\end{aligned}
$$

and more generally,

$$
\begin{aligned}
&\mathbb{P}\left\{\sum_{i=1}^n Y_i\le t, Y_1,\dots,Y_n \text{ are rejected}, \sum_{i=1}^{n+1}Y_i>t\right\} = \\
&\quad = \frac{(c-\lambda)^n}{c^n}\int_0^t\int_0^{t-y_1}\cdots\int_0^{t-\Sigma_{i=1}^{n-1}y_i}\int_{t-\Sigma_{i=1}^n y_i}^{\infty} c^n e^{-c\Sigma_{i=1}^n y_i}\cdot ce^{-cy_{n+1}}dy_{n+1}\cdots dy_1 \\
&\quad = (c-\lambda)^n\int_0^t\int_0^{t-y_1}\cdots\int_0^{t-\Sigma_{i=1}^{n-1}y_i} e^{-c\Sigma_{i=1}^n y_i}\cdot\left[-e^{-cy_{n+1}}\right]_{t-\Sigma_{i=1}^n y_i}^{\infty}dy_n\cdots dy_1 \\
&\quad = (c-\lambda)^n\int_0^t\int_0^{t-y_1}\cdots\int_0^{t-\Sigma_{i=1}^{n-1}y_i} e^{-c\Sigma_{i=1}^n y_i}\cdot e^{-c(t-\Sigma_{i=1}^n y_i)}dy_n\cdots dy_1 \\
&\quad = (c-\lambda)^n\int_0^t\int_0^{t-y_1}\cdots\int_0^{t-\Sigma_{i=1}^{n-1}y_i} e^{-ct}dy_n\cdots dy_1 \\
&\quad = (c-\lambda)^ne^{-ct}\int_0^t\int_0^{t-y_1}\cdots\int_0^{t-\Sigma_{i=1}^{n-2}y_i}(t-\sum_{i=1}^{n-1}y_i)dy_{n-1}\cdots dy_1 \\
&\quad = (c-\lambda)^ne^{-ct}\int_0^t\int_0^{t-y_1}\cdots\int_0^{t-\Sigma_{i=1}^{n-3}y_i}\left[-\frac{1}{2}(t-\sum_{i=1}^{n-1}y_i)^2\right]_0^{t-\Sigma_{i=1}^{n-2}y_i}dy_{n-2}\cdots dy_1 \\
&\quad = (c-\lambda)^ne^{-ct}\int_0^t\int_0^{t-y_1}\cdots\int_0^{t-\Sigma_{i=1}^{n-3}y_i}\frac{1}{2}(t-\sum_{i=1}^{n-2}y_i)^2dy_{n-2}\cdots dy_1 \\
&\quad = (c-\lambda)^ne^{-ct}\int_0^t\int_0^{t-y_1}\cdots\int_0^{t-\Sigma_{i=1}^{n-4}y_n}\frac{1}{2}\left[-\frac{1}{3}(t-\sum_{i=1}^{n-2}y_i)^3\right]_0^{t-\Sigma_{i=1}^{n-3}y_i}dy_{n-3}\cdots dy_1
\end{aligned}
$$

$$
\begin{aligned}
&= (c-\lambda)^n e^{-ct} \int_0^t \int_0^{t-y_1} \cdots \int_0^{t-\Sigma_{i=1}^{n-4} y_i} \frac{1}{2\cdot 3}(t-\sum_{i=1}^{n-3} y_i)^3 dy_{n-3}\cdots dy_1 \\
&= \cdots \\
&= (c-\lambda)^n e^{-ct} \int_0^t \frac{1}{(n-1)!}(t-y_1)^{n-1} dy_1 \\
&= (c-\lambda)^n e^{-ct} \frac{1}{(n-1)!}\left[-\frac{1}{n}(t-y_1)^n\right]_0^t \\
&= (c-\lambda)^n e^{-ct} \frac{1}{n!} t^n
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\mathbb{P}\{Y>t\} &= e^{-ct} + (c-\lambda)e^{-ct}t + (c-\lambda)^2 e^{-ct}\frac{1}{2}t^2 + (c-\lambda)^3 e^{-ct}\frac{1}{6}t^3 + \cdots \\
&= e^{-ct}\sum_{n=0}^{\infty}(c-\lambda)^n \frac{t^n}{n!} \\
&= e^{-ct}\cdot e^{(c-\lambda)t} \\
&= e^{-\lambda t} \\
&= \mathbb{P}\{T>t\}
\end{aligned}
$$

which completes the proof of Proposition 4.2 □

These propositions are particularly useful in case we consider $p_P(dt,[0,\Lambda])$, with Λ not a constant. The specific extension of interest is where Λ is a function of a stochastic process state, e.g., $\Lambda = \Lambda(\xi_t)$. If $\{\xi_t\}$ is a stochastic process we cannot, at times $< t$, directly generate exponential random variables with intensity $1/\Lambda(\xi_t)$, since the value of $\Lambda(\xi_t)$ may not be known until we are at time t. However, the combined use of Propositions 4.1 and 4.2 allows to generate points $(\{\tau\},\{z\})$ from this Poisson random measure in a straightforward manner: Consider a sequence (Y_i,U_i) of pairs where $Y_i \sim \text{Exp}(C)$ and $U_i \sim U[0,C]$. Then use U_1 to accept or reject Y_1: If $U_1 \le \Lambda(\xi_{Y_1})$ then accept, otherwise reject. In case of rejection, use U_2 to accept or reject Y_2: If $U_2 \le \Lambda(\xi_{Y_1+Y_2})$ then accept, otherwise reject, etc. If Y_q, $q \ge 1$, is the first sample that is accepted, then due to Proposition 4.2, we find that $\tau = \sum_{n=1}^{q} Y_n$ and due to Proposition 4.1, we find that $z = U_q$.

The above reasoning can be extended to cases where the mark space can be written as $\mathbf{Z} = \mathbf{Z}_1 \times \underline{\mathbf{Z}}$, where $\mathbf{Z}_1$ is bounded, e.g., $\mathbf{Z}_1 = [0,C]$, and where the intensity measure is $\mathbb{E}\{p_P(dt,dz)\} = dt\cdot dz_1 \cdot \mu(d\underline{z})$. Here, z_1 is the first component of a mark $z \in \mathbf{Z}$, $\underline{z}$ collects the other components, and μ is a probability measure. Using $\mu(\underline{\mathbf{Z}}) = 1$ we find again that $\mathbb{E}\{p_P((0,t],\mathbf{Z})\} = t\cdot C$ and that the arrivals of τ_n are Poisson with intensity C. The generation of points from $p_P(dt,(0,\Lambda(\xi_t)]\times\underline{\mathbf{Z}})$ is similar as above, except that sequences of triples $(Y_i,U_i,\underline{Z}_i)$ are generated where $Y_i \sim \text{Exp}(C)$,

$U_i \sim U[0, C]$ and $\underline{Z}_i \sim \mu$. As long as $U_i > \Lambda(\xi_{\Sigma_{n=1}^{i} Y_n})$, reject the triple. As soon as $U_q \leq \Lambda(\xi_{\Sigma_{n=1}^{q} Y_n})$ then the Poisson random measure has generated a point $(\{\tau\}, \{z_1\}, \{\underline{z}\})$, with $\tau = \sum_{n=1}^{q} Y_n$, $z_1 = U_q$, and $\underline{z} = \underline{Z}_q$.

4.3 Stochastically and dynamically coloured Petri nets

This section presents a definition of *stochastically and dynamically coloured Petri net* (SDCPN).

Definition 4.1 (Stochastically and dynamically coloured Petri net)**.** *An SDCPN is a collection of elements* $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{W}, \mathcal{G}, \mathcal{D}, \mathcal{F})$*, together with an SDCPN execution prescription which makes use of a sequence* $\{U_i; i = 0, 1, \ldots\}$ *of independent uniform* $U[0, 1]$ *random variables, of independent sequences of mutually independent standard Brownian motions* $\{B_t^{i,P}; i = 1, 2, \ldots\}$ *of appropriate dimensions, one sequence for each place* P*, and of five rules R0–R4 that solve enabling conflicts.*

The execution of an SDCPN defines a sample path of a stochastic process which is a random hybrid vector formed by the collection of colours of all tokens, and by the places in which they reside. This is similar as for DCPN. The main difference with DCPN is that the token colour functions are no longer defined by ordinary differential equations, but by stochastic differential equations.

The formal SDCPN definition provided below is organised as follows:

- Section 4.3.1 defines the SDCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$).
- Section 4.3.2 explains the SDCPN execution.
- Section 4.3.3 explains how the SDCPN execution defines a unique stochastic process.

4.3.1 SDCPN elements

The SDCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) are defined as follows:

- $\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{G}$, $\mathcal{D}$, and $\mathcal{F}$ are as for DCPN, see Section 3.3.1.
- $\mathcal{W} = \{\mathcal{W}_P; P \in \mathcal{P}, \mathcal{C}(P) \neq \mathbb{R}^0\}$ is a set of token colour matrix functions. For each place $P \in \mathcal{P}$ for which $\mathcal{C}(P) \neq \mathbb{R}^0$, it contains a measurable mapping $\mathcal{W}_P : \mathcal{C}(P) \to \mathbb{R}^{n(P) \times h(P)}$ that defines the diffusion coefficient of a stochastic differential equation for the colour of a token in place P, where $h : \mathcal{P} \to \mathbb{N}$, and $n : \mathcal{P} \to \mathbb{N}$ is such that $\mathcal{C}(P) = \mathbb{R}^{n(P)}$. It is

assumed that $\mathcal{W}_P$ and $\mathcal{V}_P$ satisfy conditions that ensure a probabilistically unique solution of each stochastic differential equation.[1]

For the places, transitions and arcs, the graphical notation is as for DCPN, see Figure 3.1.

4.3.2 SDCPN execution

The execution of an SDCPN is similar to the execution of a DCPN, following a prescription of the initiation, the token colour evolution, the transition enabling, and the transition firing. The execution makes use of a sequence $\{U_i; i = 0, 1, \ldots\}$ of independent uniform random variables on $[0, 1]$ and it provides a series of increasing stopping times, $\tau_0 < \tau_1 < \tau_2 < \cdots$, with for $t \in (\tau_i, \tau_{i+1})$ a fixed number of tokens per place and per token a colour which is the solution of a differential equation. The main difference with DCPN execution is that this differential equation is now a stochastic rather than an ordinary one, by inclusion of a diffusion term. For this, it makes use of independent sequences of mutually independent standard Brownian motions $\{B_t^{i,P}; i = 1, 2, \ldots\}$ of appropriate dimensions, one sequence for each place P.

Initiation The initiation of SDCPN is equal to the initiation of DCPN: The probability measure $\mathcal{I}$ characterises an initial marking at time $\tau_0 = 0$.

Token colour evolution The process of token colour evolution between transition firings in SDCPN is different from the evolution in DCPN: For each token in each place P for which $\mathcal{C}(P) \neq \mathbb{R}^0$: if the colour of this token is equal to C_τ^P at time $t = \tau$, and if this token is still in this place at time $t > \tau$, then the colour C_t^P of this token equals the probabilistically unique solution of the stochastic differential equation $dC_t^P = \mathcal{V}_P(C_t^P)dt + \mathcal{W}_P(C_t^P)dB_t^{i,P}$ with initial condition C_τ^P, and with $\{B_t^{i,P}\}$ an $h(P)$-dimensional standard Brownian motion. This gives $C_t^P = C_\tau^P + \int_\tau^t \mathcal{V}_P(C_s^P)ds + \int_\tau^t \mathcal{W}_P(C_s^P)dB_s^{i,P}$. The first token, if any, in place P uses Brownian motion $\{B_t^{1,P}\}$; the second token, if any, uses Brownian motion $\{B_t^{2,P}\}$, etc. Each token in a place for which $\mathcal{C}(P) = \mathbb{R}^0$ remains without colour.

Transition enabling, and Transition firing Pre-enabling, enabling and firing of SDCPN transitions is according to the same procedures and rules as for DCPN, including rules R0–R4. In order to keep track of the identity of individual tokens, the tokens in a place are ordered, according to the same rules as for DCPN.

[1] Note that in earlier SDCPN definitions, e.g., [EB06], it was assumed that $\mathcal{V}_P$ and $\mathcal{W}_P$ satisfy local Lipschitz condition. This condition has now been relaxed to probabilistic uniqueness of solution of the related stochastic differential equation(s).

4.3.3 SDCPN stochastic process

The SDCPN stochastic process is defined in the same way as the DCPN stochastic process: The marking of the SDCPN is given by the numbers of tokens in the places and the associated colour values of these tokens. Due to the uniquely defined order of the tokens, the marking is unique except possibly when one or more transitions fire (particularly, immediate transitions fire without delay hence a sequence of immediate transitions firing will generate a sequence of markings at the same time instant). The SDCPN marking at each time instant can be mapped to a probabilistically unique SDCPN stochastic process $\{M_t, C_t\}$, as follows:

For any $t \geq \tau_{k-1}$, $k = 1, 2, \ldots$, let a token distribution be characterised by the vector $M'_t = (M'_{1,t}, \ldots, M'_{|\mathcal{P}|,t})$, where $M'_{i,t} \in \mathbb{N}$ denotes the number of tokens in place P_i at time t and $1, \ldots, |\mathcal{P}|$ refers to a unique ordering of places adopted for SDCPN. At times $t \in (\tau_{k-1}, \tau_k)$ when no transition fires, the token distribution is unique and the SDCPN discrete process state M_t is defined to be equal to M'_t. The associated colours of these tokens are uniquely gathered in a column vector C_t which first contains all colours of tokens in place P_1, next (i.e., below it) all colours of tokens in place P_2, etc, until place $P_{|\mathcal{P}|}$, where $1, \ldots, |\mathcal{P}|$ refers to a unique ordering of places adopted for SDCPN. Within a place the colours of the tokens are ordered according to the unique ordering of tokens within their place defined for SDCPN (see under SDCPN execution above).

If at time $t = \tau_k$ ($k \geq 0$) one or more transitions fire, then the set of applicable token distributions is collected in $\widetilde{M}_{\tau_k} = \{M'_{\tau_k} \mid M'_{\tau_k}$ is a token distribution at time $\tau_k\}$, and the SDCPN discrete process state at time τ_k is uniquely defined by $M_{\tau_k} = \{M'_{\tau_k} \mid M'_{\tau_k} \in \widetilde{M}_{\tau_k}$ and no transitions are enabled in $M'_{\tau_k}\}$; in words, M_{τ_k} is the token distribution that occurs after all transitions that fire at time τ_k have been fired. The associated colours of these tokens are gathered in a column vector C_{τ_k} in the same way as described above. This construction ensures that the process $\{M_t, C_t\}$ has limits from the left and is continuous from the right, i.e., it satisfies the càdlàg property. If at a time t when one or more transitions fire, the process $\{M_t\}$ jumps to the same value again, and only C_t makes a jump, then the càdlàg property for $\{C_t\}$ (hence for $\{M_t, C_t\}$) is still maintained due to the timing construction of $\{M_t\}$ above and the direct coupling of $\{C_t\}$ with $\{M_t\}$.

4.4 Hybrid stochastic differential equations

This section presents, following [Blo03] and [BBEP03], a definition of *hybrid stochastic differential equation* (HSDE) on a hybrid state space and gives conditions under which the HSDE has a pathwise unique solution. This pathwise unique solution is referred to as *HSDE solution process* or *general stochastic hybrid process* (GSHP).

We work with a complete stochastic basis $(\Omega, \Im, \{\Im_t\}, \mathbb{P}, \mathbb{T})$, in which a complete probability space $(\Omega, \Im, \mathbb{P})$ is equipped with a right-continuous filtration $\{\Im_t\}$ on the positive time line

$\mathbb{T} = \mathbb{R}^+$. This stochastic basis is endowed with a probability measure μ_{θ_0,X_0} for the initial state, an independent h-dimensional standard Wiener process $\{W_t\}$ and an independent homogeneous Poisson random measure $p_P(dt, dz)$ on $\mathbb{T} \times \mathbb{R}^{q+1}$.

Definition 4.2 (Hybrid stochastic differential equation)**.** *A HSDE on stochastic basis* $(\Omega, \Im, \{\Im_t\}, \mathbb{P}, \mathbb{T})$*, is defined as a set of equations (4.4.1)-(4.4.8) in which a collection of elements (*$\mathbb{M}$*,* E*,* f*,* g*,* μ_{θ_0,X_0}*,* Λ*,* ψ*,* ρ*,* μ*,* p_P*,* $\{W_t\}$*) appear.*

This section is organised as follows:

- Section 4.4.1 explains the elements and the equations that define HSDE on a hybrid state space.
- Section 4.4.2 shows, following [Blo03] and [BBEP03], that under a number of HSDE conditions H1–H8, the HSDE has a pathwise unique solution which is a semi-martingale.

4.4.1 HSDE elements and equations

This section presents the elements and equations that define an HSDE on a hybrid state space. The elements ($\mathbb{M}$, E, f, g, μ_{θ_0,X_0}, Λ, ψ, ρ, μ, p_P, $\{W_t\}$) are defined as follows:

- $\mathbb{M} = \{\vartheta_1, \ldots, \vartheta_N\}$ is a finite set, $N \in \mathbb{N}$, $1 \leq N < \infty$.
- $E = \{\{\theta\} \times E_\theta; \theta \in \mathbb{M}\}$ is the hybrid state space, where for each $\theta \in \mathbb{M}$, E_θ is an open subset of $\mathbb{R}^n$ with boundary ∂E_θ. The boundary of E is given by $\partial E = \{\{\theta\} \times \partial E_\theta; \theta \in \mathbb{M}\}$.
- $f : \mathbb{M} \times \mathbb{R}^n \to \mathbb{R}^n$ is a measurable mapping.
- $g : \mathbb{M} \times \mathbb{R}^n \to \mathbb{R}^{n \times h}$ is a measurable mapping.
- $\mu_{\theta_0,X_0} : \Omega \times \mathcal{B}(E) \to [0, 1]$ is a probability measure for the initial random variables θ_0, X_0, which are defined on the stochastic basis; μ_{θ_0,X_0} is assumed to be invertible (i.e., its quantile function is assumed to exist).
- $\Lambda : \mathbb{M} \times \mathbb{R}^n \to [0, \infty)$ is a measurable mapping.
- $\psi : \mathbb{M} \times \mathbb{M} \times \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^n$ is a measurable mapping such that $x + \psi(\vartheta, \theta, x, \underline{z}) \in E_\vartheta$ for all $x \in E_\theta$, $\underline{z} \in \mathbb{R}^q$, and $\vartheta, \theta \in \mathbb{M}$.
- $\rho : \mathbb{M} \times \mathbb{M} \times \mathbb{R}^n \to [0, \infty)$ is a measurable mapping such that $\sum_{i=1}^N \rho(\vartheta_i, \theta, x) = 1$ for all $\theta \in \mathbb{M}, x \in \mathbb{R}^n$.
- $\mu : \Omega \times \mathbb{R}^q \to [0, 1]$ is a probability measure which is assumed to be invertible.

- $p_P : \Omega \times \mathbb{T} \times \mathbb{R}^{q+1} \to \{0, 1\}$ is a homogeneous Poisson random measure on the stochastic basis, independent of (θ_0, X_0). The intensity measure of $p_P(dt, dz)$ equals $dt \cdot \mu_L(dz_1) \cdot \mu(d\underline{z})$, where $z = \text{Col}\{z_1, \underline{z}\}$ and μ_L is the Lebesgue measure.
- $W : \Omega \times \mathbb{T} \to \mathbb{R}^h$ such that $\{W_t\}$ is an h-dimensional standard Wiener process on the stochastic basis, and independent of (θ_0, X_0) and p_P.

Using these elements, the HSDE process $\{\theta_t^*, X_t^*\}$ is defined as follows:

$$\theta_t^* = \theta_t^k \text{ for all } t \in [\tau_k^b, \tau_{k+1}^b), k = 0, 1, 2, \ldots \tag{4.4.1}$$

$$X_t^* = X_t^k \text{ for all } t \in [\tau_k^b, \tau_{k+1}^b), k = 0, 1, 2, \ldots \tag{4.4.2}$$

Hence $\{\theta_t^*, X_t^*\}$ consists of a concatenation of processes $\{\theta_t^k, X_t^k\}$ which are defined by (4.4.3)-(4.4.8) below. If the system (4.4.1)-(4.4.8) has a solution in probabilistic sense, then the process $\{\theta_t^*, X_t^*\}$ is referred to as *HSDE solution process* or *GSHP*.

$$d\theta_t^k = \sum_{i=1}^{N} (\vartheta_i - \theta_{t-}^k) p_P(dt, (\Sigma_{i-1}(\theta_{t-}^k, X_{t-}^k), \Sigma_i(\theta_{t-}^k, X_{t-}^k)] \times \mathbb{R}^q) \tag{4.4.3}$$

$$dX_t^k = f(\theta_t^k, X_t^k)dt + g(\theta_t^k, X_t^k)dW_t + \int_{\mathbb{R}^q} \psi(\theta_t^k, \theta_{t-}^k, X_{t-}^k, \underline{z}) p_P(dt, (0, \Lambda(\theta_{t-}^k, X_{t-}^k)] \times d\underline{z}) \tag{4.4.4}$$

with $\theta_0^0 = \theta_0$, $X_0^0 = X_0$ and with Σ_0 through Σ_N measurable mappings satisfying, for $\theta \in \mathbb{M}$, $\vartheta_j \in \mathbb{M}$, $x \in \mathbb{R}^n$:

$$\Sigma_i(\theta, x) = \begin{cases} \Lambda(\theta, x) \sum_{j=1}^{i} \rho(\vartheta_j, \theta, x) & \text{if } i > 0 \\ 0 & \text{if } i = 0 \end{cases} \tag{4.4.5}$$

In addition, for $k = 0, 1, 2, \ldots$, with $\tau_0^b = 0$:

$$\tau_{k+1}^b \triangleq \inf\{t > \tau_k^b \mid (\theta_t^k, X_t^k) \in \partial E\} \tag{4.4.6}$$

$$\mathbb{P}\{\theta_{\tau_{k+1}^b}^{k+1} = \vartheta, X_{\tau_{k+1}^b}^{k+1} \in A \mid \theta_{\tau_{k+1}^b-}^k = \theta, X_{\tau_{k+1}^b-}^k = x\} = Q(\{\vartheta\} \times A; \theta, x) \tag{4.4.7}$$

for $A \in \mathcal{B}(\mathbb{R}^n)$, where Q is given by

$$Q(\{\vartheta\} \times A; \theta, x) = \rho(\vartheta, \theta, x) \int_{\mathbb{R}^q} \mathbf{1}_{\{(x + \psi(\vartheta, \theta, x, \underline{z})) \in A\}} \mu(d\underline{z}) \tag{4.4.8}$$

4.4.2 HSDE solution

This subsection shows that under a set of sufficient conditions H1-H8, the HSDE (4.4.1)-(4.4.8) has a pathwise unique solution. Note that the existence of a pathwise unique solution guarantees

the existence of a unique solution in probabilistic sense.

Proposition 4.3. *Let conditions H1-H8 below hold true. Let* $(\theta_0^*(\omega), X_0^*(\omega)) = (\theta_0, X_0) \in E$ *for all* ω*. Then for every initial condition* (θ_0, X_0)*, Equations (4.4.1)-(4.4.8) have a pathwise unique solution* $\{\theta_t^*, X_t^*\}$ *which is càdlàg and adapted and is a semi-martingale assuming values in the hybrid state space* E*.*

H1 For all $\theta \in \mathbb{M}$ there exists a constant $K(\theta)$ such that for all $x \in \mathbb{R}^n$, $|f(\theta, x)|^2 + \|g(\theta, x))\|^2 \leq K(\theta)(1 + |x|^2)$, where $|a|^2 = \sum_i (a_i)^2$ and $||b||^2 = \sum_{i,j} (b_{ij})^2$.

H2 For all $r \in \mathbb{N}$ and for all $\theta \in \mathbb{M}$ there exists a constant $L_r(\theta)$ such that for all x and y in the ball $\{z \in \mathbb{R}^n \mid |z| \leq r + 1\}$, $|f(\theta, x) - f(\theta, y)|^2 + \|g(\theta, x) - g(\theta, y)\|^2 \leq L_r(\theta)|x - y|^2$.

H3 For each $\theta \in \mathbb{M}$, the mapping $\Lambda(\theta, \cdot) : \mathbb{R}^n \to [0, \infty)$ is continuous and bounded, with upper bound a constant R_Λ.

H4 For all $(\theta, \vartheta) \in \mathbb{M}^2$, the mapping $\rho(\vartheta, \theta, \cdot) : \mathbb{R}^n \to [0, \infty)$ is continuous.

H5 For all $r \in \mathbb{N}$ there exists a constant $M_r(\theta)$ such that

$$\sup_{|x| \leq r} \int_{\mathbb{R}^q} |\psi(\vartheta, \theta, x, \underline{z})| \mu(d\underline{z}) \leq M_r(\theta), \text{ for all } \vartheta, \theta \in \mathbb{M}$$

H6 $|\psi(\theta, \theta, x, \underline{z})| = 0$ or > 1 for all $\theta \in \mathbb{M}$, $x \in \mathbb{R}^n$, $\underline{z} \in \mathbb{R}^q$.

H7 $\{(\theta_t^*, X_t^*)\}$ hits the boundary ∂E a finite number of times on any finite time interval.

H8 $|\vartheta_i - \vartheta_j| > 1$ for $i \neq j$ and $\vartheta_i, \vartheta_j \in \mathbb{M}$, with $|\cdot|$ a suitable metric well defined on $\mathbb{M}$. We could for example take for ϑ_i the ith unit vector of length N; this yields $|\vartheta_i - \vartheta_j| = \sqrt{2} > 1$ for $i \neq j$ and $|\vartheta_i| = 1$ for all i.

Proof. [Blo03] has used [LM76] to prove a version of Proposition 4.3 where $E = \mathbb{M} \times \mathbb{R}^n$, i.e., there are no boundaries with instantaneous jumps (forced jumps). Subsequently, [BBEP03] have proven the proposition under H1-H8 and the additional condition that $\{\tau_k^b\}$ is a sequence of predictable stopping times. [KB05a, Kry06] have shown that this additional condition can be removed. The complete proof of Proposition 4.3 is provided in [BE09]. □

4.5 Hybrid stochastic differential equations into stochastically and dynamically coloured Petri nets

This section shows that under a few conditions, each hybrid stochastic differential equation can be represented by a stochastically and dynamically coloured Petri net, in such a way that the

HSDE-process and the SDCPN-process are probabilistically equivalent.

Theorem 4.4 (HSDE into SDCPN)**.** *Consider an arbitrary HSDE (4.4.1)-(4.4.8) with elements ($\mathbb{M}$, E, f, g, μ_{θ_0,X_0}, Λ, ψ, ρ, μ, p_P, $\{W_t\}$). If for each θ the stochastic differential equation $dX_t = f(\theta, X_t)dt + g(\theta, X_t)dW_t$ has a unique solution in probabilistic sense and if Λ is bounded, then the elements of this HSDE can be mapped into an SDCPN ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) satisfying R0–R4. If the resulting SDCPN is executed on a probability space endowed with sequences of standard Brownian motions (one sequence for each place) then the resulting SDCPN process and the HSDE solution process are probabilistically equivalent.*

Proof. Consider an arbitrary HSDE (4.4.1)-(4.4.8) with elements ($\mathbb{M}$, E, f, g, θ_0, X_0, Λ, ψ, ρ, μ, p_P, $\{W_t\}$. Next, we prove Theorem 4.4 by the following steps:

- (Construction of SDCPN elements.) First, we assume that the stochastic differential equations defined by f and g have probabilistically unique solutions and that Λ is bounded. We characterise SDCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) in terms of HSDE elements ($\mathbb{M}$, E, f, g, θ_0, X_0, Λ, ψ, ρ, μ, p_P, $\{W_t\}$). The thus constructed SDCPN is referred to as SDCPN$^{\text{HSDE}}$.

- (Probabilistic equivalence.) The execution of the SDCPN$^{\text{HSDE}}$ elements provides the SDCPN$^{\text{HSDE}}$ stochastic process. We verify that SDCPN rules R0–R4 hold true for the SDCPN$^{\text{HSDE}}$. Finally, we show that the SDCPN$^{\text{HSDE}}$ stochastic process is probabilistically equivalent to the stochastic process defined by the original HSDE (4.4.1)-(4.4.8).

Notice that Theorem 4.4 does not assume that HSDE conditions H1-H8 are necessarily satisfied.

4.5.1 Construction of SDCPN$^{\text{HSDE}}$ elements

We assume that for each θ the stochastic differential equation $dX_t = f(\theta, X_t)dt + g(\theta, X_t)dW_t$ has a unique solution in probabilistic sense and that Λ is bounded. We provide an into-mapping that characterises SDCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) in terms of HSDE elements ($\mathbb{M}$, E, f, g, θ_0, X_0, Λ, ψ, ρ, μ, p_P, $\{W_t\}$).

$\mathcal{P} = \{P_\theta; \theta \in \mathbb{M}\}$. Hence, for each $\theta \in \mathbb{M}$, there is one place P_θ. The places are ordered $P_{\vartheta_1}, \ldots, P_{\vartheta_N}$ according to $\mathbb{M} = \{\vartheta_1, \ldots, \vartheta_N\}$. Since $\mathbb{M}$ is finite, $\mathcal{P}$ is finite as well, which satisfies SDCPN definitions.

$\mathcal{T} = \mathcal{T}_G \cup \mathcal{T}_D \cup \mathcal{T}_I$, with $\mathcal{T}_I = \emptyset$, $\mathcal{T}_G = \{T^G_\theta; \theta \in \mathbb{M}\}$, $\mathcal{T}_D = \{T^D_\theta; \theta \in \mathbb{M}\}$. Hence, for each $\theta \in \mathbb{M}$ there is one guard transition T^G_θ and one delay transition T^D_θ.

$\mathcal{A} = \mathcal{A}_O \cup \mathcal{A}_E \cup \mathcal{A}_I$, with $|\mathcal{A}_I| = 0$, $|\mathcal{A}_E| = 0$, and $|\mathcal{A}_O| = 2N + 2N^2$, where $N = |\mathbb{M}|$. Hence, there are no inhibitor arcs or enabling arcs in this SDCPN$^{\text{HSDE}}$ constructed, and the number of ordinary arcs is $2N + 2N^2$.

$\mathcal{N}$**:** The node function maps each arc in $\mathcal{A} = \mathcal{A}_O$ to a pair of nodes. These connected pairs of nodes are: $\{(P_\theta, T_\theta^G); \theta \in \mathbb{M}\} \cup \{(P_\theta, T_\theta^D); \theta \in \mathbb{M}\} \cup \{(T_\theta^G, P_\vartheta); \theta, \vartheta \in \mathbb{M}\} \cup \{(T_\theta^D, P_\vartheta); \theta, \vartheta \in \mathbb{M}\}$. Hence, each place P_θ ($\theta \in \mathbb{M}$) has two outgoing arcs: one to guard transition T_θ^G and one to delay transition T_θ^D. Each transition has N outgoing arcs: one arc to each place in $\mathcal{P}$.

$\mathcal{S} = \{\mathbb{R}^n\}$.

$\mathcal{C}$**:** For all $\theta \in \mathbb{M}$, $\mathcal{C}(P_\theta) = \mathbb{R}^n$

$\mathcal{I}$**:** For all $\theta_0 \in \mathbb{M}$ and $X_0 \in \mathcal{C}(P_{\theta_0})$, $\mathcal{I}(M^{\theta_0}, X_0) = \mu_{\theta_0, X_0}(\theta_0, X_0)$, where M^θ is the $|\mathcal{P}|$-dimensional vector that has a one at the element corresponding to place P_θ and zeros elsewhere. Hence, with probability $\mu_{\theta_0, X_0}(\theta_0, X_0)$, place P_{θ_0} initially gets one token with colour X_0 while all other places initially get zero tokens.

$\mathcal{V}$**:** For all $\theta \in \mathbb{M}$, $\mathcal{V}_{P_\theta}(\cdot) = f(\theta, \cdot)$.

$\mathcal{W}$**:** For all $\theta \in \mathbb{M}$, $\mathcal{W}_{P_\theta}(\cdot) = g(\theta, \cdot)$. Since it was assumed that for each θ the stochastic differential equation $dX_t = f(\theta, X_t)dt + g(\theta, X_t)dW_t$ has a unique solution in probabilistic sense, we find that the same holds true for $dC_t = \mathcal{V}_{P_\theta}(C_t)dt + \mathcal{W}_{P_\theta}(C_t)dB_t^{i,P_\theta}$, which satisfies SDCPN definitions.

$\mathcal{G}$**:** For all $\theta \in \mathbb{M}$, $\mathcal{G}_{T_\theta^G} = E_\theta$.

$\mathcal{D}$**:** For all $\theta \in \mathbb{M}$, $\mathcal{D}_{T_\theta^D}(\cdot) = \Lambda(\theta, \cdot)$. Since we assumed that Λ is bounded, say $\Lambda(\theta, \cdot) \leq R_\Lambda$, we find that $\mathcal{D}_{T_\theta^D}(\cdot)$ is bounded as well, and its upperbound is $R_\mathcal{D} = R_\Lambda$. This boundedness implies local integrability, which satisfies SDCPN definitions.

$\mathcal{F}$**:** For all $T \in \mathcal{T}$, define $e_T^{\vartheta'}$ as the vector of length N containing a one at the component corresponding with the arc from transition T to place $P_{\vartheta'}$ and zeros elsewhere. Then for all $\theta \in \mathbb{M}$, and for $T \in \{T_\theta^G, T_\theta^D\}$, $\mathcal{F}_T(e_T^{\vartheta'}, x'; x) = F_T^Q(\vartheta', x'; \theta, x)$, for all $x \in E_\theta \cup \partial E_\theta$, $\vartheta' \in \mathbb{M}$ and $x' \in E_{\vartheta'}$, where F_T^Q is defined through

$$F_T^Q(\{\vartheta'\} \times A'; \theta, x) = \rho(\vartheta', \theta, x) \int_{\mathbb{R}^q} \mathbf{1}_{\{(x + \psi(\vartheta', \theta, x, \underline{z})) \in A'\}} \mu(d\underline{z}) \tag{4.5.9}$$

4.5.2 Probabilistic equivalence

The execution of the SDCPN$^{\text{HSDE}}$ elements on a probability space endowed with sequences of standard Brownian motions $\{B_t^{i,P}; i = 1, 2, \ldots\}$ provides a SDCPN$^{\text{HSDE}}$ stochastic process. This execution makes use of an independent sequence $\{U_i; i = 0, 1, \ldots\}$ of independent uniform $U[0, 1]$ random variables and the rules R0–R4.

We show that the SDCPN$^{\text{HSDE}}$ stochastic process is probabilistically equivalent to the stochastic process defined by the original HSDE. This is done by showing:

- Equivalence of initial states
- Equivalence of continuous evolution until first jump
- Equivalence of time of jumps
- Equivalence of size of jumps
- Equivalence of processes after the first jump

Equivalence of initial states The initial marking of the SDCPN$^{\text{HSDE}}$ is defined by $\mathcal{I}(M^{\theta_0}, X_0) = \mu_{\theta_0,X_0}(\theta_0, X_0)$, where M^θ is the N-dimensional vector that has a one at the element corresponding to place P_θ and zeros elsewhere. Therefore, with probability $\mathcal{I}(M^{\theta_0}, X_0)$, at time $t = \tau_0$ there is one token in place P_{θ_0} which has colour X_0. The initial state of the HSDE is (θ_0, X_0) with probability $\mu_{\theta_0,X_0}(\theta_0, X_0)$. Due to the mapping between the places $P_\theta \in \mathcal{P}$ and the modes $\theta \in \mathbb{M}$, the initial states of SDCPN$^{\text{HSDE}}$ and HSDE are probabilistically equivalent.

Equivalence of continuous evolution until first jump The continuous part of the SDCPN$^{\text{HSDE}}$ stochastic process equals the vector that collects all token colours. Since there is only one token in the constructed SDCPN$^{\text{HSDE}}$ at all times, this vector equals the colour of this single token. Until the first jump, this colour follows the stochastic differential equation $dC_t^{P_{\theta_0}} = \mathcal{V}_{P_{\theta_0}}(C_t^{P_{\theta_0}})dt + \mathcal{W}_{P_{\theta_0}}(C_t^{P_{\theta_0}})dB_t^{i,P_{\theta_0}}$ which has probabilistically unique solution $C_t^{P_{\theta_0}}$.

In the original HSDE solution process, the continuous process until the first jump follows stochastic differential equation $dX_t^0 = f(\theta_t^0, X_t^0)dt + g(\theta_t^0, X_t^0)dW_t + \int_{\mathbb{R}^q} \psi(\theta_t^0, \theta_{t-}^0, X_{t-}^0, \underline{z})\, p_P(dt, (0, \Lambda(\theta_{t-}^0, X_{t-}^0)] \times d\underline{z})$ where $d\theta_t^0 = \sum_{i=0}^N (\vartheta_i - \theta_{t-}^0) p_P(dt, (\Sigma_{i-1}(\theta_{t-}^0, X_{t-}^0), \Sigma_i(\theta_{t-}^0, X_{t-}^0)] \times \mathbb{R}^q)$. Until the first jump, the Poisson terms in the stochastic differential equations above are equal to zero. What remains is: $d\theta_t^0 = 0$ and $dX_t^0 = f(\theta_t^0, X_t^0)dt + g(\theta_t^0, X_t^0)dW_t$, which are assumed to have a probabilistically unique solution θ_t^0 and X_t^0.

Due to equivalence of initial states $M^{\theta_0} \equiv \theta_0$ and $C_0 = X_0$, equivalence of drift coefficients $\mathcal{V}_{P_{\theta_0}}(\cdot) = f(\theta_0, \cdot)$, equivalence of diffusion coefficients $\mathcal{W}_{P_{\theta_0}}(\cdot) = g(\theta_0, \cdot)$ and probabilistic

equivalence of $\{B_t^{i,P}\}$ and $\{W_t\}$, as long as no jumps occur, we derive that for $t \geq \tau_0 = 0$, $M^{\theta_0} = \theta_t^0$ and $X_t^0 = C_t^{P_{\theta_0}}$ in probabilistic sense.

Equivalence of time of jumps For the SDCPN$^{\text{HSDE}}$, for each arbitrary place in which the initial token may reside, two transitions are pre-enabled: a guard transition and a delay transition. If either of them becomes enabled and fires, then the other becomes disabled. The time until the guard transition is enabled is $t_*^G(M^{\theta_0}, C_0) \triangleq \inf\{t - \tau_0 > 0 \mid C_t^{P_{\theta_0}} \in \partial\mathcal{G}_{T_{\theta_0}^G}\}$. The time until the delay transition is enabled is $\sigma_1^{T_{\theta_0}^D} = D_{T_{\theta_0}^D}^{qf}(U_1)$, with $D_{T_{\theta_0}^D}^{qf}(u) = \inf\{t - \tau_0 \mid \exp(-\int_{\tau_0}^t \mathcal{D}_{T_{\theta_0}^D}(C_s^{P_{\theta_0}})ds) \leq u\}$ and $U_1 \sim U[0,1]$.

The verification if SDCPN rules R0–R4 hold true in the construction above is according to the same reasoning as for DCPN$^{\text{PDP}}$, see Section 3.5.2: Since there are no immediate transitions in the constructed SDCPN$^{\text{HSDE}}$ instantiation, rule R0 holds true. Since there is only one token in the constructed SDCPN$^{\text{HSDE}}$ instantiation, R1–R3 also hold true. Rule R4 is in effect when for particular θ, transitions T_θ^G and T_θ^D become enabled at exactly the same time. Since $\mathcal{D}_{T_\theta^D}$ is integrable, the probability that this occurs is zero, yielding that R4 holds with probability one. However, if this event should occur, then due to $\mathcal{F}_{T_\theta^G} = \mathcal{F}_{T_\theta^D}$, the application of rule R4 has no effect on the path of the SDCPN$^{\text{HSDE}}$ stochastic process.

For HSDE, from Equation (4.4.6), using $k = 0$ and $\tau_0^b = \tau_0$, the time at which the continuous state first hits the boundary of its state space is $\tau_1^b \triangleq \inf\{t > \tau_0 \mid (\theta_t^0, X_t^0) \in \{\{\theta\} \times \partial E_\theta; \theta \in \mathbb{M}\}\}$. It is easily seen that as long as $\theta_t^0 = \theta_0$, then due to $X_t^0 = C_t^{P_{\theta_0}}$ and the equality $\partial\mathcal{G}_{T_{\theta_0}^G} = \partial E_{\theta_0}$, we have that $\inf\{t > \tau_0 \mid (\theta_t^0, X_t^0) \in \{\{\theta\} \times \partial E_\theta; \theta \in \mathbb{M}\}\} = \tau_0 + \inf\{t - \tau_0 > 0 \mid C_t^{P_{\theta_0}} \in \partial\mathcal{G}_{T_{\theta_0}^G}\}$, hence $\tau_1^b = \tau_0 + t_*^G(M^{\theta_0}, C_0)$. However, there is a possibility that at some time $\tau_1^p < \tau_1^b$, the HSDE solution process state makes a jump due to the Poisson random measure generating a point: Consider Equations (4.4.3) and (4.4.4), for $k = 0$. A jump is generated when $\sum_{i=1}^N (\vartheta_i - \theta_{t-}^0) p_P(dt, (\Sigma_{i-1}(\theta_{t-}^0, X_{t-}^0), \Sigma_i(\theta_{t-}^0, X_{t-}^0)] \times \mathbb{R}^q) \neq 0$ or when $\int_{\mathbb{R}^q} \psi(\theta_t^0, \theta_{t-}^0, X_{t-}^0, \underline{z}) p_P(dt, (0, \Lambda(\theta_{t-}^0, X_{t-}^0)] \times d\underline{z}) \neq 0$, or both. Consider the Poisson random measure in Equation (4.4.4), i.e., $p_P(dt, (0, \Lambda(\theta_{t-}^0, X_{t-}^0)] \times d\underline{z})$, which is equal to zero, except at singular times when it generates a multivariate point $(\{\tau_1^p\}, \{z_1\}, \{\underline{z}\})$. In Section 4.2 it is explained that due to the Poisson random measure being homogeneous and due to $\Lambda(\theta_{t-}^0, X_{t-}^0) \leq R_\Lambda$, the point $(\{\tau_1^p\}, \{z_1\}, \{\underline{z}\})$ is generated as follows: Generate a triple $(\varepsilon_1, \nu_1, \underline{\nu}_1)$, with $\varepsilon_1 \sim \text{Exp}(R_\Lambda)$, $\nu_1 \sim U[0, R_\Lambda]$ and $\underline{\nu}_1 \sim \mu$. Accept this triple if $\nu_1 \leq \Lambda(\theta_{\tau_0+\varepsilon_1-}^0, X_{\tau_0+\varepsilon_1-}^0)$, otherwise reject it. If it is accepted then $\tau_1^p = \tau_0 + \varepsilon_1$, $z_1 = \nu_1$ and $\underline{z} = \underline{\nu}_1$. If it is not accepted then another triple $(\varepsilon_2, \nu_2, \underline{\nu}_2)$ is generated with $\varepsilon_2 \sim \text{Exp}(R_\Lambda)$, $\nu_2 \sim U[0, R_\Lambda]$ and $\underline{\nu}_2 \sim \mu$, and this triple is accepted if $\nu_2 \leq \Lambda(\theta_{\tau_0+\varepsilon_1+\varepsilon_2-}^0, X_{\tau_0+\varepsilon_1+\varepsilon_2-}^0)$. If it is accepted then $\tau_1^0 = \tau_0 + \varepsilon_1 + \varepsilon_2$, $z_1 = \nu_2$ and $\underline{z} = \underline{\nu}_2$. If it is not accepted then another triple $(\varepsilon_3, \nu_3, \underline{\nu}_3)$ is generated, and so on. Hence if $(\varepsilon_r, \nu_r, \underline{\nu}_r)$ is the first triple that is accepted then $\tau_1^p = \tau_0 + \sum_{n=1}^r \varepsilon_n$ and $z_1 = \nu_r$ and $\underline{z} = \underline{\nu}_r$. Due to Proposition 4.2

we find that the interarrival times of the triples accepted through this mechanism are exponential with intensity Λ. In addition, due to $\mathcal{D}_{T_\theta^D}(\cdot) = \Lambda(\theta, \cdot)$, we find that $\tau_1^p - \tau_0$ is probabilistically equivalent to $\sigma_1^{T_{\theta_0}^D}$.

For HSDE, the time of the first jump is equal to the minimum of τ_1^b and τ_1^p. Due to the reasoning above, this time of first jump is probabilistically equivalent to the time of first jump of the SDCPN$^{\text{HSDE}}$.

Equivalence of size of jumps For the SDCPN$^{\text{HSDE}}$, the jump size is determined by the firing measure $\mathcal{F}_T$ of the enabled transition T: for all $\theta \in \mathbb{M}$ and $T \in \{T_\theta^G, T_\theta^D\}$, $\mathcal{F}_T(e_T^{\vartheta'}, x'; x) = F_T^Q(\vartheta', x'; \theta, x)$, for all $x \in E_\theta \cup \partial E_\theta$, $\vartheta' \in \mathbb{M}$ and $x' \in E_{\vartheta'}$, where F_T^Q is defined through

$$F_T^Q(\{\vartheta'\} \times A'; \theta, x) = \rho(\vartheta', \theta, x) \int_{\mathbb{R}^q} \mathbf{1}_{\{(x+\psi(\vartheta', \theta, x, \underline{z})) \in A'\}} \mu(d\underline{z})$$

For HSDE, the size of jumps is generated as follows: In case of a jump generated by Poisson random measure at time $t = \tau_1^p$, the size of jump in $\{\theta_t^0\}$ is given by

$$\theta_{\tau_1^p}^0 - \theta_{\tau_1^p-}^0 = \sum_{i=1}^{N} (\vartheta_i - \theta_{\tau_1^p-}^0) p_P(dt, (\Sigma_{i-1}(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0), \Sigma_i(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0)] \times \mathbb{R}^q)$$

and the size of jump in $\{X_t^0\}$ is given by

$$X_{\tau_1^p}^0 - X_{\tau_1^p-}^0 = \int_{\mathbb{R}^q} \psi(\theta_{\tau_1^p}^0, \theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0, \underline{z}) p_P(dt, (0, \Lambda(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0)] \times d\underline{z})$$

Now use that the Poisson random measure has generated a point $(\{\tau_1^p\}, \{z_1\}, \{\underline{z}\})$, with $z_1 = \nu_r$ and $\underline{z} = \underline{\nu}_r$ as described above. Random variable z_1 is used as follows: Notice that by Equation (4.4.5) and definition of ρ, for all $\theta \in \mathbb{M}$ and all $x \in \mathbb{R}^n$, the interval $(0, \Lambda(\theta, x)]$ is divided into subintervals $(\Sigma_{i-1}(\theta, x), \Sigma_i(\theta, x)]$, i.e., $(0, \Lambda(\theta, x)] = (\Sigma_0(\theta, x), \Sigma_1(\theta, x)] \cup (\Sigma_1(\theta, x), \Sigma_2(\theta, x)] \cup \cdots \cup (\Sigma_{N-1}(\theta, x), \Sigma_N(\theta, x)]$, where $\Sigma_0(\theta, x) = 0$ and $\Sigma_N(\theta, x) = \Lambda(\theta, x)$. The ith interval, i.e., $(\Sigma_{i-1}(\theta, x), \Sigma_i(\theta, x)]$ has a weight $\rho(\vartheta_i, \theta, x) = (\Sigma_i(\theta, x) - \Sigma_{i-1}(\theta, x))/\Lambda(\theta, x)$, with $\sum_{i=1}^N \rho(\vartheta_i, \theta, x) = 1$. Due to $z_1 \in (0, \Lambda(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0)]$, there exists $j \in \{1, \ldots, N\}$ such that $z_1 \in (\Sigma_{j-1}(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0), \Sigma_j(\theta_{\tau_1^p}^0, X_{\tau_1^p}^0)]$. This makes $p_P(dt, (\Sigma_{i-1}(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0), \Sigma_i(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0)] \times \mathbb{R}^q) = 1$ if $i = j$ and $= 0$ for $i \neq j$. Therefore, $\theta_{\tau_1^p}^0 - \theta_{\tau_1^p-}^0 = \vartheta_j - \theta_{\tau_1^p-}^0$, i.e., at time τ_1^p, θ_t jumps from $\theta_{\tau_1^p-}^0 = \theta_0$ to $\theta_{\tau_1^p}^0 = \vartheta_j$. Next, the random variable $\underline{z}$ is used to determine $X_{\tau_1^p}^0 - X_{\tau_1^p-}^0$, i.e., in $(\{\tau_1^p\}, \{z_1\}, \{\underline{z}\})$, $p_P(dt, (0, \Lambda(\theta_{t-}^0, X_{t-}^0)] \times d\underline{z}) = 1$ and is zero elsewhere. Therefore, $X_{\tau_1^p}^0 - X_{\tau_1^p-}^0 = \psi(\vartheta_j, \theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0, \underline{z})$. This gives that at time τ_1^p, X_t^0 jumps from $X_{\tau_1^p-}^0$ to $X_{\tau_1^p-}^0 + \psi(\vartheta_j, \theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0, \underline{z})$. From this, we find that the probability for (θ_t^0, X_t^0) to jump into $(\{\vartheta_j\}, A)$ $(= \{\vartheta_j\} \times A)$, given that the state right before the jump is $(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0)$, is

equal to the probability that z_1 is in $(\Sigma_{j-1}(\theta^0_{\tau_1^p-}, X^0_{\tau_1^p-}), \Sigma_j(\theta^0_{\tau_1^p-}, X^0_{\tau_1^p-})]$, times the probability that $X^0_{\tau_1^p-} + \psi(\vartheta_j, \theta^0_{\tau_1^p-}, X^0_{\tau_1^p-}, \underline{z})$ is in A. This probability is equal to

$$\rho(\vartheta_j, \theta^0_{\tau_1^p-}, X^0_{\tau_1^p-}) \int_{\mathbb{R}^q} \mathbf{1}_{\{(X^0_{\tau_1^p-} + \psi(\vartheta_j, \theta^0_{\tau_1^p-}, X^0_{\tau_1^p-}, \underline{z})) \in A\}} \mu(d\underline{z})$$

which is equal to $Q(\{\vartheta_j\} \times A; \theta^0_{\tau_1^p-}, X^0_{\tau_1^p-})$, according to Equation (4.4.8).

For boundary hitting type of jumps, the size of jump is given by Equation (4.4.7), i.e.,

$$\mathbb{P}\{\theta^1_{\tau_1^b} = \vartheta, X^1_{\tau_1^b} \in A \mid \theta^0_{\tau_1^b-} = \theta, X^0_{\tau_1^b-} = x\} = Q(\{\vartheta\} \times A; \theta, x)$$

This shows that the jump size mechanisms for Poisson random measure type of jumps and boundary hitting type of jumps are the same. Also note that for all ϑ', x', θ and x, and $T \in \mathcal{T}_D \cup \mathcal{T}_G$, $F^Q_T(\vartheta', x'; \theta, x) = Q(\vartheta', x'; \theta, x)$. This means that the SDCPN$^{\text{HSDE}}$ state after the jump and the HSDE solution process state after the jump are probabilistically equivalent.

Equivalence of processes after the first jump From $\tau_1 = \min\{\tau_1^b, \tau_1^p\}$ onwards, the probabilistic equivalence of the HSDE and SDCPN$^{\text{HSDE}}$ processes is shown in the same way. If $\tau_1 = \tau_1^p$, then Equations (4.4.3) and (4.4.4) are used for $k = 0$; if $\tau_1 = \tau_1^b$ then these equations are used for $k = 1$. From stopping time τ_{n-1} to stopping time τ_n the HSDE-process and the associated SDCPN$^{\text{HSDE}}$ process have probabilistically equivalent paths and probabilistically equivalent stopping times. Due to the unique definition of the SDCPN$^{\text{HSDE}}$ stochastic process at times when transitions fire, the SDCPN$^{\text{HSDE}}$ state at stopping times is also equivalent to the HSDE-process state at the stopping times and both processes are càdlàg.

This completes the proof of Theorem 4.4. □

4.6 Stochastically and dynamically coloured Petri nets into hybrid stochastic differential equations

This section shows that under a few conditions, any stochastically and dynamically coloured Petri net can be represented by a hybrid stochastic differential equation, in such a way that the SDCPN-process and the HSDE-process are probabilistically equivalent.

Theorem 4.5 (SDCPN into HSDE)**.** *Consider an arbitrary SDCPN* $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{W}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ *satisfying R0–R4. If the initial marking does not enable a transition, if none of the transition firings enable a guard transition, if the delay rates* $\mathcal{D}_T$ *are bounded, and if the number of tokens remains finite for* $t \to \infty$*, then this SDCPN can be mapped into an HSDE (4.4.1)-(4.4.8) with elements* $(\mathbb{M}, E, f, g, \mu_{\theta_0, X_0}, \Lambda, \psi, \rho, \mu, p_P, \{W_t\})$*, provided* μ *is given. If the original*

SDCPN is executed on a probability space which is endowed with sequences of standard Brownian motions (one sequence for each place), then the resulting HSDE solution process and the SDCPN process are probabilistically equivalent. If in addition, conditions S1-S6 below are satisfied, then conditions H1-H8 are satisfied for the resulting HSDE.

S1 *For all $r \in \mathbb{N}$ and for all $P \in \mathcal{P}$, there exist K_P^v, $L_{r,P}^v$, K_P^w and $L_{r,P}^w$ such that for all $c \in \mathcal{C}(P)$ and any a, b in the ball $\{z \in \mathcal{C}(P) \mid |z| \leq r+1\}$,*

- $|\mathcal{V}_P(c)|^2 \leq K_P^v(1+|c|^2)$
- $|\mathcal{V}_P(b) - \mathcal{V}_P(a)|^2 \leq L_{r,P}^v |b-a|^2$
- $\|\mathcal{W}_P(c)\|^2 \leq K_P^w(1+|c|^2)$
- $\|\mathcal{W}_P(b) - \mathcal{W}_P(a)\|^2 \leq L_{r,P}^w |b-a|^2$.

S2 *If (M_t, C_t) denotes the SDCPN marking at time t and τ denotes a time at which one or more transitions fire, then there exists $R_S(M_t) < \infty$ such that $|\overline{C}_\tau - \overline{C}_{\tau-}| = 0$ or $\in (1, R_S(M_{\tau-})]$. Here, $\overline{C}_t$ denotes a vector C_t with a sufficient number of zeros added so that $\overline{C}_\tau$ and $\overline{C}_{\tau-}$ have the same number of vector elements.*

S3 *For all $T \in \mathcal{T}_D$, $\mathcal{D}_T$ is continuous.*

S4 *In a finite time interval, each guard transition is expected to fire a finite number of times.*

S5 *For all $T \in \mathcal{T}$, $c \in \mathcal{C}(P(A_{in,OE}(T)))$ and $e \in \{0,1\}^{|A_{out}(T)|}$, $\mathcal{F}_T(e, \cdot; c)$ is continuous.*

S6 *If $\mathcal{M} = \{M_t \mid (M_t, C_t)$ is a reachable marking, $t \geq 0\}$ is the set of reachable token distributions, then for all $M^i, M^j \in \mathcal{M}$, $M^i \neq M^j$, $|M^i - M^j| > 1$.*

Proof. Consider an arbitrary SDCPN $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{W}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ that satisfies rules R0–R4. Next we prove Theorem 4.5 by the following steps:

- (Construction of HSDE elements.) It is assumed that the initial marking does not enable a transition, that none of the transition firings enable a guard transition, that the delay rates are bounded, and that the number of tokens remains finite for $t \to \infty$. We characterise the HSDE elements $(\mathbb{M}, E, f, g, \theta_0, X_0, \Lambda, \psi, \rho, \mu, p_P, \{W_t\})$ in terms of SDCPN elements, provided μ is given. The thus constructed HSDE is referred to as HSDE$^{\text{SDCPN}}$.

- (Probabilistic equivalence.) The solution of the HSDE$^{\text{SDCPN}}$ (4.4.1)-(4.4.8) provides the HSDE$^{\text{SDCPN}}$-process, i.e., the GSHP. We show that the HSDE$^{\text{SDCPN}}$ stochastic process is probabilistically equivalent to the stochastic process defined by the original SDCPN.

- (Verification of H1-H8.) Finally, we show that if conditions S1-S6 are satisfied for the original SDCPN, then HSDE conditions H1-H8 hold true for the constructed HSDE$^{\text{SDCPN}}$.

With this mapping, the constructed HSDE$^{\text{SDCPN}}$-process discrete state θ_t will be a vector of length $|\mathcal{P}|$, which counts the number of tokens in each place at time t. The HSDE$^{\text{SDCPN}}$-process continuous state X_t will be formed by a vector that contains the colours of all tokens in the SDCPN at time t. This X_t evolves through time according to the combined token colour functions; the HSDE$^{\text{SDCPN}}$-process jumps correspond with SDCPN transitions firing, which may change the distribution of tokens among places (θ_t) and the colours of the tokens (X_t).

4.6.1 Construction of HSDE$^{\text{SDCPN}}$ elements

It is assumed that the initial marking does not enable a transition, that none of the transition firings enable a guard transition, that the delay rates $\mathcal{D}_T$ are bounded, and that the number of tokens remains finite for $t \to \infty$. We provide an into-mapping that characterises HSDE$^{\text{SDCPN}}$ elements ($\mathbb{M}$, E, f, g, θ_0, X_0, Λ, ψ, ρ, μ, p_P, $\{W_t\}$) in terms of SDCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$).

The construction is largely similar to the construction of PDP$^{\text{DCPN}}$ elements in terms of DCPN elements (Section 3.6). However, in addition to the diffusion term and the Poisson random measure, there is one other difference between PDP and HSDE-processes: For PDP, the dimension of the continuous-valued process $\{X_t\}$ is a function of the current value of θ_t, i.e., $X_t \in \mathbb{R}^{d(\theta)}$ if $\theta_t = \theta$. For HSDE, $X_t \in \mathbb{R}^n$, with n a constant. In the following, we first construct all HSDE$^{\text{SDCPN}}$ elements in a similar way as for PDP$^{\text{DCPN}}$, and subsequently, we add a sufficient number of zeros to some elements in order to create a constant dimension n for $\{X_t\}$.

$\mathbb{M}$ The characterisation of $\mathbb{M}$ in terms of SDCPN elements is by means of the reachability graph (RG), in the same way as for the characterisation of PDP$^{\text{DCPN}}$ elements in terms of DCPN elements, see Section 3.6. The nodes in the RG are written as row vectors $(m_1, \ldots, m_{|\mathcal{P}|})$, where m_i is the number of tokens in place P_i. These nodes are sometimes referred to as 'token distributions'. Arrows between nodes are labelled by transitions, and indicate how the number of tokens in the places change due to transition firings. Then $\mathbb{M}$ is composed of the non-vanishing nodes, i.e., is composed of those nodes in the reachability graph that do not enable an immediate transition. Due to the condition that the number of tokens remains finite for $t \to \infty$, $\mathbb{M}$ is a finite set; $N = |\mathbb{M}|$.

E For each $\theta \in \mathbb{M}$, corresponding with node $m = (m_1, \ldots, m_{|\mathcal{P}|})$ in the RG, define $d(\theta) = \sum_{i=1}^{|\mathcal{P}|} m_i n(P_i)$, where $n(P_i)$ is defined through $\mathcal{C}(P_i) = \mathbb{R}^{n(P_i)}$. If under token distribution θ, no guard transitions are pre-enabled, then $E_\theta = \mathbb{R}^{d(\theta)}$. If under token distribution θ, one or more guard transitions are pre-enabled, then $E_\theta = \mathbb{R}^{d(\theta)} \setminus \partial E_\theta$, where ∂E_θ is constructed as follows:

Without loss of generality, suppose that under token distribution θ, the multi-set of pre-enabled guard transitions is $T_1, \ldots, T_k$. This set may contain one transition multiple times, if such transition evaluates multiple input token vectors in parallel. Suppose $\{P_{i_1}, \ldots, P_{i_{r_i}}\} = P(A_{in,OE}(T_i))$ are the input places of T_i that are connected to T_i by means of ordinary or enabling arcs. This set may contain one place multiple times if such place is connected to T_i by multiple arcs (input arcs of T_i). Define $d_i = \sum_{j=1}^{r_i} n(P_{i_j})$, then $\partial E_\theta = \partial \mathcal{G}'_{T_1} \cup \ldots \cup \partial \mathcal{G}'_{T_k}$, where $\mathcal{G}'_{T_i} = [[\mathcal{G}_{T_i} \times \mathbb{R}^{d(\theta)-d_i}]] \in \mathbb{R}^{d(\theta)}$. Here $[[\cdot]]$ denotes a special ordering of all vector elements: Vector elements are ordered according to the unique ordering of places and to the unique ordering of tokens within their place defined for SDCPN. Finally, $E = \{\{\theta\} \times E_\theta; \theta \in \mathbb{M}\}$.

f For each $\theta \in \mathbb{M}$ and $x \in E_\theta$, $f(\theta, x) = \text{Col}_{i=1}^{|\mathcal{P}|} \left\{\text{Col}_{j=1}^{m_i}\{\mathcal{V}_{P_i}(c_{ij})\}\right\}$, where $x = \text{Col}_{i=1}^{|\mathcal{P}|} \left\{\text{Col}_{j=1}^{m_i}\{c_{ij}\}\right\}$ and θ corresponds to $(m_1, \ldots, m_{|\mathcal{P}|})$. Since $\mathcal{V}_P$ are measurable mappings, f is measurable.

g**:** For each $\theta \in \mathbb{M}$ and $x \in E_\theta$,
$g(\theta, x) = \text{Row}\{\text{Diag}_{i=1}^{|\mathcal{P}|} \left\{\text{Diag}_{j=1}^{m_i}\{\mathcal{W}_{P_i}(c_{ij})\}\right\}, O^{\Sigma_{i=1}^{|\mathcal{P}|}(m_i^{max}-m_i)h(P_i)}\}$, where

- $x = \text{Col}_{i=1}^{|\mathcal{P}|} \left\{\text{Col}_{j=1}^{m_i}\{c_{ij}\}\right\}$
- $O^{\Sigma_{i=1}^{|\mathcal{P}|}(m_i^{max}-m_i)h(P_i)}$ is a square matrix of dimension $(\Sigma_{i=1}^{|\mathcal{P}|}(m_i^{max} - m_i)h(P_i)) \times (\Sigma_{i=1}^{|\mathcal{P}|}(m_i^{max} - m_i)h(P_i))$ that contains only zeros. In the $g(\theta, \cdot)$ constructed above it is put to the right of the block that contains the matrices $\mathcal{W}_{P_i}$.
- $m_i^{max} = \max_{\theta \in \mathbb{M}}\{m_i \mid \theta = (m_1, \ldots, m_{|\mathcal{P}|})\}$ is the maximum number of tokens that exists in place P_i. This maximum m_i^{max} exists due to the condition that for $t \to \infty$ the number of tokens remains finite.

Since $\mathcal{W}_P$ are measurable mappings, g is measurable.

μ_{θ_0,X_0}**:** $\mu_{\theta_0,X_0}(M_0, C_0) = \mathcal{I}(M_0, C_0)$ for all M_0 and C_0, where $M_0 = (M_{1,0}, \ldots, M_{|\mathcal{P}|,0})$, with $M_{i,0}$ the initial number of tokens in place P_i, with the places ordered according to the unique ordering adopted for SDCPN, and $C_0 \in \mathbb{R}^{d(\theta_0)}$ containing the colours of these tokens. Due to the condition that no transitions are enabled in the initial marking (which prevents vanishing token distributions to be current at the initial time), the constructed M_0 and C_0 are uniquely defined, and $M_0 \in \mathbb{M}$ and $C_0 \in E_{\theta_0}$.

Λ**:** For each $\theta \in \mathbb{M}$ and $x \in E_\theta$, $\Lambda(\theta, x) = \sum_{n=1}^{k} \mathcal{D}_{T_n}(c^{T_n})$, where $T_1, \ldots, T_k$ refers to the multi-set of transitions in $\mathcal{T}_D$ that, under token distribution θ, are pre-enabled, and c^{T_n} are the respective elements of x that are used to pre-enable these transitions. This set $T_1, \ldots, T_k$ may contain one transition multiple times, if multiple input token vectors are evaluated in

parallel. If the set of pre-enabled delay transitions is empty in θ, then $\Lambda(\theta, \cdot) = 0$. Since $\mathcal{D}_T$ are locally integrable and bounded, Λ is measurable and bounded. The upperbound of Λ is $R_\Lambda = \max_{\theta \in \mathbb{M}}\{k \cdot R_\mathcal{D}$; the number of elements in the multi-set of transitions in $\mathcal{T}_D$ that, under token distribution θ, are pre-enabled $= k\}$.

ψ, ρ, μ We make use of the assumption that μ is given. As part of the construction, define a probability measure $P_Q(\theta', A; \theta, x)$, the value of which equals the probability that if a jump occurs, and if the value of the HSDE-process just prior to the jump is (θ, x), then the value of the HSDE-process just after the jump is in $\{\theta'\} \times A$. Probability $P_Q(\theta', A; \theta, x)$ is characterised in terms of the SDCPN by the reachability graph (RG), elements $\mathcal{D}$, $\mathcal{G}$ and Rules R0–R4 and the set $\mathcal{F}$. This is done in four steps, precisely following the characterisation of the PDP transition measure Q in terms of DCPN elements in Appendix 3.9. Next, we characterise ψ and ρ in terms of the results:

For HSDE, due to Equation (4.4.7), the probability that given a jump from (θ, x), the state after the jump is in (θ', A) is given by $Q(\{\theta'\} \times A; \theta, x)$ hence we find that $P_Q = Q$. Here, Q is given by Equation (4.4.8):

$$Q(\{\theta'\} \times A; \theta, x) = \rho(\theta', \theta, x) \int_{\mathbb{R}^q} \mathbf{1}_{\{(x+\psi(\theta', \theta, x, \underline{z})) \in A\}} \mu(d\underline{z})$$

From this, we find

$$\rho(\theta', \theta, x) = Q(\{\theta'\} \times \mathbb{R}^n; \theta, x)$$

ρ is measurable due to Q being a probability measure. Next write, for any x',

$$\begin{aligned} Q(\{\theta'\}, x'; \theta, x) &= \rho(\theta', \theta, x) \cdot \mathbb{P}\{x + \psi(\theta', \theta, x, \underline{z}) = x'\} \\ &= \rho(\theta', \theta, x) \cdot \mathbb{P}\{\psi(\theta', \theta, x, \underline{z}) = x' - x\} \\ &= \rho(\theta', \theta, x) \cdot \mathbb{P}\{\underline{z} = \psi^{qf}_{\theta', \theta, x}(x' - x)\} \\ &= \rho(\theta', \theta, x) \cdot \mu(\psi^{qf}_{\theta', \theta, x}(x' - x)) \end{aligned}$$

where $\psi^{qf}_{\theta', \theta, x}$ is such that $\mu_L\{u \mid \psi^{qf}_{\theta', \theta, x}(u) \in B\} = \psi(\theta', \theta, x, B)$. Therefore,

$$\mu(\psi^{qf}_{\theta', \theta, x}(x' - x)) = \frac{Q(\{\theta'\}, x'; \theta, x)}{\rho(\theta', \theta, x)}$$

and

$$\psi^{qf}_{\theta', \theta, x}(x' - x)) = \mu^{qf}\left(\frac{Q(\{\theta'\}, x'; \theta, x)}{\rho(\theta', \theta, x)}\right)$$

Hence with this, ψ is defined, and it is measurable due to measurability of Q and ρ.

Note that the definition of firing function yields $x+\psi(\vartheta,\theta,x,\underline{z}) \in E_\vartheta$ since the firing function of a transition delivers tokens in its output places with colours corresponding to the colour type of the output places.

p_P: The Poisson random measure is constructed as follows: p_P is such that at exponentially distributed intervals with intensity R_Λ it produces a "mark" $(z_1, \underline{z})$, where $z_1 \sim U[0, R_\Lambda]$ and $\underline{z} \sim \mu$. Here, R_Λ is the upperbound of Λ defined above.

$\{W_t\}$: This is generated according to the standard mechanism to generate Wiener processes. An h-dimensional Wiener process is constructed by collecting a number of $h = \sum_{i=1}^{|\mathcal{P}|} m_i^{max} h(P_i)$ independent one-dimensional Wiener processes in a vector.

Adding zeros and transforming discrete state vectors We add a sufficient number of zeros to some of the elements in order to create a constant dimension for the HSDE$^{\text{SDCPN}}$ hybrid state space. Denote $n = \max_\theta d(\theta)$, 0^a as a column vector of zeros in $\mathbb{R}^a$ and $0^{a\times b}$ as a matrix of zeros in $\mathbb{R}^{a\times b}$, then E is redefined as $E = \{\{\theta\} \times (E_\theta \times \mathbb{R}^{n-d(\theta)}); \theta \in \mathbb{M}\}$; f is redefined as Col$\{f, 0^{n-d(\theta)}\}$; g is redefined as Col$\{g, 0^{(n-d(\theta))\times\Sigma_i m_i^{max}\cdot h(P_i)}\}$; X_0 is redefined as Col$\{X_0, 0^{n-d(\theta)}\}$ and ψ is redefined as Col$\{\psi, 0^{n-d(\theta)}\}$.

This shows that all HSDE$^{\text{SDCPN}}$ elements can be characterised uniquely in terms of SDCPN elements.

4.6.2 Probabilistic equivalence

Subsequently, we show that if the original SDCPN is executed on a probability space endowed with sequences of standard Brownian motions (one sequence for each place) then the solution of the constructed HSDE$^{\text{SDCPN}}$ delivers a stochastic process which is probabilistically equivalent to the process defined by the original SDCPN. This is done by showing:

- Equivalence of initial states
- Equivalence of continuous evolution until first jump
- Equivalence of time of jumps
- Equivalence of size of jumps
- Equivalence of processes after the first jump

Equivalence of initial states The initial HSDE$^{\text{SDCPN}}$-process state (θ_0, X_0) at $t = \tau_0$ is equivalent to the initial SDCPN state through the mapping constructed above. If $\mathcal{I}^{qf}$ denotes the quantile function of $\mathcal{I}$ and $\mu^{qf}_{\theta_0,X_0}$ denotes the quantile function of μ_{θ_0,X_0}, then the random variable $(M_0, C_0) = \mathcal{I}^{qf}(U)$ is equivalent to the random variable $(\theta_0, X_0) = \mu^{qf}_{\theta_0,X_0}(U)$. Due to equivalence between $\mathcal{I}$ and μ_{θ_0,X_0}, the initial states are probabilistically equivalent.

Equivalence of continuous evolution until first jump By the unique mapping of SDCPN elements into HSDE elements, for $t > \tau_0$, up until the first jump, the HSDE$^{\text{SDCPN}}$ state is probabilistically equivalent to the original SDCPN state: The continuous part of the SDCPN marking is composed of the colours of all tokens in all places in a specific unique order. For the constructed HSDE$^{\text{SDCPN}}$, the continuous state is also composed of these colours and in the same order.

At times t when no jump occurs, the HSDE$^{\text{SDCPN}}$-process evolves according to f and g and the SDCPN-process evolves according to $\mathcal{V} = \{\mathcal{V}_P; P \in \mathcal{P}\}$ and $\mathcal{W} = \{\mathcal{W}_P; P \in \mathcal{P}\}$. As long as no jump occurs, the stochastic differential equations defining the SDCPN token colours are driven by Brownian motions. The stochastic differential equations defining the HSDE continuous state are driven by Wiener processes. The collection of Brownian motions and the Wiener process are probabilistically equivalent in the sense that they are both Gaussian with the same mean and variance (see Appendix A). Through the mappings between f and $\mathcal{V}$ and between g and $\mathcal{W}$ developed above, and due to the equivalence of the Brownian motions and the Wiener process used, these evolutions provide probabilistically equivalent processes, i.e., for all $t > \tau_0$, until the first jump, $X_t = C_t$ in probabilistic sense.

Equivalence of time of jumps The times of jumps are generated by forced jumps and spontaneous jumps. In SDCPN, the forced jumps are represented by guard transitions; in HSDE, the forced jumps are represented by continuous state space boundary hits. Due to the mapping between the boundary of the HSDE$^{\text{SDCPN}}$ state space ∂E_θ and the transition guards of the guard transitions $\{\partial\mathcal{G}_T; T \in \mathcal{T}_G\}$, the HSDE$^{\text{SDCPN}}$ forced jumps and the SDCPN forced jumps occur at the same time. The HSDE$^{\text{SDCPN}}$ spontaneous jumps are generated by a Poisson random measure that uses a rate Λ. Due to definition of Poisson random measure, the time until the next jump is exponential with intensity Λ. The SDCPN spontaneous jumps are generated by the delay transitions that use rates $\{\mathcal{D}_T; T \in \mathcal{T}_D\}$. Each pre-enabled delay transition T is enabled after an exponential time with intensity $\mathcal{D}_T$. The time until the first delay transition enabling is also exponential, with an intensity equal to the sum of all $\mathcal{D}_T$ of pre-enabled delay transitions. Due to the constructed mapping between Λ and $\{\mathcal{D}_T; T \in \mathcal{T}_D\}$, the time of spontaneous jump is therefore according to the same rate for both HSDE$^{\text{SDCPN}}$ and SDCPN.

Equivalence of size of jumps At times when a jump occurs, the HSDE$^{\text{SDCPN}}$-process makes a jump generated by ψ, ρ and μ, while the SDCPN-process makes a jump generated by $\mathcal{F}$. Through the mapping between ψ, ρ, μ and $\mathcal{F}$, these jumps provide probabilistically equivalent processes.

Equivalence of processes after the first jump After the first jump, equivalence is shown in a similar way as above.

4.6.3 Verification of H1-H8

Finally, we show that if additionally, conditions S1-S6 are satisfied for the original SDCPN, then HSDE conditions H1-H8 are satisfied for the resulting HSDE$^{\text{SDCPN}}$.

H1 Condition H1 reads: For all $\theta \in \mathbb{M}$ there exists a constant $K(\theta)$ such that for all $x \in \mathbb{R}^n$, $|f(\theta,x)|^2 + \|g(\theta,x))\|^2 \leq K(\theta)(1+|x|^2)$, where $|a|^2 = \sum_i (a_i)^2$ and $||b||^2 = \sum_{i,j}(b_{ij})^2$.

This condition H1 is verified as follows: from the construction of f and g above we have
$f(\theta,x) = \text{Col}_{i=1}^{|\mathcal{P}|}\left\{\text{Col}_{j=1}^{m_i}\{\mathcal{V}_{P_i}(c_{ij})\}\right\}$
and $g(\theta,x) = \text{Row}\{\text{Diag}_{i=1}^{|\mathcal{P}|}\left\{\text{Diag}_{j=1}^{m_i}\{\mathcal{W}_{P_i}(c_{ij})\}\right\}, O^{\sum_{i=1}^{|\mathcal{P}|}(m_i^{max}-m_i)h(P_i)}\}$,
where $x = \text{Col}_{i=1}^{|\mathcal{P}|}\left\{\text{Col}_{j=1}^{m_i}\{c_{ij}\}\right\}$.

Use that under condition S1, for all $P \in \mathcal{P}$, there exist K_P^v and K_P^w such that for all $c \in \mathcal{C}(P)$, $|\mathcal{V}_P(c)|^2 \leq K_P^v(1+|c|^2)$ and $\|\mathcal{W}_P(c)\|^2 \leq K_P^w(1+|c|^2)$.

This gives

$$
\begin{aligned}
&|f(\theta,x)|^2 + \|g(\theta,x)\|^2 = \\
&\quad = \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i}(\mathcal{V}_{P_i}(c_{ij}))^2 + \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i}(\mathcal{W}_{P_i}(c_{ij}))^2 \\
&\quad \leq \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i} K_{P_i}^v(1+|c_{ij}|^2) + \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i} K_{P_i}^w(1+|c_{ij}|^2) \\
&\quad = \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i}(K_{P_i}^v + K_{P_i}^w)(1+|c_{ij}|^2) \\
&\quad \leq K(\theta)(1+|x|^2)
\end{aligned}
$$

where $K(\theta) = \max\{\max_i(K_{P_i}^v + K_{P_i}^w), \sum_{i=1}^{|\mathcal{P}|} m_i(K_{P_i}^v + K_{P_i}^w)\}$.

H2 Condition H2 reads: For all $r \in \mathbb{N}$ and for all $\theta \in \mathbb{M}$ there exists a constant $L_r(\theta)$ such that for all x and y in the ball $\{z \in \mathbb{R}^n \mid |z| \leq r+1\}$, $|f(\theta,x) - f(\theta,y)|^2 + \|g(\theta,x) - g(\theta,y)\|^2 \leq L_r(\theta)|x-y|^2$.

This condition H2 is verified as follows: Use that under condition S1, for all $r \in \mathbb{N}$ and all $P \in \mathcal{P}$, there exist $L^v_{r,P}$ and $L^w_{r,P}$ such that for all $a, b \in \{z \in \mathcal{C}(P) |\ |z| \leq r+1\}$, $|\mathcal{V}_P(b) - \mathcal{V}_P(a)|^2 \leq L^v_{r,P}|b-a|^2$ and $\|\mathcal{W}_P(b) - \mathcal{W}_P(a)\|^2 \leq L^w_{r,P}|b-a|^2$. Then

$$
\begin{aligned}
&|f(\theta,x) - f(\theta,y)|^2 + \|g(\theta,x) - g(\theta,y)\|^2 = \\
&\quad = \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i} |\mathcal{V}_{P_i}(b_{ij}) - \mathcal{V}_{P_i}(a_{ij})|^2 + \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i} \|\mathcal{W}_{P_i}(b_{ij}) - \mathcal{W}_{P_i}(a_{ij})\|^2 \\
&\quad \leq \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i} L^v_{r,P_i}|b_{ij} - a_{ij}|^2 + \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i} L^w_{r,P_i}|b_{ij} - a_{ij}|^2 \\
&\quad = \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i} (L^v_{r,P_i} + L^w_{r,P_i})|b_{ij} - a_{ij}|^2 \\
&\quad \leq \sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{m_i} L_r(\theta)|b_{ij} - a_{ij}|^2 \\
&\quad = L_r(\theta)|x-y|^2
\end{aligned}
$$

where $L_r(\theta) = \max_i\{L^v_{r,P_i} + L^w_{r,P_i}\}$.

H3 Condition H3 reads: For each $\theta \in \mathbb{M}$, $\Lambda(\theta,\cdot) : \mathbb{R}^n \to [0,\infty)$ is continuous and bounded with upper bound a constant R_Λ.

This condition H3 is verified as follows: Use that for all $T \in \mathcal{T}_D$, $\mathcal{D}_T$ is continuous (condition S3) and bounded. We find that Λ is continuous and bounded as well, with $R_\Lambda = \max_{\theta\in\mathbb{M}}\{k \cdot R_{\mathcal{D}}$; the number of elements in the multi-set of transitions in $\mathcal{T}_D$ that, under token distribution θ, are pre-enabled $= k\}$.

H4 Condition H4 reads: For all $\theta, \vartheta \in \mathbb{M}$, the mapping $\rho(\vartheta,\theta,\cdot) : \mathbb{R}^n \to [0,\infty)$ is continuous.

This condition H4 is verified as follows: In the construction of ρ it was derived that $\rho(\vartheta,\theta,x) = Q(\{\vartheta\} \times \mathbb{R}^n; \theta, x)$. Q is constructed from elements $\mathcal{D}$, $\mathcal{G}$ and $\mathcal{F}$, and is continuous due to $\mathcal{D}_T$ and $\mathcal{F}_T$ being continuous for all $T \in \mathcal{T}$ (conditions S3 and S5).

H5 Condition H5 reads: For all $r \in \mathbb{N}$ there exists a constant $M_r(\theta)$ such that

$$
\sup_{|x|\leq r} \int_{\mathbb{R}^q} |\psi(\vartheta,\theta,x,\underline{z})|\mu(d\underline{z}) \leq M_r(\theta), \text{ for all } \vartheta,\theta \in \mathbb{M}
$$

To verify this condition notice that the above inequality means that the expected size of the jump in X_t should be bounded, i.e., if θ_t jumps from θ to ϑ and X' denotes the continuous state after the jump, then $\sup_{|x|\leq r} \mathbb{E}\{|X' - x|\} \leq M_r(\theta)$, for all $\vartheta, \theta \in \mathbb{M}$, $x \in E_\theta$, $X' \in E_\vartheta$ in the domain of ψ.

This follows from condition S2 if one realises that if the HSDE is modelled by an SDCPN then the size of the jump is determined by the combination of firing functions of transitions firing at the jump. Since condition S2 ensures that the size of the jump in the token colours is finite, there exists $M_r(\theta)$ that ensures that the size of the jump in the HSDE is bounded by $M_r(\theta)$.

H6 Condition H6 reads: ψ satisfies: $|\psi(\theta, \theta, x, \underline{z})| = 0$ or > 1 for all $\theta \in \mathbb{M}$, $x \in \mathbb{R}^n$, and $\underline{z} \in \mathbb{R}^q$.

This condition H6 follows from condition S2 if one notes that $|\psi(\theta, \theta, x, \underline{z})|$ provides the size of the jump in X_t (if θ_t jumps from θ back to θ).

H7 Condition H7 reads: $\{(\theta_t^*(\omega), X_t^*(\omega))\}$ hits the boundary $\partial E \triangleq \{\{\theta\} \times \partial E_\theta; \theta \in \mathbb{M}\}$ a finite number of times on any finite time interval

This condition H7 follows from condition S4.

H8 Condition H8 reads: $|\vartheta_i - \vartheta_j| > 1$ for $i \neq j$, with $|\cdot|$ a suitable metric well defined on $\mathbb{M}$.

This condition follows from condition S6.

This completes the proof of Theorem 4.5. □

4.7 Discussion of conditions of Theorem 4.5

This section discusses the conditions for Theorem 4.5 (SDCPN into HSDE).

4.7.1 Discussion on finite number of tokens

The first set of conditions for Theorem 4.5 is that the initial marking does not enable a transition, that none of the transition firings enable a guard transition, that the delay rates are bounded, and that for $t \to \infty$ the number of tokens remains finite. Here, the condition on the initial marking can be easily verified. The condition on the immediate enabling of guard transitions is discussed in Section 4.7.5, which refers to Section 3.7.4. The condition on bounded delay rates is discussed in Subsection 4.7.4. We discuss the condition on finite number of tokens next.

This condition has been introduced in Theorem 4.5 in order to guarantee that the reachability graph (RG) of the SDCPN exists and is finite, i.e., the SDCPN is bounded (see Subsection 2.2.2). The RG and its nodes are used in the construction of HSDE$^{\text{SDCPN}}$ elements $\mathbb{M}$, f and g, and indirectly also in the construction of E, Λ, ρ and ψ. A finite RG makes this construction easier.

A sufficient condition for a SDCPN to be bounded is if its initial marking contains a finite number of tokens (note this is satisfied due to definition of $\mathcal{I}$), and each transition, when firing, produces a number of tokens equal to the number of tokens removed by the firing. Note that for

most SDCPN applications that we made in practice, these sufficient conditions are satisfied (see also Chapter 5) hence there has been no real practical reason to explore options where the number of tokens could grow to infinity.

In Section 3.7.1 the same condition was discussed for DCPN as part of the theorem mapping DCPN into PDP. The difference between PDP and HSDE on this point is that PDP allows the discrete state space to be countable, whereas HSDE requires the discrete state space to be finite. In Section 3.7.1 we provided a discussion that argued that the condition of finite number of tokens could be relaxed for DCPN. Due to the more stringent HSDE condition on the size of the discrete state space, this more relaxed condition does not hold true for SDCPN if it is mapped to an HSDE.

4.7.2 Discussion on Condition S1 (growth and local Lipschitz)

Condition S1 reads: for all $r \in \mathbb{N}$ and for all $P \in \mathcal{P}$, there exist K_P^v, $L_{r,P}^v$, K_P^w and $L_{r,P}^w$ such that for all $c \in \mathcal{C}(P)$ and any a, b in the ball $\{z \in \mathcal{C}(P) \mid |z| \leq r+1\}$,

- $|\mathcal{V}_P(c)|^2 \leq K_P^v(1+|c|^2)$
- $|\mathcal{V}_P(b) - \mathcal{V}_P(a)|^2 \leq L_{r,P}^v|b-a|^2$
- $\|\mathcal{W}_P(c)\|^2 \leq K_P^w(1+|c|^2)$
- $\|\mathcal{W}_P(b) - \mathcal{W}_P(a)\|^2 \leq L_{r,P}^w|b-a|^2$.

This condition has been introduced in order to satisfy HSDE conditions H1 and H2. Note that these conditions are standard sufficient conditions for a stochastic differential equation to have a pathwise unique solution. Since the stochastic differential equations in SDCPN are already assumed to have probabilistically unique solutions, this condition S1 is not too stringent.

4.7.3 Discussion on Condition S2 (bounded jumps)

Condition S2 reads: If (M_t, C_t) denotes the SDCPN marking at time t and τ denotes a time at which one or more transitions fire, then there exists $R_S(M_t) < \infty$ such that $|\overline{C}_\tau - \overline{C}_{\tau-}| = 0$ or $\in (1, R_S(M_{\tau-})]$. Here, $\overline{C}_t$ denotes a vector C_t with a sufficient number of zeros added so that $\overline{C}_\tau$ and $\overline{C}_{\tau-}$ have the same number of vector elements.

This condition has been introduced in order to satisfy HSDE conditions H5 and H6, which are conditions on the mapping ψ that determines the size of jumps in the continuous state. Due to condition H5, this size of jump should either be zero or be larger than one, and due to condition H6, the expected size of jump should be bounded. Condition S2 ensures that both H5 and H6 are satisfied. Due to the complicated relation between $\mathcal{F}$ and ψ it is not possible to translate this to straightforward sufficient conditions on $\mathcal{F}$, rather than on C_t.

4.7.4 Discussion on Condition S3 (continuous and bounded delays)

Condition S3 reads: For all $T \in \mathcal{T}_D$, $\mathcal{D}_T$ is continuous and bounded with upperbound $R_{\mathcal{D}}$.

This condition is easily verified. Note that a delay transition for which $\mathcal{D}_T = \infty$ does not add value to the SDCPN definition since in effect such transition is an immediate transition. Therefore, the boundedness condition is a reasonable restriction.

4.7.5 Discussion on Condition S4 (finite number of firings)

Condition S4 reads: In a finite time interval, each guard transition is expected to fire a finite number of times.

A version of this condition was also posed on DCPN in the mapping from DCPN to PDP. For a discussion we refer to Section 3.7, which also discusses the condition on the immediate enabling of guard transitions.

4.7.6 Discussion on Condition S5 (continuous firing measures)

Condition S5 reads: For all $T \in \mathcal{T}$, $c \in \mathcal{C}(P(A_{in,OE}(T)))$ and $e \in \{0,1\}^{|A_{out}(T)|}$, $\mathcal{F}_T(e, \cdot; c)$ is continuous.

This condition has been introduced in addition to S3 to ensure that $\rho(\vartheta, \theta, \cdot)$ is continuous. Note that $\mathcal{F}_T$ is a probability measure hence is measurable. Its continuity puts a verifyable restriction on it.

4.7.7 Discussion on Condition S6 (distinguishable token distributions)

Condition S6 reads: If $\mathcal{M} = \{M_t \mid (M_t, C_t)$ is a reachable marking, $t \geq 0\}$ is the set of reachable token distributions, then for all $M^i, M^j \in \mathcal{M}$, $M^i \neq M^j$, $|M^i - M^j| > 1$.

This condition was introduced to satisfy HSDE condition H8, which states that for all $\vartheta_i, \vartheta_j \in \mathbb{M}$, $|\vartheta_i - \vartheta_j| > 1$. Due to the construction of ϑ_i in the mapping from SDCPN to HSDE, i.e., $\vartheta_i = M^i$, we find that ϑ_i is constructed as the vector that counts the numbers of tokens in each of the places. From this, we find that $|\vartheta_i - \vartheta_j| \geq 1$. The strict inequality $|\vartheta_i - \vartheta_j| > 1$ is not automatically satisfied. However, it can be ensured if we redefine ϑ_i to be, e.g., the ith unit vector of size $|\mathbb{M}| = N$. For reduced readability, the latter step is not done in the actual proof, but it is a formality.

4.8 Equivalence between SDCPN and stochastic hybrid automata

In Sections 4.5 and 4.6, equivalence relations have been established between SDCPN and hybrid stochastic differential equations (HSDE). This section shows that similar equivalence relations exist between SDCPN and a particular class of *stochastic hybrid automaton*, referred to as *general stochastic hybrid system* (GSHS), [BL06]. A GSHS collects the elements that define the hybrid state space, the continuous evolution mechanism and the jump mechanism. In [BL06], it is shown that the execution of a GSHS defines a GSHP. In the remainder of this section we first describe GSHS and explain how these relate to HSDE. Subsequently, we formulate the equivalence relations between GSHS and SDCPN.

4.8.1 Definition of GSHS and its execution

This section presents, following [BL06], a definition of *general stochastic hybrid system* (GSHS) and its execution. Next, it explains the differences with HSDE.

Definition 4.3 (General stochastic hybrid system)**.** *A GSHS is an automaton (*$\mathbf{K}$*,* d*,* $\mathcal{X}$*,* f*,* g*,* Init*,* λ*,* Q*), where*

- $\mathbf{K}$ *is a countable set.*
- $d : \mathbf{K} \to \mathbb{N}$ *maps each* $\theta \in \mathbf{K}$ *to a natural number.*
- $\mathcal{X} : \mathbf{K} \to \{E_\theta; \theta \in \mathbf{K}\}$ *maps each* $\theta \in \mathbf{K}$ *to an open subset* E_θ *of* $\mathbb{R}^{d(\theta)}$*. With this, the hybrid state space is given by* $E \triangleq \{\{\theta\} \times E_\theta; \theta \in \mathbf{K}\}$.
- $f : E \to \{\mathbb{R}^{d(\theta)}; \theta \in \mathbf{K}\}$ *is a vector field.*
- $g : E \to \{\mathbb{R}^{d(\theta)\times h}; \theta \in \mathbf{K}\}$ *is a matrix field, with* $h \in \mathbb{N}$.
- Init: $\mathcal{B}(E) \to [0, 1]$ *is an initial probability measure, with* $\mathcal{B}(E)$ *the Borel* σ*-algebra on* E.
- $\lambda : E \to \mathbb{R}^+$ *is a jump rate function.*
- $Q : \mathcal{B}(E) \times (E \cup \partial E) \to [0, 1]$ *is a GSHS transition measure, where* $\partial E \triangleq \{\{\theta\} \times \partial E_\theta; \theta \in \mathbf{K}\}$ *is the boundary of* E*, in which* ∂E_θ *is the boundary of* E_θ.

Definition 4.4 (GSHS execution)**.** *A stochastic process* $\{\theta_t, X_t\}$ *is called a* GSHS execution *if there exists a sequence of stopping times* $0 = \tau_0 < \tau_1 < \tau_2 \cdots$ *such that for each* $k \in \mathbb{N}$*:*

- (θ_0, X_0) *is an* E*-valued random variable extracted according to probability measure* Init.

- *For $t \in [\tau_k, \tau_{k+1})$, $\theta_t = \theta_{\tau_k}$ and $X_t = X_t^k$, where for $t \geq \tau_k$, X_t^k is a solution of the stochastic differential equation $dX_t^k = f(\theta_{\tau_k}, X_t^k)dt + g(\theta_{\tau_k}, X_t^k)dB_t^{\theta_{\tau_k}}$ with initial condition $X_{\tau_k}^k = X_{\tau_k}$, and where $\{B_t^\theta\}$ is h-dimensional standard Brownian motion for each $\theta \in \mathbf{K}$.*

- *$\tau_{k+1} = \tau_k + \sigma_k$, where σ_k is chosen according to a survivor function given by $F(t) = \mathbf{1}_{\{t<\tau^*\}} \cdot \exp(-\int_0^t \lambda(\theta, X_s^k)ds)$. Here, $\tau^* = \inf\{t > \tau_k \mid X_t^k \in \partial E_{\theta_{\tau_k}}\}$ and $\mathbf{1}$ is indicator function.*

- *The probability distribution of $(\theta_{\tau_{k+1}}, X_{\tau_{k+1}})$, i.e., the hybrid state right after the jump, is governed by the law $Q(\cdot; (\theta_{\tau_k}, X_{\tau_{k+1}-}))$.*

[BL06] show that under assumptions G1-G4 below, a GSHS execution is a strong Markov process and has the càdlàg property (right continuous with left hand limits).

G1 $f(\theta, \cdot)$ and $g(\theta, \cdot)$ are Lipschitz continuous and bounded. This yields that for each initial state (θ, x) at initial time τ there exists a pathwise unique solution X_t to $dX_t = f(\theta, X_t)dt + g(\theta, X_t)dB_t$, where $\{B_t\}$ is h-dimensional standard Brownian motion.

G2 $\lambda : E \to \mathbb{R}^+$ is a measurable function such that for all $\xi \in E$, there is $\epsilon(\xi) > 0$ such that $t \to \lambda(\theta_t, X_t)$ is integrable on $[0, \epsilon(\xi)[$.

G3 For each fixed $A \in \mathcal{B}(E)$, the map $\xi \to Q(A; \xi)$ is measurable and for any $(\theta, x) \in E \cup \partial E$, $Q(\cdot; \theta, x)$ is a probability measure.

G4 If $N_t = \sum_k \mathbf{1}_{\{t \geq \tau_k\}}$, then it is assumed that for every starting point (θ, x) and for all $t \in \mathbb{R}^+$, $\mathbb{E}N_t < \infty$. This means, there will be a finite number of jumps in finite time.

Note that HSDE and GSHS have a lot of similarities. Both concatenate different solutions of stochastic differential equations with hybrid jumps at each moment of switching to another stochastic differential equation. Hence the differences are of a rather technical nature. Below, the technical differences are collected between GSHS and its GSHP execution, versus HSDE and its GSHP solution:

1. For GSHS, the discrete state space is a countable space of discrete variables. For HSDE, the discrete state space is a finite set.

2. For GSHS, the continuous state is Euclidean with a dimension dependent on θ. For HSDE, the continuous state is Euclidean with constant dimension n.

3. The times of spontaneous jump of the GSHS execution are driven by a survivor function which imposes a stochastic basis. For HSDE, the times of spontaneous jumps are driven by a Poisson random measure endowed upon a given stochastic basis.

4. For GSHS, the size of jump is driven by a transition measure Q. For HSDE, the jump size is determined by probability measure μ and measurable mappings ψ and ρ.

5. GSHS involves $|\mathbf{K}|$ Brownian motions. HSDE involves one Wiener process only.

6. For GSHS, the drift and diffusion coefficient are assumed (globally) Lipschitz and bounded. For HSDE, the drift and dissusion coefficient are locally Lipschitz and are allowed to grow with the continuous state.

For 1) and 2), GSHS has as advantage of being more general than HSDE. HSDE however has significant advantages regarding issues 3)-6): Regarding 3)-5), HSDE has the advantage that this allows to establish the *semi-martingale property*. The semi-martingale property of GSHS execution is unknown, which prohibits the use of Itô's differentiation rule for semi-martingales. Regarding 6), HSDE removes the particular restriction of GSHS which excludes *jump linear systems*. For GSHS, this restriction unfortunately excludes most existing work on stochastic hybrid systems.

4.8.2 Equivalence relations between SDCPN and GSHS

This section formulates the equivalence relations between SDCPN and GSHS.

Theorem 4.6 (GSHS into SDCPN)**.** *Consider an arbitrary GSHS (*$\mathbf{K}$, d, $\mathcal{X}$, f, g, Init, λ, Q*) with a finite domain* $\mathbf{K}$*. If for each* θ *and initial value* X_0*, the stochastic differential equation* $dX_t = f(\theta, X_t)dt + g(\theta, X_t)dB_t$ *has a unique solution in probabilistic sense, then the elements of this GSHS can be mapped into the elements of an SDCPN (*$\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$*) satisfying R0-R4. If the resulting SDCPN is executed on a probability space endowed with sequences of standard Brownian motions (one sequence for each place), then the resulting SDCPN process and the GSHS execution are probabilistically equivalent.*

Proof. See [EB06]. This proof is similar to the proof of Theorem 3.1 of PDP into DCPN, with the addition of Brownian motion in the continuous evolutions. □

Theorem 4.7 (SDCPN into GSHS)**.** *Consider an arbitrary SDCPN (*$\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$*) satisfying R0-R4. If in the initial marking no transition is enabled, if a transition firing does not enable guard transitions, and if the number of tokens remains finite for* $t \to \infty$*, then the elements of this SDCPN can be mapped into the elements of a GSHS (*$\mathbf{K}$, d, $\mathcal{X}$, f, g, Init, λ, Q*). If the original SDCPN is executed on a probability space endowed with sequences of standard Brownian motions (one sequence for each place) then the resulting GSHS execution and the SDCPN process are probabilistically equivalent.*

Proof. See [EB06]. This proof is similar to the proof of Theorem 3.2 of DCPN into PDP, with the addition of Brownian motion in the continuous evolutions. □

Notice that the proofs of equivalence between SDCPN and GSHS actually resemble those of equivalence between DCPN and PDP rather than those of equivalence between SDCPN and HSDE. This is due to the time of jump mechanism of both GSHS and PDP being a survivor function, while HSDE uses Poisson random measure as an integrated part of the stochastic differential equations.

4.9 Concluding remarks

General stochastic hybrid processes (GSHP) extend PDP by the inclusion of diffusion. Diffusion exists in air transport operations for example in the form of stochastic variations around position and velocity of an aircraft, due to, e.g., weather, navigation or surveillance uncertainties, or engine power fluctuations. GSHP can be defined in several ways. In [Blo03] and [BBEP03], GSHP are defined as the solution process of hybrid stochastic differential equation (HSDE) on a hybrid state space. In [BL06], they are defined as the execution of a general stochastic hybrid system (GSHS).

This chapter has extended DCPN to stochastically and dynamically coloured Petri net (SDCPN) and has shown that under some mild conditions, any SDCPN can be mapped into the elements of an HSDE, such that the SDCPN process and the resulting HSDE process are probabilistically equivalent. Moreover, it has shown that any HSDE can be mapped into an SDCPN such that the HSDE process and the resulting SDCPN process are probabilistically equivalent. Similar equivalence relations are shown between SDCPN and GSHS. This implies that SDCPN, HSDE and GSHS are bisimilar (see also Section 3.8).

To our best knowledge, SDCPN is the only hybrid Petri net that incorporates diffusion.

The key result of this chapter is that this is the first time that proof of equivalence between GSHP and Petri nets has been established. This significantly extends the modelling power hierarchy of [MT94], [MFT00] in terms of Petri nets and Markov processes, see Figure 1.1. The SDCPN-inherited modelling power can be used to model GSHP. In addition, GSHP, HSDE and GSHS theoretical results like stochastic analysis, stability and control theory, now also apply to SDCPN processes.

Because of this, for accident risk modelling in air transport operations, in, e.g., [BBB$^+$01, BKB03, BBK$^+$05] SDCPNs are adopted for their specification power and for their GSHP inherited stochastic analysis power. Here, SDCPN are used as a basis for a Monte Carlo simulation, and the GSHP inherited stochastic analysis properties are used to make the simulations and analysis more efficient.

Chapter 5

Compositional specification of SDCPN

5.1 Introduction

By the very nature of air transport operations, the decision-makers are highly distributed: per aircraft there is a crew of pilots, and per air traffic control centre there are many human operators. In addition, the safety-related decision-making process involves interactions of these humans with each other and with a random and often unpredictable environment (e.g., varying wind, thunderstorms), a large set of procedural rules and guidelines, many technical and automation support systems, decision-makers at airline operation centres, etc. These aspects make the specification of an unambiguous mathematical model of air transport operations a very challenging task.

In order to capture the characteristics of air transport operations, Chapters 3 and 4 developed dynamically coloured Petri net (DCPN) and stochastically and dynamically coloured Petri net (SDCPN), and showed equivalence between DCPN and piecewise deterministic Markov process (PDP) and equivalence between SDCPN and general stochastic hybrid processes (GSHP). With this, in cases where GSHP (PDP) is an appropriate mathematical formalism to use for a particular application, equipped with the proper analysis tools, SDCPN (DCPN) can be formally used instead of GSHP (PDP) with the advantage of the graphical modelling support. The DCPN and SDCPN formalisms have been successfully used in practical air transport applications, (e.g., [BKB03], [BSEP03]). However, it was found that when being used for modelling more and more complex multi-agent hybrid systems, the compositional specification power of SDCPN, but also of Petri net classes in general, reaches its limitations. More specifically, the following problems were identified:

Problem A. For the modelling of a compositional Petri net for complex systems, a hierarchical approach is necessary that separates local modelling issues from compositional or interaction modelling issues.

Problem B. Often the addition of an interconnection between two local Petri nets leads to a necessary duplication of transitions and arcs within a local Petri net.

Problem C. The number of interconnections between the different local Petri nets tends to grow quadratically with the size of the Petri net[1].

This chapter aims to solve these problems for SDCPN, by combining and adopting approaches from the literature, see Section 2.6, that solve problem A for other classes of Petri net, and by developing novel approaches to solve problems B and C. It is noted that all approaches are similarly applicable to DCPN. It is also noted that these approaches further increase the modelling power of SDCPN, while maintaining the stochastic analysis power of GSHP.

To solve problem A, the compositional specification of an SDCPN for a complex operation starts with developing a separate local Petri net (LPN) for each *agent* that exists in the operation (e.g., air traffic controller, pilot, navigation and surveillance equipment). Counterparts of LPNs in literature are the modules of [FKK97], the pages of [HJS90] and the components of [Kin97]. Next, the LPNs are interconnected by arcs, and where necessary, by additional places and transitions. For manageability, the following restriction is posed: *the interactions between LPNs are not allowed to change the number of tokens in an LPN*. This restriction is also posed by [FKK97], but not by [HJS90] or [Kin97]. Two types of interconnections between places and transitions in different LPNs are introduced that ensure that this restriction holds true: (1) *Enabling arc* (or *inhibitor arc*) from one place in one LPN to one transition in another LPN; and (2) *Interaction Petri net* (IPN) from one (or more) transition(s) in one LPN to one (or more) transition(s) in another LPN by means of ordinary arcs. Enabling and inhibitor arcs have been used widely in Petri net literature, including [FKK97] for inhibitor arcs and [FAP97] for both types. They have the property that no tokens are moved along them at the firing of a transition. IPNs are similar to the interconnection blocks of [FKK97]. If an IPN consists of one place only, then the connection of two LPNs through an IPN also has some similarity with place fusion, see, e.g., [HJS90] or [Kin97], except that our IPN will not change the number of tokens in its connecting LPNs.

Each LPN is surrounded by a box, following, e.g., [FAP97] or [Kin97]. This alone, however, does not solve problems B and C. Hence, following ideas of statecharts [Har87], arcs are introduced that initiate at and/or end on the edge of an LPN-box, including a well-defined meaning in terms of interconnections between LPN-boxes. To the best of our knowledge, this element, which solves problems B and C, has no counterpart in the Petri net literature. The meaning of these interconnections from or to an edge of a box allows several arcs or transitions to be represented by only one arc or transition. In that sense, there is a relation with transition fusion used by [HJS90] and with module folding used by [FKK97].

[1] A good example of a cluttering of interconnections can be found in [TTV06, Figure 7].

The mathematical formalisation of the approach thus developed is referred to as an SDCPN^{imt}, where *imt* refers to *interconnection mapping types*. This formalisation includes the derivation of unambiguous rules in the use of interconnection mapping types and in transforming an SDCPN^{imt} to an SDCPN.

This chapter is organised as follows: Section 5.2 describes how an SDCPN can be specified in a logical sequence of local Petri nets for each entity of an agent, and explains how the entities of agents are connected without drastically changing the structure of low-level entities. This solves problem A above. Section 5.3 defines new Petri net interconnection mapping types which avoid the internal duplication problem (problem B) and the problem of cluttering interconnections (problem C). Section 5.4 formally extends the SDCPN definition to SDCPN^{imt}, to include these new interconnection mapping types, and discusses the relation between SDCPN^{imt} and GSHP. Finally, Section 5.5 gives concluding remarks.

5.2 Local Petri nets-based specification of an SDCPN

The compositional specification of a stochastically and dynamically coloured Petri net for a complex process with many different interacting agents such as exist in air transport operations (e.g., air traffic controllers, pilots, navigation and surveillance equipment), is a bottom-up process. Prior to starting this compositional process, per agent the relevant low-level functional entities have to be identified based on expert domain knowledge of that agent. The compositional specification idea is then first to specify one small Petri net per functional entity of an agent, and refer to this as a local Petri net (LPN). Next, the interactions between these LPNs are specified.

The specification of the various elements of one LPN is explained in Subsection 5.2.1; this has to be accomplished for all LPNs. Subsection 5.2.2 describes how the interconnections between these LPNs are established.

5.2.1 Specification of local Petri net

Specification of elements $\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$

First, places (drawn as circles) are identified for the LPN. These places may represent operational or physical conditions (nominal modes and non-nominal modes). Next, the transitions are identified: If between two (or more) places, say P_1 and P_2, a mode switch might occur, then one transition (rectangle) is drawn, with arcs (arrows) connecting the places and the transition. A transition may have multiple input places, if a switch requires multiple pre-conditions to hold true. In SDCPN there is also freedom in the range of post-conditions: a transition may fire zero or one tokens to each output place. This is in contrast with most other Petri nets classes, in which tokens

are fired to all output places.

If the LPN graph has been specified, the places are gathered in the set of places $\mathcal{P}$, the transitions are gathered in the set of transitions $\mathcal{T}$, and the arcs are gathered in the set of arcs $\mathcal{A}$. The node function $\mathcal{N}$ specifies for each arc which place and transition it connects.

Specification of elements $\mathcal{S}$, $\mathcal{C}$, $\mathcal{V}$ and $\mathcal{W}$

A complex stochastic dynamic process such as in air transport operations cannot be described by places, transitions and arcs alone. Some processes may be continuous rather than discrete, and the location of the tokens or the time of firing of the transitions may be dependent of the value of such continuous valued processes.

In SDCPN, a place can be associated with a particular stochastic differential equation (SDE), and a token can have a value. From a particular initial value onwards, this token value equals the solution of the SDE associated with the place in which the token resides. After a transition has fired, its output tokens will follow the SDEs associated with the output places. Note that the identification of several SDEs for one original discrete state will necessitate the introduction of several new places (and transitions) since each place can only be associated with one SDE.

If all SDEs have been specified, their coefficients are collected in sets $\mathcal{V} = \{\mathcal{V}_P; P \in \mathcal{P}\}$ and $\mathcal{W} = \{\mathcal{W}_P; P \in \mathcal{P}\}$: For each P, $\mathcal{V}_P$ equals the drift coefficient of P's SDE, and $\mathcal{W}_P$ equals the diffusion coefficient. The function $\mathcal{C}$ specifies the dimensions of all SDEs: If for $P \in \mathcal{P}$, $\mathcal{V}_P(\cdot) \in \mathbb{R}^n$, then $\mathcal{C}(P) = \mathbb{R}^n$; this makes that a token in P has a value in $\mathbb{R}^n$, with n being P-specific. The set $\mathcal{S}$ is defined to be the collection $\{\mathcal{C}(P); P \in \mathcal{P}\}$. Some places may not be associated with an SDE; for these places $\mathcal{C}(P) = \mathbb{R}^0 \triangleq \emptyset$, $\mathcal{V}_P$ and $\mathcal{W}_P$ are not defined, and the tokens will be the 'normal' black dots.

Specification of elements $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$ and $\mathcal{I}$ in local Petri net terms

Next, for each transition $T \in \mathcal{T}$, one should determine whether it is a guard transition, a delay transition or an immediate transition. A guard transition fires if the combined colours of its input tokens reach a boundary value. A delay transition fires after a random delay, hence models a duration. An immediate transition fires without delay. The guard transitions are collected in the set $\mathcal{T}_G$, the delay transitions are collected in $\mathcal{T}_D$ and the immediate transitions are collected in the set $\mathcal{T}_I$. Subsequently, the guards $\mathcal{G}$ (by means of the boundary values on the token colours) and the delays $\mathcal{D}$ (i.e., the jump intensity for the delays) are specified in detail. The firing measure $\mathcal{F}$ describes the colours of the tokens fired by a transition into its output places, given the colours of the tokens in the input places. Finally, the initial marking $\mathcal{I}$ describes which place(s) of the LPN initially contain one or more token(s) and describes the initial colour values of these tokens, hence

it describes the initial state of the process modelled by the SDCPN. It is possible to define several or a continuum of initial states and specify the choice by means of a probability measure.

5.2.2 Interconnections between LPNs

The interconnections between the LPNs have to be specified in a way that allows to start at the lowest level and then step by step to go up to the highest level, and such that an interconnection at a higher level does not imply a significant change at a lower level[2]. In this subsection, specific types of interconnections are identified that support such specification in a comprehensive way.

Following [FKK97], one step in enabling a systematic bottom-up specification of a Petri net is to ensure that each LPN always contains exactly one token. For air transport types of applications it often is useful to allow multiple tokens to be within one LPN, e.g., one for each aircraft. Hence, for SDCPN we relaxed the one-token-principle to the following requirement: *all interconnections between LPNs shall be such that the number of tokens in an LPN is not directly influenced by these interconnections*. Subsequently we identified two types of interconnections that satisfied our above requirement:

- Enabling arc (or inhibitor arc) from one place in one LPN to one transition in another LPN.
- Interaction Petri Net (IPN) from one (or more) transition(s) in one LPN to one (or more) transition(s) in another LPN.

Enabling and inhibitor interconnections are illustrated in Figures 5.1 and 5.2, respectively. Note that in these figures, each LPN is surrounded by a box. This boxing idea has also been used by, e.g., [FAP97] and [Kin97].

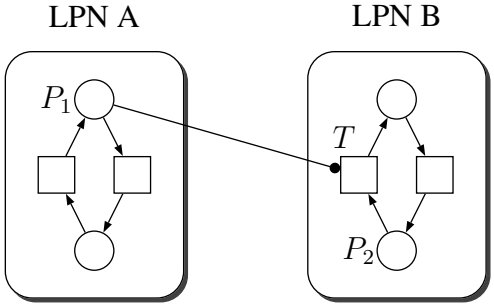


Figure 5.1 Illustration of an enabling arc from one place in LPN A to one transition in LPN B. Transition T can only fire if both its input places, P_1 and P_2, contain a token. However, upon firing, T does not remove the token from the place P_1 to which it is connected by an enabling arc

The transition at the tip of the enabling or inhibitor arc (i.e., transition T in LPN B in Figures 5.1 and 5.2) can only fire if the process modelled by LPN A is in a particular state or marking, and

[2]The typical exception to this is caused by non-local influences on $\mathcal{G}$, $\mathcal{D}$ and $\mathcal{F}$. We come back to this issue later.

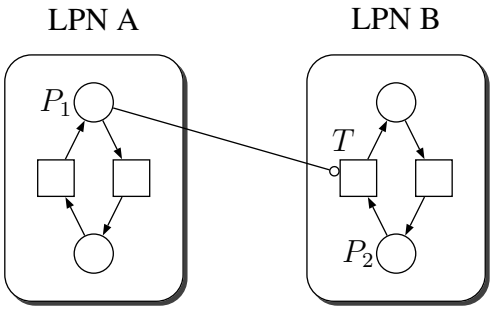


Figure 5.2 Illustration of an inhibitor arc from one place in LPN A to one transition in LPN B. Transition T can only fire if place P_1 to which it is connected by an inhibitor arc does not contain a token (and place P_2 does)

it may use the information existing in this marking of LPN A. For example, it may appear that the guard or delay of transition T in LPN B is dependent of the colour of the token in place P_1 in LPN A; however, the transition should not disturb the local process within LPN A. To model this, the Petri net graph needs to be extended with an enabling arc from place P_1 to transition T in order to get access to this information without consuming the token from P_1. Note that in addition, the firing measure and the guard or delay of the transition, which in the previous subsection has only been defined locally, needs to be adapted to allow for the extended number of input tokens. Since tokens are not consumed through enabling arcs at a transition firing, the state of LPN A is not changed through this firing. [FAP97] uses enabling arcs like this to model synchronisation. [FKK97] uses generalised stochastic Petri nets which support inhibitor arcs but do not support enabling arcs; however, [FKK97] does allow tokens of other modules be consumed and immediately placed back, which is similar[3] to using an enabling arc. Here, inhibitor arcs are generally used to model priority constructs and enabling arcs are used to model synchronisation constructs.

An *interaction Petri net* (IPN) consists of one or more places, and zero or more transitions. It connects, by means of ordinary arcs, one or more transition(s) in one LPN with one or more transition(s) in another LPN. If necessary, additional enabling or inhibitor arcs can be used to further connect the IPN with places or transitions in LPNs. An example of an IPN is given in Figure 5.3. It can be easily verified that an IPN does not influence the number of tokens in the LPNs it connects.

Interaction Petri nets are used where enabling or inhibitor arcs are insufficient to model the interconnection between two agents. For example, an IPN can temporarily hold on to state information from its input LPN (i.e., LPN A in Figure 5.3) while the state of this input LPN itself evolves further. Also, IPNs can be used to connect two transitions, while enabling or inhibitor arcs always connect a place with a transition. Note that our IPNs are similar to the interconnection blocks of [FKK97]. The connection of two LPNs through an interaction Petri net also has some similarity with place fusion, see, e.g., [HJS90] or [Kin97], except that our interaction Petri net will

[3]Though not the same: In case of timed events, placing a token back could be associated with sampling a new time delay for this token, which may be different from the delay it had.

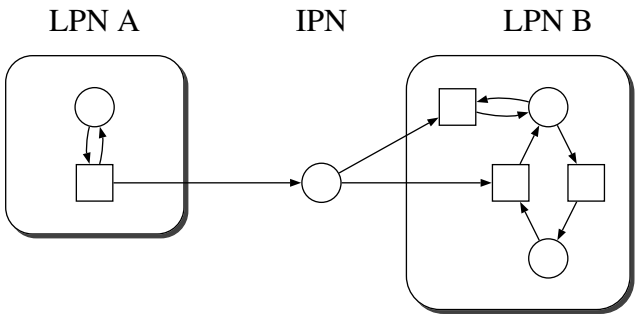


Figure 5.3 Illustration of an interaction Petri net from one transition in LPN A to two transitions in LPN B. The IPN exists of at least one place and connects two LPNs by ordinary arcs. If necessary, additional enabling or inhibitor arcs can be used to further connect the IPN with LPNs

not change the number of tokens in its connecting LPNs.

5.3 Interconnection mapping types

For a complex operation with many interactions between agents, the coupling of the LPNs might lead to a combinatorial growth of the number of interconnections with the size of the Petri net. To avoid this combinatorial growth as much as possible, this section develops eight hierarchical clustering and interconnection mapping approaches. These developments are based on how [Har87] composes statecharts and are referred to as *interconnection mapping types* I through VIII:

1. Interconnection mapping types I and II are defined to avoid possible duplication of transitions and arcs within LPNs caused by specifying interconnections between LPNs (see Section 5.3.1).

2. Interconnection mapping types III, IV and V are defined to avoid cluttering of interconnections between places and transitions of different LPNs (see Section 5.3.2).

3. Interconnection mapping types VI and VII define interconnections from or to hierarchical clusters of LPNs, which reduce the cluttering of interconnections (see Section 5.3.3).

4. Interconnection mapping type VIII avoids a duplication of transitions and arcs within an LPN and duplication of arcs between LPNs (see Section 5.3.4).

5. And finally, several combinations of interconnection mapping types I through VIII are derived (see Section 5.3.5).

The remainder of this section defines and illustrates these interconnection mapping types in more detail. In Section 5.4, the SDCPN formal definition is extended to include these interconnection mapping types.

5.3.1 Avoid duplication of transitions and arcs within an LPN

Some interconnections between LPNs lead to duplication of transitions and arcs within one of these LPNs. Suppose that a transition from place P_3 to place P_4 should only occur if P_1 *or* P_2 contains a token (or both), see Figure 5.4. Since a transition can only become enabled if all its input places contain a token, the use of only one transition between P_3 and P_4, with additional input places P_1 and P_2, would model an *and*-relation (i.e., both P_1 *and* P_2 contain a token) instead of an *or* relation. To model an *or* relation, it is necessary to use two transitions T_1 and T_2 between P_3 and P_4; one with P_1 as additional input place, and the other with P_2 as additional input place.

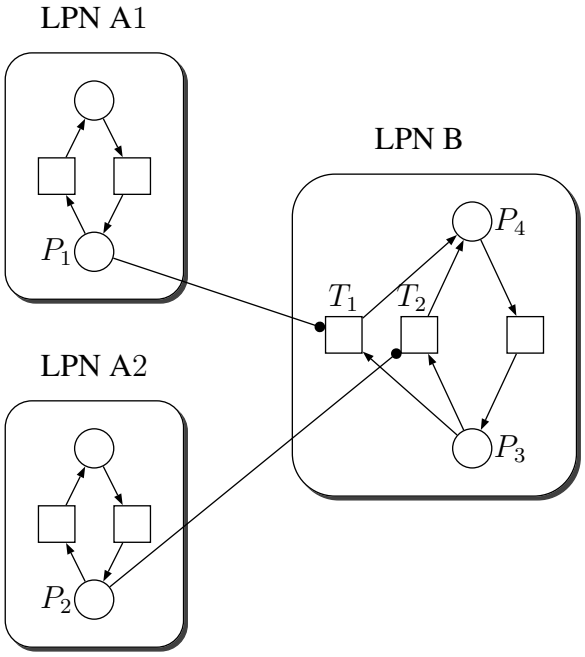


Figure 5.4 Illustration of duplication of transitions within an LPN

In most cases, since the duplicated transitions were one transition in the original LPN, they do not have an essentially different meaning. In particular, since the interconnections are mostly introduced to be able to make use of colours of tokens residing in other LPNs, these duplicated transitions will have *the same guard or delay and the same firing measure*. This makes that graphical duplication leads to reduced readability. This subsection presents some interconnection mapping types to avoid such graphical duplication.

Remark 5.1. *One may notice that in addition to the* 'or' *relation in Figure 5.4 one may think of other types of relations, such as* 'and'*,* 'exclusive or'*. Modelling an* and *relation is virtually standard practice in Petri net modelling: a transition that has two input places is enabled if both the first and the second input place contains a token. No duplication of arcs or transitions is necessary to model this. An* exclusive or *could be modelled by adding inhibitor arcs. For example, in Figure 5.4, if additional inhibitor arcs are drawn from P_1 to T_2 and from P_2 to T_1, then a transition from*

P_3 to P_4 can only then occur if either P_1 or P_2 contains a token, not if both contain a token. This situation does not appear to occur very often in practical applications, therefore no dedicated interconnection shorthand is defined for it. (The construct itself can still be used, though.)

Remark 5.2. *In an interconnection mapping type where a transition may be defined to represent a number, say s, of 'duplicated transitions', this transition is considered to be replaced by s of its copies. Each copy gets the same guard or delay function and the same firing measure as the original transition, and is connected to other places in the same way as the original transition was, if we exclude the arcs defining the interconnection mapping type considered (i.e., the enabling arcs in case of Figure 5.4).*

LPN interconnection mapping type I

Definition and example. Consider a set of s enabling arcs initiating at s places $P_1, \ldots, P_s$ that have the same colour type, merging into one arc and ending on one transition T that is in a different LPN than the s places $P_1, \ldots, P_s$. This type of arc is called *merging arc*. The merging point is denoted by a small black square.

Interconnection mapping type I means that transition T represents s actual duplicated transitions $T_1, \ldots, T_s$, and that s enabling arcs $A_1, \ldots, A_s$ are drawn; arc A_i connects place P_i with duplicated transition T_i ($i = 1, \ldots, s$). Figure 5.5 shows an example of this interconnection mapping type, for $s = 2$. Interconnection mapping type I is not defined with inhibitor or ordinary arcs instead of enabling arcs.

Interpretation and practical use. The most common application for interconnection mapping type I is a situation in which transition T in Figure 5.5 can only be enabled if place P_1 *or* place P_2 is current (or both). For example, T represents a mode switch from nominal operations to non-nominal operations, P_1 represents a non-nominal state for supporting system A1 and place P_2 ditto for system A2. Transition T switches to non-nominal operations if supporting system A1 or supporting system A2 goes into a non-nominal state (or both). This situation can only be modelled if transition T is duplicated into T_1 and T_2. If T would have both P_1 and P_2 as input places, it would only be enabled if both P_1 and P_2 contain a token, and this would model an *and* situation. However, since in effect it is still only one transition, an interconnection mapping type notational shorthand that presents it as such is a logical solution.

LPN interconnection mapping type II

Definition and example. Consider an enabling arc initiating at the edge of an LPN-box and ending on a transition T in another LPN-box.

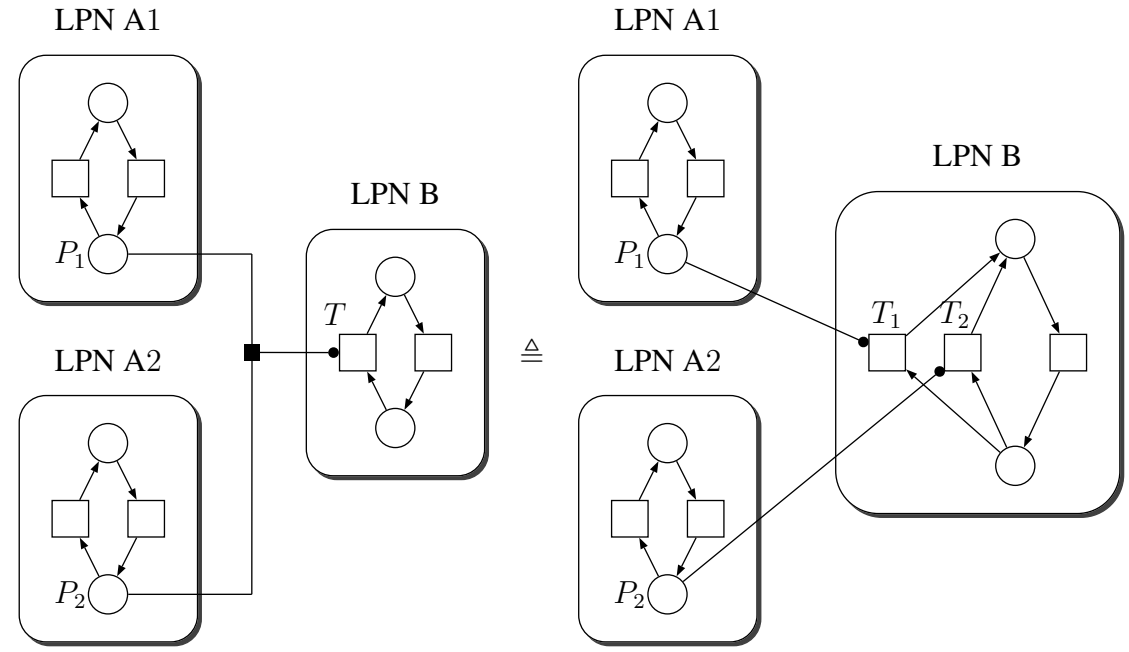


Figure 5.5 LPN interconnection mapping type I. The point where several arcs merge into one arc is represented by a small black square

Interconnection mapping type II means that the enabling arc represents s actual enabling arcs $A_1, \ldots, A_s$, and that transition T represents s duplications $T_1, \ldots, T_s$, where s is the number of places in the first LPN. These s places $P_1, \ldots, P_s$ should have the same colour type. Arc A_i connects place P_i with duplicated transition T_i ($i = 1, \ldots, s$). Figure 5.6 shows an example of this interconnection mapping type, for $s = 2$. Interconnection mapping type II is not defined with inhibitor or ordinary arcs instead of enabling arcs.

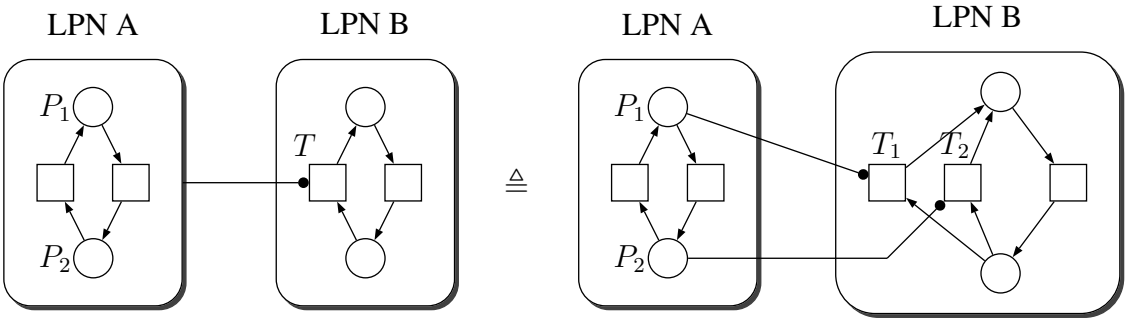


Figure 5.6 LPN interconnection mapping type II

Interpretation and practical use. The primary use for this structure is if transition T (or, rather, its guard or delay) wants to make use of the colour of the token in LPN A, wherever this token resides, and without consuming this token. This situation can only be modelled by duplication of T into T_1 and T_2, since otherwise, because there is usually a token in either P_1 or P_2, not in both,

transition T would never be enabled.

5.3.2 Avoid cluttering of interconnections between LPNs

Interconnection mapping types I and II avoid the duplication problem, but there also exist other types of interconnections that appear cluttered due to the many enabling arcs and IPNs between places and transitions of different LPNs. The result becomes unreadable. This subsection presents some interconnection mapping types to avoid this:

- Interconnection mapping types III-A and III-B can be applied to avoid cluttering of enabling arcs or inhibitor arcs.
- Interconnection mapping types IV and V can be applied to avoid IPNs cluttering.

LPN interconnection mapping types III-A and III-B

Definition and example. For interconnection mapping type III-A, consider an enabling arc initiating at a place P in one LPN and ending on the edge of another LPN's box.

This means that the enabling arc represents s actual enabling arcs $A_1, \ldots, A_s$, where s is the number of transitions in the second LPN. These transitions are referred to as $T_1, \ldots, T_s$. Arcs $A_1, \ldots, A_s$ all initiate at place P, and arc A_i ends on transition T_i ($i = 1, \ldots, s$). Figure 5.7 shows an example of interconnection mapping type III-A, for $s = 2$.

Interconnection mapping type III-B is similar to III-A, but with inhibitor arcs instead of enabling arcs. A version with ordinary arcs is not defined.

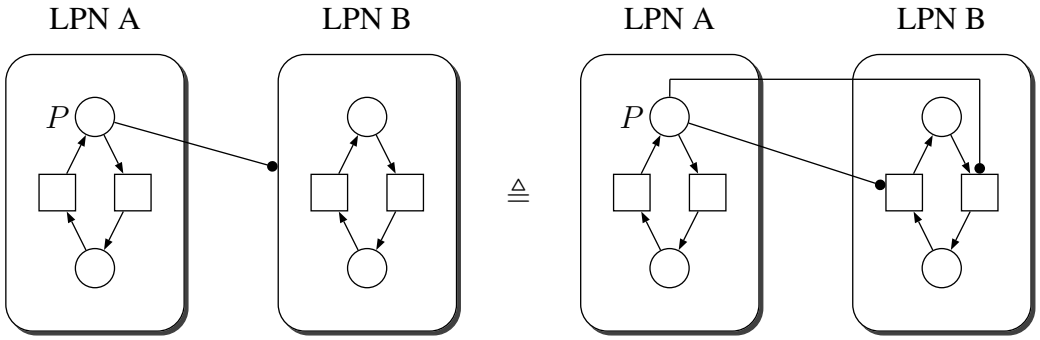


Figure 5.7 LPN interconnection mapping type III-A. For type III-B, the enabling arcs are replaced by inhibitor arcs

Interpretation and practical use. A practical use for interconnection mapping type III-A is where any of the transitions in LPN B can only fire if place P in LPN A is current. In such case these transitions in LPN B can also make use of the colour of the token in place P. For type

III-B, any of the transitions in LPN B can only fire if place P in LPN A is *not* current. The use of III-B with respect to III-A is mainly practical if LPN A contains more than two places and if the transitions in LPN B do not need to make use of the colour of the token in place P.

LPN interconnection mapping type IV

Definition and example. Consider an ordinary arc initiating at the edge of an LPN-box and ending on a place P within an IPN.

Interconnection mapping type IV means that the ordinary arc represents s actual ordinary arcs $A_1, \ldots, A_s$, where s is the number of transitions in the LPN-box. These transitions are referred to as $T_1, \ldots, T_s$. Arc A_i initiates at transition T_i ($i = 1, \ldots, s$) and all arcs $A_1, \ldots, A_s$ end on place P.

Figure 5.8 shows an example of this interconnection mapping type, for $s = 2$. Interconnection mapping type IV is not defined with enabling or inhibitor arcs instead of ordinary arcs.

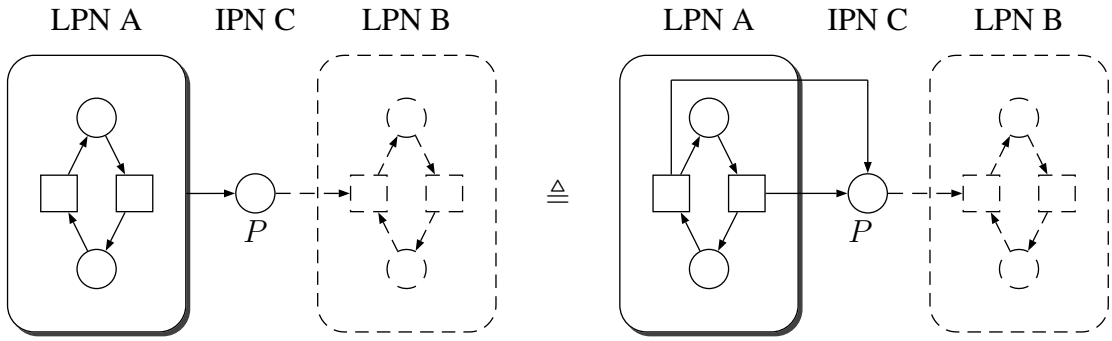


Figure 5.8 LPN interconnection mapping type IV. LPN B has been 'dashed-out' since it does not make part of the interconnection mapping type

Interpretation and practical use. An application for this interconnection mapping type is where there is need for 'asynchronous information exchange'. For example, each time a transition (any transition) in LPN A fires, it sends some information to LPN B by means of a token into IPN C. LPN B may not want to make immediate use of this information, e.g., in case it needs to finish some other local actions first, and in the mean time LPN A may also want to continue with its own actions instead of waiting for LPN B to be ready. Therefore, the token stays in IPN C until LPN B is ready for it, after which the token is finally consumed by a transition in LPN B.

LPN interconnection mapping type V

Definition and example. Consider an ordinary arc that starts from a place P within an IPN and ends on the edge of an LPN-box.

Interconnection mapping type V means that the ordinary arc represents s actual ordinary arcs $A_1, \ldots, A_s$, where s is the number of transitions in the LPN. These transitions are referred to as $T_1, \ldots, T_s$. All arcs $A_1, \ldots, A_s$ initiate at place P, and arc A_i ends on transition T_i ($i = 1, \ldots, s$). Figure 5.9 shows an example of this interconnection mapping type, for $s = 2$. Interconnection mapping type V is not defined with enabling or inhibitor arcs instead of ordinary arcs.

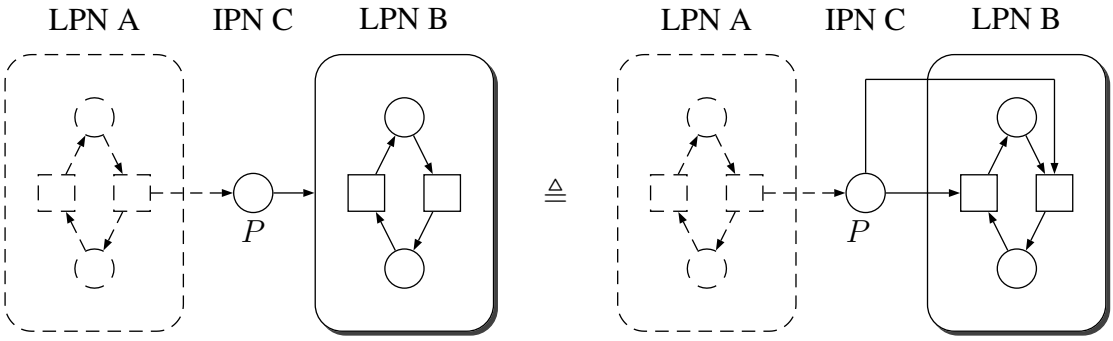


Figure 5.9 LPN interconnection mapping type V

Interpretation and practical use. Practical use for this interconnection mapping type V is similar as for type IV, except that now the information in IPN C is used by any of the transitions in LPN B, whichever is enabled first.

5.3.3 Clustering of LPNs

We saw that some of the previously defined interconnection mapping types represent shorthand notations for situations where a transition needs access to the token in another LPN, wherever it resides, or for situations where a token is used by another LPN, by whichever transition is enabled first. These situations can also be extended to multiple LPNs. In this subsection, we define enabling arcs that go from or to a *cluster* of LPNs. This is done following interconnection mapping types VI and VII. Note that the definitions of these types have been modified slightly with respect to the definitions given in [EKBK06] (i.e., the original paper on which this chapter was based).

LPN interconnection mapping types VI-A, VI-B and VI-C

Definition and example. For interconnection mapping type VI-A, consider one LPN A with a place P, and a set of s LPNs Bi ($i = 1, \ldots, s$), which set is enclosed by a large box, referred to as *cluster-box*. Also consider an enabling arc that initiates at place P in LPN A and ends on the edge of the cluster-box with the set of LPNs Bi.

Interconnection mapping type VI-A means that the enabling arc represents s actual enabling arcs $A_1, \ldots, A_s$, with arc A_i initiating at P and ending on the edge of LPN Bi, $(i = 1, \ldots, s)$. Note

that repeated use is made of interconnection mapping type III-A, which is defined from a place to the edge of an LPN-box.

A version with inhibitor arcs instead of enabling arcs is also defined, which is referred to as type VI-B. Interconnection mapping type VI can also be defined from a place within an IPN to a cluster-box of LPNs, by means of an ordinary arc; this type is referred to as type VI-C. Figure 5.10 shows an example of interconnection mapping type VI-A, for $s = 2$.

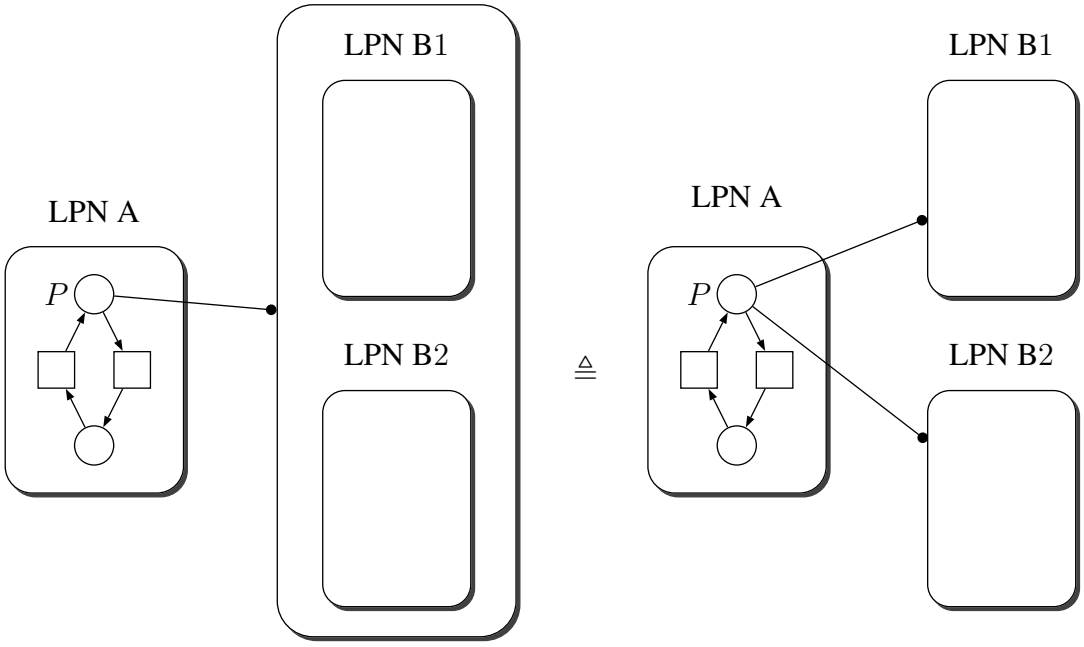


Figure 5.10 LPN interconnection mapping type VI-A; this type makes repeated use of type III-A. For type VI-B, the enabling arcs are replaced by inhibitor arcs; this type makes repeated use of type III-B. For type VI-C, the enabling arcs are replaced by ordinary arcs and LPN A is replaced by an IPN; this type makes repeated use of type V

Interpretation and practical use. This situation occurs where all transitions in several LPNs can only be enabled if place P of another LPN is current. In such a case these transitions can also make use of the colour of the token in P. As such, interconnection mapping type VI-A is an extension of interconnection mapping type III-A. For type VI-B, any of the transitions in several LPNs can only fire if place P in LPN A is not current. Type VI-B is an extension of type III-B. Similarly, the use of VI-B with respect to VI-A is mainly practical if LPN A contains more than two places. Type VI-C is an extension of type V, with the interpretation that the information in the IPN is used by any of the transitions in the cluster-box, whichever is enabled first.

LPN interconnection mapping type VII

Definition and example. Consider a set of s LPNs Ai ($i = 1, \ldots, s$) which is enclosed by a cluster-box, and there is one LPN B (not in the cluster-box) which contains a transition T. Also consider an enabling arc that initiates at the edge of the cluster-box with the set of LPNs Ai ($i = 1, \ldots, s$) and ends on transition T.

Interconnection mapping type VII means that the enabling arc represents s actual enabling arcs $A_1, \ldots, A_s$, and that transition T represents s duplications $T_1, \ldots, T_s$. Arc A_i initiates at the edge of LPN Ai and ends on duplication T_i ($i = 1, \ldots, s$). All places in all LPNs Ai should have the same colour type. Interconnection mapping type VII is not defined with ordinary or inhibitor arcs instead of enabling arcs. Figure 5.11 shows an example of this interconnection mapping type, for $s = 2$.

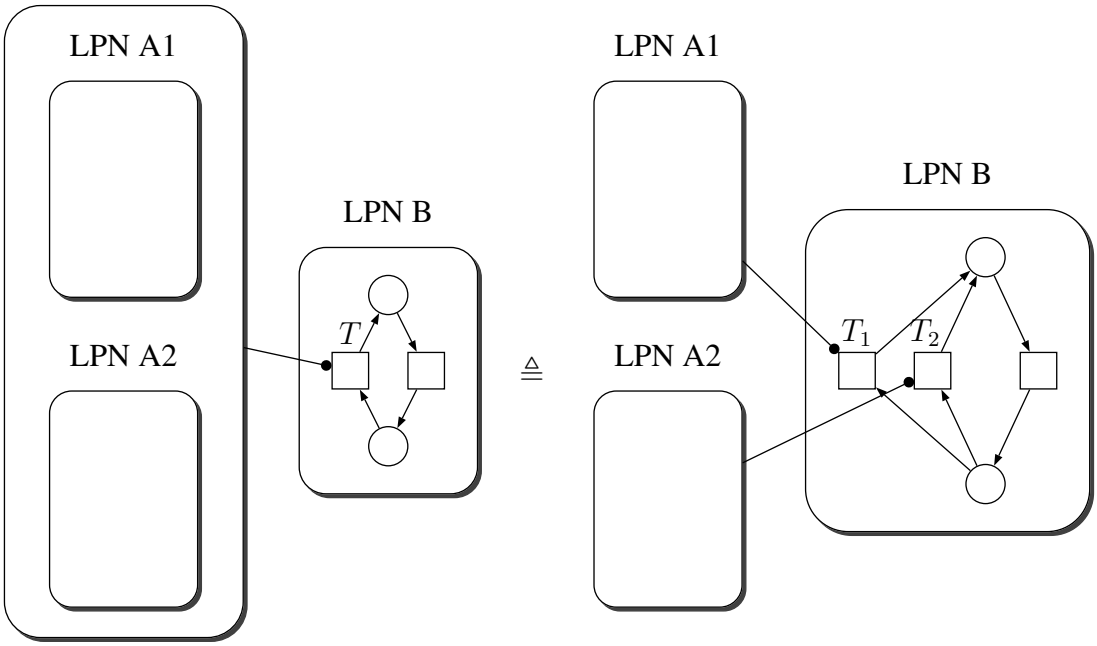


Figure 5.11 LPN interconnection mapping type VII; this type makes repeated use of type II

Interpretation and practical use. This is an extension of interconnection mapping type II: Transition T wants to make use of the colours of the tokens in all LPNs Ai, wherever they reside, and without consuming these tokens.

5.3.4 Avoid duplication and cluttering within an LPN

Finally, we introduce an additional interconnection mapping type which avoids duplication of transitions and arcs within an LPN, and consequently cluttering of arcs between LPNs:

LPN interconnection mapping type VIII

Definition and example. Consider an ordinary arc initiating at the edge of an LPN-box and ending on a transition T inside the *same* LPN-box.

Interconnection mapping type VIII means that this ordinary arc represents s actual ordinary arcs $A_1, \ldots, A_s$, and the transition represents s duplications $T_1, \ldots, T_s$, where s is the number of places in the LPN-box. These s places are referred to as $P_1, \ldots, P_s$ and they all should have the same colour type. Arc A_i initiates at place P_i and ends at duplicated transition T_i ($i = 1, \ldots, s$).

This interconnection mapping type VIII is not defined for enabling arcs or inhibitor arcs instead of ordinary arcs. Figure 5.12 illustrates how it avoids both the duplication of transitions and arcs within an LPN, and the consequential duplication of arcs between LPNs, for $s = 3$.

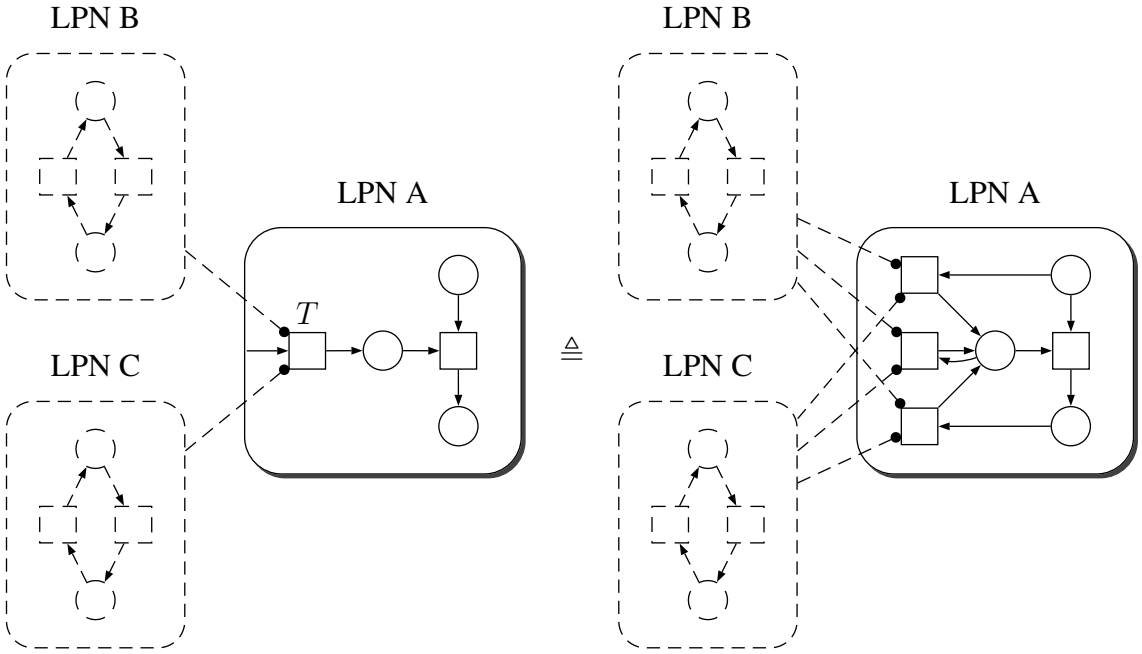


Figure 5.12 LPN interconnection mapping type VIII, which avoids duplication of arcs and transitions within an LPN and duplication of arcs between LPNs

Interpretation and practical use. This situation occurs if a transition T needs access to the token in its own LPN, wherever it resides.

5.3.5 Combinations of interconnection mapping types

Interconnection mapping types can also be combined, such as

- type II with III-A (i.e., an enabling arc from an LPN-box to another LPN-box),

- type VII with III-A (an enabling arc from a cluster-box to an LPN-box; this combination makes repeated use of type II),
- type VI-A with II (an enabling arc from an LPN-box to a cluster-box; this combination makes repeated use of type III-A), or
- type VII with III-A, VI-A and II (enabling arc from a cluster-box to a cluster-box).

Note that type II could also be combined with type I (i.e., a merging arc from a set of LPN-boxes to a transition), but since in effect this combination is the same as type VII (i.e., meaning that enabling arcs are drawn from each LPN-box to duplications of transitions), and since combining type I with other types appears to pose restrictions on the order in which the interconnection mapping types are applied, it is omitted as an additional combination. In addition, one could argue about including in the list the combination of type IV with type V (ordinary arcs from an LPN-box to another LPN-box via an IPN). Since this really is a sequence of types rather than a new combination, it is also omitted from the list.

An illustration of combination of II with III-A is given in Figure 5.13.

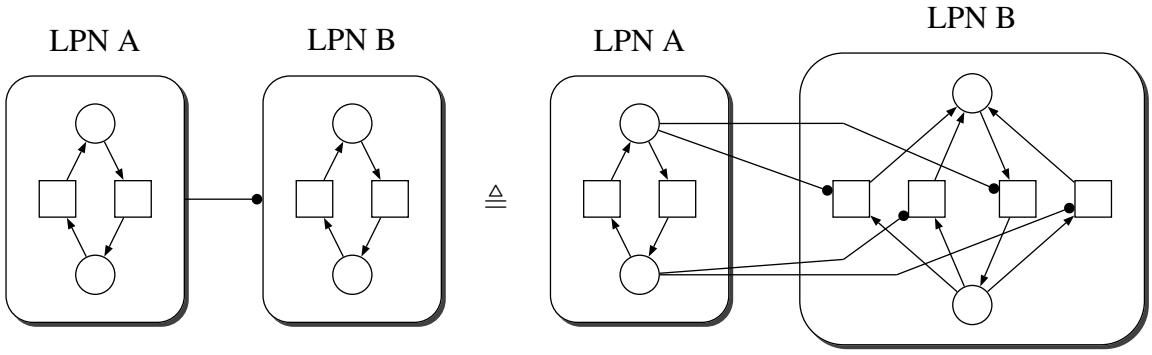


Figure 5.13 LPN interconnection mapping types II and III-A combined. This situation occurs if any of the transitions in LPN B needs to make use of the colour of the token in LPN A, wherever this token resides

Remark 5.3. *The interconnection mapping types introduced in this section can also be used for other types of Petri nets than SDCPN (or DCPN), provided that these other types of Petri nets support the same graphical elements as SDCPN (or DCPN), such as enabling arcs. If this is not the case, the interconnection mapping types might be used with ordinary arcs instead of enabling arcs, but then the restriction that the number of tokens in an LPN cannot be changed by the interconnections must be removed.*

It is also noted that in practice, the use of interconnection mapping types II and VII and their combinations with other types will mainly be relevant if the Petri net contains distinguishable (i.e., coloured) tokens, since otherwise there would not be a distinction between the duplicated

transitions. Similarly, interconnection mapping types V and VI are mainly relevant for timed and/or coloured nets, since the use of IPNs is typical for the transport of information and for modelling a buffer in which this information can be temporarily stored.

Remark 5.4. *It can be noticed from the developments introduced in this section, that the interconnection mapping types significantly increase the possibilities for drawing arcs between types of nodes. In an SDCPN, an ordinary arc can be drawn only between a place and a transition or vice versa; an enabling arc or inhibitor arc can be drawn from a place to a transition only. The interconnection types allow arcs to be drawn between places, transitions and various types of boxes. However, there are still a few restrictions.*

In order to determine what is allowed and what is not, we made an analysis in which we first identified the types of nodes available for initiating or ending an arc (i.e., place in an LPN, place in an IPN, transition in an LPN, transition in an IPN, LPN-box, cluster-box), and next, for each ordered combination of nodes and for each type of arc available (i.e., ordinary arc, enabling arc, inhibitor arc, merging arc) we determined if this arc could be drawn between this combination of nodes according to the interconnection mapping rules defined, including their combinations, and if additional types should be defined. For this analysis, we used that an arc between two places, or between two transitions is never allowed, and neither is an enabling arc or inhibitor arc from a transition or to a place. For all other combinations we analysed if an interconnection mapping type should be (or could be made) available to provide a formal meaning for this combination. The analysis is provided in the appendix, Section 5.6. The result of this analysis is incorporated in the following section.

5.4 Extension of SDCPN with interconnection mapping types I through VIII

This subsection extends the formal SDCPN definition of Section 4.3 to include the interconnection mapping types identified in Section 5.3. The extension is referred to as SDCPN$^{\text{imt}}$.

Definition 5.1 (SDCPN$^{\text{imt}}$)**.** *An SDCPN^{imt} is a collection of elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{B}^{\text{imt}}$, $\mathcal{A}^{\text{imt}}$, $\mathcal{L}^{\text{imt}}$, $\mathcal{N}^{\text{imt}}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) together with five rules R5–R9 which prescribe how the graphical elements may be connected, and an execution prescription which makes use of a sequence $\{U_i; i = 0, 1, \ldots\}$ of independent uniform $U[0, 1]$ random variables, of independent sequences of mutually independent standard Brownian motions $\{B_t^{i,P}; i = 1, 2, \ldots\}$ of appropriate dimensions, one sequence for each place, and of five rules R0–R4 that solve enabling conflicts.*

Section 5.4.1 explains the SDCPN$^{\text{imt}}$ elements and the rules R5–R9, Section 5.4.2 explains the

execution, Section 5.4.3 discusses the relationship between SDCPN$^{\text{imt}}$ and general stochastic hybrid process (GSHP).

5.4.1 SDCPN$^{\text{imt}}$ elements

Elements $\mathcal{P}$, $\mathcal{T}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$ are as in the definition of SDCPN (Section 4.3). The other elements and Rules R5–R9 are outlined below:

$\mathcal{A}^{\text{imt}} = \mathcal{A} \cup \mathcal{A}_m$ is the set of arcs in the SDCPN$^{\text{imt}}$. It equals the set of arcs $\mathcal{A} = \mathcal{A}_o \cup \mathcal{A}_e \cup \mathcal{A}_i$ as defined for SDCPN (Section 4.3), extended with a set of merging arcs $\mathcal{A}_m$.

A merging arc is a set of $s \geq 2$ enabling arcs merging into one enabling arc, where s is finite but can be different for each merging arc. The merging point is denoted by a small black square.

$\mathcal{B}^{\text{imt}} = \mathcal{B}_L \cup \mathcal{B}_C \cup \mathcal{B}_S$ is a set of boxes which consists of a set $\mathcal{B}_L$ of LPN-boxes, a set $\mathcal{B}_C$ of cluster-boxes, and one element $\mathcal{B}_S$ which is a box that will be drawn around all elements of the SDCPN$^{\text{imt}}$ except itself.

Each box in $\mathcal{B}^{\text{imt}}$ is drawn as a rectangle with rounded corners. Note that at this definition level, each element of $\mathcal{B}^{\text{imt}}$ is just an empty box. The box function $\mathcal{L}^{\text{imt}}$ (see definition next) will specify the actual contents (i.e., the places and transitions or other boxes) of each box in $\mathcal{B}^{\text{imt}}$.

$\mathcal{L}^{\text{imt}} : \mathcal{P} \cup \mathcal{T} \cup \mathcal{B}_L \cup \mathcal{B}_C \rightarrow \mathcal{B}^{\text{imt}}$ is a box function which specifies the contents of each box in $\mathcal{B}^{\text{imt}}$: $\mathcal{L}^{\text{imt}}$ maps each place in $\mathcal{P}$ to one box in $\mathcal{B}^{\text{imt}}$, each transition in $\mathcal{T}$ to one box in $\mathcal{B}^{\text{imt}}$, and each box in $\mathcal{B}_L$ to one box in $\mathcal{B}_C \cup \mathcal{B}_S$. All cluster-boxes (in $\mathcal{B}_C$) are mapped to $\mathcal{B}_S$. Places (and transitions) that form IPNs are not mapped to an LPN-box (in $\mathcal{B}_L$) but can be mapped to a cluster-box (in $\mathcal{B}_C$), or else, to $\mathcal{B}_S$, and at least two LPN-boxes should be mapped to each cluster-box. There is no hierarchy of cluster-boxes within cluster-boxes.

$\mathcal{L}^{\text{imt}}$ maps each element of $\mathcal{P} \cup \mathcal{T} \cup \mathcal{B}_L \cup \mathcal{B}_C$ to the box immediately enclosing it. Hence, for each LPN-box in $\mathcal{B}_L$, the box function specifies which places in $\mathcal{P}$ and which transitions in $\mathcal{T}$ are drawn in it to form an LPN; for each cluster-box in $\mathcal{B}_C$ it specifies which (at least two) LPN-boxes in $\mathcal{B}_L$ are drawn in it to form a cluster of LPNs. Some places (and transitions) are not inside any LPN-box; these form the IPNs. It is, however, possible that IPNs are part of a cluster-box (although they are not part of an LPN-box), and if not, they are in box $\mathcal{B}_S$. Similarly, not all LPN-boxes need to be inside a cluster-box; those that are not, are assigned to $\mathcal{B}_S$. It is noted that $\mathcal{B}_S$ is usually not physically drawn. The introduction of this box $\mathcal{B}_S$ was necessary to be able to map *all* SDCPN$^{\text{imt}}$ places and transitions to a box, even if they are not part of an LPN-box or cluster-box.

$\mathcal{N}^{\text{imt}}$ is a node function which maps each arc in $\mathcal{A}_m$ to an ordered pair of which the first component is a set of places (but not in IPNs), and the second component is a transition. Furthermore, $\mathcal{N}^{\text{imt}}$ maps each arc in $\mathcal{A} = \mathcal{A}_o \cup \mathcal{A}_e \cup \mathcal{A}_i$, to an ordered pair of nodes, where a node is a place, a transition, an LPN-box or a cluster-box. Multiple arcs between the same pair of nodes are allowed (but not both an inhibitor arc and another type of arc). There are five rules R5–R9 to be adhered to in order to assure unambiguity. These rules have been derived by analysis in the appendix, Section 5.6.

R5 Ordinary arcs can only be drawn from a place to a transition or vice versa within the same LPN-box, from a place in an IPN to a transition, from a transition to a place in an IPN, from a place in an IPN to an LPN-box, from an LPN-box to a place in an IPN, from a place in an IPN to a cluster-box that does not contain this place, or from an LPN-box to a transition in the same LPN-box.

R6 Enabling arcs can only be drawn from any place to any transition, from a place in an LPN to another LPN-box or to a cluster-box that does not contain the place's LPN, from an LPN-box to a transition in another LPN, from a cluster-box to a transition in an LPN that is not in this cluster-box, or between two boxes (i.e., LPN-LPN, LPN-cluster, cluster-LPN or cluster-cluster), provided the LPN at one end of the arc is not in the cluster at the other end of the arc.

R7 Inhibitor arcs can only be drawn from any place to any transition, or from a place in an LPN to an LPN-box or cluster-box. If two nodes are already connected to each other by an inhibitor arc, they cannot also be connected (in the same direction) by another arc.

R8 Merging arcs can only be drawn from a set of places (but not in IPNs) to a transition that is in another LPN than these places.

R9 If an arc initiates at a set of places (as in case of merging arc), or at the edge of an LPN-box or cluster-box that contains multiple places, then all these places should have the same colour type as specified by $\mathcal{C}$.

5.4.2 SDCPN$^{\text{imt}}$ execution

The execution of an SDCPN$^{\text{imt}}$ satisfies the execution rules as described in Section 4.3, including the rules R0–R4, once the SDCPN$^{\text{imt}}$ has been uniquely transformed to an SDCPN according to the interconnection mapping types defined in Section 5.3. However, in most cases, one will want to avoid having to perform this transformation, since it will require having to draw a cluttered Petri net graph. Therefore, this section also derives the execution rules for the situation in which the interconnection mapping types have been used.

Since the number of places and the token colour functions are not affected by the interconnection mapping types, the execution of the SDCPN$^{\text{imt}}$ is exactly the same as for SDCPN as long as the tokens stay in their current place: the colour of a token while residing in a place P evolves according to a stochastic differential equation that is governed by the colour functions $\mathcal{V}_P$ (for the drift coefficient) and $\mathcal{W}_P$ (for the diffusion coefficient).

For a transition to be allowed to remove and produce tokens, it first has to be pre-enabled (i.e., have a token in each of its input places), and next its guard or delay condition (if applicable) needs to hold true. This is similar to the situation in SDPCN; however, due to the use of the interconnection mapping types, some transitions actually represent several duplications, and the SDCPN$^{\text{imt}}$ graph may not always give direct insight into whether a transition is pre-enabled, and by which combination of token colours. Therefore, rules are given next to provide this insight.

Note that the input tokens of a transition can be of two kinds: 'local input tokens', i.e., residing in places that are within the same LPN the transition makes part of, and 'external input tokens', i.e., residing in places that are outside the LPN the transition makes part of. Both kinds of tokens determine whether and when a transition is enabled. However, below, to keep the description brief, only the external input tokens are considered.

- **Transition has incoming ordinary and/or enabling and/or inhibitor arcs that are connected to places only.**
  Pre-enabling and enabling is according to the 'normal' SDCPN rules.

- **Transition has an incoming merging arc (see interconnection mapping type I).**
  Transition is pre-enabled if it has a token in at least one of the places connected to this merging arc.
  Transition is enabled if it is enabled by this input token according to the rules as described for SDCPN (e.g., its guard evaluates to true, or its delay has passed). If there are several tokens in these connected places, the transition guard or delay function uses their colours in parallel for its evaluation. By rule R9, all connected places have the same colour type.

- **Transition has an (enabling) incoming arc that is connected with an LPN-box (see interconnection mapping type II).**
  Transition is pre-enabled if there is at least one token somewhere in this input LPN-box (and this is usually the case).
  Transition is enabled if it is enabled by this input token as described for SDCPN. If there are several tokens in the input LPN-box, the transition guard or delay function uses their colours in parallel for its evaluation. By rule R9, all these input tokens have the same colour type.

- **Transition has an (enabling) incoming arc that is connected with a cluster-box (see interconnection mapping type VII).**

Transition is pre-enabled if there is at least one token somewhere in this input cluster-box (and this is usually the case).
Transition is enabled if it is enabled by this input token as described for SDCPN. If there are several tokens in the input cluster-box, the transition guard or delay function uses their colours in parallel for its evaluation. By rule R9, all these input tokens have the same colour type.

- **Transition has an (ordinary) incoming arc that is connected with the LPN-box the transition makes part of itself (see interconnection mapping type VIII).**
  Transition is pre-enabled if there is at least one token somewhere in this LPN-box (and this is usually the case).
  Transition is enabled if it is enabled by this input token as described for SDCPN. If there are several tokens in the LPN-box, the transition guard or delay function uses their colours in parallel for its evaluation. By rule R9, all these input tokens have the same colour type.

- **Transition makes part of an LPN-box or of a cluster-box that has an input place by means of an enabling arc (see interconnection mapping type III-A or VI-A).**
  Transition is pre-enabled if there is a token in this input place.
  Transition is enabled if it is enabled by this input token as described for SDCPN. If there are several tokens in the input place, the transition guard or delay function uses their colours in parallel for its evaluation.

- **Transition makes part of an LPN-box or of a cluster-box that has an input place by means of an inhibitor arc (see interconnection mapping type III-B or VI-B).**
  Transition is pre-enabled if there is no token in this input place.
  Enabling of the transition is not further affected by external interconnections.

- **Transition makes part of an LPN-box or of a cluster-box that has an input place that is in an IPN (see interconnection mapping type V or VI-C).**
  Transition is pre-enabled if there is a token in this input place.
  Transition is enabled if it is enabled by this input token as described for SDCPN. If there are several tokens in the input place, the transition guard or delay function uses their colours in parallel for its evaluation.

- **Transition makes part of an LPN-box or of a cluster-box that has an input LPN-box (see interconnection mapping type III-A combined with II, or type VI-A (making repeated use of III-A) combined with II)**
  Transition is pre-enabled if there is at least one token somewhere in the input LPN-box (and this is usually the case).

Transition is enabled if it is enabled by this input token as described for SDCPN. If there are several tokens in the input LPN-box, the transition guard or delay function uses their colours in parallel for its evaluation. By rule R9, all these input tokens have the same colour type.

- **Transition makes part of an LPN-box or of a cluster-box that has an input cluster-box (see interconnection mapping type VII (making repeated use of II) combined with III-A, or see type VI-A combined with III-A, II and VII)**
  Transition is pre-enabled if there is at least one token somewhere in the input cluster-box (and this is usually the case).
  Transition is enabled if it is enabled by this input token as described for SDCPN. If there are several tokens in the input cluster-box, the transition guard or delay function uses their colours in parallel for its evaluation. By rule R9, all these input tokens have the same colour type.

When a transition is enabled, it produces output as determined by its firing measure $\mathcal{F}$. Since the set of output places of a transition is clear from the SDCPN$^{\text{imt}}$ graph, even if the transition actually represents a number of duplications, this output can be determined following the SDCPN rules. A possible exception is the case where interconnection mapping type IV is applied, in which an ordinary arc from an LPN-box to a place in an IPN represents s actual ordinary arcs $A_1, \ldots, A_s$, where s is the number of transitions in the LPN-box; these transitions are referred to as $T_1, \ldots, T_s$. Although in the SDCPN graph version, there is an arc A_i from T_i to P, in the SDCPN$^{\text{imt}}$ graph, there is no arc from T_i to P; there is only an arc from the edge of T_i's LPN-box to P. This means that each transition in the LPN has an additional output place which is not directly visible from the SDCPN$^{\text{imt}}$ graph, but which may expect output from the transition's firing measure. Hence, interconnection mapping type IV deserves special consideration in this respect. In cases where problems can be expected, it is recommended to not use interconnection mapping type IV.

5.4.3 Relation between SDCPN$^{\text{imt}}$ and GSHP

In this section we showed that SDCPN$^{\text{imt}}$ is an extension of SDCPN, by the inclusion of an additional arc type (merging arc), by the inclusion of different types of boxes that surround sets of places and transitions or other boxes, and by the inclusion of a set of rules R5–R9 that allow arcs also to be drawn from or to the edges of these boxes. More specifically, an SDCPN$^{\text{imt}}$ reduces to an SDCPN if $\mathcal{A}_m = \emptyset$ (hence there are no merging arcs) and $\mathcal{B}^{\text{imt}} = \emptyset$ (hence there are no LPN-boxes or cluster-boxes). With this, an SDCPN is a special case of an SDCPN$^{\text{imt}}$ and thus there exists an into-mapping from the set of SDCPN to the set of SDCPN$^{\text{imt}}$. Through the unambiguous definitions of the interconnection mapping types in Section 5.3 and the rules R5 through R9 in Subsection 5.4.1, each given SDCPN$^{\text{imt}}$ can also be uniquely transformed to an SDCPN. This is

done by applying the transformation rules of Section 5.3 to each occurrence of interconnection mapping types use. Since each interconnection mapping type has a unique definition, which is in terms of SDCPN rules, the resulting SDCPN is unique and according to the definition of SDCPN.

In Chapter 4 it was proven that, under a few conditions, an SDCPN is probabilistically equivalent to a particular powerful class of stochastic hybrid process, named general stochastic hybrid process (GSHP). Section 4.7 provided a discussion on the conditions. Since each SDCPN^{imt} can be uniquely transformed to an SDCPN, the equivalence between SDCPN^{imt} and GSHP also exists, and if the conditions hold true for SDCPN, they will also hold true for SDCPN^{imt}.

5.5 Concluding remarks

For the compositional specification of a multi-agent hybrid system this chapter has introduced a hierarchical extension of the compositional specification power of Petri nets, which avoids the need for all kinds of low-level changes once making connections at a higher model level. Moreover, the problem of combinatorial growth of the number of interconnections with the size of the Petri net is remedied. The usage of the interconnection mapping types improves simplicity and readability. In addition, once the interpretation and practical use of the interconnection mapping types have been mastered, their use improves resilience against modelling errors. In Chapter 6, the effectiveness of the SDCPN-based compositional specification is illustrated for an air transport example.

The extension with interconnection mapping types further increases the modelling power of SDCPN, while maintaining the SDCPN analysis properties. A key property of SDCPN is the existence of equivalence relations with GSHP. This implies that the compositional modelling power of SDCPN^{imt} is combined with the stochastic analysis power of GSHP.

5.6 Appendix: Analysis of interconnection mapping types allowed

This appendix systematically analyses which arcs can be used to connect which SDCPN^{imt} nodes. The objective is to ensure that all defined interconnection mapping types make sense and that no important types are forgotten.

Tables 5.1 and 5.2 summarise the interconnection mapping types and their possible combinations, give brief descriptions, and analyse if the same interconnection mapping type may also be introduced with other types of arcs (e.g., inhibitor instead of enabling arc). Tables 5.3 and 5.4 address the analysis problem from an arcs perspective by systematically considering all combinations of SDCPN^{imt} nodes and analysing which types of arcs can be used to connect these

combinations.

The codes used in these tables, i.e., (*), (**), (1), (2), . . . , are explained below:

***** These combinations are not allowed due to basic Petri net rules: no arcs allowed from place to place; no arcs allowed from transition to transition; no enabling or inhibitor arcs allowed leaving a transition and/or ending at a place.

****** These combinations are not logical or compatible, e.g., arc from place *in IPN* to transition *in same LPN*.

(1) These combinations draw an arc A from a place P to the edge of an LPN-box that the place makes part of. One could imagine a meaning to this combination, being that this arc represents s arcs $A_1, \ldots, A_s$, where s is the number of transitions also existing in the LPN-box, and where for $i = 1, \ldots, s$, arc A_i is drawn from P to T_i. This would mean that each transition inside the LPN-box would have place P as input place. In case A is an ordinary arc, the token in P would be consumed upon firing of any enabled transition. In case A is an enabling arc, any transition would have constant access to the colour information of the token in place P. In case A is an inhibitor arc, no transition could be enabled as long as place P contains a token.

These combinations could be interesting to include by means of an additional interconnection mapping type. However, if the number of transitions inside the LPN-box is limited, the added value is also limited: the use of the interconnection mapping type would only lead to reduction of a few arcs within an LPN, there is no reduction in the interconnections between LPNs. The effort of having to additionally 'master' the meaning of this interconnection mapping type does not weigh up to the positive effects. Therefore, for the time being, these combinations are not included as a formal interconnection mapping type shorthand.

(2) These combinations draw an arc A between a place P in an LPN or in an IPN to the edge of a cluster-box the place makes part of. One could imagine a meaning to this combination, similar to the one for item (1), being that this arc represents s arcs $A_1, \ldots, A_s$, where s is the number of transitions also existing in the cluster-box, and where for $i = 1, \ldots, s$, arc A_i is drawn from P to T_i. This would mean that each transition inside the cluster-box would have place P as input place.

These combinations could also be interesting to include by means of an additional interconnection mapping type. However, the connection from a place in an LPN to the edge of a cluster-box by means of any type of arc would require this arc to cross the edge of the LPN-box the place makes part of, which would probably reduce readability rather than increase it. The added value of the connection from a place in an IPN to the edge of a cluster-box is limited since it is similar to connections from this place to the edges of each LPN within

the same cluster-box. Since the number of LPNs within a cluster-box is usually limited, the reduction in number of arcs is marginal with such interconnection mapping type. Therefore, for the time being, these combinations are not included as a formal interconnection mapping type.

(3) These combinations draw an ordinary arc from a transition T in an IPN or LPN to the edge of an LPN-box or cluster-box. One could imagine a meaning to this combination, being that this arc represents s ordinary arcs $A_1, \ldots, A_s$, where s is the number of places existing in the LPN-box or cluster-box, and where for $i = 1, \ldots, s$, arc A_i is drawn from T to P_i. This would mean that T may possibly fire tokens to all these places. Due to the restriction that the number of tokens in an LPN should not be affected by interconnections, these combinations between T and the edge of a box that T does not make part of cannot be allowed. This leaves the case where an ordinary arc is drawn from a transition to the edge of the LPN that the transition makes part of. For reasons similar to those provided at item (1), the added value of defining an interconnection mapping type for this is limited. Therefore, for the time being, these combinations are not included as a formal interconnection mapping type.

(4) These combinations draw an arc A from the edge of an LPN-box to a place P inside the same LPN-box. One could imagine a meaning to this combination, being that this arc represents s arcs $A_1, \ldots, A_s$, where s is the number of transitions T_i also existing in the LPN-box, and where for $i = 1, \ldots, s$, arc A_i is drawn from T_i to P. This means that each transition in the LPN, upon firing, may produce a token for P. For reasons similar to those provided at item (1), the added value of defining an interconnection mapping type for this is limited. Therefore, for the time being, these combinations are not included as a formal interconnection mapping type.

(5) These combinations draw an ordinary arc from an LPN-box or a cluster-box to a place P in another LPN. The meaning of this combination would be that each transition in the LPN-box or cluster-box that is at the beginning of the arc, upon firing may produce a token for P. This would affect the number of tokens in this place due to interconnections, hence will not be allowed.

(6) These combinations draw an arc from the edge of an LPN-box or a cluster-box to a transition T in an IPN. One could imagine a meaning to this combination, being that this arc represents s arcs $A_1, \ldots, A_s$, where s is the number of places existing in the LPN-box or cluster-box, and where for $i = 1, \ldots, s$, T_i is duplication of T, and arc A_i is drawn from P_i to T_i.
Due to IPN rules, such arcs cannot be ordinary arcs, hence the possibility for enabling or inhibitor arcs remains. This situation could be seen as a special case of interconnection

mapping type II, except that here the transition at the end of the arc is in an IPN rather than an LPN. It would require transitions to be duplicated inside an IPN, and such IPN, or at least part of it containing the transition, would practically have become an LPN. Therefore, for the time being, this combination is not included as a formal interconnection mapping type.

(7) These combinations draw an arc from the edge of an LPN-box to itself or from the edge of a cluster-box to itself. No added value could be identified from this.

(8) These combinations draw an arc from the edge of an LPN-box to the cluster-box surrounding the LPN-box, or vice versa. No added value could be identified from this.

(9) This combination draws an ordinary arc from the edge of a cluster-box to a place that makes part of an LPN inside the same cluster-box. This is similar to item (4), except that in this case, ordinary arcs are drawn from all transitions in the cluster-box to the place. For transitions that are inside the same LPN-box as the place, this is equal to the situation of item (4), and for transitions that are outside the LPN-box (but inside the cluster), this situation cannot be allowed since the number of tokens would be changed due to interconnections. Therefore, for the time being, this combination is not included as a formal interconnection mapping type.

(10) This combination draws an ordinary arc from a cluster-box to a place in an IPN. This IPN can be located either inside the same cluster-box or outside it. One could imagine a meaning to this combination, which could be seen as an extension of interconnection mapping type IV, being that this arc represents s ordinary arcs $A_1, \ldots, A_s$, where s is the number of LPNs inside the cluster-box, and where for $i = 1, \ldots, s$, arc A_i is drawn from LPNi to P. Such interconnection mapping type would then make repeated use of type IV.
Since the number of LPNs inside a cluster-box is usually limited, the added value of this new type is limited. Therefore, for the time being, this combination is not included as a formal interconnection mapping type.

(11) This combination draws an arc from the edge of a cluster-box to a transition T that makes part of an LPN inside the same cluster-box. This could be seen as an extension of interconnection mapping type VIII, which draws an arc from an LPN-box to a transition inside the same box. The meaning of this new combination would be that if the cluster-box contains s places $P_1, \ldots, P_s$, the arc A represents s actual arcs $A_1, \ldots, A_s$ and for $i = 1, \ldots, s$, arc A_i connects P_i to duplicated transition T_i. For the case where A is an ordinary arc and where the places P_i are not in the same LPN as T_i, this would lead to the number of tokens affected by interconnections, which is not allowed. For places P_i inside the same LPN as T_i, the situation is reduced to type VIII, which is not defined for enabling or inhibitor arcs.

Therefore, for the time being, this combination is not included as a formal interconnection mapping type.

(12) A merging arc is only defined from a set of places to a transition that is in another LPN than these places. Many variations of this type could be imagined, e.g., from a set of places to a transition in the same LPN, or to a transition in an IPN, or to an LPN-box or cluster-box. In addition, merging arcs could initiate at a set of transitions instead of a set of places, at a set of boxes instead of places, or even at a mix of places and boxes.
In Section 5.3.5 it was argued that a merging arc version that starts at a set of LPN-boxes and ends on a transition is in effect similar to an enabling arc from a cluster-box holding these LPNs, to the transition, which is imt VII. Moreover, combining a merging arc with other types of imt appears to pose restrictions on the order in which the imt's are applied: imt I can be used only after all other imt's have been used to obtain an SDCPN$^{\text{imt}}$ graph, and if the SDCPN$^{\text{imt}}$ graph is transformed to an SDCPN graph, imt I needs to be applied first. This will also hold if there is a mix of places and LPNs at the beginning of the merging arc. An enabling arc cannot start at a transition, so if the merging arc starts at a set of transitions, it should become a merging arc of ordinary arcs, and it should end at a place in an IPN. The meaning of this would be that the merging arc from s transitions to a particular place would simply replace s ordinary arcs. No transitions are duplicated in this situation, hence there is no real reduction of cluttering. Therefore, for the time being, other combinations for merging arcs are not included as a formal interconnection mapping type.

From Tables 5.3 and 5.4, SDCPN$^{\text{imt}}$ rules R5 through R8 in Section 5.4.1 can be easily derived: For Rule R5, which specifies the nodes between which ordinary arcs can be drawn, look up the rows that have 'ordinary' in the first column, and gather all column headings that give 'yes'. For rule R6 (enabling arcs), rule R7 (inhibitor arcs) and rule R8 (merging arcs), a similar procedure can be followed.

Rule R9 can be derived from the second column in Tables 5.1 and 5.2. Here, one may see that for each use of interconnection mapping types, all places that are at the beginning of the arc (including the places that are inside a box that is at the beginning of the arc) are required to be of the same colour type. If this set of places contains only one element, the requirement holds true automatically and is not explicitly posed. The reason for inclusion of this restriction is that the guard or delay function of a transition that is at the end of the arc (sometimes indirectly as part of a box) uses the colours of the input tokens, hence implicitly expects these token colours to be of a particular type. If the transition is replaced by a number of duplications, then each duplication gets the same guard or delay function (if applicable), hence will expect the same type of input token colours.

Table 5.1 Summary of interconnection mapping types (imt) and their combinations. Last column motivates whether some imt's may also be defined for other types of arcs

imt	Shorthand notation	Meaning	Remarks
I	Merging arc initiating at set of places $P_1, \ldots, P_m$, ending at transition T in other LPN; for all k, ℓ: $\mathcal{C}(P_k) = \mathcal{C}(P_\ell)$	For $i = 1, \ldots, m$, $\mathcal{N}(A_i) = (P_i, T_i)$, with A_i enabling arc and T_i duplication of T	Not defined for ordinary arc (since then # tokens affected by interconnections) or inhibitor arc (the merging arc construction is mainly used to model an *or* relation, which is not practically applicable for inhibitor arcs)
II	Enabling arc from LPN-box that contains places $P_1, \ldots, P_m$ to transition T in other LPN; for all k, ℓ: $\mathcal{C}(P_k) = \mathcal{C}(P_\ell)$	For $i = 1, \ldots, m$, $\mathcal{N}(A_i) = (P_i, T_i)$, with A_i enabling arc and T_i duplication of T	Not defined for ordinary arc (since then # tokens affected by interconnections) or inhibitor arc (imt II mainly used for token colour access)
III-A	Enabling arc from place P to LPN-box that contains transitions $T_1, \ldots, T_n$	For $i = 1, \ldots, n$, $\mathcal{N}(A_i) = (P, T_i)$, with A_i enabling arc	Not defined for ordinary arc (since then # tokens affected by interconnections) but is defined for inhibitor arcs, see imt III-B
III-B	Inhibitor arc from place P to LPN-box that contains transitions $T_1, \ldots, T_n$	For $i = 1, \ldots, n$, $\mathcal{N}(A_i) = (P, T_i)$, with A_i inhibitor arc	See at imt III-A
IV	Ordinary arc from LPN-box that contains transitions $T_1, \ldots, T_n$ to place P in IPN	For $i = 1, \ldots, n$, $\mathcal{N}(A_i) = (T_i, P)$, with A_i ordinary arc	Not defined for enabling or inhibitor arcs since these arcs are not allowed from transition to place
V	Ordinary arc from place P in IPN to LPN-box that contains transitions $T_1, \ldots, T_n$	For $i = 1, \ldots, n$, $\mathcal{N}(A_i) = (P, T_i)$, with A_i ordinary arc	Not defined for enabling arc (since then the IPN will overflow and the whole reason of using IPN rather than LPN makes no sense) or inhibitor arcs (ditto)
VI-A	Enabling arc from place P in LPN to cluster-box that contains LPNs $L_1, \ldots, L_s$, where L_j contains n_j transitions	For $i = 1, \ldots, \sum_{j=1}^{s} n_j$, $\mathcal{N}(A_i) = (P, T_i)$, with A_i enabling arc and $\sum_{j=1}^{s} n_j$ the number of transitions in the cluster	Not defined for ordinary arcs, except if P is in an IPN, see imt VI-C. This imt VI-A is extension of imt III-A (so is similarly applicable for inhibitor arcs, see imt VI-B)
VI-B	Inhibitor arc from place P in LPN to cluster-box that contains LPNs $L_1, \ldots, L_s$, where L_j contains n_j transitions	For $i = 1, \ldots, \sum_{j=1}^{s} n_j$, $\mathcal{N}(A_i) = (P, T_i)$, with A_i inhibitor arc and $\sum_{j=1}^{s} n_j$ the number of transitions in the cluster	See at imt VI-A
VI-C	Ordinary arc from place P in IPN to cluster-box that contains LPNs $L_1, \ldots, L_s$, where L_j contains n_j transitions	For $i = 1, \ldots, \sum_{j=1}^{s} n_j$, $\mathcal{N}(A_i) = (P, T_i)$, with A_i ordinary arc and $\sum_{j=1}^{s} n_j$ the number of transitions in the cluster	This imt VI-C is extension of imt V (so not applicable for enabling or inhibitor arcs)

Table 5.2 Summary of interconnection mapping types and combinations, cont.

imt	Shorthand notation	Meaning	Remarks
VII	Enabling arc from cluster-box that contains LPNs $L_1, \ldots, L_s$ to transition T in LPN L, where L_j contains m_j places. If places in the cluster are $P_1, \ldots, P_{\sum_{j=1}^{s} m_j}$ then for all k, ℓ: $\mathcal{C}(P_k) = \mathcal{C}(P_\ell)$	For $i = 1, \ldots, \sum_{j=1}^{s} m_j$, $\mathcal{N}(A_i) = (P_i, T_i)$, with A_i enabling arc and T_i duplication of T	Extension of imt II, so not defined for ordinary or inhibitor arcs
VIII	Ordinary arc from LPN-box that contains places $P_1, \ldots, P_m$ to a transition inside this same LPN-box; for all k, ℓ: $\mathcal{C}(P_k) = \mathcal{C}(P_\ell)$	For $i = 1, \ldots, m$, $\mathcal{N}(A_i) = (P_i, T_i)$, with A_i ordinary arc and T_i duplication of T	Not defined for enabling arcs (since then possibly the LPN will overflow with tokens) or inhibitor arcs (since this makes no sense)
II + III-A	Enabling arc from LPN-box that contains places $P_1, \ldots, P_m$ to LPN-box that contains transitions $T_1, \ldots, T_n$; for all k, ℓ: $\mathcal{C}(P_k) = \mathcal{C}(P_\ell)$	For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, $\mathcal{N}(A_{ij}) = (P_i, T_{j_i})$, with A_{ij} enabling arc and T_{j_i} duplication of T_j	Not defined for ordinary or inhibitor arcs since these are not allowed for imt II
VI-A + II (+ III-A)	Enabling arc from LPN-box that contains places $P_1, \ldots, P_m$ to cluster-box that contains transitions $T_1, \ldots, T_n$ (organised in different LPNs); for all k, ℓ: $\mathcal{C}(P_k) = \mathcal{C}(P_\ell)$	For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, $\mathcal{N}(A_{ij}) = (P_i, T_{j_i})$, with A_{ij} enabling arc and T_{j_i} duplication of T_j	Not defined for ordinary or inhibitor arcs since these are not allowed for imt II. Note that the 'meaning' of imt VI-A uses imt III-A multiple times. Therefore, this could also be seen as a combination of imt VI-A and II and III-A
VII + III-A (+ II)	Enabling arc from cluster-box that contains places $P_1, \ldots, P_m$ (organised in different LPNs) to LPN-box that contains transitions $T_1, \ldots, T_n$; for all k, ℓ: $\mathcal{C}(P_k) = \mathcal{C}(P_\ell)$	For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, $\mathcal{N}(A_{ij}) = (P_i, T_{j_i})$, with A_{ij} enabling arc and T_{j_i} duplication of T_j	Not defined for ordinary or inhibitor arcs since these are not allowed for imt VII. Note that the 'meaning' of imt VII uses imt II multiple times. Therefore, this could also be seen as a combination of imt VII and II and III-A
VI + II + VII + III-A	Enabling arc from cluster-box that contains places $P_1, \ldots, P_m$ (organised in different LPNs) to cluster-box that contains transitions $T_1, \ldots, T_n$ (organised in different LPNs); for all k, ℓ: $\mathcal{C}(P_k) = \mathcal{C}(P_\ell)$	For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, $\mathcal{N}(A_{ij}) = (P_i, T_{j_i})$, with A_{ij} enabling arc and T_{j_i} duplication of T_j	Not defined for ordinary or inhibitor arcs since these are not allowed for imt VII or imt II

Table 5.3 Evaluation of whether different types of arcs are allowed to be drawn between different combinations of nodes

Arc	To → From ↓	Place in same LPN	Place in (other) LPN	Place in same IPN	Place in (other) IPN	Trans in same LPN	Trans in (other) LPN	Trans in same IPN	Trans in (other) IPN	Same LPN-box	(Other) LPN-box	Same Cluster-box	(Other) Cluster-box
ordinary	Place in LPN	No: *	No: *	No: * and **	No: *	Yes: basic PN rule	No: affects # tokens	No: **	No: affects # tokens	No: (1)	No: imt III not def for ordinary, see Table 5.1	No: (2)	No: imt VI not def for ordinary, see Table 5.1
enabling	Place in LPN	No: *	No: *	No: * and **	No: *	Yes: basic PN rule	Yes: LPN interconn rule	No: **	Yes: IPN rule	No: (1)	Yes: imt III-A	No: (2)	Yes: imt VI-A
inhibitor	Place in LPN	No: *	No: *	No: * and **	No: *	Yes: basic PN rule	Yes: LPN interconn rule	No: **	Yes: IPN rule	No: (1)	Yes: imt III-B	No: (2)	Yes: imt VI-B
ordinary	Place in IPN	No: * and **	No: *	No: *	No: *	No: **	Yes: IPN rule	Yes: basic PN rule	Yes: basic PN rule (IPNs merge)	No: **	Yes: imt V	No: (2)	Yes: imt VI-C
enabling	Place in IPN	No: * and **	No: *	No: *	No: *	No: **	Yes: IPN rule	Yes: basic PN rule	Yes: IPN rule	No: **	No: imt V not def for enabling, see Table 5.1	No: (2)	No: imt VI-C not def for enabling, see Table 5.1
inhibitor	Place in IPN	No: * and **	No: *	No: *	No: *	No: **	Yes: IPN rule	Yes: basic PN rule	Yes: IPN rule	No: **	No: imt V not def for inhibitor, see Table 5.1	No: (2)	No: imt VI-C not def for inhibitor, see Table 5.1
ordinary	Transition in LPN	Yes: basic PN rule	No: affects # tokens	No: **	Yes: IPN rule	No: *	No: *	No: * and **	No: *	No: (3)	No: (3)	No: (3)	No: (3)
enabling	Transition in LPN	No: *	No: *	No: * and **	No: *	No: *	No: *	No: * and **	No: *	No: *	No: *	No: *	No: *
inhibitor	Transition in LPN	No: *	No: *	No: * and **	No: *	No: *	No: *	No: * and **	No: *	No: *	No: *	No: *	No: *
ordinary	Transition in IPN	No: * and **	No: affects # tokens	Yes: basic PN rule	Yes: basic PN rule (IPNs merge)	No: * and **	No: *	No: *	No: *	No: **	No: (3)	No: (3)	No: (3)
enabling	Transition in IPN	No: * and **	No: *	No: *	No: *	No: * and **	No: *	No: *	No: *	No: * and **	No: *	No: *	No: *
inhibitor	Transition in IPN	No: * and **	No: *	No: *	No: *	No: * and **	No: *	No: *	No: *	No: * and **	No: *	No: *	No: *

Table 5.4 Evaluation of whether different types of arcs are allowed to be drawn between different combinations of nodes, cont.

Arc	To → From ↓	Place in same LPN	Place in (other) LPN	Place in same IPN	Place in (other) IPN	Trans in same LPN	Trans in (other) LPN	Trans in same IPN	Trans in (other) IPN	Same LPN-box	(Other) LPN-box	Same Cluster-box	(Other) Cluster-box
ordinary	LPN-box	No: (4)	No: (5)	No: **	Yes: imt IV	Yes: imt VIII	No: imt II not def for ordinary, see Table 5.1	No: **	No: (6)	No: (7)	No: imt II+III not def for ordinary, see Table 5.2	No: (8)	No: imt VI+II not def for ordinary, see Table 5.2
enabling	LPN-box	No: *	No: *	No: * and **	No: *	No: imt VIII not def for enabling, see Table 5.2	Yes: imt II	No: **	No: (6)	No: (7)	Yes: imt II+III-A	No: (8)	Yes: imt VI-A+II
inhibitor	LPN-box	No: *	No: *	No: * and **	No: *	No: imt VIII not def for inhibitor, see Table 5.2	No: imt II not def for inhibitor, see Table 5.1	No: **	No: (6)	No: (7)	No: imt II+III not def for inhibitor, see Table 5.2	No: (8)	No: imt VI+II not def for inhibitor, see Table 5.2
ordinary	Cluster-box	No: (9)	No: (5)	No: (10)	No: (10)	No: (11)	No: imt VII not def for ordinary, see Table 5.2	No: (6)	No: IPN rule	No: (8)	No: imt VII+III not def for ordinary, see Table 5.2	No: (7)	No: imt VI + III + VII + II not def for ordinary, see Table 5.2
enabling	Cluster-box	No: *	No: *	No: *	No: *	No: (11)	Yes: imt VII	No: (6)	No: (6)	No: (8)	Yes: imt VII+III-A	No: (7)	Yes: imt VI-A+ III+ VII+II
inhibitor	Cluster-box	No: *	No: *	No: *	No: *	No: (11)	No: imt VII not def for inhibitor, see Table 5.2	No: (6)	No: (6)	No: (8)	No: imt VII+III not def for inhibitor, see Table 5.2	No: (7)	No: imt VI+ III+ VII+II not def for inhibitor, see Table 5.2
merging	Set of places (see also (12))	No: *	No: *	No: *	No: *	No: (12)	Yes: imt I	No: (12)	No: IPN rule	No: (12)	No: (12)	No: (12)	No: (12)

Chapter 6

Analysis of DCPN and SDCPN

Since their initiation at the National Aerospace Laboratory NLR[1], DCPN, SDCPN and SDCPN^{imt} have been further developed and extended, with the versions in this thesis as result. Throughout the years, DCPN, SDCPN and SDCPN^{imt} have been used numerous times to model and analyse air transport operations, both at NLR and beyond[2]. Therefore, a lot of modelling and analysis experience has been developed. Most often, the analysis interests are related to reachability questions. For example, an SDCPN^{imt} model of an air transport operation is analysed to determine probability distributions for the distance between aircraft. Since these models tend to be large[3], and since the most interesting distances (being the small ones) tend to occur rarely, the analysis is by means of Monte Carlo simulations, which are made more efficient due to the stochastic analysis properties of PDP and GSHP.

Unfortunately, this thesis has no room to illustrate the analysis of a full-scale example SDCPN. Instead, this chapter gives some simple examples of DCPN and SDCPN, illustrates their mapping to PDP and GSHP, illustrates analysis tools for these formalisms, and illustrates the effectiveness of the SDCPN^{imt} approach.

6.1 Analysis of classical Petri net properties for SDCPN

Section 2.2.2 of this thesis explained that the analysis of classical types of Petri net, such as place/transition net (P/T net), mainly focuses on studying properties like *boundedness* (i.e., the set of reachable markings is finite), *reachability* (i.e., can a particular marking be reached from the initial marking) and *liveness* (i.e., will a particular transition or a set of transitions fire again). For P/T nets, many of these properties have been proven to be decidable, i.e., an effective algorithm

[1] The first version DCPN was published in 1997, [EBK97].

[2] Examples of applications outside NLR are at Old Dominion University (Norfolk, Virginia, USA) [HGG07], and the Universities of São Paolo (Brazil) [OHC06, OCBB07], Tokyo (Japan) [IEBB09] and Belgrade (Serbia) [NVT+10].

[3] Typical examples cover between 50 and 400 pages of Petri net graphs and details of components.

exists that determines if the P/T net has the property or not, Section 2.2.2. For many extensions of P/T net, decidability of these properties quickly disappears. The proof of this exploits the Church-Turing thesis and Rice's theorem, from which it can be shown that non-trivial properties like boundedness, reachability, and liveness are undecidable for Petri net classes that are Turing-complete. Here, Turing-completeness is shown by proving that the Petri net class considered can emulate a universal Turing machine, Section 2.2.2.

Sihombing, [Sih05] gives an algorithm that provides sufficient conditions for boundedness of a special class of DCPN, i.e., DCPN *without inhibitor arcs or enabling arcs*. Since this algorithm does not give necessary conditions for boundedness, it is not guaranteed to give a decision. In fact, it can be shown that necessary conditions will not exist, not even for this special class of DCPN. This is due to Turing-completeness of DCPN, which is proven below.

Theorem 6.1 (DCPN Turing-complete)**.** *Dynamically coloured Petri nets are Turing-complete.*

Proof. This is by construction of a DCPN^{Turing} that emulates a universal Turing machine. The constructed DCPN^{Turing} has one place, one immediate transition and two ordinary arcs, such that the place is both input and output place to the transition. The place contains a token, the colour of which contains the Turing machine's tape, the position of the tape head, and the current state. The DCPN^{Turing} transition is always enabled. The firing measure of the DCPN^{Turing} transition emulates the Turing machine's action table, i.e., it reads from the input token the symbol at the tape and the current state, and it produces an output token that has a colour according to the output of the Turing machine action table. □

Obviously, Theorem 6.1 holds true for SDCPN and SDCPN$^{\text{imt}}$ as well.

With Theorem 6.1, the Church-Turing thesis and Rice's theorem, boundedness, reachability and liveness are undecidable for DCPN. The undecidability of these properties for DCPN can also be shown indirectly, through the following theorems from literature, which prove Turing-completeness of special cases of DCPN:

- P/T nets with inhibitor arcs are Turing-complete. Proof: [Age74].
- P/T nets extended with priority firing rules are Turing-complete. Proof: [Hac76].
- GSPN (generalised stochastic Petri nets) are Turing-complete, even if the set of inhibitor arcs is empty. Proof: [Cia87].

Note that by this, undecidability of DCPN properties is proven to be due to several of the DCPN features independently, meaning that in special cases of DCPN where one or two of these features are not used (e.g., DCPN without inhibitor arcs), Turing-completeness still holds true.

It was shown in Chapter 5 that for practical applications, boundedness of SDCPN$^{\text{imt}}$ is supported by allowing one token per local Petri net, and connecting LPN in such a way that the connections do not change the number of tokens locally. This leaves posing restrictions on the number of tokens in IPNs (interaction Petri nets, see Figure 5.3). This guarantees that boundedness, in fact one of the conditions for SDCPN$^{\text{imt}}$ to be equivalent to GSHP, is satisfied. Next, stochastic analysis tools for GSHP can be used to answer further questions, see, e.g., [BBK$^+$05].

6.2 Example SDCPN and mapping to HSDE and GSHS

This section gives a simple example SDCPN model and its mapping to HSDE and to GSHS.

6.2.1 Aircraft evolution example

Assume the deviation of an aircraft from its intended path depends on the mode of operation of two of its aircraft systems: the engine system, and the navigation system. Each of these aircraft systems can be in one of two modes: *Working* (i.e., functioning properly) or *Not working* (i.e., operating in some degradation or failure mode). Both systems switch between their modes independently and with exponentially distributed sojourn times, with finite rates δ_3 (engine repaired), δ_4 (engine fails), δ_5 (navigation repaired) and δ_6 (navigation fails). The mode of operation of these systems has the following effect on the aircraft path: if both systems are *Working*, the aircraft evolves in *Nominal* mode and the position Y_t and velocity S_t of the aircraft are determined by the vector solution of $dX_t = \mathcal{V}_1(X_t)dt + \mathcal{W}_1 dW_t$, where $X_t = (Y_t, S_t)'$. If either one, or both, of the systems is *Not working*, the aircraft evolves in *Non-nominal* mode and the position and velocity of the aircraft are determined by the vector solution of $dX_t = \mathcal{V}_2(X_t)dt + \mathcal{W}_2 dW_t$. The factors $\mathcal{W}_1$ and $\mathcal{W}_2$ are determined by wind fluctuations. Initially, the aircraft has position Y_0 and velocity S_0, while both its systems are *Working*. The evaluation of this process may be stopped when the aircraft has *Landed*, i.e., its vertical position and velocity are equal to zero.

This simple aircraft evolution example illustrates the kind of difficulty encountered when modelling a realistic problem directly as an HSDE (or a GSHS). Mathematically one would define three discrete valued processes $\{\theta_t^1\}$, $\{\theta_t^2\}$, $\{\theta_t^3\}$, and an $\mathbb{R}^6$-valued process $\{X_t\}$:

- $\{\theta_t^1\}$ represents the aircraft evolution mode assuming values in {*Nominal*, *Non-nominal*, *Landed*};
- $\{\theta_t^2\}$ represents the navigation mode assuming values in {*Working*, *Not working*};
- $\{\theta_t^3\}$ represents the engine mode assuming values in {*Working*, *Not working*};
- $\{X_t\}$ represents the three-dimensional position and three-dimensional velocity of the aircraft

The process $\{\theta_t, X_t\}$, with $\theta_t = \text{Col}\{\theta_t^1, \theta_t^2, \theta_t^3\}$, is not an HSDE (or GSHS), since some θ_t combinations (such as Col{*Nominal*, *Not working*, *Not working*}) lead to immediate jumps, which is not allowed for HSDE (or GSHS) since at times of immediate jumps the HSDE (or GSHS) state would not be associated to a unique state value.

6.2.2 SDCPN model for the aircraft evolution example

This subsection models the previously described aircraft evolution example by an SDCPN. The first step is to identify Petri nets for each of the separate entities that exist in the air transport operation. These individual Petri nets are referred to as local Petri nets (LPN). The entities identified are: Aircraft evolution, Engine system, and Navigation system. This gives three local Petri nets. The resulting graphs are given in Figure 6.1.

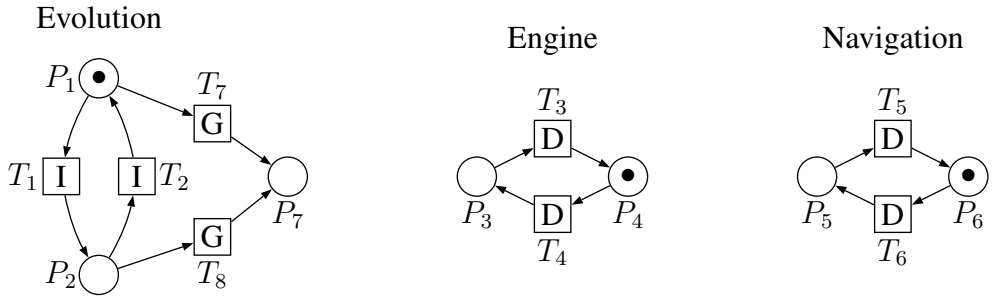


Figure 6.1 Local Petri nets for the aircraft operations example. Place P_1 models Evolution *Nominal*, P_2 models Evolution *Non-nominal*, P_3 models Engine system *Not working*, P_4 models Engine system *Working*, P_5 models Navigation system *Not working*, P_6 models Navigation system *Working*, P_7 models aircraft has *Landed*

The interactions between the Engine and Navigation local Petri net and the Evolution local Petri net are modelled by coupling the local Petri nets by additional arcs (and, if necessary, additional places or transitions). Aircraft evolution switches to Non-nominal (Transition T_1 fires) if Engine and/or Navigation are in *Not working* (if P_3 and/or P_5 have a token). Evolution switches back to Nominal if both Engine and Navigation are *Working* (both P_4 and P_6 have a token). Here, undesired removal of a token from one local Petri net by a transition of another local Petri net is prevented by using enabling arcs instead of ordinary arcs for the interactions. The resulting graph is presented in Figure 6.2. Notice that transition T_1 has to be replaced by two transitions T_{1a} and T_{1b} in order to allow both the Engine and the Navigation LPNs to influence transition T_1 separately from each other[4].

The graph in Figure 6.2 completely defines SDCPN elements $\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{N}$, where $\mathcal{T}_G = \{T_7, T_8\}$, $\mathcal{T}_D = \{T_3, T_4, T_5, T_6\}$ and $\mathcal{T}_I = \{T_{1a}, T_{1b}, T_2\}$. The other SDCPN elements are specified below.

[4]Note that this duplication can be avoided by using a merging arc from P_3 and P_4 to T_1, see SDCPN$^{\text{imt}}$.

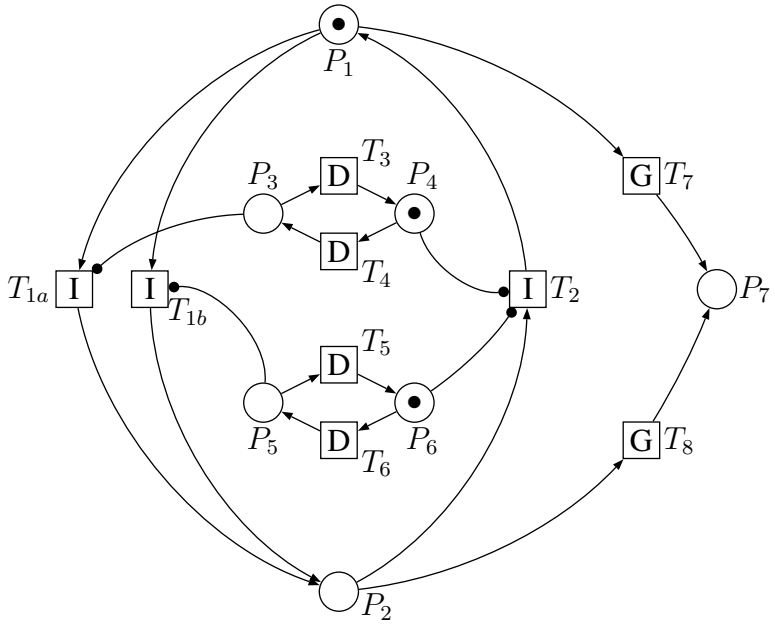


Figure 6.2 Local Petri nets integrated into one Petri net

$\mathcal{S}$: Two colour types are defined; $\mathcal{S} = \{\mathbb{R}^0, \mathbb{R}^6\}$.

$\mathcal{C}$: $\mathcal{C}(P_1) = \mathcal{C}(P_2) = \mathcal{C}(P_7) = \mathbb{R}^6$, hence $n(P_1) = n(P_2) = n(P_7) = 6$. The first three colour components of a token in P_1, P_2 or P_7 model the longitudinal, lateral and vertical position of the aircraft, the last three components model the corresponding velocities. Moreover, $\mathcal{C}(P_3) = \mathcal{C}(P_4) = \mathcal{C}(P_5) = \mathcal{C}(P_6) = \mathbb{R}^0 \triangleq \emptyset$ hence $n(P_3) = n(P_4) = n(P_5) = n(P_6) = 0$.

$\mathcal{I}$: With probability one, Place P_1 initially has a token with colour $X_0 = (Y_0, S_0)'$, with $Y_0 \in \mathbb{R}^2 \times (0, \infty)$ and $S_0 \in \mathbb{R}^3 \setminus \text{Col}\{0, 0, 0\}$. Places P_4 and P_6 initially each have a token without colour.

$\mathcal{V}, \mathcal{W}$: The token colour functions for places P_1, P_2 and P_7 are determined by $(\mathcal{V}_1, \mathcal{W}_1)$, $(\mathcal{V}_2, \mathcal{W}_2)$, and $(\mathcal{V}_7, \mathcal{W}_7)$, respectively, where $(\mathcal{V}_7, \mathcal{W}_7) = (0, 0)$. For places P_3 – P_6 there is no token colour function.

$\mathcal{G}$: Transitions T_7 and T_8 have a guard that is defined by $\mathcal{G}_{T_7} = \mathcal{G}_{T_8} = \mathbb{R}^2 \times (0, \infty) \times \mathbb{R}^2 \times (0, \infty)$.

$\mathcal{D}$: The jump rates for transitions T_3, T_4, T_5 and T_6 are $\mathcal{D}_{T_3}(\cdot) = \delta_3$, $\mathcal{D}_{T_4}(\cdot) = \delta_4$, $\mathcal{D}_{T_5}(\cdot) = \delta_5$ and $\mathcal{D}_{T_6}(\cdot) = \delta_6$, respectively.

$\mathcal{F}$: Each transition has a unique output place, to which it fires a token with a colour (if applicable) equal to the colour of the token removed, i.e., for all T, $\mathcal{F}_T(1, \cdot; \cdot) = 1$.

6.2.3 Mapping to HSDE and to GSHS

In this subsection, the SDCPN for the aircraft evolution example is mapped to an HSDE, following the construction in the proof of Theorem 4.5. Next, the mapping to GSHS is also explained.

The first step is to construct the state space $\mathbb{M}$ for the HSDE discrete process $\{\theta_t\}$. This is done by identifying the SDCPN *reachability graph*. Nodes in the reachability graph provide the number of tokens in each of the SDCPN places. Arrows connect these nodes as they represent transitions firing. The SDCPN of Figure 6.2 has seven places hence the reachability graph has elements that are vectors of length 7. Since there is always one token in the set of places $\{P_1, P_2, P_7\}$, one token in $\{P_3, P_4\}$ and one token in $\{P_5, P_6\}$, the reachability graph has $3 \times 2 \times 2 = 12$ nodes, see Figure 6.3. However, four nodes are excluded from $\mathbb{M}$: nodes $(1,0,1,0,0,1,0)$, $(0,1,0,1,0,1,0)$ and $(1,0,0,1,1,0,0)$ enable immediate transitions (i.e., are vanishing nodes), and node $(1,0,1,0,1,0,1,0)$ cannot be reached since it requires the enabling of a delay transition that is competing with an immediate transition, while due to SDCPN rule R0, an immediate transition always gets priority. Therefore, $\mathbb{M}$, i.e., the state space for $\{\theta_t\}$, consists of the remaining 8 nodes $\{V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8\}$, which are specified in Table 6.1.

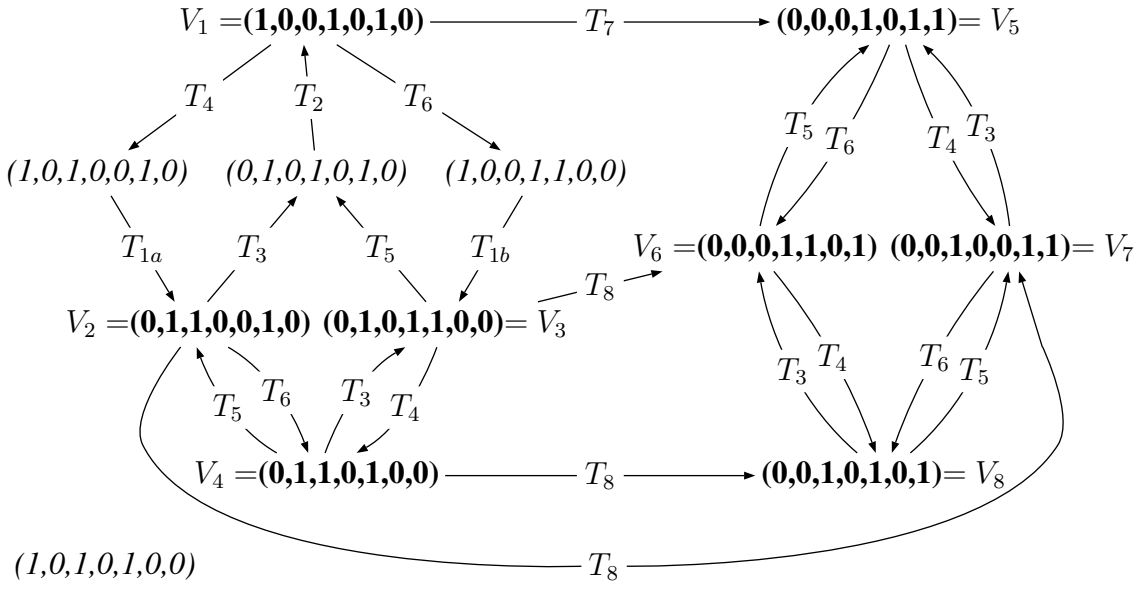


Figure 6.3 Reachability graph for the SDCPN of Figure 6.2. The nodes in bold type face correspond with the elements of $\mathbb{M}$.

Since initially there is a token in places P_1, P_4 and P_6, the initial mode θ_0 equals $\theta_0 = V_1 = (1,0,0,1,0,1,0)$. The HSDE initial continuous state value equals the vector containing the initial colours of all initial tokens. Since the initial colour of the token in Place P_1 equals X_0, and the tokens in places P_4 and P_6 have no colour, the HSDE initial continuous state value equals

Table 6.1 Discrete modes in $\mathbb{M}$

Node	Engine	Navigation	Evolution
$V_1 = (1,0,0,1,0,1,0)$	*Working*	*Working*	*Nominal*
$V_2 = (0,1,1,0,0,1,0)$	*Not working*	*Working*	*Non-nominal*
$V_3 = (0,1,1,0,1,0,0)$	*Not working*	*Not working*	*Non-nominal*
$V_4 = (0,1,0,1,1,0,0)$	*Working*	*Not working*	*Non-nominal*
$V_5 = (0,0,0,1,0,1,1)$	*Working*	*Working*	*Landed*
$V_6 = (0,0,1,0,0,1,1)$	*Not working*	*Working*	*Landed*
$V_7 = (0,0,1,0,1,0,1)$	*Not working*	*Not working*	*Landed*
$V_8 = (0,0,0,1,1,0,1)$	*Working*	*Not working*	*Landed*

$\text{Col}\{X_0, \emptyset, \emptyset\} = X_0$.

The drift coefficient $f(\theta, \cdot)$ is given by $f(\theta, \cdot) = \mathcal{V}_1(\cdot)$ for $\theta = V_1$, $f(\theta, \cdot) = \mathcal{V}_2(\cdot)$ for $\theta \in \{V_2, V_3, V_4\}$, and $f(\theta, \cdot) = 0$ otherwise. For the diffusion coefficient, $g(\theta, \cdot) = \mathcal{W}_1$ for $\theta = V_1$, $g(\theta, \cdot) = \mathcal{W}_2$ for $\theta \in \{V_2, V_3, V_4\}$, and $g(\theta, \cdot) = 0$ otherwise, see Table 6.2.

The hybrid state space is determined by the transition guards: $E = \{\theta\} \times E_\theta; \theta \in \mathbb{M}\}$, where for $\theta \in \{V_1, V_2, V_3, V_4\}$, $E_\theta = \mathbb{R}^2 \times (0, \infty) \times \mathbb{R}^2 \times (0, \infty)$ and for $\theta \in \{V_5, V_6, V_7, V_8\}$, $E_\theta = \mathbb{R}^6$.

The jump rate $\Lambda(\theta, \cdot)$ is determined from the enabling rates corresponding with the set of delay transitions in $\mathcal{T}_D$ that, under token distribution θ, are pre-enabled. At each time, always two delay transitions are pre-enabled: either T_3 or T_4 and either T_5 or T_6. Hence $\Lambda(\theta, \cdot) = \sum_{i=j,k} \mathcal{D}_{T_i}(\cdot)$ if T_j and T_k are pre-enabled. See Table 6.2 for the resulting Λ's.

Table 6.2 Example HSDE components $f(\theta, \cdot)$, $g(\theta, \cdot)$ and $\Lambda(\theta, \cdot)$ as a function of θ

θ	$f(\theta, \cdot)$	$g(\theta, \cdot)$	$\Lambda(\theta, \cdot)$
V_1	$\mathcal{V}_1(\cdot)$	$\mathcal{W}_1$	$\delta_4 + \delta_6$
V_2	$\mathcal{V}_2(\cdot)$	$\mathcal{W}_2$	$\delta_3 + \delta_6$
V_3	$\mathcal{V}_2(\cdot)$	$\mathcal{W}_2$	$\delta_3 + \delta_5$
V_4	$\mathcal{V}_2(\cdot)$	$\mathcal{W}_2$	$\delta_4 + \delta_5$
V_5	0	0	$\delta_4 + \delta_6$
V_6	0	0	$\delta_3 + \delta_6$
V_7	0	0	$\delta_3 + \delta_5$
V_8	0	0	$\delta_4 + \delta_5$

For the determination of elements ψ, ρ and μ, we first construct a probability measure P_Q, by making use of the reachability graph, the sets $\mathcal{D}$, $\mathcal{G}$ and $\mathcal{F}$ and the rules R0–R4. In Table 6.3, $P_Q(\theta', x'; \theta, x) = p$ denotes that if (θ, x) is the value of the HSDE state before the hybrid jump, then, with probability p, (θ', x') is the value of the HSDE state immediately after the jump.

Next, ψ, ρ and μ are determined as follows: Since the continuous valued process jumps to the same value with probability 1, we find $\psi(V_i, V_j, x, \underline{z}) = 0$ for all V_i, V_j, x, $\underline{z}$. Moreover, $\rho(V_i, V_j, x) = P_Q(V_i, x, V_j, x)$ and μ may be any given invertible probability measure.

Table 6.3 Example P_Q

For $x \notin \partial E_{V_1}$:	$P_Q(V_2, x; V_1, x) = \frac{\delta_4}{\delta_4+\delta_6}$,	$P_Q(V_4, x; V_1, x) = \frac{\delta_6}{\delta_4+\delta_6}$
For $x \in \partial E_{V_1}$:	$P_Q(V_5, x; V_1, x) = 1$	
For $x \notin \partial E_{V_2}$:	$P_Q(V_3, x; V_2, x) = \frac{\delta_6}{\delta_3+\delta_6}$,	$P_Q(V_1, x; V_2, x) = \frac{\delta_3}{\delta_3+\delta_6}$
For $x \in \partial E_{V_2}$:	$P_Q(V_6, x; V_2, x) = 1$	
For $x \notin \partial E_{V_3}$:	$P_Q(V_4, x; V_3, x) = \frac{\delta_3}{\delta_3+\delta_5}$,	$P_Q(V_2, x; V_3, x) = \frac{\delta_5}{\delta_3+\delta_5}$
For $x \in \partial E_{V_3}$:	$P_Q(V_7, x; V_3, x) = 1$	
For $x \notin \partial E_{V_4}$:	$P_Q(V_3, x; V_4, x) = \frac{\delta_4}{\delta_4+\delta_5}$,	$P_Q(V_1, x; V_4, x) = \frac{\delta_5}{\delta_4+\delta_5}$
For $x \in \partial E_{V_4}$:	$P_Q(V_8, x; V_4, x) = 1$	
For all x:	$P_Q(V_6, x; V_5, x) = \frac{\delta_4}{\delta_4+\delta_6}$,	$P_Q(V_8, x; V_5, x) = \frac{\delta_6}{\delta_4+\delta_6}$
For all x:	$P_Q(V_7, x; V_6, x) = \frac{\delta_6}{\delta_3+\delta_6}$,	$P_Q(V_5, x; V_6, x) = \frac{\delta_3}{\delta_3+\delta_6}$
For all x:	$P_Q(V_8, x; V_7, x) = \frac{\delta_3}{\delta_3+\delta_5}$,	$P_Q(V_6, x; V_7, x) = \frac{\delta_5}{\delta_3+\delta_5}$
For all x:	$P_Q(V_7, x; V_8, x) = \frac{\delta_4}{\delta_4+\delta_5}$,	$P_Q(V_5, x; V_8, x) = \frac{\delta_5}{\delta_4+\delta_5}$

With this, the SDCPN of the aircraft evolution example is uniquely mapped to an HSDE.

The mapping of SDCPN into GSHS is very similar to the mapping of SDCPN into HSDE. The main difference is that instead of elements ψ, ρ and μ, GSHS uses an element Q which is a transition measure for the size of jumps. We find that this Q is equal to measure P_Q constructed in Table 6.3.

Note that if, in addition, we want to make use of the HSDE properties of Proposition 4.3, e.g., the resulting HSDE solution process being pathwise unique and a semi-martingale, we need to make sure that HSDE conditions H1-H8 are satisfied. It is shown below that they are, under sufficient condition S1 for the example SDCPN.

S1 For all $r \in \mathbb{N}$ and for all $P \in \mathcal{P}$, there exist K_P^v, $L_{r,P}^v$, K_P^w and $L_{r,P}^w$ such that for all $c \in \mathcal{C}(P)$ and any a, b in the ball $\{z \in \mathcal{C}(P) \mid |z| \leq r+1\}$,

- $|\mathcal{V}_P(c)|^2 \leq K_P^v(1+|c|^2)$
- $|\mathcal{V}_P(b) - \mathcal{V}_P(a)|^2 \leq L_{r,P}^v|b-a|^2$
- $\|\mathcal{W}_P(c)\|^2 \leq K_P^w(1+|c|^2)$
- $\|\mathcal{W}_P(b) - \mathcal{W}_P(a)\|^2 \leq L_{r,P}^w|b-a|^2$.

We verify that under condition S1, HSDE conditions H1-H8 hold true in this example.

H1: From the construction of f and g above we have for $\theta = V_1$: $|f(\theta,x)|^2 + \|g(\theta,x)\|^2 = |\mathcal{V}_1(x)|^2 + \|\mathcal{W}_1(x)\|^2 \leq K_{P_1}^v(1+|x|^2) + K_{P_1}^w(1+|x|^2) = K(\theta)(1+|x|^2)$, with $K(\theta) = (K_{P_1}^v + K_{P_1}^w)$. For $\theta = V_2, V_3, V_4$ the verification is with replacing $\mathcal{V}_1$, $\mathcal{W}_1$ by $\mathcal{V}_2$, $\mathcal{W}_2$.

H2: From the construction of f and g above we have for $\theta = V_1$: $|f(\theta,x) - f(\theta,y)|^2 + \|g(\theta,x) - g(\theta,y)\|^2 = |\mathcal{V}_1(x) - \mathcal{V}_1(y)|^2 + \|\mathcal{W}_1(x) - \mathcal{W}_1(y)\|^2 \leq L^v_{r,P_1}|x-y|^2 + L^w_{r,P_1}|x-y|^2 = L_r(\theta)|x-y|^2$ with $L_r(\theta) = L^v_{r,P_1} + L^w_{r,P_1}$. For $\theta = V_2, V_3, V_4$ replace $\mathcal{V}_1$, $\mathcal{W}_1$ by $\mathcal{V}_2$, $\mathcal{W}_2$.

H3: Since $\delta_3, \ldots, \delta_6$ are constant, for all θ, $\Lambda(\theta,\cdot)$ is bounded and continuous, with upper bound $R_\Lambda = \max\{\delta_4+\delta_6, \delta_3+\delta_6, \delta_3+\delta_5, \delta_4+\delta_5\}$.

H4: Since for all θ, ϑ, $P_Q(\vartheta,\cdot;\theta,x)$ is constant, we find $\rho(\vartheta,\theta,x) = P_Q(\vartheta,x,\theta,x)$ is continuous.

H5 and H6: These are satisfied due to $\psi(V_i,V_j,x,\underline{z}) = 0$ for all V_i, V_j, x, $\underline{z}$.

H7: This condition holds due to $\delta_3, \ldots, \delta_6$ being finite and the fact that in this SDCPN example, there is no firing sequence of more than one guard transition.

H8: This condition holds for all $V_1, \ldots, V_8$, with metric $|a|^2 = \sum_i (a_i)^2$.

Thanks to this mapping we can now use HSDE stochastic analysis tools to analyse the GSHP that is defined by the execution of the SDCPN model for the example.

From a mathematical perspective, the HSDE (or GSHS) model has advantages, such as a direct support in stochastic process (or automata formal methods) theory. However, the HSDE (or GSHS) model does not show the structure of the SDCPN. Because of this, the SDCPN model of Subsection 6.2.2 is simpler to comprehend and to verify against the aircraft evolution example description of Subsection 6.2.1. These complementary advantages from both perspectives tend to increase with the complexity of the considered operation.

6.3 Example DCPN and its analysis by means of PDP stochastic analysis tools

As another illustrative example, consider two opposite streams of air traffic at the same flight level (see Figure 6.4).

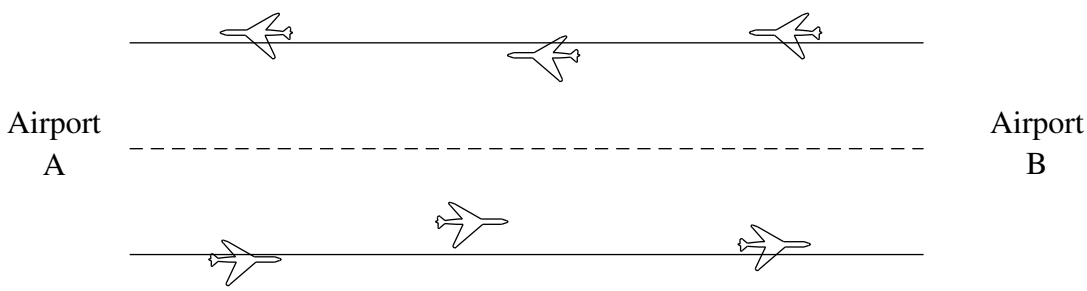


Figure 6.4 Top view of two opposite direction parallel lanes at the same flight level

The behaviour of these aircraft is modelled by DCPN, and we are interested in the behaviour of two aircraft i and j relative to each other. This behaviour is represented by process $\{\kappa_t^{ij}\}$. Due to the equivalence between DCPN and PDP, and due to the strong Markov property of PDP, the execution of a DCPN is also a strong Markov process. Therefore, we are able to capture the behaviour of these two aircraft at a stopping time τ^{ij}, and we are able to make the analysis much more efficient.

Stopping time used

Let τ^{ij} be the first moment of overlap in along-lane direction between aircraft i and aircraft j, i.e.,

$$\tau^{ij} \triangleq \min\{t_1, \inf\{t \geq t_1 - t^H; \left|y_{1,t}^{ij}\right| \leq d_1^{ij} + \Delta\}\} \tag{6.3.1}$$

with t_1 the time horizon, t^H equals one hour being the time unit of interest, $y_{1,t}^{ij}$ the distance between the centres of the aircraft in the direction of the parallel lanes, d_1^{ij} the average length of the two aircraft, and Δ a small positive value.

DCPN model

A DCPN model for this example will consist of several local Petri nets that interact, and will be developed according to the steps outlined in Chapter 5. For illustration purposes, we will give a simple example DCPN, consisting of three interacting local Petri nets (LPN) Evolution, Engine, and Display, see Figure 6.5.

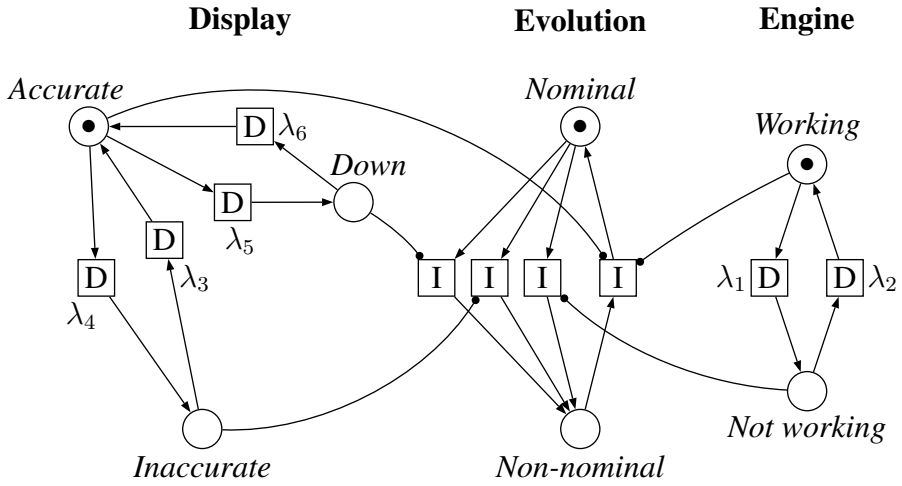


Figure 6.5 Three local Petri nets that interact. The delay transitions fire with rate λ as indicated

The evolution of an aircraft (middle LPN in Figure 6.5) can be in two modes: *Nominal* (the aircraft flies according to expectations) and *Non-nominal* (the aircraft may deviate from expectations). The aircraft position and velocity follow a differential equation which is different in

the *Nominal* and in the *Non-nominal* case. The switchings between these modes are controlled by two other LPNs, aircraft Display (on the left hand side in Figure 6.5) and aircraft Engine (on the right hand side). The Display has three modes: *Accurate*, *Inaccurate* and *Down*, between which it switches according to transition-dependent but colour-independent exponential rates as indicated in the figure by λ_3 through λ_6. The Engine has two modes *Working* and *Not working*, also with exponential switchings, with rates λ_1 and λ_2. As is modelled by the immediate transitions and the enabling arcs, in case of a token in *Down*, *Inaccurate*, or *Not working*, aircraft Evolution switches to *Non-nominal*; in case of a token in both *Accurate* and *Working*, aircraft Evolution switches back to *Nominal*.

Representation of behaviour as a process

The behaviour of the aircraft is determined by a differential equation for the position and velocity of the aircraft, which is different if it is in *Nominal* evolution mode compared to *Non-nominal* evolution mode. To capture the relative behaviour of the two aircraft, we are interested in the probability, at the selected stopping time, that both aircraft are in *Nominal* modes, the probability that both aircraft are in *Non-nominal* mode, and the probability of two mixed mode classes. However, it can be expected that if an aircraft is in *Non-nominal* mode for a long time, and switches to *Nominal* right before it meets another aircraft, that this cannot really be counted as a 'Nominal' encounter. For this reason, we assume that an encounter is considered *Nominal* if during the whole 120 second period before the aircraft meet each other, their evolution is in *Nominal* mode. The *Non-nominal* and mixed mode cases are defined analogously. This gives four classes:

- κ_{11} denoting an encounter in which both aircraft i and j have been in *Nominal* mode during the last 120 seconds before they pass each other;
- κ_{12} denoting an encounter in which aircraft i has been in *Nominal* mode during the last 120 seconds, and aircraft j has been in *Non-nominal* mode some time during the last 120 seconds;
- κ_{21} denoting an encounter in which aircraft j has been in *Nominal* mode during the last 120 seconds, and aircraft i has been in *Non-nominal* mode some time during the last 120 seconds; and
- κ_{22} denoting an encounter in which both aircraft i and j have been in *Non-nominal* mode some time during the last 120 seconds.

Evaluation of $\mathbb{P}\{\kappa^{ij}_{\tau^{ij}} = \kappa\}$, $\kappa \in \{\kappa_{11}, \kappa_{12}, \kappa_{21}, \kappa_{22}\}$

To calculate the probability of occurrence of each $\kappa \in \{\kappa_{11}, \kappa_{12}, \kappa_{21}, \kappa_{22}\}$, we make use of Markov chain analysis techniques.

Since the firings of the transitions in the Engine LPN are independent of the other LPNs and since both transitions are delay transitions with switching rates independent of colour and time, the Engine LPN is equivalent to a Markov chain with states $[\textit{Working}, \textit{Not working}]$. The same holds for the Display LPN with states $[\textit{Accurate}, \textit{Inaccurate}, \textit{Down}]$.

If the row vector $\pi(t)$ denotes the probability density of a Markov chain at time t, i.e., the ith component of $\pi(t)$ defines the probability that the Markov chain is in its ith state at time t, $\pi(t)$ satisfies the forward Kolmogorov equation

$$\frac{d}{dt}\pi(t) = \pi(t) \cdot \Upsilon(t) \tag{6.3.2}$$

with the infinitesimal generator $\Upsilon(t)$ a square matrix with components $q_{k\ell}(t)$ defined by:

- $q_{k\ell}(t)$ is the rate at which the Markov chain moves from state k to state ℓ at time t, $k \neq \ell$
- $-q_{kk}(t)$ is the rate at which the Markov chain departs from state k at time t, if it is in k at time t

Under conditions of no guard transitions and no time or colour dependent delays and firing measures, the infinitesimal generator $\Upsilon(t)$ is independent of time, $\Upsilon(t) = \Upsilon$, and Equation (6.3.2) has solution $\pi(t) = \pi(0)\exp(\Upsilon \cdot t)$, where

$$\exp(\Upsilon \cdot t) \triangleq I + \Upsilon \cdot t + \Upsilon^2 \cdot \frac{t^2}{2!} + \Upsilon^3 \cdot \frac{t^3}{3!} + \cdots \tag{6.3.3}$$

For the DCPN example considered it follows that

$$\Upsilon_{Engine} = \begin{bmatrix} -\lambda_1 & \lambda_1 \\ \lambda_2 & -\lambda_2 \end{bmatrix} \text{ and } \Upsilon_{Display} = \begin{bmatrix} -\lambda_4 - \lambda_5 & \lambda_4 & \lambda_5 \\ \lambda_3 & -\lambda_3 & 0 \\ 0 & \lambda_6 & -\lambda_6 \end{bmatrix}$$

We assume that the initial state $\pi(0)$ of these two LPNs is according to a steady state distribution, which can be found from $0 = \frac{d}{dt}\pi(t) = \pi(0) \cdot \Upsilon$, which gives

$$\pi_{Engine}(0) = \begin{bmatrix} \frac{\lambda_2}{\lambda_1+\lambda_2} & \frac{\lambda_1}{\lambda_1+\lambda_2} \end{bmatrix} \text{ and}$$

$$\pi_{Display}(0) = \begin{bmatrix} \frac{\lambda_3\lambda_6}{\lambda_4\lambda_6+\lambda_3\lambda_6+\lambda_3\lambda_5} & \frac{\lambda_4\lambda_6}{\lambda_4\lambda_6+\lambda_3\lambda_6+\lambda_3\lambda_5} & \frac{\lambda_3\lambda_5}{\lambda_4\lambda_6+\lambda_3\lambda_6+\lambda_3\lambda_5} \end{bmatrix}$$

This means that at time $t = 0$, the token in Engine is in place *Working* with probability $\frac{\lambda_2}{\lambda_1+\lambda_2}$ and is in *Not working* with probability $\frac{\lambda_1}{\lambda_1+\lambda_2}$. Analogously for Display places *Accurate*, *Inaccurate* and *Down*.

We assume $\lambda_1 = 4 \cdot 10^{-8}$, $\lambda_2 = 3 \cdot 10^{-3}$, $\lambda_3 = 2 \cdot 10^{-3}$, $\lambda_4 = 4 \cdot 10^{-7}$, $\lambda_5 = 4 \cdot 10^{-8}$, and $\lambda_6 = 3 \cdot 10^{-3}$, which gives $\pi_{Engine}(0) = [0.99999, 1.33 \cdot 10^{-5}]$ and $\pi_{Display}(0) = [0.99979, 2.00 \cdot 10^{-4}, 1.33 \cdot 10^{-5}]$.

Next, we evaluate $\mathbb{P}\{\kappa^{ij}_{\tau^{ij}} = \kappa\}$ for $\kappa \in \{\kappa_{11}, \kappa_{12}, \kappa_{21}, \kappa_{22}\}$. The only challenge is that $\pi(120)$ given by equation $\pi(t) = \pi(0)\exp(\Upsilon \cdot t)$ for $t = 120$ gives us the probability density of the Evolution modes at time $t = 120$, but this gives us no knowledge of whether Evolution has been in *Non-nominal* any time during the previous 120 seconds. A way out is to make the *Down*, *Inaccurate* and *Not working* modes 'absorbing', i.e., delete the transitions labelled by λ_2, λ_3 and λ_6, and see if after the 120 second period the Markov chain is still in a *Nominal* mode. If it is, it cannot have been in *Non-nominal* during these 120 seconds because there is no way out of there.

The infinitesimal generators for the thus modified Markov chains for LPNs Engine and Display are:

$$\Upsilon'_{Engine} = \begin{bmatrix} -\lambda_1 & \lambda_1 \\ 0 & 0 \end{bmatrix} \text{ and } \Upsilon'_{Display} = \begin{bmatrix} -\lambda_4 - \lambda_5 & \lambda_4 & \lambda_5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Solving (e.g., by means of MATLAB software) for $\pi'(120) = \pi(0)\exp(\Upsilon' \cdot 120)$, gives $\pi'_{Engine}(120) = \pi_{Engine}(0)\exp(\Upsilon'_{Engine} \cdot 120) = [0.99998, 1.81 \cdot 10^{-5}]$ and $\pi'_{Display}(120) = \pi_{Display}(0)\exp(\Upsilon'_{Display} \cdot 120) = [0.99973, 2.48 \cdot 10^{-4}, 1.81 \cdot 10^{-5}]$. Then, the probability of one aircraft staying in *Nominal* mode during the entire 120 seconds equals the product of the first components of $\pi'_{Engine}(120)$ and $\pi'_{Display}(120)$, which is 0.99972 and the probability of one aircraft having been in *Non-nominal* mode some time during these 120 seconds equals $1 - 0.99972 = 2.84 \cdot 10^{-4}$. Since these modes for one aircraft are independent from the modes of other aircraft, we get $\mathbb{P}\{\kappa^{ij}_{\tau^{ij}} = \kappa_{11}\} = (0.99972)^2 = 0.99943$, $\mathbb{P}\{\kappa^{ij}_{\tau^{ij}} = \kappa_{12}\} = 0.99972 \cdot 2.84 \cdot 10^{-4} = 2.84 \cdot 10^{-4}$, $\mathbb{P}\{\kappa^{ij}_{\tau^{ij}} = \kappa_{21}\} = 2.84 \cdot 10^{-4} \cdot 0.99972 = 2.84 \cdot 10^{-4}$, $\mathbb{P}\{\kappa^{ij}_{\tau^{ij}} = \kappa_{22}\} = (2.84 \cdot 10^{-4})^2 = 8.08 \cdot 10^{-8}$.

Note that all values used are considered to be illustrative only.

For more extended examples we refer to, e.g., [EBB07, BKB$^+$07].

6.4 Example illustrating the effectiveness of SDCPN$^{\text{imt}}$

This section aims to show the effectiveness of the interconnection mapping types developed in Chapter 5, by means of application to a particular air transport example. This example application is taken from [BKKB04], which has used the SDCPN$^{\text{imt}}$ formalism to model an operation referred to as *free flight* (also referred to as *self separation assurance*), and to subsequently determine the accident risk for this operation. Free flight is a concept where pilots are allowed to select their aircraft trajectory freely at real time, at the cost of being responsible for preventing conflicts with other aircraft. This is in contrast with conventional operations, where this responsibility is with

a dedicated ground-based air traffic control. The free flight operation can be organised in several ways, i.e., different rules and procedures can be envisioned. The operation considered in [BKKB04] is the one adopted in [Kle05].

6.4.1 LPNs of the free flight air transport example

In the free flight air transport example, the airspace is an en-route airspace without fixed routes and without an active ATC (air traffic control) giving instructions. The pilots can try to optimise their trajectory, due to the enlarged freedom to choose path and altitude. The pilots are only limited by their responsibility to maintain airborne separation, in which they are assisted by a system called ASAS (airborne separation assistance system). Each aircraft knows, through its navigation, guidance and control systems, its own state (position, velocity). It also knows the state information of other aircraft, which is sent via data-communication links. ASAS processes all this information and uses it to detect conflicts, to determine conflict resolution manoeuvres and to present an advisory to the pilots. All aircraft flying in the airspace considered are assumed to be properly equipped with ASAS.

A large number of agents is involved in the free flight operation; in the setting chosen for the initial risk assessment in [BKKB04], the following agents are taken into account:

- A pilot-flying in each aircraft, i.e., the pilot who is actually controlling the trajectory followed,
- A pilot-non-flying in each aircraft, i.e., the pilot who handles other tasks such as communication
- A number of systems and entities per aircraft, such as the aircraft's position evolution and the conflict management support systems,
- A number of global systems and entities, such as the communication frequencies and the satellite system.

It was judged sufficient to specify the following number of LPNs for the agents:

- 6 LPNs for each pilot-flying,
- 2 LPNs for each pilot-non-flying,
- 36 LPNs for the systems and entities of each aircraft,
- 7 LPNs for global systems and the environment.

The actual number of LPNs in the whole model then depends on the number N of aircraft involved, and equals $7 + N \cdot (6 + 2 + 36)$.

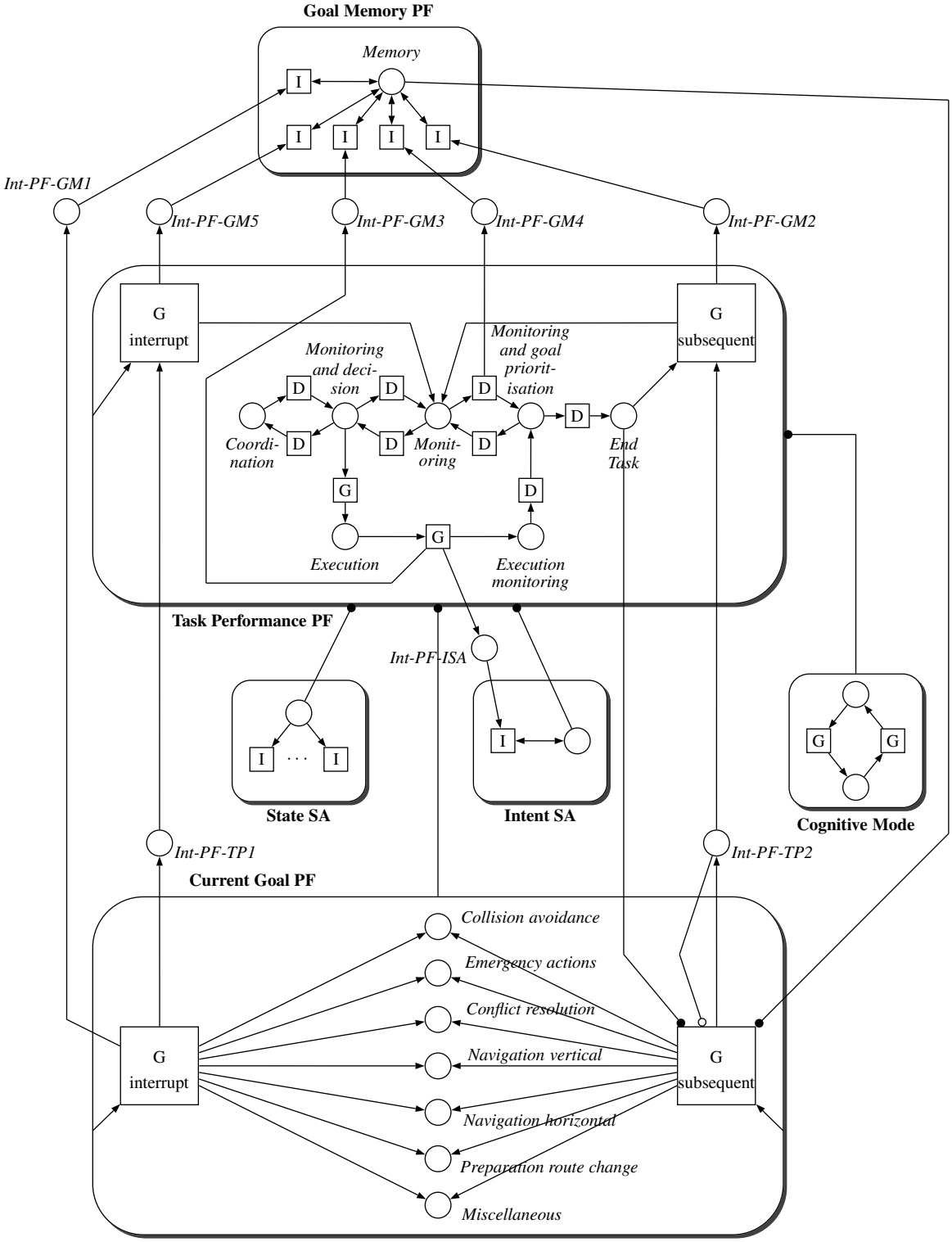


Figure 6.6 The agent pilot-flying in free flight is modelled by 6 different LPNs, interconnected by arcs and IPNs

6.4.2 Interconnected LPNs of 'pilot-flying'

This subsection illustrates, for the specific free flight air transport example, a Petri net model for the pilot-flying as agent. A graphical representation of the six LPNs modelling the pilot-flying is given in Figure 6.6, which is taken from [BKKB04]. The human-machine-interface where sound or visual clues might indicate that attention should be paid to a particular issue, is represented by an LPN that does not belong to the pilot-flying as agent and is therefore not depicted in the figure. Similarly, the arcs to or from any other agent are not shown in Figure 6.6.

To get an understanding of the six different LPNs, take as a starting point the LPN 'current goal' (at the bottom of the figure) as it represents the objective the pilot-flying is currently working on. Examples of such goals are 'collision avoidance', 'conflict resolution' and 'horizontal navigation'. For each of these goals, the pilot executes a number of tasks in a prescribed or conditional order, represented in the LPN 'task performance'. Examples of such tasks are 'monitoring and decision', 'execution' and 'execution monitoring'. If all relevant tasks for the current goal are considered executed, the pilot chooses another goal, thereby using his memory (where goals deserving attention might be stored, represented by the LPN 'goal memory') and the human-machine-interface.

Whereas the LPNs 'current goal', 'task performance', and 'goal memory' are important in the modelling of *which* task the pilot-flying is executing, the other three LPNs are important in the modelling on *how* the pilot-flying is executing the tasks. The LPN 'state SA', where SA stands for *situation awareness*, represents the relevant perception of the pilot about the states of elements in his environment, e.g., whether he is aware of an engine failure. The LPN 'intent SA' represents the intent, e.g., whether he needs to leave the free flight airspace. The LPN 'cognitive mode' represents whether the pilot is in an opportunistic mode, leading to a high but error-prone throughput, or in a tactical mode, leading to a moderate throughput with a low error probability.

6.4.3 Effectiveness of imt approach for example

From Figure 6.6, it will be clear that there are many interactions between the individual LPNs, which are depicted as enabling or inhibitor arcs and as IPNs with one place only. Interconnection mapping types I, IV, V, VI and VII have not been used, while type II has been used twice, type III four times, and type VIII three times. Table 6.4 shows that without the use of these interconnection mapping types the figure really would be cluttered with duplicated transitions and arcs within LPNs, and with connections drawn between LPNs. The use of the interconnection mapping types makes that the figure is still readable.

Another interesting observation is the vastness of the dimension of the state space for this example. The pilot-flying (PF) agent consists of 6 LPNs, three of which have 1 place, two have 7 places and one has 2 places. In addition, there are 8 IPNs of 1 place each. In [BKB$^+$07, Table 10.1]

Table 6.4 Numbers of Petri net elements before and after application of interconnection mapping types. The number of places (i.e., 19 places within LPNs and 8 places in IPNs) does not change due to the interconnection mapping types

Number of elements	In Figure 6.6	Without interconnection mapping types
Within LPNs	27 transitions	279 transitions
	66 arcs	642 arcs
Between LPNs	16 ordinary arcs	293 ordinary arcs
	7 enabling arcs	1023 enabling arcs
	1 inhibitor arc	7 inhibitor arcs
Total	117 elements	2244 elements

it has been analysed that in total this provides a discrete state space of

$$|\mathbb{M}^{PF}| = 1^{3+8} \cdot 7^2 \cdot 2^1 = 98,$$

when limited to the pilot-flying agent. Each LPN contains one token at each time instant. For ten LPNs/IPNs, the colour type of their places is $\mathbb{R}$, for two LPNs/IPNs, the colour type of their places is $\mathbb{R}^2$, for one LPN, the colour type is $\mathbb{R}^3$ and for one LPN, the colour type is $\mathbb{R}^5$. This yields that the maximum dimension of the continuous state space is

$$\sum_{i=1}^{|\mathcal{P}^{PF}|} n(P_i) = 10 \cdot 1 + 2 \cdot 2 + 1 \cdot 3 + 1 \cdot 5 = 22,$$

when limited to the pilot-flying agent. For the SDCPN of the whole free flight air transport example, [BKB$^+$07, Table 10.3] analyses that the discrete state space is

$$|\mathbb{M}| \approx 16 \cdot (0.777 \cdot 10^{12})^A,$$

where A is the number of aircraft considered. The maximum dimension of the continuous state space of the whole SDCPN is

$$\sum_{i=1}^{|\mathcal{P}|} n(P_i) = 126A + 21A^2.$$

For $A = 8$, these numbers come down to

$$|\mathbb{M}| \approx 16 \cdot (0.777 \cdot 10^{12})^8 \approx 2.13 \cdot 10^{96}$$

$$\sum_{i=1}^{|\mathcal{P}|} n(P_i) = 126 \cdot 8 + 21 \cdot 8^2 = 2352.$$

It must be clear that it will be practically impossible to model such operation directly as a PDP or an HSDE (or GSHS).

Chapter 7

Conclusions

7.1 Main results of this thesis

For a large range of complex applications, governments and industries invest in the development of innovative new operations and systems existing of many distributed components that interact in a dynamic way with many uncertainties. Before any such system can be introduced into practice, an evaluation needs to have shown that both the system and the way it is used in its new context realize the applicable objectives. If the new complex system is in its interactions similar to a previous system, such investigation can be done by analysis judgement of capable and experienced experts who judge local behaviour and implicitly assume that the interactions are working as before. If the complex system is very different from the old system, then this expert judgement approach falls short. A valuable alternative is to develop a mathematical model that incorporates the interactions, analyse this model, mobilise domain experts to evaluate where the model is representative for reality and where it needs improvement, and learn to understand how the real system works by learning how the model works. This implies a growing need for modelling and analysis formalisms. These formalisms should be able to capture the complexities of the operation, such as dynamics, multi-dimensional continuous processes, discontinuities (jumps), stochastics (randomness and uncertainties), and complex interactions, while at the same time, they should support an effective and efficient analysis.

Petri nets, see Chapter 2, have shown to be useful for developing models of various complex applications. Typical Petri net features are concurrency and synchronisation mechanism, hierarchical and modular construction, and natural expression of causal dependencies, in combination with graphical and equational representation. Numerous extensions to the basic formalism have been developed that combine different modelling features in an integrated way, including various hybrid state Petri net versions, which combine discrete and continuous system aspects. For the analysis part of the problem, Petri net properties can be studied with the objective to understand

the properties of the original system. Examples of such properties are boundedness, reachability, and liveness. For basic classes of Petri net, such as place/transition net, the problem of decidability of these properties are a main field of study. For stochastic classes of Petri net, these properties quickly become become undecidable but transformations to, e.g., continuous-time Markov chains can be used so that analysis tools for Markovian formalisms become available.

In order to combine the advantages of the Petri net modelling formalisms and those of the Markovian analysis formalism, [MT94] and [MFT00] started the development of establishing formal connections between Petri nets and stochastic processes. Their result is a hierarchy of various dependability models based on their modelling power. At the left-hand-side of this power hierarchy are Petri net models, with *generalised stochastic Petri nets* (GSPN) and *deterministic and stochastic Petri nets* (DSPN) at the top. At the right-hand-side of this power hierarchy are *continuous-time Markov chains* (CTMC) and *semi-Markov processes* at the top. Arrows between different formalisms indicate that mappings exist, i.e., that the elements of one formalism can be represented in terms of the elements of the other formalism, such that the executions, i.e., their solutions as a stochastic process, are equivalent. In this thesis, based on a series of studies [EB03, EB05, EB06, EB10a, EB10b] we developed an extension of this power hierarchy in probabilistic modelling, see Figure 7.1.

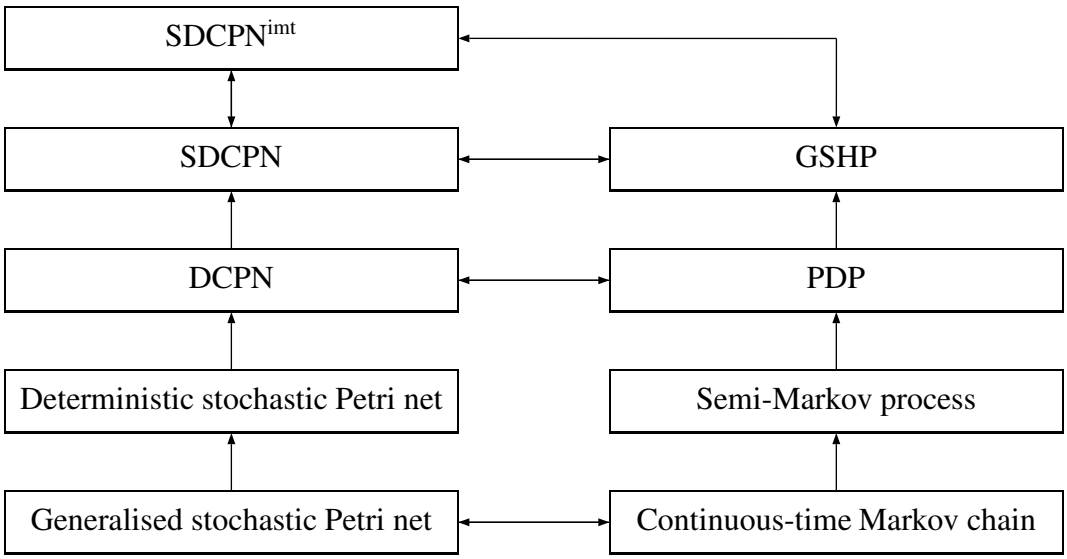


Figure 7.1 Power hierarchy among various model types. An arrow from a model to another model indicates that the second model has more modelling power than the first model.

At the left hand side of this power hierarchy, we extended DSPN to *dynamically coloured Petri nets* (DCPN) and further to *stochastically and dynamically coloured Petri nets* (SDCPN) and to *SDCPN with interconnection mapping types* (SDCPN$^{\text{imt}}$). At the right hand side of the power hierarchy we extended semi-Markov processes to *piecewise deterministic Markov processes* (PDP)

and further to *general stochastic hybrid processes* (GSHP).

DCPN (Section 3.3), SDCPN (Section 4.3) and SDCPN$^{\text{imt}}$ (Section 5.4) are hybrid Petri net classes in which the tokens have Euclidean-valued colours that change through time (*dynamically*) while the tokens reside in their place. For DCPN, these colours follow ordinary differential equations, for SDCPN and SDCPN$^{\text{imt}}$, the colours follow stochastic differential equations, hence include diffusion. The extension of SDCPN with interconnection mapping types SDCPN$^{\text{imt}}$ significantly simplifies the Petri net graphs, while maintaining the properties of SDCPN. The strength of DCPN, SDCPN and SDCPN$^{\text{imt}}$ is their compositional specification power, which makes available a hierarchical modelling approach that separates local modelling issues from global modelling issues (Section 5.2), while making effective use of Petri net features such as natural expression of causal dependencies, concurrency and synchronisation mechanism, modular construction and graphical representation.

PDP (Section 3.4) have been introduced by [Dav84, Dav93] as the most general class of continuous-time hybrid state Markov processes which include both discrete and continuous processes, except diffusion. The strength of PDP is in its unique solution and strong Markov properties, the càdlàg property (right continuous with left hand limits), and in the existence of the extended generator, [Dav93].

GSHP extend PDP by incorporation of diffusion. GSHP can be defined in several ways. One is by defining GSHP as the solution process of a *hybrid stochastic differential equation* (HSDE) (Section 4.4). A second is by defining GSHP as the execution of a *general stochastic hybrid system* (GSHS) (Section 4.8.1). The HSDE approach towards studying GSHP has been developed in a series of complementary studies [Blo03, BBEP03, Kry06, KBB07]. The GSHS approach has been developed in [BL04a, BL06]. The strength of HSDE is in its pathwise unique solution which is càdlàg and adapted and is a semi-martingale. This allows the use of Itô's differentiation rule for semi-martingales and the existence of the extended generator, e.g., [BBEP03, Ell82, EAM95]. The strength of GSHS is in its strong Markov property and the càdlàg property, and in the connection to formal methods within automata theory, e.g., [BL04b]. Main differences between HSDE and GSHS are (Section 4.8.1) that the semi-martingale property of GSHS execution is unknown, and that HSDE removes a particular restriction of GSHS which excludes *jump linear systems*.

This thesis has proven that there exist equivalence relations between the formalisms above. More specifically, it has shown:

- how DCPN can be mapped into the elements of PDP and the other way around (Theorems 3.1 and 3.2),
- how SDCPN can be mapped into HSDE and the other way around (Theorems 4.4 and 4.5)
- how SDCPN can be mapped into GSHS and the other way around (Theorems 4.6 and 4.7),

and

- how SDCPN can be extended to SDCPN^{imt} while maintaining SDCPN properties (Sections 5.3 and 5.4).

In addition, the thesis has proven that these mappings are such, that the resulting processes on both sides are *probabilistically equivalent*.

In [Van04, BLB05] such relation between elements is referred to as *bisimilarity*. Theorems 3.1 and 3.2 imply that DCPN and the PDP elements are bisimilar. Theorems 4.4 and 4.5 imply that SDCPN and HSDE are bisimilar. Theorems 4.6 and 4.7 imply that SDCPN and GSHS are bisimilar. The implications are that GSHS and HSDE are also bisimilar.

The implications of these bisimilarity relations are that the strengths of several formalisms are combined. Figure 7.2 shows the relations between the formalisms, and the key tools available for each of them.

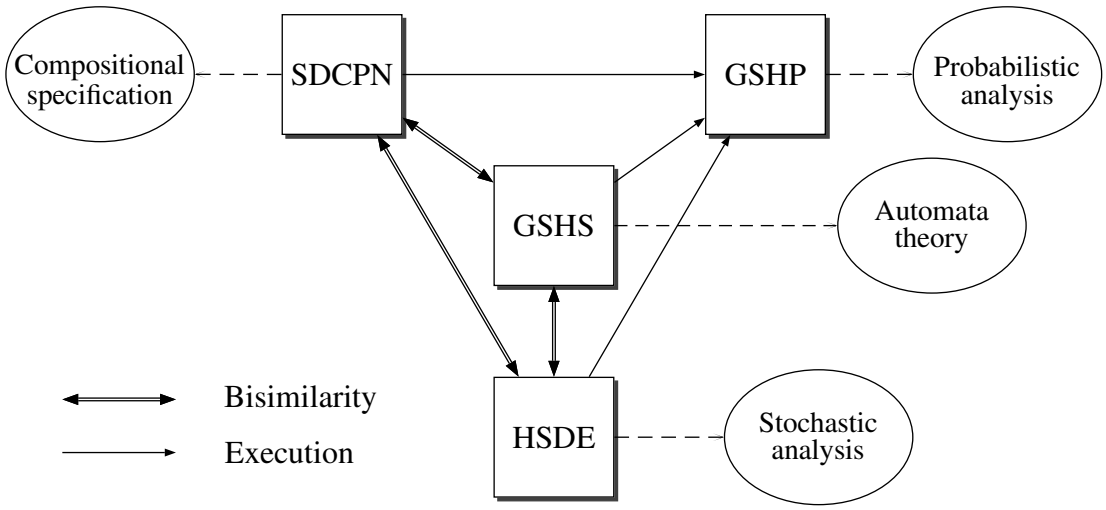


Figure 7.2 Relationship between SDCPN, GSHS, GSHP and HSDE, and their key properties and advantages. Bisimilarity relations have two-directional arrows.

The bisimilarities between SDCPN, GSHS and HSDE mean that each of them inherits the strengths of the other two formalisms. The compositional specification power of SDCPN makes it relatively easy to develop a model for a complex system with multiple interactions, due to SDCPN features such as natural expression of causal dependencies, concurrency and synchronisation mechanism, hierarchical and modular construction and graphical representation. Subsequently, in the analysis stage three alternative approaches can be taken. The first is through direct execution of SDCPN and evaluation through, e.g., Monte Carlo simulation. The second is through mapping the SDCPN into HSDE and evaluating its solution, with the advantages of stochastic analysis for semi-martingales. The third is mapping the SDCPN into a GSHS and evaluating its execution, with the advantages of connection to formal methods in automata theory and to optimal control

theory. With the GSHP resulting from any of these three means, properties become available such as convergence of discretisation, existence of limits, existence of event probabilities, strong Markov properties, and reachability analysis [Dav93, EK86, BL06].

Chapter 6 illustrated with simple examples how the strengths of these approaches work out in practice.

7.2 Further study

Since their initiation, the class of DCPN, SDCPN and SDCPN^{imt} has been used as one of the main compositional modelling formalisms within an accident risk assessment feedback methodology at the National Aerospace Laboratory NLR. Through the years, each of the steps, methods and techniques used within this methodology has been subject of further research and development, driven by demand from practical applications and the wish to improve. When restricting to SDCPN-related developments, a few issues can be identified for further study:

- The analysis power of PDP and HSDE has been exploited to make Monte Carlo simulations of the SDCPN model more efficient, see, e.g., [KB05b, BKB$^+$07, Kry06]. However, the use of formal methods in automata theory, e.g., [BL04b], to support analysis of SDCPN is still an underdeveloped area.

- For classical Petri nets, such as place/transition net, a lot of research is available on studying properties like boundedness, reachability and liveness. For many Petri net extensions, including SDCPN, these properties quickly become undecidable. However, given the equivalence relations between SDCPN and GSHP, such properties could be addressed from a stochastic hybrid processes point of view, following, e.g., [BL03, PC04, BBK07, BLL08],

- For many Petri net classes, software is available to help in the construction, verification and simulation of models, and in, e.g., the automatic generation of reachability graphs, see, e.g., [RH10]. For new classes like SDCPN, such software support is not commercially available. A few years ago, an initiative was taken at NLR to develop an 'Automatic code generator' for DCPN, [Ovi03]. A feasible tool was developed for a special case of DCPN, but further research will be necessary to develop it further, and extend it to support of SDCPN and SDCPN^{imt}.

Bibliography

[ABB+85] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. On Petri nets with stochastic timing. In *Proceedings international workshop on timed Petri nets, Torino, Italy*, pages 80–87. IEEE computer society press, July 1985. (Cited on page 24).

[ABC+91] M. Ajmone Marsan, G. Balbo, G. Chiola, G. Conte, S. Donatelli, and G. Franceschinis. An introduction to generalized stochastic Petri nets. *International journal of microelectronics and reliability*, 31(4):699–725, 1991. (Cited on page 24).

[ABD98] M. Ajmone Marsan, A. Bobbio, and S. Donatelli. Petri nets in performance analysis: an introduction. *Lectures on Petri nets I: basic models*, 1491:211–256, 1998. (Cited on page 24).

[AC87] M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In G. Rozenberg, editor, *Advances in Petri nets*, volume 266 of *Lecture notes in computer science (LNCS)*, pages 132–145. Springer-Verlag, Berlin, 1987. (Cited on page 25).

[ACB84] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM transactions on computer systems*, 2(2):93–122, May 1984. (Cited on pages 24 and 25).

[AF73] T. Agerwala and M. Flynn. Comments on capabilities, limitations and correctness of Petri nets. In G.J. Lipovski and S.A. Szygenda, editors, *Proceedings of the 1st annual symposium on computer architecture (ISCA)*, volume 1, pages 81–86. ACM press, 1973. (Cited on page 24).

[Age74] T. Agerwala. A complete model for representing the coordination of asynchronous processes. Technical report 32, Johns Hopkins university, Hopkins computer science program, research, Baltimore, Maryland, USA, July 1974. (Cited on pages 24 and 138).

[AHR00] W.M.P. Van der Aalst, K.M. Van Hee, and H.A. Reijers. Analysis of discrete-time stochastic Petri nets. *Statistica Neerlandica*, 54(2):237–255, 2000. (Cited on pages 23 and 24).

[Ajm89] M. Ajmone Marsan. Stochastic Petri nets: an elementary introduction. In *Advances in Petri nets*, volume 424 of *Lecture notes in computer science (LNCS)*, pages 1–29. Springer-Verlag, Berlin, 1989. (Cited on page 24).

[AK77] T. Araki and T. Kasami. Some decision problems related to the reachability problem for Petri nets. *Theoretical computer science*, 3:85–104, 1977. (Cited on page 20).

[AKZ98] P. Antsaklis, X. Koutsoukos, and J. Zaytoon. On hybrid control of complex systems: a survey. *European journal of automation*, 32:1023–1045, 1998. Presented at 3rd international conference ADMP, automation of mixed processes: dynamic hybrid systems, pages 1–8, Reims, France, March 1998. (Cited on page 25).

[Bag93] A. Bagchi. *Optimal control of stochastic systems*. M.J. Grimble, series editor, Series in systems and control engineering. Prentice Hall International, 1993. (Cited on page 179).

[Bal01] G. Balbo. Introduction to stochastic Petri nets. In E. Brinksma, H. Hermanns, and J.-P. Katoen, editors, *Lectures on formal methods and performance analysis (FMPA)*, volume 2090 of *Lecture notes in computer science (LNCS)*, pages 84–155. Springer-Verlag Berlin Heidelberg, 2001. (Cited on page 24).

[BB93] G.J. Bakker and H.A.P. Blom. Air traffic collision risk modeling. In *Proceedings 32nd IEEE conference on decision and control (CDC), San Antonio, Texas, USA*, volume 2, pages 1464–1469, 1993. (Cited on page 195).

[BBB+01] H.A.P. Blom, G.J. Bakker, P.J.G. Blanker, J. Daams, M.H.C. Everdij, and M.B. Klompstra. Accident risk assessment for advanced ATM. In G.L. Donohue and A.G. Zellweger, editors, *Air transportation systems engineering AIAA*, volume 193 of *Progress in astronautics and aeronautics*, pages 463–480. American institute of aeronautics and astronautics (AIAA), Reston, Virginia, USA, 2001. Presented at 2nd USA/Europe air traffic management R&D seminar, Orlando, Florida, USA, 1998. (Cited on pages 65 and 103).

[BBD+99] A. Battke, A. Borusan, J. Dehnert, H. Ehrig, C. Ermel, M. Gajewsky, K. Hoffmann, B. Hohberg, J. Juhas, S. Lembke, A. Martens, J. Padberg, W. Reisig, T. Vesper, H. Weber, and M. Weber. Initial realization of the "Petri Net Baukasten". Informatik-Bericht 129, Humboldt Universität zu Berlin, Institut für Informatik, October 1999. (Cited on page 11).

[BBEP03] H.A.P. Blom, G.J. Bakker, M.H.C. Everdij, and M.N.J. Van der Park. Stochastic analysis background of accident risk assessment for air traffic management. Hybridge report, D2.2, July 2003. http://hosted.nlr.nl/public/hosted-sites/hybridge/. (Cited on pages 3, 6, 71, 79, 80, 82, 103, 157, and 196).

[BBK+05] H.A.P. Blom, G.J. Bakker, J. Krystul, M.H.C. Everdij, B. Klein Obbink, and M.B. Klompstra. Sequential Monte Carlo simulation of collision risk in free flight air traffic. Hybridge report, D9.4, August 2005. http://hosted.nlr.nl/public/hosted-sites/hybridge/. (Cited on pages 103 and 139).

[BBK07] H.A.P. Blom, G.J. Bakker, and J. Krystul. Probabilistic reachability analysis for large scale stochastic hybrid systems. In *Proceedings 46rd IEEE conference on decision and control (CDC), New Orleans, Louisiana, USA*, December 2007. (Cited on page 159).

[BE09] H.A.P. Blom and M.H.C. Everdij. General stochastic hybrid process as pathwise unique solution of hybrid stochastic differential equation on a hybrid state space. iFly report, D7.2x, 2009. http://ifly.nlr.nl/. (Cited on page 82).

[BFF+95] E. Best, H. Fleischhack, W. Frączak, R.P. Hopkins, H. Klaudel, and E. Pelz. A class of composable high level Petri nets. In G. De Michelis and M. Diaz, editors, *Proceedings 16th international workshop on application and theory of Petri nets (ATPN), Torino, Italy*, volume 935 of *Lecture notes in computer science (LNCS)*, pages 103–120. Springer-Verlag, 1995. (Cited on page 29).

[BKB03] H.A.P. Blom, M.B. Klompstra, and G.J. Bakker. Accident risk assessment of simultaneous converging instrument approaches. *Air traffic control quarterly*, 11(2):123–155, 2003. (Cited on pages 103 and 105).

[BKB+07] H.A.P. Blom, J. Krystul, G.J. Bakker, M.B. Klompstra, and B. Klein Obbink. Free flight collision risk estimation by sequential Monte Carlo simulation. In C.G. Cassandras and J. Lygeros, editors, *Stochastic hybrid systems: recent developments and research trends*, volume 24 of *Control engineering*, chapter 10, pages 247–281. Taylor & Francis Group/ CRC Press, November 2007. (Cited on pages 149, 152, 153, and 159).

[BKKB04] G.J. Bakker, B. Klein Obbink, M.B. Klompstra, and H.A.P. Blom. DCPN specification of a free flight air traffic operation, working document. Technical report, National Aerospace Laboratory NLR, Amsterdam, The Netherlands, August 2004. (Cited on pages 34, 149, 150, and 152).

[BL03] M.L. Bujorianu and J. Lygeros. Reachability questions in piecewise deterministic Markov processes. In O. Maler and A. Pnueli, editors, *Proceedings 6th international workshop on*

hybrid systems: computation and control (HSCC), Prague, Czech Republic, volume 2623 of *Lecture notes in computer science (LNCS)*, pages 126–140. Springer, April 2003. (Cited on pages 159 and 196).

[BL04a] M.L. Bujorianu and J. Lygeros. General stochastic hybrid systems. In *Proceedings 12th IEEE Mediterranean conference on control and automation (MED), Kusadasi, Aydin, Turkey*, June 2004. (Cited on page 157).

[BL04b] M.L. Bujorianu and J. Lygeros. General stochastic hybrid systems: modelling and optimal control. In *Proceedings 43rd IEEE conference on decision and control (CDC), Nassau, Bahamas*, December 2004. (Cited on pages 157 and 159).

[BL06] M.L. Bujorianu and J. Lygeros. Toward a general theory of stochastic hybrid systems. In H.A.P. Blom and J. Lygeros, editors, *Stochastic hybrid systems: theory and safety critical applications*, volume 337 of *Lectures notes in control and information sciences (LNCIS)*, pages 3–30. Springer, 2006. (Cited on pages 3, 6, 71, 100, 101, 103, 157, and 159).

[BLB05] M.L. Bujorianu, J. Lygeros, and M.C. Bujorianu. Different approaches on bisimulation for stochastic hybrid systems. In M. Morari and L. Thiele, editors, *Proceedings 8th international workshop on hybrid systems: computation and control (HSCC), Zürich, Switzerland*, volume 3414 of *Lecture notes in computer science (LNCS)*, pages 198–214, 2005. (Cited on pages 65 and 158).

[BLGP03] M.L. Bujorianu, J. Lygeros, W. Glover, and G. Pola. A stochastic hybrid system modelling framework. Technical report, University of Cambridge and University of L'Aquila, May 2003. (Cited on pages 3 and 196).

[BLL08] M.L. Bujorianu, J. Lygeros, and R. Langerak. Reachability analysis of stochastic hybrid systems by optimal control. In M. Egerstedt and B. Mishra, editors, *Proceedings 11th international workshop on hybrid systems: computation and control (HSCC), St. Louis, Missouri, USA*, volume 4981 of *Lectures notes in computer science (LNCS)*, pages 610–613. Springer-Verlag, 2008. (Cited on page 159).

[Blo90] H.A.P. Blom. *Bayesian estimation for decision-directed stochastic control*. PhD thesis, Delft University of Technology, 1990. (Cited on page 71).

[Blo03] H.A.P. Blom. Stochastic hybrid processes with hybrid jumps. In *Proceedings IFAC conference on analysis and design of hybrid system (ADHS), Saint-Malo, Brittany, France*, pages 361–365, June 2003. (Cited on pages 6, 71, 79, 80, 82, 103, 157, and 196).

[BSC+93] G. Balbo, M. Silva, G. Chiola, J. Campos, et al. The timed (coloured) Petri net formalism: position paper. In *Proceedings workshop on formalisms, principles, and state-of-the-art, Erlangen, Germany*, volume 14, Band 26 of *Arbeitsberichte des Instituts für Mathematische Maschinen und Datenverarbeitung (Informatik)*, pages 3–60, 1993. (Cited on pages 10, 11, and 25).

[BSEP03] H.A.P. Blom, S.H. Stroeve, M.H.C. Everdij, and M.N.J. Van der Park. Human cognition performance model to evaluate safe spacing in air traffic management. In D. Harris and H.C. Muir, editors, *Human factors and aerospace safety: an international journal*, volume 3, number 1, pages 59–82. Ashgate publishing, 2003. (Cited on page 105).

[Buj05] M.L. Bujorianu. A statistical inference method for the stochastic reachability analysis. In *Proceedings 44th IEEE conference on decision and control (CDC), Seville, Spain*, December 2005. (Cited on page 66).

[Cia87] G. Ciardo. Toward a definition of modeling power for stochastic Petri net models. In *Proceedings international workshop on Petri nets and performance models, Madison, Wisconsin, USA*, volume 796, pages 54–62. IEEE computer society press, 1987. (Cited on pages 24 and 138).

[CL93] G. Ciardo and C. Lindemann. Analysis of deterministic and stochastic Petri nets. In *Proceedings 5th international workshop on Petri nets and performance models (PNPM), Toulouse, France*, pages 160–169. IEEE computer society press, October 1993. (Cited on page 25).

[CL99] C.G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer academic publishers, September 1999. (Cited on pages 10 and 11).

[CR83] J.E. Coolahan Jr. and N. Roussopoulos. Timing requirements for time-driven systems using augmented Petri nets. *IEEE transactions on software engineering*, SE-9(5):603–616, September 1983. (Cited on page 23).

[DA87] R. David and H. Alla. Continuous Petri nets. In *Proceedings 8th international conference on application and theory of Petri nets (ATPN), Zaragoza, Spain*, volume 815 of *Lecture notes on computer science (LNCS)*, pages 275–294. Springer-Verlag, 1987. (Cited on page 25).

[DA94] R. David and H. Alla. Petri nets for modeling of dynamic systems - a survey. *Automatica*, 30(2):175–202, 1994. (Cited on pages 10, 11, and 24).

[Dav84] M.H.A. Davis. Piecewise deterministic Markov processes: a general class of non-diffusion stochastic models. *Journal royal statistical society (B)*, 46(3):353–388, 1984. (Cited on pages 3, 5, 33, 157, and 195).

[Dav93] M.H.A. Davis. *Markov models and optimization*, volume 49 of *Monographs on statistics and applied probability*. Chapman and Hall, London, 1993. (Cited on pages 3, 6, 33, 36, 44, 45, 47, 48, 58, 63, 157, 159, 179, and 195).

[Dav97] R. David. Modeling of hybrid systems using continuous and hybrid Petri nets. In *Proceedings international workshop on Petri nets and performance models, St Malo, France*, pages 47–58, Los Alamitos, California, USA, June 1997. IEEE computer society. (Cited on page 26).

[DFGS07] M. Dotoli, M.P. Fanti, A. Giua, and C. Seatzu. First-order hybrid Petri nets. An application to distributed manufacturing systems. *Nonlinear analysis: hybrid systems and applications*, 2007. (Cited on page 31).

[DK96] I. Demongodin and N.T. Koussoulas. Modeling dynamic systems through Petri nets. In *Proceedings IEEE international conference on systems, man and cybernetics (SMC) – Computational engineering in systems applications (CESA), IMACS multiconference, symposium on discrete events and manufacturing systems, Lille, France*, pages 279–284, July 1996. (Cited on page 26).

[DK98] I. Demongodin and N.T. Koussoulas. Differential Petri nets: representing continuous systems in a discrete-event world. In *IEEE transactions on automatic control*, volume 43, number 4, pages 573–579. Institute of electrical and electronics engineers (IEEE), New York, New York, USA, 1998. (Cited on page 26).

[EAM95] R.J. Elliott, L. Aggoun, and J.B. Moore. *Hidden Markov models: estimation and control*, volume 29 of *Applications of mathematics: stochastic modelling and applied probability*. Springer-Verlag, 1995. (Cited on page 157).

[EB00] M.H.C. Everdij and H.A.P. Blom. Piecewise deterministic Markov processes represented by dynamically coloured Petri nets. Technical publication TP-2000-428, National Aerospace Laboratory NLR, Amsterdam, The Netherlands, 2000. (Cited on page 196).

[EB02] M.H.C. Everdij and H.A.P. Blom. How to specify a DCPN instantiation. Memorandum LL-2002-008, National Aerospace Laboratory NLR, Amsterdam, The Netherlands, 2002. (Cited on page 197).

[EB03] M.H.C. Everdij and H.A.P. Blom. Petri nets and hybrid state Markov processes in a power-hierarchy of dependability models. In *Proceedings IFAC conference on analysis and design of hybrid system (ADHS), Saint-Malo, Brittany, France*, pages 355–360, June 2003. (Cited on pages 5 and 156).

[EB05] M.H.C. Everdij and H.A.P. Blom. Piecewise deterministic Markov processes represented by dynamically coloured Petri nets. In S. Jacka, editor, *Stochastics: an international journal of probability and stochastic processes*, volume 77, number 1, pages 1–29. Taylor & Francis, February 2005. (Cited on pages 6, 38, 156, and 196).

[EB06] M.H.C. Everdij and H.A.P. Blom. Hybrid Petri nets with diffusion that have into-mappings with generalised stochastic hybrid processes. In H.A.P. Blom and J. Lygeros, editors, *Stochastic hybrid systems: theory and safety critical applications*, volume 337 of *Lectures notes in control and information sciences (LNCIS)*, pages 31–63. Springer, 2006. (Cited on pages 6, 38, 78, 102, 156, and 196).

[EB08] M.H.C. Everdij and H.A.P. Blom, editors. Safety methods database. Maintained since 2004 by National Aerospace Laboratory NLR, The Netherlands, 2008. http://www.nlr.nl/-documents/flyers/SATdb.pdf. (Cited on page 2).

[EB10a] M.H.C. Everdij and H.A.P. Blom. Bisimulation relations between automata, stochastic differential equations and Petri nets. In M. Bujorianu and M. Fisher, editors, *Proceedings workshop on Formal Methods for Aerospace (FMA)*, volume 20 of *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, pages 1–15, 2010. (Cited on pages 7, 156, and 196).

[EB10b] M.H.C. Everdij and H.A.P. Blom. Hybrid state Petri nets which have the analysis power of stochastic hybrid systems and the formal verification power of automata. In Pawel Pawlewski, editor, *Petri nets: applications*, chapter 12, pages 227–252. IN-TECH, 2010. ISBN 978-953-307-047-6, http://sciyo.com/books/show/title/petri-nets-applications. (Cited on pages 6, 7, 156, and 196).

[EBB07] M.H.C. Everdij, H.A.P. Blom, and G.J. Bakker. Modelling lateral spacing and separation for airborne separation assurance using Petri nets. *Simulation: transactions of the society for modeling and simulation international, special issue on air transportation*, 83(5):401–414, May 2007. (Cited on page 149).

[EBK97] M.H.C. Everdij, H.A.P. Blom, and M.B. Klompstra. Dynamically coloured Petri nets for air traffic management safety purposes. In M. Papageorgiou and A. Pouliezos, editors,

Proceedings 8th IFAC symposium on transportation systems, Chania, Greece, pages 184–189, June 1997. (Cited on pages 137 and 196).

[EK86] S.N. Ethier and T.G. Kurtz. *Markov processes, characterization and convergence*. Wiley series in probability and mathematical statistics. John Wiley & Sons, New York, 1986. (Cited on pages 159 and 179).

[EKB96] M.H.C. Everdij, M.B. Klompstra, and H.A.P. Blom. Development of mathematical techniques for ATM safety analysis. MUFTIS work package report 3.2, Final report on safety model, Part 2 NLR-TR-96197, National Aerospace Laboratory NLR, Amsterdam, The Netherlands, 1996. (Cited on page 195).

[EKBF96] M.H.C. Everdij, M.B. Klompstra, H.A.P. Blom, and O.N. Fota. Evaluation of hazard analysis techniques for application to en route ATM. MUFTIS work package report 3.2, Final report on safety model, Part 1 NLR-TR-96196, National Aerospace Laboratory NLR, Amsterdam, The Netherlands, 1996. (Cited on page 195).

[EKBK06] M.H.C. Everdij, M.B. Klompstra, H.A.P. Blom, and B. Klein Obbink. Compositional specification of a multi-agent system by stochastically and dynamically coloured Petri nets. In H.A.P. Blom and J. Lygeros, editors, *Stochastic hybrid systems: theory and safety critical applications*, volume 337 of *Lectures notes in control and information sciences (LNCIS)*, pages 325–350. Springer, 2006. (Cited on pages 6, 117, and 197).

[Ell82] R.J. Elliott. *Stochastic calculus and applications*, volume 18 of *Applications of mathematics: Stochastic modelling and applied probability*. Springer-Verlag, 1982. (Cited on page 157).

[EN94] J. Esparza and M. Nielsen. Decidability issues for Petri nets. In *Petri net newsletter*, volume 47, pages 5–23. Gesellschaft für Informatik (GI), special interest group on Petri nets and related system models, Bonn, Germany, 1994. (Cited on pages 18 and 19).

[Esp98] J. Esparza. Decidability and complexity of Petri net problems - an introduction. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri nets I: basic models*, volume 1491 of *Lecture notes in computer science (LNCS)*, pages 374–428. Springer-Verlag, 1998. (Cited on pages 18 and 19).

[FAP97] M. Fernandes, M. Adamski, and A.J. Proença. VHDL generation from hierarchical Petri net specifications of parallel controller. *IEE proceedings: computers and digital techniques*, 144:127–137, March 1997. (Cited on pages 29, 106, 109, and 110).

[FG97] H. Fleischhack and B. Grahlmann. A Petri net semantics for B(PN)^2 with procedures. In *Proceedings 2nd workshop on parallel and distributed software engineering (PDSE), Boston, Massachusetts, USA*, IEEE computer society, pages 15–27, May 1997. (Cited on page 29).

[FKK97] N.O. Fota, M. Kaâniche, and K. Kanoun. A modular and incremental approach for building complex stochastic Petri net models. In *Proceedings 1st international conference on mathematical methods in reliability (MMR), Bucharest, Romania*, pages 151–158, September 1997. Rapport LAAS No97224. (Cited on pages 30, 106, 109, and 110).

[FKK98] N.O. Fota, M. Kaâniche, and K. Kanoun. Dependability evaluation of an air traffic control computing system. In *Proceedings 3rd IEEE international computer performance and dependability symposium (IPDS), Durham, North Carolina, USA*, pages 206–215, 1998. (Cited on page 32).

[FM94] M. Felder and A. Morzenti. A temporal logic approach to implementation and refinement of timed Petri nets. In D. Gabbay, editor, *Proceedings 1st international conference on temporal logic (ICTL), Bonn, Germany*, pages 365–381. Springer-Verlag, New York, July 1994. (Cited on page 24).

[FN84] G. Florin and S. Natkin. Définition formelle des réseaux de Petri stochastiques. Research report, Conservatoire national des arts et métiers (CNAM), Paris, France, 1984. (Cited on page 24).

[Frö04] S. Fröschle. *Decidability and coincidence of equivalences for concurrency*. PhD thesis, Laboratory for foundations of computer science, School of informatics, University of Edinburgh, UK, 2002, Graduation date: May 2004. (Cited on page 21).

[GAM91] M.K. Ghosh, A. Arapostathis, and S.I. Marcus. An optimal control problem arising in flexible manufacturing systems. In *Proceedings 30th IEEE conference on decision and control (CDC), Brighton, UK*, pages 1884–1849, 1991. (Cited on page 3).

[Gen86] H.J. Genrich. Predicate/transition 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, Germany*, volume 254 of *Lecture notes in computer science (LNCS)*, pages 207–247. Springer-Verlag, 1986. (Cited on page 22).

[Giu06] A. Giua. Bibliography on hybrid Petri nets, 2006. http://bode.diee.unica.it/ ∼hpn/. (Cited on page 25).

[GL81] H.J. Genrich and K. Lautenbach. System modelling with high-level Petri nets. *Theoretical computer science, special issue on semantics of concurrent computation*, 13:109–136, 1981. (Cited on page 22).

[GSB99] M. Gribaudo, M. Sereno, and A. Bobbio. Fluid stochastic Petri nets: an extended formalism to include non-markovian models. In *Proceedings 8th international workshop on Petri nets and performance models (PNPM99), Zaragoza, Spain*, pages 74–81, October 1999. (Cited on page 26).

[GU96] A. Giua and E. Usai. High-level hybrid Petri nets: a definition. In *Proceedings 35th IEEE conference on decision and control (CDC), Kobe, Japan*, pages 148–150, 1996. (Cited on page 27).

[GU98] A. Giua and E. Usai. Modeling hybrid systems by high-level Petri nets. *European journal of automation APII-JESA*, 32(9–10):1209–1231, December 1998. (Cited on page 27).

[GV03] C. Girault and R. Valk, editors. *Petri nets for systems engineering. A guide to modeling, verification, and applications*. Springer-Verlag, Berlin Heidelberg New York, 2003. (Cited on page 10).

[Haa02] P.J. Haas. *Stochastic Petri nets, modelling, stability, simulation*. Springer-Verlag, New York, 2002. (Cited on pages 23, 24, and 25).

[Hac75] M. Hack. *Decidability questions for Petri nets*. PhD thesis, Massachusetts institute of technology (MIT), department of electrical engineering, Cambridge, Massachusetts, USA, December 1975. (Cited on pages 19 and 20).

[Hac76] M. Hack. Petri net languages. Technical report MIT-LCS-TR-159, Massachusetts institute of technology (MIT), laboratory for computer science, Cambridge, Massachusetts, USA, March 1976. (Cited on page 138).

[Han93] H. Hanisch. Analysis of place/transition nets with timed arcs and its application to batch process control. In M. Ajmone Marsan, editor, *Proceedings 14th international conference on application and theory of Petri nets (ATPN), Chicago, Illinois, USA*, volume 691 of *Lecture notes in computer science (LNCS)*, pages 282–299. Springer-Verlag, 1993. (Cited on page 24).

[Har87] D. Harel. Statecharts: a visual formalism for complex systems. *Science of computer programming*, 8:231–274, 1987. (Cited on pages 106 and 111).

[HGG07] H. Herencia-Zapana, O.R. González, and W.S. Gray. Dynamically colored Petri net representation of nonlinear sampled-data systems with embedded recovery algorithms. In *Proceedings 46th IEEE conference on decision and control (CDC), New Orleans, Louisiana, USA*, pages 97–102, 2007. (Cited on page 137).

[HJS90] P. Huber, K. Jensen, and R.M. Shapiro. Hierarchies in coloured Petri nets. In G. Rozenberg, editor, *Advances in Petri nets*, volume 483 of *Lecture notes in computer science (LNCS)*, pages 313–341. Springer-Verlag, Berlin Heidelberg New York, 1990. Also Chapter 7 in: K. Jensen and G. Rozenberg, editors, *High-level Petri nets; theory and application*, Springer-Verlag, 1991. (Cited on pages 28, 106, and 110).

[HLS00] J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In N. Lynch and B.H. Krogh, editors, *Proceedings 3rd international workshop on hybrid systems: computation and control (HSCC), Pittsburgh, Pennsylvania, USA*, volume 1790 of *Lecture notes in computer science (LNCS)*, pages 160–173. Springer Verlag, April 2000. (Cited on page 3).

[HYB05] HYBRIDGE project website. Maintained by National Aerospace Laboratory NLR, Amsterdam, The Netherlands, 2005. http://hosted.nlr.nl/public/hosted-sites/hybridge/. (Cited on page 196).

[IEBB09] E. Itoh, M.H.C. Everdij, G.J. Bakker, and H.A.P. Blom. Speed control for airborne separation assistance in continuous descent arrivals. In *Proceedings 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO), Hilton Head, South Carolina, USA*, September 2009. (Cited on page 137).

[ITWM08] Netherlands Inspection Transport and Water Management. Civil aviation safety data 1993–2007. Den Haag, The Netherlands, 2008. http://www.verkeerenwaterstaat.nl/Images/200810073%20bijlage.doc_tcm195-227432.pdf. (Cited on page 1).

[JE02] J.W. Janneck and R. Esser. Higher-order Petri net modelling - techniques and applications. In *Proceedings 23rd international conference on application and theory of Petri nets (ATPN) – workshop on software engineering and formal methods, Adelaide, Australia*, volume 2360 of *Lecture notes in computer science (LNCS)*. Springer-Verlag, 2002. (Cited on page 29).

[Jen81] K. Jensen. Coloured Petri nets and the invariant method. *Theoretical computer science*, 14:317–336, 1981. (Cited on page 22).

[Jen90] K. Jensen. Coloured Petri nets: a high-level language for system design and analysis. In G. Rozenberg, editor, *Advances in Petri nets*, volume 483 of *Lecture notes in computer*

science (LNCS), pages 342–416. Springer-Verlag, Berlin Heidelberg New York, 1990. (Cited on pages 22 and 28).

[Jen92] K. Jensen. *Coloured Petri nets: basic concepts, analysis methods and practical use*, volume 1. Springer-Verlag, 1992. (Cited on page 36).

[KB05a] J. Krystul and H.A.P. Blom. Generalised stochastic hybrid processes as strong solutions of stochastic differential equations. Technical report, University of Twente and National Aerospace Laboratory NLR, June 2005. (Cited on pages 3 and 82).

[KB05b] J. Krystul and H.A.P. Blom. Sequential Monte Carlo simulation of rare event probability in stochastic hybrid systems. In P. Horacek, M. Simandl, and P. Zitek, editors, *Proceedings 16th IFAC world congress, 4-8 July 2005, Prague, Czech Republic*. Elsevier, Oxford, 2005. (Cited on page 159).

[KBB07] J. Krystul, H.A.P. Blom, and A. Bagchi. Stochastic differential equations on hybrid state spaces. In C.G. Cassandras and J. Lygeros, editors, *Stochastic hybrid systems*, number 24 in Control engineering series, pages 15–45. Taylor and Francis/CRC Press, 2007. (Cited on pages 71 and 157).

[KD99] D. Karabeg and R. Djurhuus. Algoritmer og effektivitet. Kompendium nr. 51 til IN210, Universitetet i Oslo, Institutt for informatikk, August 1999. (Cited on pages 17 and 18).

[Kel75] R.M. Keller. A fundamental theorem of asynchronous parallel computation. In T.Y. Feng, editor, *Parallel processing*, volume 24 of *Lecture notes in computer science (LNCS)*, pages 102–112. Springer-Verlag, 1975. (Cited on page 19).

[Kin97] E. Kindler. A compositional partial order semantics for Petri net components. In P. Azema and G. Balbo, editors, *Proceedings 18th international conference on application and theory of Petri nets (ATPN), Toulouse, France*, volume 1248 of *Lecture notes in computer science (LNCS)*. Springer-Verlag, June 1997. (Cited on pages 29, 106, 109, and 110).

[KL00] S. Kurkovsky and R. Loganantharaj. Modeling of, and reasoning with recurrent events with imprecise durations. In *Proceedings 13th international conference on the industrial and engineering application of artificial intelligence and expert systems (IEA/AIE), New Orleans, Louisiana, USA*, pages 272–283, 2000. (Cited on page 23).

[Kle05] B. Klein Obbink. Description of advanced operation: free flight. Hybridge report, D9.1, March 2005. http://hosted.nlr.nl/public/hosted-sites/hybridge/. (Cited on page 150).

[KM69] R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of computer and system sciences*, 3(2):147–195, 1969. (Cited on page 18).

[KO00] K. Kanoun and M. Ortalo-Borrel. Fault-tolerant system dependability-explicit modeling of hardware and software component-interactions. *IEEE transactions on reliability*, 49(4):363–376, December 2000. (Cited on page 32).

[Koo05] H.-Y.B. Koo. *A meta-language for systems architecting*. PhD thesis, Massachusetts institute of technology (MIT), engineering systems division, Cambridge, Massachusetts, USA, January 2005. (Cited on page 18).

[Kos82] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Proceedings 14th ACM symposium on theory of computing (STOC), San Francisco, California, USA*, pages 267–281. ACM press, 1982. (Cited on page 19).

[Kry06] J. Krystul. *Modelling of stochastic hybrid systems with applications to accident risk assessment*. PhD thesis, University of Twente, The Netherlands, September 2006. (Cited on pages 3, 71, 82, 157, and 159).

[LAD91] J. Le Bail, H. Alla, and R. David. Hybrid Petri nets. In *Proceedings 1st European control conference (ECC), Grenoble, France*, pages 1472–1477, 1991. (Cited on page 26).

[Lip76] R.J. Lipton. The reachability problem requires exponential space. Research report 62, Yale university, department of computer science, New Haven, Connecticut, USA, 1976. (Cited on page 19).

[LJSE99] J. Lygeros, K.H. Johansson, S. Sastry, and M. Egerstedt. On the existence of executions of hybrid automata. In *Proceedings 38th IEEE conference on decision and control (CDC), Phoenix, Arizona, USA*, pages 2249–2254, 1999. (Cited on page 35).

[LM76] J.P. Lepeltier and B. Marchal. Problème des martingales et équations différentielles stochastiques associées à un opérateur intégro-différentiel. *Annales de l'Institute Henri Poincaré*, Section B - XII(1):43–103, 1976. (Cited on page 82).

[LT05] C. Lesire and C. Tessier. Particle Petri nets for aircraft procedure monitoring under uncertainty. In G. Ciardo and P. Darondeau, editors, *Proceedings 26th international conference on application and theory of Petri nets (ATPN), Miami, Florida, USA*, volume 3536 of *Lecture notes in computer science (LNCS)*, pages 329–348. Springer-Verlag, June 2005. (Cited on pages 27 and 32).

[Mad02] N. Madras. *Lectures on Monte Carlo methods*, volume 16 of *Fields institute monographs, Fields institute for research in mathematical sciences*. American Mathematical Society, 2002. (Cited on page 36).

[May81] E.W. Mayr. An algorithm for the general Petri net reachability problem (preliminary version). In *Proceedings 13th annual ACM symposium on theory of computing, Milwaukee, Wisconsin, USA*, pages 238–246. Association for computing machinery, May 1981. (Cited on page 19).

[May84] E.W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM journal of computing*, 13(3):441–460, August 1984. (Cited on page 19).

[MF76] P.M. Merlin and D.J. Farber. Recoverability of communication protocols: implications of a theoretical study. *IEEE transactions on communications*, 24(9):1036–1043, September 1976. (Cited on page 23).

[MFT00] J.K. Muppala, R.M. Fricks, and K.S. Trivedi. Techniques for system dependability evaluation. In W. Grasman, editor, *Computational probability*, pages 445–480. Kluwer academic publishers, The Netherlands, 2000. (Cited on pages 4, 5, 65, 103, and 156).

[MM81] E.W. Mayr and A.R. Meyer. The complexity of the finite containment problem for Petri nets. In *Journal of the ACM*, volume 28, number 3, pages 561–576. ACM press, 1981. (Cited on page 19).

[Mol81] M.K. Molloy. *On the integration of delay and throughput measures in distributed processing models*. PhD thesis, University of California, Los Angeles, California, USA, 1981. (Cited on pages 23 and 24).

[MT94] M. Malhotra and K.S. Trivedi. Power-hierarchy of dependability-model types. *IEEE transactions on reliability*, R-43(3):493–502, 1994. (Cited on pages 4, 5, 65, 103, and 156).

[Mur89] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989. (Cited on pages 10, 11, 18, 19, and 20).

[Nat80] S. Natkin. *Les réseaux de Petri stochastiques*. Thèse de docteur ingénieur, CNAM, Paris, France, June 1980. (Cited on page 23).

[NH04] E. Németh and K.M. Hangos. Multi-scale process model description by generalized hierarchical CPN models. Research report SCL-002/2004, Process control research group, MTA SZTAKI computer and automation research institute, Hungarian academy of sciences, Hungary, 2004. (Cited on page 28).

[NN73] J.D. Noe and G.J. Nutt. Macro E-nets representation of parallel systems. *IEEE transactions on computers*, 31(9):718–727, August 1973. (Cited on page 23).

[NVT+10] F. Netjasov, A. Vidosavljevic, V. Tosic, M.H.C. Everdij, and H.A.P. Blom. Stochastically and dynamically coloured Petri net model of ACAS operations. In D. Zellweger, V. Duong, and D. Lovell, editors, *Proceedings 4th international conference on research in air transportation (ICRAT), Budapest, Hungary*, June 2010. (Cited on page 137).

[Obe06] H. Oberheid. A coloured Petri net model of cooperative arrival planning in air traffic control. In K. Jensen, editor, *Proceedings of the 7th workshop and tutorial on practical use of coloured Petri nets and the CPN tools, department of computer science, University of Aarhus, Denmark*, volume PB-579, pages 177–196, 2006. (Cited on page 32).

[OCBB07] Í.R. de Oliveira, P.S. Cugnasca, H.A.P. Blom, and G.J. Bakker. Modelling and estimation of separation criteria for airborne time-based spacing operation. In *Proceedings 7th air traffic management (ATM) seminar, Barcelona, Spain*, 2007. paper 137. (Cited on page 137).

[OHC06] Í.R. de Oliveira, R.M. Honda, and P.S. Cugnasca. Risk analysis of airborne spacing in approach sequencing. *Journal of the Brazilian air transportation research society*, 2(2):63–81, 2006. (Cited on page 137).

[Øk02] B. Øksendal. *Stochastic differential equations. An introduction with applications*. Springer-Verlag, fifth edition, corrected printing edition, 2002. (Cited on pages 48 and 62).

[OÜRS07] H.O. Oberheid, M. Übbing-Rumke, and D. Söffker. Cooperative arrival management in air traffic control - a coloured Petri net model of sequence planning. *Software tools for technology transfer*, 2007. (Cited on page 32).

[Ovi03] B.A. Oving. VALILEO 2002, volume 3: GNSS dependability tool Part 1: Requirements identification on DCPN-based modelling for dependability assessment. Technical Report NLR-CR-2002-456-VOL-3-PT-1, National Aerospace Laboratory NLR, January 2003. (Cited on page 159).

[Pad99] J. Padberg. The "Petri Net Baukasten": an application-oriented Petri net technology. In H. Weber, H. Ehrig, and W. Reisig, editors, *Proceedings international colloquium on Petri net technologies for modelling communication based systems*. Fraunhofer Gesellschaft ISST, October 1999. (Cited on page 11).

[PC04] M. Prandini and M.C. Campi. Reachability analysis for probabilistic hybrid systems with application to air traffic management. Hybridge report, D3.1, November 2004. http://hosted.nlr.nl/public/hosted-sites/hybridge/. (Cited on page 159).

[Pet62] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, Germany, 1962. Schriften des IIM, number 2. (Cited on page 9).

[Pet66] C.A. Petri. Communication with automata. Technical report volume 1, number RADC-TR-65-377, New York: Griffiss air force base, 1966. English translation. (Cited on page 9).

[Rac78] C. Rackoff. The covering and boundedness problem for vector addition systems. *Theoretical computer science*, 6:223–231, 1978. (Cited on page 19).

[Ram73] C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets*. PhD thesis, Massachusetts institute of technology (MIT), Cambridge, Massachusetts, USA, 1973. (Cited on page 23).

[RH80] C.V. Ramamoorthy and G.S. Ho. Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE transactions on software engineering*, 6(5):440–449, September 1980. (Cited on page 23).

[RH10] H. Rölke and F. Heitmann, editors. Petri nets world, website, maintained by the TGI (Theoretische Grundlagen der Informatik) group at the University of Hamburg, Germany, 2010. http://www.informatik.uni-hamburg.de/TGI/PetriNets/. (Cited on pages 4, 9, 10, and 159).

[Ric53] H.G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions American mathematics society*, 74:358–366, 1953. (Cited on page 18).

[She05] R.J. Sheehan. *The ICICLE (interaction computing in a constructionist learning environment) programming environment for children*. PhD thesis, University of Auckland, New Zealand, 2005. (Cited on page 18).

[Sif77] J. Sifakis. Use of Petri nets for performance evaluation. In H. Beilner and E. Gelenbe, editors, *Proceedings 3rd international symposium IFIP, W.G. 7.3, Measuring, modelling and evaluating computing systems, Bonn-Bad Godesberg, Germany*, pages 75–93. Elsevier science, Amsterdam, 1977. (Cited on page 23).

[Sih05] S.K. Sihombing. Dynamically coloured Petri nets: properties and their decidability. Master's thesis, University of Twente, faculty of engineering mathematics, Enschede, The Netherlands, July 2005. (Cited on page 138).

[SR02] M. Silva and L. Recalde. Petri nets and integrality relaxations: a view of continuous Petri nets. *IEEE transactions on systems, man, and cybernetics, part C*, 32(4):314–327, 2002. (Cited on page 26).

[Str05] S.N. Strubbe. *Compositional modelling of stochastic hybrid systems*. PhD thesis, University of Twente, The Netherlands, December 2005. (Cited on page 66).

[Sud97] T.A. Sudkamp. *Languages and machines: an introduction to the theory of computers*. Addison-Wesley Longman publishing Co., Inc., Boston, Massachusetts, USA, 2nd edition, 1997. (Cited on page 16).

[SV05a] S.N. Strubbe and A.J. Van der Schaft. Bisimulation for communicating piecewise deterministic Markov processes (CPDPs). In M. Morari and L. Thiele, editors, *Proceedings 8th international workshop on hybrid systems: computation and control (HSCC), Zürich, Switzerland*, volume 3414 of *Lecture notes in computer science (LNCS)*, pages 623–639, 2005. (Cited on page 66).

[SV05b] S.N. Strubbe and A.J. Van der Schaft. Stochastic semantics for communicating piecewise deterministic Markov processes. In *Proceedings 44th IEEE conference on decision and control, and the European control conference, Seville, Spain*, pages 6103–6108, December 2005. (Cited on page 66).

[TK93] K.S. Trivedi and V.G. Kulkarni. FSPNs: fluid stochastic Petri nets. In M. Ajmone Marsan, editor, *Proceedings 14th international conference on application and theory of Petri nets (ATPN), Chicago, Illinois, USA*, volume 691 of *Lecture notes in computer science (LNCS)*, pages 24–31. Springer-Verlag, Heidelberg, 1993. (Cited on page 26).

[TTV06] G.J. Tsinarakis, N.C. Tsourveloudis, and K.P. Valavanis. Modeling, analysis, synthesis, and performance evaluation of multioperational production systems with hybrid timed Petri nets. *IEEE transactions on automation science and engineering*, 3(1):29–46, January 2006. (Cited on pages 26 and 106).

[Tur36] A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. In *Proceedings London mathematical society*, pages 230–265, 1936. (Cited on page 17).

[Van04] A.J. Van der Schaft. Equivalence of dynamical systems by bisimulation. *IEEE transactions on automatic control*, 49(12):2160–2172, 2004. (Cited on pages 65 and 158).

[VJMV04] E. Villani, F. Junqueira, P.E. Miyagi, and R. Valette. Petri net and OO for the modular analysis of an aircraft landing system. In *Proceedings 17th international congress of mechanical engineering, São Paolo, Brazil*, ABCM symposium series in mechatronics, pages 570–579, 2004. (Cited on pages 28 and 32).

[VN92] N. Viswanadham and Y. Narahari. *Performance modeling of automated manufacturing systems*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1992. (Cited on page 20).

[WER+03] H. Weber, H. Ehrig, W. Reisig, A. Borusan, S. Lembke, J. Dehnert, M. Weber, A. Martens, J. Padberg, C. Ermel, and A. Qemali. The Petri Net Baukasten of the DFG Forschergruppe Petri net technology. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, *Petri net technology for communication-based systems*, volume 2472 of *Lecture notes in computer science (LNCS)*, pages 1–21. Springer-Verlag, November 2003. (Cited on page 11).

[Wie96a] R. Wieting. Hybrid high-level nets. In J.M. Charnes, D.J. Morrice, D.T. Brunner, and J.J. Swain, editors, *Proceedings 28th winter simulation conference, Coronado, California, USA*, pages 848–855, December 1996. (Cited on page 27).

[Wie96b] R. Wieting. Modeling and simulation of hybrid systems using hybrid high-level nets. In A.G. Bruzzone and E.J.H. Kerkhoffs, editors, *Proceedings 8th European simulation symposium (ESS), Genoa, Italy*, pages 158–162, October 1996. (Cited on page 27).

[Wik10] Wikipedia, the Free Encyclopedia, 2010. http://en.wikipedia.org/wiki/. (Cited on pages 16, 17, and 18).

[YLB95] Y.Y. Yang, D.A. Linkens, and S.P. Banks. Modelling of hybrid systems based on extended coloured Petri nets. In P. Antsaklis, S. Sastry, W. Kohn, and A. Nerode, editors, *Hybrid systems II*, volume 999, pages 509–528. Springer-Verlag, December 1995. (Cited on pages 22, 23, and 27).

[Zen85] A. Zenie. Coloured stochastic Petri nets. In *Proceedings international workshop on timed Petri nets, Torino, Italy*, pages 262–271. IEEE computer society press, July 1985. (Cited on pages 22 and 25).

Appendix A

Preliminaries on stochastic processes

This thesis assumes a background knowledge in stochastic processes. There are many good textbooks that give an introduction to these processes, including [Dav93], [EK86], and [Bag93]. This appendix provides a brief overview of the main terminology and notation used.

Random variables, probability, and expectation

- Ω is a set, usually referred to as the *sample space*.
- $\Im$ is a collection of subsets of Ω, with the requirement that it is a *σ-algebra* on Ω. This means that
  - $\emptyset \in \Im$,
  - if $A \in \Im$ then also $\Omega \setminus A \in \Im$ and
  - if $A_1, A_2, \ldots \in \Im$ then $\cup_{i=1}^{\infty} A_i \in \Im$.

  Here, $\emptyset$ denotes the empty set. Note that with the requirements above, also $\Omega \in \Im$. If A is a member of $\Im$ it is called *measurable*. $(\Omega, \Im)$ is referred to as a *measurable space*.
- If S is a collection of subsets of Ω, then the *σ-algebra generated by S* and referred to as $\sigma(S)$ is the smallest σ-algebra on Ω that contains all elements of S. Special cases are:
  - The *Borel σ-algebra* on $\mathbb{R}$, referred to as $\mathcal{B}(\mathbb{R})$, is the smallest σ-algebra that contains all open intervals.
  - The *Borel σ-algebra* on $\mathbb{R}^n$, referred to as $\mathcal{B}(\mathbb{R}^n)$, is the smallest σ-algebra that contains all n-dimensional open intervals.
  - The *Borel σ-algebra* on E, referred to as $\mathcal{B}(E)$, is the smallest σ-algebra that contains all open subsets of E.

A *Borel subset* is an element of a Borel σ-algebra. A σ-algebra is *separable* if it can be generated by a countable collection of sets. A Borel measurable space is denoted $(E, \mathcal{B}(\mathcal{E}))$.

- A *measure* on $(\Omega, \Im)$ is a function $\mu : \Im \to \mathbb{R}^+$ such that
  - $\mu\{\emptyset\} = 0$,
  - if $A_1, A_2, \ldots \in \Im$ are pairwise disjoint, i.e., $A_i \cap A_j = \emptyset$ for $i \neq j$, then $\mu\{\cup_{i=1}^{\infty} A_i\} = \sum_{i=1}^{\infty} \mu\{A_i\}$.

  $(\Omega, \Im, \mu)$ is referred to as a *measure space*.

  Special cases are:
  - A *probability measure* on $(\Omega, \Im)$ is a measure $\mathbb{P} : \Im \to [0, 1]$ for which $\mathbb{P}\{\Omega\} = 1$. $(\Omega, \Im, \mathbb{P})$ is referred to as a *probability space*.
  - A *Borel measure* is any measure on a Borel measurable space, i.e., any measure on $(E, \mathcal{B}(\mathcal{E}))$.
  - The *Lebesgue measure* μ_L is the unique measure that is constructed as follows: For any subset B of $\mathbb{R}$, we define an *outer measure* μ_L^* by: $\mu_L^*\{B\} = \inf\{\mu^*\{M\} \mid M \supseteq B\}$, where M is a countable union of intervals and $\mu^*\{M\}$ is the sum of the lengths of these intervals. Then a set A is *Lebesgue measurable* if for all sets B, $\mu_L^*(B) = \mu_L^*\{A \cap B\} + \mu_L^*\{B \setminus A\}$. The Lebesgue measurable sets form a σ-algebra, and the Lebesgue measure μ_L is defined by $\mu_L\{A\} \triangleq \mu_L^*\{A\}$ for any Lebesgue measurable set A.

    For n-dimensional intervals, in the definition above M is a countable union of products of intervals and $\mu^*\{M\}$ is the sum of the product of the lengths of these intervals.

- If $(\Omega_1, \Im_1)$ and $(\Omega_2, \Im_2)$ are two measurable spaces then a function $f : \Omega_1 \to \Omega_2$ is called *measurable* if $\{\omega_1 \in \Omega_1 \mid f(\omega_1) \in A\}$ is in $\Im_1$ for every $A \in \Im_2$.

  A function $g : \mathbb{R}^n \to \mathbb{R}$ is *Borel-measurable* if every set of the form $\{x \in \mathbb{R}^n \mid g(x) \in \mathcal{B}(\mathbb{R})\}$ belongs to $\mathcal{B}(\mathbb{R}^n)$.

- A measurable set A is called a *μ-null-set* if $\mu\{A\} = 0$. A measure space $(\Omega, \Im, \mu)$ is *complete* if every subset of a μ-null set is measurable. The Lebesgue measure (or, more precisely, the Lebesgue measure space) is complete whereas the Borel measure is not. An event A is *$\mathbb{P}$-almost sure* if $\mathbb{P}\{A\} = 1$.

- A *random variable* on $(\Omega, \Im)$ is a measurable function from $(\Omega, \Im)$ to $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$. In other terms, a random variable is a function $X : \Omega \to \mathbb{R}$ such that $\{\omega \in \Omega \mid X(\omega) \leq x\} \in \Im$ for every $x \in \mathbb{R}$. A *random vector* is a vector $X = (X_1, \ldots, X_n)'$ such that X_i is a random

variable for each i. The probability that X takes on values in a set A is given by $\mathbb{P}\{X \in A\} \triangleq \mathbb{P}\{\omega \in \Omega \mid X(\omega) \in A\}$.

- The *probability distribution function* $F_X : \mathbb{R} \to [0,1]$ of a random variable X is given by $F_X(x) = \mathbb{P}\{\omega \in \Omega \mid X(\omega) \leq x\}$ (or by $F_X(x) = \mathbb{P}\{X \leq x\}$ for short).

- The *survivor function* $\Gamma_X : \mathbb{R} \to [0,1]$ of a random variable X is given by $\Gamma_X(x) = \mathbb{P}\{X > x\}$. If X represents time then the survivor function represents the probability that X occurs at least after time x (it survives until at least time x). Note that $\Gamma_X(x) = 1 - F_X(x)$.

- If $f : \mathbb{R} \to \mathbb{R}$ is a function and if X is a random variable with probability distribution function F_X then the *expectation* of $f(X)$ is given by

$$\mathbb{E}\{f(X)\} = \int_{-\infty}^{\infty} f(x) dF_X(x)$$

- A random variable X on a probability space $(\Omega, \Im, \mathbb{P})$ is *integrable* if $\int_\Omega |X(\omega)| d\mathbb{P}(\omega)$ exists and is finite.

Stochastic processes

- A *stochastic process* with index set $\mathbb{T}$ and state space $(E, \mathcal{B}(E))$ (a Borel measurable space) defined on a probability space $(\Omega, \Im, \mathbb{P})$ is a function X defined on $\mathbb{T} \times \Omega$ with values in E such that for each $t \in \mathbb{T}$, $X(t, \cdot) : \Omega \to E$ is an E-valued random variable (or random vector), i.e., $\{\omega \mid X(t,\omega) \in B\} \in \Im$ for every $B \in \mathcal{B}(E)$.

  Throughout this appendix, we take $\mathbb{T} = \mathbb{R}^+ = [0, \infty)$, i.e., the positive time line. Generally, $X(t,\omega)$ is denoted as $X_t(\omega)$ and $X(t,\cdot)$ is denoted as $X(t)$ or as X_t.

  X is *measurable* if $X : [0,\infty) \times \Omega \to E$ is $\mathcal{B}[0,\infty) \times \Im$-measurable.

  X is *continuous* if for every $\omega \in \Omega$, $X(\cdot, \omega)$ is continuous. Similar definitions apply for *almost surely continuous*, *right continuous*, *left continuous*.

  The function $X(\cdot, \omega)$ is called the *sample path* of the process at ω.

- A collection $\{\Im_t\} = \{\Im_t, t \in \mathbb{T}\}$ of σ-algebras of sets in $\Im$ is a *filtration* if $\Im_t \subset \Im_{t+s}$ for $t, s \in \mathbb{T}$. Intuitively, $\Im_t$ corresponds to the information known to an observer at time t. We say $\{\Im_t\}$ is *complete* if $(\Omega, \Im, \mathbb{P})$ is complete and $\{N \in \Im \mid \mathbb{P}(N) = 0\} \subset \Im_0$. For a process X, the natural filtration $\{\Im_t^X\}$ is defined by the filtration induced by X, i.e., $\Im_t^X = \sigma(X(s) \mid s \leq t)$, i.e., $\Im_t^X$ is the information obtained by observing X up to time t. A process X is *adapted* to a filtration $\{\Im_t\}$ (or X is $\{\Im_t\}$-adapted) if $X(t)$ is $\Im_t$-measurable

for each $t \geq 0$. This means that the filtration gives us enough information to find the value of the process. Since $\Im_t$ is increasing in t, X is $\{\Im_t\}$-adapted if and only if $\Im_t^X \subset \Im_t$ for each $t \geq 0$.

$(\Omega, \Im, \{\Im_t\}, \mathbb{P}, \mathbb{T})$ is referred to as a *filtered probability space* or *stochastic basis*.

- A *stopping time* τ with respect to a filtration $\{\Im_t\}$ is a random variable $\tau : \Omega \to [0, \infty)$ such that $\{\tau \leq t\} \triangleq \{\omega \in \Omega \mid \tau(\omega) \leq t\} \in \Im_t$ for every $t \in \mathbb{T}$. In words, this says that we should be able to decide on the basis of the information on the process up to time t, whether the stopping time has passed. Or: a stopping time is a random time at which 'something detectable happens'.

- Let $\{X(t) \mid t \geq 0\}$ be a stochastic process defined on a probability space $(\Omega, \Im, \mathbb{P})$ with values in E and let $\Im_t^X = \sigma(X(s) \mid s \leq t)$. Then X is a *Markov process* if $\mathbb{P}\{X(t+s) \in A \mid \Im_t^X\} = \mathbb{P}\{X(t+s) \in A \mid X(t)\}$ for all $s, t \geq 0$ and $A \in \mathcal{B}(E)$.

  A Markov process is called *time-homogeneous* if $\mathbb{P}\{X(t+s) \in A \mid X(t)\}$ depends only on s and not on t.

  A Markov process has the *strong Markov property* if it has the Markov property for any stopping time τ.

- A function $Q(t, x, A)$ defined on $[0, \infty) \times E \times \mathcal{B}(E)$ is a time-homogeneous transition function if

  - $Q(t, x, \cdot)$ is a probability measure for fixed $t \in [0, \infty)$, $x \in E$
  - $Q(0, x, \cdot) = \delta_x$ (unit mass at x), $x \in E$
  - $Q(\cdot, \cdot, A) \in B([0, \infty) \times E)$, $A \in \mathcal{B}(E)$, where $B([0, \infty) \times E)$ denotes the space of bounded functions on $[0, \infty) \times E$
  - $Q(t+s, x, A) = \int Q(s, y, A) Q(t, x, dy)$, $s, t \geq 0$, $x \in E$, $A \in \mathcal{B}(E)$

- The probability measure μ given by $\mu(A) = \mathbb{P}\{X(0) \in A\}$ is called the initial distribution of X.

- A *Wiener process* relative to a right continuous filtration $\{\Im_t\}$ is a continuous adapted real-valued process $\{W_t\}$ such that $W_0 = 0$ and for every t,

  - $\mathbb{E}\{W_t\} = 0$
  - $\mathbb{E}\{W_t^2\} < \infty$
  - $W_t - W_s$ is independent of the σ-algebra $\Im_s$ for all $s \leq t$

A Wiener process is said to be *standard* if $\mathbb{E}\{W_t^2\} = t$. It can be shown that $W_t - W_s \sim N(0, t-s)$, i.e., Gaussian with mean zero and variance $t-s$. In addition, $\mathbb{E}\{W_t W_s\} = \min\{t, s\}$ and W is a Markov process. A *h-dimensional Wiener process* is an h-vector of independent Wiener processes. A process B_t is called a *standard Brownian motion* if there exists a stochastic basis, say $(\Omega, \Im, \{\Im_t\}, \mathbb{P}, \mathbb{T})$, such that B_0 is $\Im_0$-measurable and $\{B_t - B_0\}$ is a standard Wiener process relative to $\{\Im_t\}$.

- If $\{X_t\}$ is a stochastic process on filtration $\{\Im_t\}$ then the pair $\{(X_t, \Im_t) \mid t \geq 0\}$ is called a *martingale* if $\mathbb{E}\{|X_t|\} < \infty$ for all $t \geq 0$ and $\mathbb{E}\{X_t \mid \Im_s\} = X_s$ for all $s < t$. This means that the expected value of the process at any time in the future, given all information about the process up to the present time, is equal to the present value of the process. The stochastic process $\{X_t\}$ is called a *local martingale* if there exists a sequence of stopping times $\tau_k : \Omega \to \mathbb{R}^+$ such that the τ_k are almost surely increasing: $\mathbb{P}\{\tau_k < \tau_{k+1}\} = 1$; the τ_k diverge almost surely: $\mathbb{P}\{\tau_k \to \infty \text{ as } k \to \infty\} = 1$; and the stopped process $X_t^{\tau_k} \triangleq X_{\min\{t, \tau_k\}}$ is a martingale for every k. A stochastic process $\{X_t\}$ is called a *semi-martingale* if its trajectories are *càdlàg*[1] (i.e., right-continuous and with left limits), and if it can be represented in the form $X_t = M_t + V_t$, where $\{(M_t, \Im_t)\}$ is a local martingale and $\{(V_t, \Im_t)\}$ is a process of *finite variation*, i.e., $\int_0^t dV_s(\omega) < \infty, t > 0, \omega \in \Omega$. Note that this decomposition is unique for continuous processes, but not in general.

- A stochastic process X adapted to a filtration $\{\Im_t\}$ is a *diffusion* if it is a strong Markov process with respect to $\{\Im_t\}$, homogeneous in time, and has continuous sample paths.

- A *Poisson process* is a process $\{N(t) \mid t \geq 0\}$, where $N(t)$ is the number of events that have occurred up to time t (starting from time 0), which has a Poisson distribution: $\mathbb{P}\{N(t) = k\} = \frac{e^{-\lambda t}(\lambda t)^k}{k!}$, where λ is the mean number of events per unit time, sometimes referred to as jump rate. The sojourn time between two events that are generated by a Poisson process is exponentially distributed: If t_k denotes the time of the kth event, then $\mathbb{P}\{t_k - t_{k-1} > t\} = e^{-\lambda t}$. A Poisson Point process is a collection of Poisson random variables indexed by intervals.

Poisson random measure

- The *predicable σ-algebra* is the σ-algebra $\digamma_P$ on $\mathbb{T} \times \Omega$, that is generated by all adapted left-continuous processes. A stochastic process is *predictable* if it is $\digamma_P$-measurable. The *optional σ-algebra* is the σ-algebra $\digamma_O$ on $\mathbb{T} \times \Omega$, that is generated by all adapted right-

[1] Continue à droite, limitée à gauche.

continuous processes. A stochastic process is *optional* if it is $\mathcal{F}_O$-measurable. Let $\mathcal{F}_C$ denote the σ-algebra on $\mathbb{T} \times \Omega$ that is generated by all adapted càdlàg processes.

- A *random measure* $p(\cdot; dt, dz)$ or $p(dt, dz)(\cdot)$ on $\mathbb{R}^+ \times \mathbf{Z}$ is a family $\{p(\omega; dt, dz) \mid \omega \in \Omega\}$ of non-negative measures on $\mathcal{B}(\mathbb{R}^+) \times \mathcal{B}(\mathbf{Z})$ such that $p(\omega; \{0\} \times \mathbf{Z}) = 0$ for all ω.

- Let p be a random measure and let $V(\cdot)$ denote a $\mathcal{F}_C \times \mathcal{B}(\mathbf{Z})$-measurable mapping of $\Omega \times \mathbb{R}^+ \times \mathbf{Z}$ into $\mathbb{R}$, then define the $\mathbb{R} \cup \{\infty\}$-valued integral process $\{[V * p]_t\}$ as follows:

$$[V * p]_t(\omega) = \begin{cases} \int_{[0,t]\times\mathbf{Z}} V(\omega, s, z) p(\omega; ds, dz) & \text{if this integral } < \infty \\ 0 & \text{otherwise} \end{cases}$$

  A random measure p is called *optional* if the process $\{[V * p]_t\}$ is optional for every $\mathcal{F}_C \times \mathcal{B}(\mathbf{Z})$-measurable mapping V. A random measure is called *predictable* if the process $\{[V * p]_t\}$ is predictable for every $\mathcal{F}_P \times \mathcal{B}(\mathbf{Z})$-measurable mapping V. A random measure is said to be $\mathcal{F}_P \times \mathcal{B}(\mathbf{Z})$-$\sigma$-finite if there exists an $\mathcal{F}_P \times \mathcal{B}(\mathbf{Z})$-measurable partition (A_i) of $\Omega \times \mathbb{R}^+ \times \mathbf{Z}$, such that each $[\mathbf{1}_{A_i} * p]_\infty$ is integrable.

- An *integer valued random measure* is an optional $\mathcal{F}_P \times \mathcal{B}(\mathbf{Z})$-$\sigma$-finite random measure $p(\omega; dt, dz)$ satisfying:

  - $p(\omega; \{t\}, \mathbf{Z}) \leq 1$, for every ω
  - for each $A \in \mathcal{B}(\mathbb{R}^+) \times \mathcal{B}(\mathbf{Z})$, $p(\cdot; A)$ assumes values in $\mathbb{N} \cup \{\infty\}$.

- The *intensity measure* $\nu(dt, dz)$ of an integer valued random measure is defined by $\nu(A) = \mathbb{E}\{p(\cdot; A)\}$, where $\mathbb{E}\{\cdot\}$ denotes expectation. ν is said to be σ-finite if there exists a sequence of sets $A_i \in \mathcal{B}(\mathbb{R}^+) \times \mathcal{B}(\mathbf{Z})$, such that $A_i \uparrow \mathbb{R}^+ \times \mathbf{Z}$ for increasing i, while $\nu(A_i) < \infty$ for every i.

- An *extended Poisson random measure* on $\mathbb{R}^+ \times \mathbf{Z}$ relative to the filtration $\Im_t$, is an integer-valued random measure $p(\omega; dt, dz)$ which satisfies:

  - its intensity measure ν is σ-finite,
  - for every $t \in \mathbb{R}^+$ and every $A \in \mathcal{B}(t, \infty) \times \mathcal{B}(\mathbf{Z})$, such that $\nu(A) < \infty$, the variable $p(\cdot; A)$ is independent of the σ-algebra $\Im_t$.

  The last item indicates that the memoryless property is satisfied.

- A *Poisson random measure* $p_P(\cdot; dt, dz)$ is an extended Poisson random measure, the intensity measure ν of which satisfies $\nu(\{t\} \times \mathbf{Z}) = 0$, for all t. A Poisson random measure is said to be *homogeneous* if its intensity measure is of the form $\nu(dt, dz) = dt \cdot \tilde{\mu}(dz)$.

Ordinary and stochastic differential equations

- A function $f : \mathbb{R} \to E$ is *continuous* if $\lim_{x \to y} f(x) = f(y)$ for all $y \in \mathbb{R}$. A function is *càdlàg* if $\lim_{x \downarrow y} f(x) = f(y)$ for all $y \in \mathbb{R}$ and $f(y-) = \lim_{x \uparrow y} f(x)$ exists for all $y \in \mathbb{R}$. A function is *absolutely continuous* if for every interval $[a, b]$ and $\epsilon > 0$ there is a $\delta > 0$ such that $\sum_{i=1}^{n}(b_i - a_i) < \delta$ implies $\sum_{i=1}^{n} |f(b_i) - f(a_i)| < \epsilon$ for any disjoint intervals (a_i, b_i), $i = 1, \ldots, n$ contained in $[a, b]$.

- A function $f : \mathbb{R}^n \to \mathbb{R}^n$ is called *Lipschitz continuous* on $A \subset \mathbb{R}^n$ if there exists a constant L such that $|f(x) - f(y)| \leq L|x - y|$, for all $x, y \in A$, where $|\cdot|$ is the *Euclidean norm*, which is defined for $x \in \mathbb{R}^n$ as $|x|^2 = \sum_{i=1}^{n} x_i^2$. f is called *locally Lipschitz* if it is Lipschitz continuous for any compact set A.

- An *ordinary differential equation* is an equation $\dot{X}(t) = f(t, X(t))$, where $\dot{X}(t) \triangleq dX(t)/dt$ refers to the derivative of X with respect to t, and $X(t)$ assumes values in $\mathbb{R}^n$. Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be locally Lipschitz, then the ordinary differential equation $\dot{X}(t) = f(t, X(t))$, $X(0) \in \mathbb{R}^n$ has a unique solution for $t \in [0, t_e)$, where $t_e \leq \infty$ is the *explosion time*, i.e., the first time such that $\lim_{t \to t_e} |X(t)| = \infty$. The explosion time is infinite if a *growth condition* is satisfied, i.e., for all x, there exists K such that $|f(t, x)| \leq K(1 + |x|)$.

- A *stochastic differential equation* is an equation $dX_t = f(t, X_t)dt + g(t, X_t)dW_t$, where $\{W_t\}$ a h-dimensional standard Wiener process, and X_t assumes values in $\mathbb{R}^n$. Here, $f : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}^n$ is a mapping, also named *drift coefficient*; $g : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}^{n \times h}$ is a matrix-valued mapping, also named *diffusion coefficient*. If f and g satisfy a locally Lipschitz condition and a growth condition, then the stochastic differential equation has a unique strong solution (or pathwise unique solution).
  Note that where necessary, the n-dimensional differential equation can be regarded as n one-dimensional stochastic differential equations $dX_t^i = f^i(t, X_t)dt + \sum_{j=1}^{b} g^{ij}(t, X_t)dW_t^j$. In addition, the time dependency can be removed by replacing $\text{Col}\{t, X_t\}$ by $Y_t \in \mathbb{R}^{n+1}$.

- The solution of an ordinary or stochastic differential equation can be denoted by the *flow* $\phi : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}^n$, where $\phi(t, x_0)$ equals the solution at time t when the differential equation is initiated at state x_0 at time zero, i.e., $\phi(t, x_0) = x_0 + \int_0^t dx_t$.

Index

Abstract

A *general stochastic hybrid process* (GSHP) is a mathematical formalism that covers most of the requirements posed by the modelling of complex operations, such as time dependencies, multi-dimensional continuous as well as discrete processes, discontinuities, randomness and model uncertainties. In addition, it is possible to study GSHP by using stochastic analysis methodologies, thereby empowering it with powerful mathematical properties. This guarantees unambiguous simulation possibility of the model and allows speeding up this simulation while keeping the model properties intact. However, using GSHP to construct a model of a complex operation is not easy. To support the modelling and the subsequent verification both by mathematical and by multiple operational domain experts, a supporting graphical modelling formalism is desired. *Petri nets* have shown to be useful for developing models of various complex applications. Typical Petri net features are concurrency and synchronisation mechanism, hierarchical and modular construction, and natural expression of causal dependencies, in combination with graphical and analytical representations.

The aim of this thesis is to combine the strengths of Petri net modelling formalisms and those of GSHP. First, *dynamically coloured Petri nets* (DCPN) are developed, and proof of equivalence is provided with *piecewise deterministic Markov processes*, which is a particular class of GSHP. Next, DCPN are extended to *stochastically and dynamically coloured Petri nets* (SDCPN), and proof of equivalence is provided with GSHP. Subsequently, SDCPN are extended to *SDCPN with interconnection mapping types* ($\text{SDCPN}^{\text{imt}}$) and proof of equivalence is provided with both SDCPN and GSHP. It is shown with illustrative air transport examples that these three classes of Petri net are very effective when it comes to the compositional modelling of operations consisting of many distributed components that behave and interact in a dynamic way with many uncertainties. With the equivalence relations between these formalisms, the properties and strengths of the various approaches are combined. The many applications of the approach developed in this thesis, executed at NLR and beyond, show that both the approach and its combined strengths are acknowledged and supported by practice.

Samenvatting

Een *general stochastic hybrid process* (GSHP) is een wiskundig formalisme waar de meeste aspecten van complexe operaties mee gemodelleerd kunnen worden, zoals tijdsafhankelijkheden, multi-dimensionale continue en discrete processen, discontinuiteiten, en model onzekerheden. Daarnaast kunnen GSHP worden bestudeerd via stochastische analyse methodes, waarmee ze krachtige wiskundige eigenschappen krijgen. Dit alles zorgt voor een unieke simulatie van het model die bovendien versneld kan worden bij gelijkblijvende modeleigenschappen. Echter, bij het modelleren van een complexe operatie zijn GSHP niet gemakkelijk in het gebruik. Om het modelleren en de aansluitende modelverificatie door zowel wiskundige experts als verschillende operationele domein experts te ondersteunen is een grafisch hulpmiddel onontbeerlijk. *Petri netten* hebben voor het ontwikkelen van modellen voor vele complexe toepassingen hun waarde reeds bewezen. Typische Petri net eigenschappen zijn mechanismes om parallelle processen en gesynchroniseerde processen te modelleren, hierarchische en modulaire constructies te bouwen, en causale verbanden op natuurlijke wijze te representeren, en dat in combinatie met zowel een grafische representatie als een representatie via wiskundige formules.

Dit proefschrift beoogt de kracht van Petri netten en GSHP te combineren. Eerst worden *dynamically coloured Petri nets* (DCPN) ontwikkeld, en er wordt bewezen dat deze equivalent zijn met *piecewise deterministic Markov processes*, een bepaalde klasse van GSHP. Vervolgens worden DCPN uitgebreid tot *stochastically and dynamically coloured Petri nets* (SDCPN), en er wordt bewezen dat deze equivalent zijn met GSHP. Als derde worden SDCPN uitgebreid tot *SDCPN with interconnection mapping types* ($\text{SDCPN}^{\text{imt}}$) en er wordt bewezen dat deze equivalent zijn met zowel SDCPN als GSHP. Met diverse aansprekende voorbeelden uit het luchtverkeer wordt geillustreerd dat deze drie Petri net klasses erg effectief zijn bij het compositioneel modelleren van operaties die bestaan uit veel gedecentreerde componenten die zich gedragen en elkaar beinvloeden op een dynamische manier met veel onzekerheden. Met de equivalentierelaties tussen de formalismes worden de kracht en eigenschappen van de verschillende aanpakken gecombineerd. De vele toepassingen van de aanpak ontwikkeld in dit proefschrift, die zowel bij het NLR als daarbuiten reeds zijn uitgevoerd, laten zien dat zowel de aanpak als zijn combinatie van goede eigenschappen door de praktijk worden bekrachtigd en ondersteund.

Acknowledgements

The SDCPN development started within the context of the project MUFTIS (Model Use and Fast Time Simulation Studies) that the National Aerospace Laboratory NLR, with partners, executed for the European Commission during 1995 and 1996. This project aimed at developing models and techniques for the evaluation of air traffic management (ATM) operations. The first part of the study, [EKBF96], revealed that commonly used models for safety risk evaluation lack the capability to address the complex dynamic interactions that occur in ATM. Therefore, the second part of the study [EKB96] was dedicated to investigating dynamic assessment techniques.

The first step was to adopt an appropriate set of equations for the risk of a collision between aircraft. The one that appeared most suitable was the one developed in [BB93], named *generalised Reich collision risk equations*. These equations use as input a probability density function for the relative position and velocity of two aircraft. Unfortunately, the precise form of this function is generally unknown, in particular under conditions of safety-related occurrences that may be applicable for future operational concepts. Therefore, a modelling formalism was required to support the evaluation of aircraft behaviour for a range of ATM operations. There were two main selection criteria for this modelling formalism: 1) It needed to have a powerful stochastic analysis support that ensured the collision risk to be uniquely evaluated; 2) It needed to have a powerful graphical support, for model development, verification and communication purposes. The stochastic analysis support was soon found in *piecewise deterministic Markov processes* (PDP), [Dav84, Dav93]. For the graphical support, Petri nets were considered as a good candidate.

The original idea to have a closer look at Petri nets came (in 1995) from Henk Blom, who came across the formalism through Nicolas Fota, co-author of the MUFTIS part 1 report (and now with Eurocontrol). Henk asked if I knew Petri nets and if I wanted to investigate whether they could be of use to us. And I did.[2] Subsequently, within MUFTIS, several classes of Petri net were identified and considered for useful elements, and a first informal mapping from PDP into Petri nets was drafted, [EKB96].

However, to take advantage of both the stochastic analysis support and the graphical support

[2] I knew Petri nets from my stay at Eindhoven University of Technology (1992-1994), and had also seen them used in practice at my six-months graduation internship for Eindhoven, which I executed at Railned in 1994.

selected, the existence of a *formal* equivalence relationship between the Petri net class used and the class of PDP was considered a must. After the MUFTIS project ended, a range of existing Petri net classes were considered as candidates for this purpose (many of which are described in Chapter 2), but for none of these classes, a mapping to PDP could be proven. This motivated the development of a new class, which was named *dynamically coloured Petri net* (DCPN)[3]. The first version was published in 1997, [EBK97], with the mapping from PDP into DCPN proven (not yet vice versa).

From this first development onwards, the DCPN definition was improved and further formalised, it was used for the modelling of several air transport applications, and at the request of users, it was enriched with more modelling power by means of additional Petri net features (such as inhibitor arcs, enabling arcs, immediate transitions). In parallel to this, through a sequence of studies, [EB00, EB05], the proof of mapping from DCPN into PDP was also established, and updated each time an additional Petri net practical modelling feature was added. The eventual result is Chapter 3.

The next extension was the formal inclusion of diffusion terms in the token colours. Diffusion exists in air transport operations for example in the form of stochastic variations around position and velocity of an aircraft, but is not covered by PDP. In early air transport modelling exercises at NLR, diffusion terms were added to DCPN terms in an ad hoc way. When [BL03, BLGP03] formally extended PDP to GSHS (general stochastic hybrid system) by inclusion of diffusion, DCPN were formally extended to SDCPN, and equivalence relations between SDCPN and GSHS were proven, [EB06]. In SDCPN, the added 'S' stands for *stochastically*.

When the work for this PhD thesis was already far advanced, I stumbled upon a few 'issues' related to the formal execution of a GSHS. There were two options: (1) Try to solve these issues; (2) Find another class of stochastic hybrid process equivalent to SDCPN. A few such classes were candidate for this at [HYB05], and the choice was made to study *hybrid stochastic differential equations* (HSDE) as developed in [Blo03, BBEP03]. The eventual result is Chapter 4. The change from GSHS to HSDE proved to be challenging; however, it also made the study more interesting, in that Chapter 4 is now not a straightforward extension of Chapter 3. In hindsight, the change was particularly fortunate since it allowed the establishment of relations between three formalisms (SDCPN, GSHS, HSDE) rather than two, which significantly broadened the scope and strength of the developments, [EB10a, EB10b], see Chapter 7.

The last extension of the formalism were features that enrich the compositional specification power of SDCPN. The development of these features started during 1999. The trigger was the desire to separate local modelling issues (e.g., model of a pilot, model of an aircraft system) from global modelling issues (e.g. model of how the pilot interacts with the aircraft). The first version

[3]After failing to find a good name, I wrote out a contest for my colleagues. The name DCPN was proposed (in January 1997) by former NLR employee Jasper Daams, who is now with the Dutch Air Traffic Control (LVNL). He won a certificate of appreciation and a box of chocolates for this winning proposal.

of the approach was documented in [EB02]. This first version was further enhanced and improved, by making use of experience obtained in practical applications. The result was named SDCPN^{imt}, where 'imt' stands for *interconnection mapping types*, see [EKBK06] and Chapter 5.

During the time I was given to work on this thesis, I took the opportunity to incorporate a few improvements both in the definition of all three Petri net classes developed, and in their proofs of equivalence to stochastic hybrid processes.

The history described above may make clear that this thesis was not written in a day, and not even in your typical four-year University stay. It may also make clear that I could not have completed this thesis without the help and support of others. Therefore, I take this opportunity to say a big thanks.

First of all, I want to thank the National Aerospace Laboratory NLR and its Air Transport Safety Institute for providing me the opportunity to write this thesis, and for giving the support throughout the years for writing intermediate results in conference and journal papers. I thank Alex Rutten in particular for arranging the financial support to write a part of this thesis during company hours. Second, I thank my promotors Arun Bagchi and Boudewijn Haverkort, as well as Anton Stoorvogel of the University of Twente for providing valuable comments on the final concept version of this thesis, and Jaroslav Krystul for providing valuable insights into stochastic processes.

I want to express my deepest gratitude to Henk Blom of NLR for his support and encouragement throughout the years. Henk is always several steps ahead of everybody, and is in hindsight always right. I am time and time again astonished by his insight and knowledge of so many diverse subjects. Without his vision and motivation, this thesis would not have been as complete as it is now, and perhaps would not have been completed at all.

I also want to thank all of my colleagues at the NLR Air Transport Safety Institute for making work look like a hobby, and for frequently showing their interest about the progress in writing this booklet. A special thanks is for Bert Bakker, Bas van Doorn, Bart Klein Obbink, Margriet Klompstra, Sybert Stroeve, and Hicham Zmarrou, and also for Jasper Daams and Marco van der Park (both formerly NLR and currently at LVNL), Ítalo de Oliveira (São Paolo University), Heber Herencia-Zapana (Old Dominion University), Fedja Netjasov (Belgrade University) and Eri Itoh (Tokyo University), for their persistence during their development of SDCPN models covering hundreds of pages of Petri net graphs and tables, and for their very valuable suggestions that ultimately improved the practical modelling power of SDCPN^{imt}. Without their enthusiasm, I would have lacked the motivation to write this thesis.

I want to thank my parents for their endless love and support and for making me wanting them to feel proud. And finally, a BIG hug goes to my main men: my husband Frans Panken and our sons David and Ruben. Thank you for your patience and support when "mummy had to work on her book again". You truly make life a joy.

Curriculum vitae

Mariken Everdij was born on 17 March 1968 in Wageningen, the Netherlands. In 1986 she received her VWO diploma from CLV in Veenendaal, after which she went to the University of Twente to study Applied Mathematics. In 1992 she obtained her MSc degree. The title of her Master's thesis was "Jump linear quadratic Gaussian control under regime uncertainties", which was completed during her final graduation project at the National Aerospace Laboratory NLR in Amsterdam. During 1992–1994 she studied at Eindhoven University of Technology and obtained a Master of Technological Design (MTD) degree in Mathematics for Industry. Part of the programme was a final graduation project, which she executed at Railned, the independent organisation for railway safety. The title of her MTD thesis was "Judging the stability of train timetables". From October 1994 she is an employee at NLR, where she has been working on developing stochastic dynamic models as basis for insight and safety analysis of enhancements in air transport operations. Part of these developments are a stochastically and dynamically Petri net formalism and a method for bias and uncertainty assessment in risk models. She is now a senior scientist at NLR's Air Transport Safety Institute, where she works on developing safety methods to address future air transport operations that involve many diverse stakeholders. Mariken is married to Frans Panken and together they have two sons: David (born in 2000) and Ruben (born in 2003).