HYBRIDGE

Distributed Control and Stochastic Analysis of Hybrid Systems
Supporting Safety Critical Real-Time Systems Design

WP2: Stochastic hybrid systems based modelling of accident risk

MODELLING HYBRID STATE MARKOV PROCESSES THROUGH STOCHASTICALLY AND DYNAMICALLY COLOURED PETRI NETS

Mariken Everdij[1] and Henk Blom[1]

30th May 2005

Version: *0.6*
Task number: *2.4*
Deliverable number: *D2.4*
Contract: *IST-2001-32460 of European Commission*

[1] National Aerospace Laboratory NLR, NL

DOCUMENT CONTROL SHEET

Title of document: *Modelling Hybrid State Markov Processes through Stochastically and Dynamically Coloured Petri Nets.*

Authors of document: *M.H.C. Everdij and H.A.P. Blom*

Deliverable number: *D2.4*

Contract: *IST-2001-32460 of European Commission*

Project: *Distributed Control and Stochastic Analysis of Hybrid Systems Supporting Safety Critical Real-Time Systems Design (HYBRIDGE)*

DOCUMENT CHANGE LOG

Version #	Issue Date	Sections affected	Relevant information
0.1	16 April 2003	All	Initial draft
0.2	26 June 2003	All	Second draft
0.3	30 June 2003	Sections 1 and 7	Consolidated draft
0.4	1 July 2003	All	Minor corrections
0.5	10 September 2004	All	Review comments used
0.6	30 May 2005	All	“SCPN” replaced by “SDCPN”

Version 1.0		Organisation	Signature/Date
Authors	M.H.C. Everdij	NLR	
	H.A.P. Blom	NLR	
Internal reviewers	M.B. Klompstra	NLR	
	P. Lezaud	CENA	
	A. Van der Schaft	TWEN	
	S. Strubbe	TWEN	

Abstract

Piecewise Deterministic Markov Processes (PDPs) are known as the largest class of strong Markov processes virtually describing all continuous-time processes not involving diffusions. In general the state space of a PDP is of hybrid type, i.e. a Kronecker product of a discrete set and a continuous-valued space. Since Stochastic Petri Nets have proven to be extremely useful in developing continuous-time Markov Chain models for complex practical discrete-valued processes, there is a clear need for a type of Petri Nets that can play a similar role for developing PDP models for complex practical problems. To fulfil this need, the report defines a Dynamically Coloured Petri Net (DCPN), and proves that there exist into-mappings between PDPs and DCPNs. Subsequently, the DCPN definition is extended to Stochastically and Dynamically Coloured Petri Net (SDCPN), and it is shown that there exist into-mappings between Generalised Stochastic Hybrid Processes (GSHP's) and SDCPNs.

Contents

1 Introduction 5

2 DCPN elements and execution 7

3 Piecewise Deterministic Markov Processes 10

4 Into-mappings between DCPN and PDP and between SDCPN and GSHP 13

5 Example DCPN and example PDP 14
- 5.1 DCPN construction and verification 14
- 5.2 Air traffic operations example 14
- 5.3 DCPN model for the air traffic operations example 15
- 5.4 PDP for the air traffic operations example 17

6 Extended power-hierarchy of dependability models 20

7 Conclusions 22

8 References 23

5 Figures

Appendices 25

A Formal definition of Dynamically Coloured Petri Nets 25

B Proof of Theorem 1 30
(1 Figure)

C Proof of Theorem 2 33
(3 Figures)

D Characterisation of Q in terms of DCPN elements 40

E Acronyms and Symbols 45

(48 pages in total)

1 Introduction

Malhotra and Trivedi (1994) and Muppala *et al.* (2000) developed a hierarchy of various dependability models based on their modelling power. This is shown in Figure 1, in which the well-known dependability models Reliability Block Diagrams and Fault Trees are at the basis of the hierarchy. The aim of this report is to extend this power hierarchy such that it includes Piecewise Deterministic Markov Processes (PDP) and Generalised Stochastic Hybrid Processes (GSHP), and PDP and GSHP related Petri Nets (see Bujorianu *et al.* (2003)).

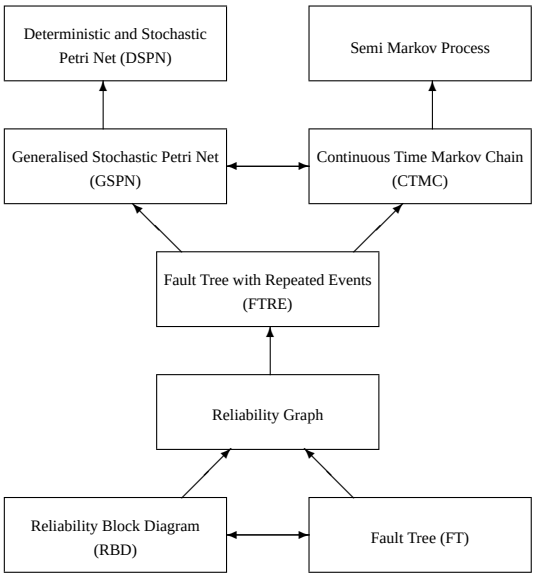

Fig. 1 Power hierarchy among various model types established by (Malhotra and Trivedi, 1994) and (Muppala et al.*, 2000). An arrow from a model to another model indicates that the second model has more modelling power than the first model.*

Davis (1984, 1993) has introduced PDPs as the most general class of continuous-time strong Markov processes which include both discrete and continuous processes, except diffusion. In his 1984 paper, Davis shows that PDP have more modelling power than Semi Markov Processes.

Petri Nets (see David and Alla (1994) for an overview) could provide an important modelling formalism for PDP processes. A Petri Net is a bipartite graph of places (possible conditions or discrete modes) and transitions (possible mode switches). Tokens, which reside in the places, model which conditions or modes are current. Several hybrid state Petri Net extensions have been developed in the past. Main classes are:

- Hybrid Petri Net (Le Bail *et al.*, 1991). Some places have a continuous amount of tokens that may be moved to other places by transitions.

- Fluid Stochastic Petri Net (FSPN) (Trivedi and Kulkarni, 1993). Some places have a continuous amount of tokens, the flow rate of which is influenced by the discrete part. The discrete part of the FSPN can be mapped to a continuous-time Markov chain.
- Extended Coloured Petri Net (ECPN) (Yang *et al.*, 1995). The token colours are real-valued vectors that may follow the solution path of a difference equation.
- High-Level Hybrid Petri Net (HLHPN) (Giua and Usai, 1996). Again, the token colours are real-valued vectors that may follow the solution path of a difference equation, but in addition, a token switch between discrete places may generate a jump in the value of the real-valued vector.
- Differential Petri Nets (Demongodin and Koussoulas, 1998). Differential places have a real-valued number of tokens and differential transitions fire with a certain speed that may also be negative.

For none of the above hybrid state Petri Nets it is clear how they relate to PDP. In order to characterise the exact relation to a PDP, a kind of hybrid state Petri Net is needed that makes direct use of the specific PDP structure. The newly developed Dynamically Coloured Petri Net (DCPN) presented in this paper does this. This makes that into-mappings between PDPs and DCPNs exist. An issue that deserves special attention when relating PDPs to Petri Nets is that for a PDP, at each moment in time, there is a unique realisation of the state, while a Petri Net may make a sequence of jumps at a single moment in time. The into-mappings between PDPs and DCPNs referred to in this paper take care of this issue.

The organisation of this paper is as follows. Section 2 defines Dynamically Coloured Petri Nets. Section 3 explains PDPs. Section 4 shows that DCPN have the same modelling power as PDP. Section 5 gives an example DCPN and an example PDP that model the same simplified air traffic situation. Section 6 shows the extended power-hierarchy of dependability models. Section 7 gives conclusions. The appendices give a formal definition of DCPN and give proofs to theorems posed in the main document.

2 DCPN elements and execution

The elements of a Dynamically Coloured Petri Net (Everdij and Blom, 2000) are given by $DCPN = (\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$, where:

- $\mathcal{P}$ is a set of places.
- $\mathcal{T}$ is a set of transitions which consists of a set $\mathcal{T}_G$ of guard transitions, a set $\mathcal{T}_D$ of delay transitions, and a set $\mathcal{T}_I$ of immediate transitions.
- $\mathcal{A}$ is a finite set of arcs, which consists of a set $\mathcal{A}_O$ of ordinary arcs, a set $\mathcal{A}_E$ of enabling arcs, and a set $\mathcal{A}_I$ of inhibitor arcs.
- $\mathcal{N}$ is a node function which maps each arc to an ordered pair of one transition and one place.
- $\mathcal{S}$ is a set of colour types for the tokens occurring in the net (a colour is the value of an object or process in Petri Net terminology).
- $\mathcal{C}$ is a colour function which maps each place to a colour type in $\mathcal{S}$.
- $\mathcal{I}$ is an initial marking which defines the set of tokens initially present, i.e., it specifies in which places they initially reside, and the colours they initially have.
- $\mathcal{V}$ is a set of place specific colour functions which describe what happens to (i.e. defines the rate of change of) the colour of a token while it resides in a specific place. It determines a token colour differential equation, which is locally Lipschitz continuous.
- $\mathcal{G}$ is a set of boolean-valued transition guards associating each transition in $\mathcal{T}_G$ with a guard function which is evaluated when the transition has a token in each of its input places. The guard function must evaluate to True before the transition is allowed to fire (i.e. remove and produce tokens). Its evaluation depends on the colours of the input tokens of the transition.
- $\mathcal{D}$ is a set of transition delays associating each transition in $\mathcal{T}_D$ with a delay function which is evaluated when the transition has a token in each of its input places. The delay function determines for how long the transition must wait before it is allowed to fire (i.e. remove and produce tokens). The firing rate depends on the colours of the input tokens of the transition.
- $\mathcal{F}$ is a set of (probabilistic) firing functions describing the quantity and colours of the tokens produced by the transitions at their firing. Its evaluation depends on the colours of the input tokens of the transition.

The set of places $\mathcal{P}$, the set of transitions $\mathcal{T}$, the set of arcs $\mathcal{A}$ and the node function $\mathcal{N}$ define a Petri Net graph. Below, the graphical representation of the elements in $\mathcal{P}$, $\mathcal{T}$ and $\mathcal{A}$ are given. The node function $\mathcal{N}$ describes how these elements are connected.

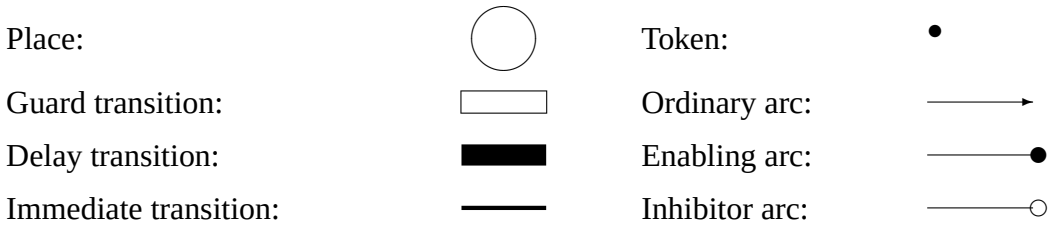

Fig. 2 DCPN graphical elements

Tokens, which reside in the places, and the associated colour values in a DCPN evolve through time quite similar as in a Coloured Stochastic Petri Nets (e.g. Haas, 2002). The main additions are that the colour of a token may evolve according to a differential equation that is governed by the colour function of the specific place where the token resides, and that guard transitions take the evolving colour values into account. More specifically, the DCPN execution rules are described below:

Tokens can be removed from places by the transitions. A transition can only remove tokens if two conditions are both satisfied.

- First condition: The transition has at least one token per ordinary arc and one token per enabling arc in each of its input places and has no token in the input places to which it is connected by an inhibitor arc. When this occurs, the colours of these input tokens, i.e. one token per incoming ordinary arc and one token per incoming enabling arc, are gathered in one big vector, named *vector of input colours*. The value of this vector may evolve with time according to the corresponding token colour functions.
- Second condition: This differs per type of transition. For immediate transitions the second condition is immediately satisfied when the first condition holds. For guard transitions the second condition holds when the vector of input colours enters a transition-specific boundary which is specified by $\mathcal{G}$. For delay transitions, the second condition holds when a particular transition-specific delay has passed, which may depend on the vector of input colours and which is specified by $\mathcal{D}$.
  If the vector of input colours is not unique (for example, if one input place contains several tokens per arc), all possible such vectors are evaluated in parallel and the transition is enabled if one of these vectors satisfies the second condition.

When the first condition holds, the transition is said to be *pre-enabled*. When both conditions hold, the transition is *enabled*. When a transition is enabled, it removes from its input places the tokens that correspond with the vector of input colours that enabled the transition. However, the transition only removes tokens along the ordinary input arcs (one token per arc);

tokens associated with enabling arcs are not removed and stay in their place. Subsequently, the transition produces a token for some or all of its output places, specified by the firing function $\mathcal{F}$. The colour of a produced token (which must be of the correct type, indicated by what $\mathcal{C}$ defines for the output place), and the place for which it is produced is also specified by the firing function $\mathcal{F}$. As for $\mathcal{G}$, $\mathcal{D}$, the evaluation of $\mathcal{F}$ may be dependent on vector of input colours that enabled the transition.

In order to avoid ambiguity, for a DCPN the following priority rules apply when two or more transitions are enabled simultaneously:

R_0 The firing of an immediate transition has priority over the firing of a guard or a delay transition.

R_1 If one transition becomes enabled by two or more disjoint sets of input tokens at exactly the same time, then it will fire these sets of tokens independently, at the same time.

R_2 If one transition becomes enabled by two or more non-disjoint sets of input tokens at exactly the same time, then the set that is fired is selected randomly.

R_3 If two or more transitions become enabled at exactly the same moment by disjoint sets of input tokens, then they will fire at the same time.

R_4 If two or more transitions become enabled at exactly the same moment by non-disjoint sets of input tokens, then the transition that will fire is selected randomly, with the same probability for each transition.

A DCPN as specified by the elements and the execution rules above determines the generation of a stochastic process up to the moment in time that the number of transition firings reaches infinity. This stochastic process is, at each time instant, composed of the ordered colours of all tokens existing in the DCPN, organised in one column vector. The precise ordering of these individual colours within the vector is described as part of the formal definition of DCPN in Appendix A.

Note that three DCPN elements are associated with drawing samples from probability distributions. These elements are the Delay function, the Guard function and the Firing function. In practice, a sample from a general probability distribution is drawn by first drawing a sample from a uniform distribution on the unit interval and then by transforming this uniform sample into a sample from the target distribution. Appendix A explains precisely how these uniform random variables are used during DCPN execution.

3 Piecewise Deterministic Markov Processes

PDP brief explanation

A Piecewise Deterministic Markov Process $\{\xi_t\}$, with $\xi_t = (\theta_t, x_t)$, is defined as follows (see Davis (1993)): For each θ in its countable domain $\mathbf{K}$, let E_θ be an open subset[1] of $\mathbb{R}^{d(\theta)}$, where d is a function that maps $\mathbf{K}$ into $\mathbb{N}$. For each $\theta \in \mathbf{K}$, consider the ordinary differential equation $\dot{x}_t = g_\theta(x_t)$, where $g_\theta : \mathbb{R}^{d(\theta)} \to \mathbb{R}^{d(\theta)}$ is a locally Lipschitz continuous function. Given an initial value $x \in E_\theta$, this differential equation has a unique solution given by the flow $\phi_{\theta,x}$. This means that if at some time instant τ the PDP state assumes value $\xi_\tau = (\theta_\tau, x_\tau)$, then, as long as no jumps occur, the PDP state at $t \geq \tau$ is given by $\xi_t = (\theta_t, x_t) = (\theta_\tau, \phi_{\theta_\tau, x_\tau}(t-\tau))$. At some moment in time, however, the PDP state value may jump. Such moment is generated by either one of the following events, depending on which event occurs first:

1. A Poisson point process with jump rate $\lambda(\theta_t, x_t)$, $t > \tau$ generates a point.
2. The piecewise continuous process x_t is about to hit the boundary ∂E_{θ_τ} of E_{θ_τ}, $t > \tau$.

At the moment when either of these events occurs, the PDP state makes a jump. The value of the PDP state right after the jump is generated by using a transition measure Q, which is the probability measure of the PDP state after the jump, given the value of the PDP state immediately before the jump. After this, the PDP state ξ_t evolves in a similar way from the new value onwards.

PDP execution

The PDP process is generated through time as follows: Suppose at time $\tau_0 \triangleq 0$ the PDP initial state is $\xi_0 = (\theta_0, x_0)$, then, if no jumps occur, the process state at $t \geq \tau_0$ is given by $\xi_t = (\theta_t, x_t) = (\theta_0, \phi_{\theta_0,x_0}(t-\tau_0))$. The complementary distribution function for the time of the first jump (i.e. the probability that the first jump occurs at least $t - \tau_0$ time units after τ_0), also named the survivor function of the first jump, is then given by:

$$G_{\xi_0}(t-\tau_0) \triangleq I_{(t-\tau_0 < t_*(\theta_0,x_0))} \cdot \exp\left\{-\int_{\tau_0}^{t} \lambda(\theta_0, \phi_{\theta_0,x_0}(s-\tau_0))ds\right\}, \tag{1}$$

where I is an indicator function and $t_*(\theta_0, x_0)$ denotes the time until the first boundary hit after $t = \tau_0$, which is given by $t_*(\theta_0, x_0) \triangleq \inf\{t - \tau_0 > 0 \mid \phi_{\theta_0,x_0}(t-\tau_0) \in \partial E_{\theta_0}\}$. The first factor in (1) is explained by the boundary hitting process: after the process state has hit the boundary, which is when $t - \tau_0 = t_*(\theta_0, x_0)$, this first factor ensures that the survivor function evaluates to zero. The second factor in (1) comes from the Poisson process: this second factor ensures

[1] Note that Davis writes $E_\theta = E_\theta^0 \cup \partial_1 E_\theta^0$, with E_θ^0 an open subset of $\mathbb{R}^{d(\theta)}$, and $\partial_1 E_\theta^0$ those points on the boundary of E_θ^0 from which E_θ^0 can be reached (by the flow ϕ), but which cannot be reached from the interior of E_θ^0.

that a jump is generated after an exponentially distributed time with a rate λ that is dependent on the PDP state.
The time τ_1 until the first jump after τ_0 is generated by drawing a sample from $G_{\xi_0}(\cdot)$. In practice, a sample from a general distribution is generated by first drawing a sample from a uniform distribution on $[0,1]$, and then using a transformation (based on the inverse of this general distribution). More formally (see Davis, 1993, Section 23), the Hilbert cube $\Omega = \prod_{i=1}^{\infty} Y_i$, with Y_i a copy of $Y = [0,1]$, provides the canonical space for a countable sequence of independent random variables U_1, U_2, ..., each having uniform $[0,1]$ distribution, defined by $U_i(\omega) = \omega_i$ for elements $\omega = (\omega_1, \omega_2, \ldots) \in \Omega$. Now, define

$$\psi_1(u, \xi_0) = \begin{cases} \inf\{t : G_{\xi_0}(t - \tau_0) \leq u\} \\ +\infty \text{ if the above set is empty} \end{cases}$$

and define $\sigma_1(\omega) = \tau_1(\omega) = \psi_1(U_1(\omega), \xi_0)$, then τ_1 is the time until the first jump.

The value of the hybrid process state to which the jump is made is generated by using the transition measure Q, which is the probability measure of the hybrid state after the jump, given the value of the hybrid state immediately before the jump. The Hilbert cube from above is again used: Let $\psi_2 : [0,1] \times (E \cup \Gamma^*) \to E$, with $E = \cup_\theta E_\theta$ and Γ^* the reachable boundary of E, be a measurable function such that $l\{u : \psi_2(u, \xi) \in B\} = Q(B, \xi)$ for B Borel measurable. Then $\xi_{\tau_1} = \psi_2(U_2(\omega), \xi)$ is a sample from $Q(\cdot, \xi)$.

With this, the algorithm to determine a sample path for the hybrid state process ξ_t, $t \geq 0$, from the initial state $\xi_0 = (\theta_0, x_0)$ on, is in two iterative steps; define $\tau_0 \triangleq 0$ and let for $k = 0$, $\xi_{\tau_k} = (\theta_{\tau_k}, x_{\tau_k})$ be the initial state, then for $k = 1, 2, \ldots$:

Step 1: Draw a sample σ_k from survivor function $G_{\xi_{\tau_{k-1}}}(\cdot)$, i.e. $\sigma_k = \psi_1(U_{2k-1}(\omega), \xi_{\tau_{k-1}})$. Then the time τ_k of the kth jump is $\tau_k = \tau_{k-1} + \sigma_k$. The sample path up to the kth jump is given by

$$\xi_t = (\theta_{\tau_{k-1}}, \phi_{\theta_{\tau_{k-1}}, x_{\tau_{k-1}}}(t - \tau_{k-1})), \quad \tau_{k-1} \leq t < \tau_k \text{ and } \tau_k \leq \infty.$$

Step 2: Draw a multi-dimensional sample ζ_k from transition measure $Q(\cdot; \xi'_{\tau_k})$, where $\xi'_{\tau_k} = (\theta_{\tau_{k-1}}, \phi_{\theta_{\tau_{k-1}}, x_{\tau_{k-1}}}(\tau_k - \tau_{k-1}))$, i.e. $\zeta_k = \psi_2(U_{2k}(\omega), \xi'_{\tau_k})$. Then, if $\tau_k < \infty$, the process state at the time τ_k of the kth jump is given by

$$\xi_{\tau_k} = \zeta_k.$$

PDP conditions

Following Section 24.8 of Davis (1993), the PDP conditions are:

C_1 g_θ is a locally Lipschitz continuous function, which, for each initial state (θ, x), determines a flow $\phi_{\theta,x}(\cdot)$. If $t_\infty(\theta, x)$ denotes the explosion time of the flow $\phi_{\theta,x}(\cdot)$, i.e. $|\phi_{\theta,x}(t)| \to \infty$ as $t \uparrow t_\infty(\theta, x)$, then it is assumed that $t_\infty(\theta, x) = \infty$ whenever $t_*(\theta, x) = \infty$. In other words, explosions are ruled out.

C_2 With $E = \cup_\theta E_\theta$, $\lambda : E \to I\!R^+$ is a measurable function such that for all $\xi \in E$, there is $\epsilon(\xi) > 0$ such that $t \to \lambda(\theta, \phi_{\theta,x}(t))$ is integrable on $[0, \epsilon(\xi)[$.

C_3 With E as above and Γ^* the reachable boundary of E, Q maps $E \cup \Gamma^*$ into the set of probability measures on $(E, \mathcal{E})$, with $\mathcal{E}$ the Borel-measurable subsets of E, while for each fixed $A \in \mathcal{E}$, the map $\xi \to Q(A; \xi)$ is measurable and $Q(\{\xi\}; \xi) = 0$.

C_4 If $N_t = \sum_k I_{(t \geq \tau_k)}$, then it is assumed that for every starting point ξ and for all $t \in I\!R^+$, $I\!E N_t < \infty$. This means, there will be a finite number of jumps in finite time.

4 Into-mappings between DCPN and PDP and between SDCPN and GSHP

An important property of DCPN is that they have similar modelling power as Piecewise Deterministic Markov processes (PDPs). This is made explicit by the two theorems below.

Theorem 1:
For any arbitrary Piecewise Deterministic Markov Process with a finite domain $\mathbf{K}$ there exists P-almost surely a pathwise equivalent process generated by a Dynamically Coloured Petri Net $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ satisfying R_0 through R_4.

Proof: See Appendix B.

Theorem 2:
For each stochastic process generated by a Dynamically Coloured Petri Net $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ satisfying R_0 through R_4 there exists a unique probabilistically equivalent Piecewise Deterministic Markov Process if the following conditions are satisfied:

D_1 There are no explosions, i.e. the time at which a token colour equals $+\infty$ or $-\infty$ approaches infinity whenever the time until the first guard transition enabling moment approaches infinity.

D_2 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

D_3 In a finite time interval, each transition is expected to fire a finite number of times.

D_4 The initial marking is such, that no immediate transition is immediately enabled.

Proof: See Appendix C.

In Bujorianu *et al.* (2003) it has been shown how a PDP is extended to a Generalised Stochastic Hybrid Process (GSHP) by allowing the inclusion of a Brownian motion term to the ordinary differential equation that describes the evolution of the continuous process x_t. In Blom *et al.* (2003) it has been shown that for GSHP the ordinary Markov property and the pathwise existence and uniqueness is well understood. In order to have a Petri Net counterpart of GSHP we allow Brownian motion terms to the place specific colour functions $\mathcal{V}$. To the resulting PN we refer as a Stochastically and Dynamically Coloured Petri Net (SDCPN). The implication is that the mappings between SDCPN and GSHP can be constructed in a similar way as those between DCPN and PDP.

5 Example DCPN and example PDP

This section gives an example DCPN model and an example PDP model of the evolution of an aircraft in one sector of airspace. However, first, we will explain how a DCPN that models a complex operation is generally constructed in several iterations.

5.1 DCPN construction and verification

A DCPN modelling a particular operation can be constructed, for example, by first identifying the discrete state space, represented by the places, the transitions and arcs, and next adding the continuous-time-based elements one by one, similar as what one would expect when modelling a PDP for such operation. However, in case of a very complex operation, with many entities that interact such as occur in air traffic, it is generally more desirable and constructive to do the DCPN modelling in several iterations, for example in a three-phased approach:

1. In the first phase, each operation entity or agent (for example, a pilot, a navigation system, an aircraft) is modelled separately by one local DCPN. Each such entity model is named a Local Petri Net (LPN).
2. In the second phase, the interactions between these entities are modelled, connecting the LPNs.
3. In the third phase, one verifies at local and global levels whether all elements of the operation have been properly modelled. If there are elements or interactions missing, a new iteration is started. In this phase, one also checks whether the conditions $D_1 - D_4$ under which a mapping to PDP is guaranteed have been fulfilled.

The advantage of such phased approach is that the LPNs can be verified separately by respective experts, without them bothering about interactions in first instance. For example, an LPN model for a navigation system can be verified by a navigational system expert; an LPN model for a pilot can be verified by a human factors expert.

5.2 Air traffic operations example

This subsection presents a very simplified representation of the evolution of an aircraft in one sector of airspace. The next subsection presents a DCPN model for this example.

Assume the deviation of this aircraft from its intended path depends on the operationality 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* (functioning properly) or *Not working* (operating in some failure mode). Both systems switch between their modes independently and on exponentially distributed times, with rates δ_3 (engine repaired), δ_4 (engine fails), δ_5 (navigation repaired) and δ_6 (navigation fails), respectively. The operationality of these

systems has the following effect on the aircraft path: if both systems are *Working*, the rate of change of the position and velocity of the aircraft is given by function $\mathcal{V}_1$ (i.e. if z_t is a vector containing this position and velocity then $\dot{z}_t = \mathcal{V}_1(z_t)$). If either one, or both, of the systems is *Not working*, the rate of change of the position and velocity of the aircraft is given by $\mathcal{V}_2$. Initially, the aircraft has a particular position x_0 and velocity v_0, while both its systems are *Working*. The evaluation of this process may be stopped when the aircraft position crosses the boundary ∂G to a neighbouring airspace sector.

5.3 DCPN model for the air traffic operations example

This subsection gives a DCPN instantiation that models the air traffic operation of the previous subsection. In order to illustrate the three-phased approach of Subsection 5.1, we first give the Local Petri Net graphs that have been identified in the first phase of the modelling. The air traffic entities identified are: Aircraft evolution, Navigation system, and Engine system. This gives us three LPNs. The resulting graphs are given in the figure below.

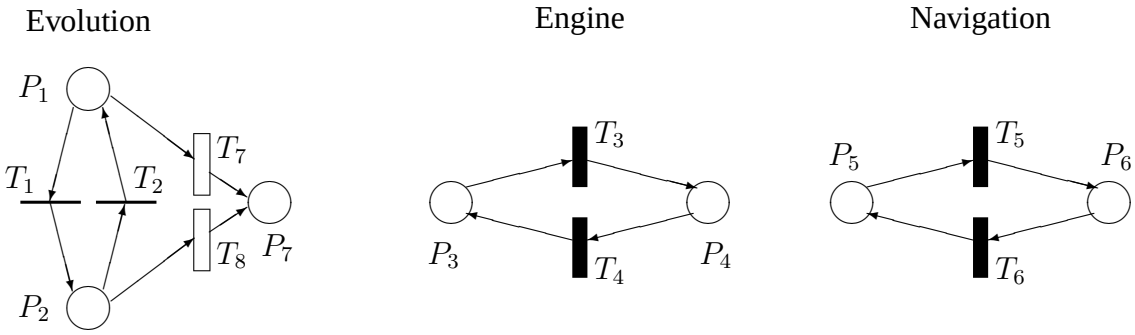

Fig. 3 Local Petri Net graphs for the aircraft operations example. Place P_1 models Evolution Nominal, P_2 models Evolution Non-nominal, P_3 models Engine system Not working, P_4 models Engine system Working, P_5 models Navigation system Not working, P_6 models Navigation system Working.

The interactions between the Engine and Navigation LPN and the Evolution LPN (i.e. execution of the second phase of DCPN instantiation) are modelled by coupling the LPNs by additional arcs (and, if necessary, additional places or transitions). Here, removal of a token from one LPN by a transition of another LPN is prevented by using enabling arcs instead of ordinary arcs for the interactions. The resulting graph is presented below. Notice that transition T_1 is replaced by two transitions T_{1a} and T_{1b}.

The graph above completely defines DCPN 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 DCPN elements are specified below.

$\mathcal{S}$: One colour type is defined; $\mathcal{S} = \{\mathbb{R}^6\}$.

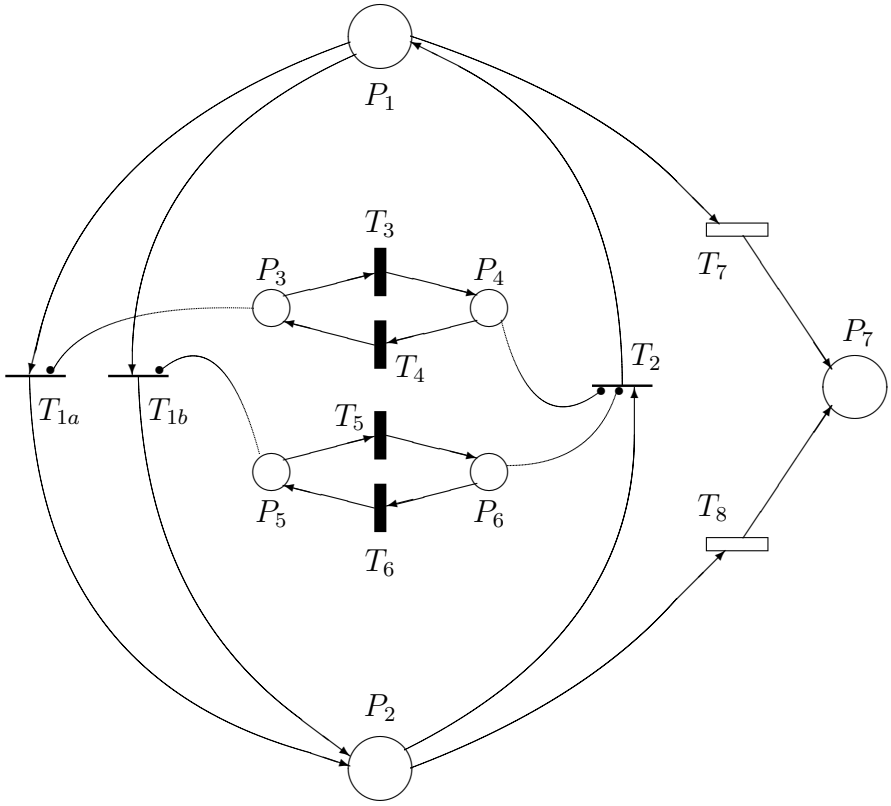

Fig. 4 Local Petri Net graphs integrated into one graph.

- $\mathcal{C}$: $\mathcal{C}(P_1) = \mathcal{C}(P_2) = \mathcal{C}(P_7) = I\!R^6$. The first three colour components model the longitudinal, lateral and vertical position of the aircraft, the last three components model the corresponding velocities. For places P_3 through P_6, no colour type needs to be defined (one might define a dummy colour type for these places, but this is not pursued further).
- $\mathcal{I}$: Place P_1 initially has a token with colour $z_0 = (x_0, v_0)' \in I\!R^6$. Places P_4 and P_6 initially each have a token with no colour (or a dummy colour).
- $\mathcal{V}$: The token colour functions for places P_1, P_2 and P_7 are defined by $\mathcal{V}_{P_1} = \mathcal{V}_1$, $\mathcal{V}_{P_2} = \mathcal{V}_2$ and $\mathcal{V}_{P_7} = 0$. For places P_3 – P_6 the token colour function is not defined (one could define a dummy function).
- $\mathcal{G}$: Transitions T_7 and T_8 have a guard that is defined by $\partial G_{T_7} = \partial G_{T_8} = \partial G \times I\!R^3$.
- $\mathcal{D}$: The jump rates for transitions T_3, T_4, T_5 and T_6 are $\delta_{T_3}(\cdot) = \delta_3$, $\delta_{T_4}(\cdot) = \delta_4$, $\delta_{T_5}(\cdot) = \delta_5$ and $\delta_{T_6}(\cdot) = \delta_6$, 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, $\mathcal{F}_T(1, \cdot; \cdot) = 1$.

As an illustration of the third phase of the DCPN instantiation development process, we check whether the conditions D_1 – D_4 have been fulfilled.

D_1 (i.e. there are no explosions: the time at which a token colour equals $+\infty$ or $-\infty$ approaches infinity whenever the time until the first guard transition enabling moment approaches infinity) is fulfilled: Since the aircraft will always leave the sector within finite time, the first guard transition will always fire within a finite time.

D_2 (i.e. 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) is also fulfilled since after each transition firing, tokens are consumed and produced for other places.

D_3 (i.e. in a finite time interval, each transition is expected to fire a finite number of times) is also fulfilled, since there are no loops connecting transitions directly back to themselves.

D_4 (i.e. the initial marking is such, that no immediate transition is immediately enabled) is also fulfilled. The set of immediate transitions is $\mathcal{T}_I = \{T_{1a}, T_{1b}, T_2\}$. T_{1a} is enabled by tokens in places P_1 and P_3, T_{1b} is enabled by tokens in places P_1 and P_5,T_2 is enabled by tokens in places P_2, P_4 and P_6. Since initially, only places P_1, P_4 and P_6 contain a token, none of these immediate transitions is immediately enabled.

Note that for more complex DCPN, the checking of these conditions will occur in two phases: first, the conditions are checked for each Local Petri Net separately; next, the interactions are checked.

5.4 PDP for the air traffic operations example

This section presents a PDP that describes the same process as modelled by a DCPN in the previous subsection, i.e. the path of an aircraft influenced by its engine and its navigation system.

For this example, the PDP mode process $\{\theta_t\}$ has three components: $\theta_t = (\theta_t^1, \theta_t^2, \theta_t^3)$', where:

θ_t^1 is the Engine system mode, taking values in {*Working, Not working*}.
θ_t^2 is the Navigation system mode, taking values in {*Working, Not working*}.
θ_t^3 is the Aircraft mission mode, taking values in {*Not completed, Completed*}.

This yields that the set $\mathbf{K}$ has $2^3 = 8$ elements $m_1, \ldots, m_8$ with:

m_1=(*Working,Working,Not completed*)
m_2=(*Not working,Working,Not completed*)
m_3=(*Not working,Not working,Not completed*)
m_4=(*Working,Not working,Not completed*)
m_5=(*Working,Working,Completed*)

m_6=(*Not working,Working,Completed*)
m_7=(*Not working,Not working,Completed*)
m_8=(*Working,Not working,Completed*).

The initial mode equals $\theta_0 = m_1$. For $\theta \in \{m_1, m_2, m_3, m_4\}$, $\partial E_\theta = \partial G \times I\!R^3$, while for $\theta \in \{m_5, m_6, m_7, m_8\}$, E_θ equals $I\!R^6$. The piecewise continuous process part $\{z_t\}$ has two components: $z_t = (x_t, v_t)$', with x_t the position and v_t the velocity of the aircraft. The first table below gives, for each $\theta \in \mathbf{K}$, the locally Lipschitz continuous function $g_\theta(\cdot)$ and the jump rates λ of the Poisson point process. In the second table below, $Q(\zeta; \xi) = p$ denotes that if ξ is the value of the PDP before the hybrid jump, then, with probability p, ζ is the value of the PDP immediately after the jump.

Table I: Example PDP components $g_\theta(\cdot)$ and λ as a function of θ

θ	$g_\theta(\cdot)$	λ
m_1	$\mathcal{V}_1(\cdot)$	$\delta_4 + \delta_6$
m_2	$\mathcal{V}_2(\cdot)$	$\delta_3 + \delta_6$
m_3	$\mathcal{V}_2(\cdot)$	$\delta_3 + \delta_5$
m_4	$\mathcal{V}_2(\cdot)$	$\delta_4 + \delta_5$
m_5	0	$\delta_4 + \delta_6$
m_6	0	$\delta_3 + \delta_6$
m_7	0	$\delta_3 + \delta_5$
m_8	0	$\delta_4 + \delta_5$

Table II: Example PDP component Q

For $z \notin \partial E_{m_1}$: $Q(m_2, z; m_1, z) = \frac{\delta_4}{\delta_4+\delta_6}$, $Q(m_4, z; m_1, z) = \frac{\delta_6}{\delta_4+\delta_6}$.
For $z \in \partial E_{m_1}$: $Q(m_5, z; m_1, z) = 1$.
For $z \notin \partial E_{m_2}$: $Q(m_3, z; m_2, z) = \frac{\delta_6}{\delta_3+\delta_6}$, $Q(m_1, z; m_2, z) = \frac{\delta_3}{\delta_3+\delta_6}$.
For $z \in \partial E_{m_2}$: $Q(m_6, z; m_2, z) = 1$.
For $z \notin \partial E_{m_3}$: $Q(m_4, z; m_3, z) = \frac{\delta_3}{\delta_3+\delta_5}$, $Q(m_2, z; m_3, z) = \frac{\delta_5}{\delta_3+\delta_5}$.
For $z \in \partial E_{m_3}$: $Q(m_7, z; m_3, z) = 1$.
For $z \notin \partial E_{m_4}$: $Q(m_3, z; m_4, z) = \frac{\delta_4}{\delta_4+\delta_5}$, $Q(m_1, z; m_4, z) = \frac{\delta_5}{\delta_4+\delta_5}$.
For $z \in \partial E_{m_4}$: $Q(m_8, z; m_4, z) = 1$.
For all z, $Q(m_6, z; m_5, z) = \frac{\delta_4}{\delta_4+\delta_6}$, $Q(m_8, z; m_5, z) = \frac{\delta_6}{\delta_4+\delta_6}$.
For all z, $Q(m_7, z; m_6, z) = \frac{\delta_6}{\delta_3+\delta_6}$, $Q(m_5, z; m_6, z) = \frac{\delta_3}{\delta_3+\delta_6}$.
For all z, $Q(m_8, z; m_7, z) = \frac{\delta_3}{\delta_3+\delta_5}$, $Q(m_6, z; m_7, z) = \frac{\delta_5}{\delta_3+\delta_5}$.
For all z, $Q(m_7, z; m_8, z) = \frac{\delta_4}{\delta_4+\delta_5}$, $Q(m_5, z; m_8, z) = \frac{\delta_5}{\delta_4+\delta_5}$.

Since the PDP is of purely mathematical nature, it is less simple to comprehend and verify by non-mathematicians than the DCPN representing the same system in the previous subsection.

6 Extended power-hierarchy of dependability models

In order to allow drawing the extended power-hierarchy, it remains to be shown that DCPN have more modelling power than the DSPN (Deterministic and Stochastic Petri Nets) at the top of the Muppala *et al.* (2000) based power hierarchy of Figure 1.

The existence of an arrow from DSPN to DCPN can be shown as follows: GSPN (Generalised Stochastic Petri Nets) are generalisations of Stochastic Petri Nets allowing transitions to have either zero firing times (immediate transitions) or exponentially distributed firing times (timed transitions). Immediate transitions which can be simultaneously enabled must have probabilities assigned. For timed transitions, the decision as to which transition fires next is decided by race; the transition with the minimal delay prior to firing will fire next. Firing of immediate transitions has priority over firing of timed transitions. Other extensions include inhibitor arcs.

A DSPN is a GSPN in which the firing delays of timed transitions may be either constant or exponential. Through the equivalence of GSPN and CTMC (Continuous Time Markov Chain) it can be easily shown that any GSPN can be written as a DCPN: Such DCPN will have constant exponential delay rates and constant colours. The extension to DSPN can also be covered by a DCPN: For each DSPN transition with a constant firing time, create a DCPN transition with a guard function that evaluates to True when the input token colour equals the DSPN transition's constant firing time plus the colour of the input token at the time the transition is pre-enabled. This input token colour has a token colour function equal to +1, and an initial colour equal to zero.

Together with the findings of Section 4 we get the power hierarchy of Figure 5.

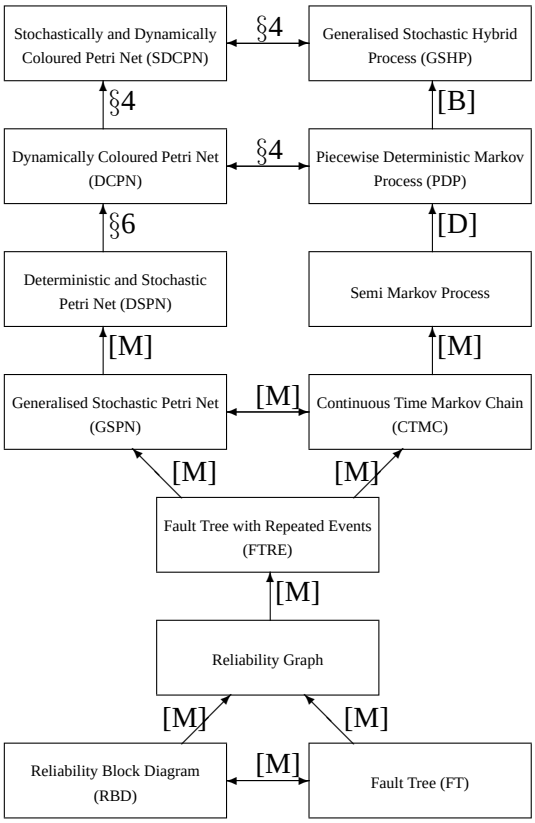

Fig. 5 Power hierarchy among various model types. An arrow from a model to another model indicates that the second model has more modelling power than the first model. Arrows labelled by [M] have been explained by (Malhotra and Trivedi, 1994) and (Muppala et al.*, 2000). The arrow from DSPN to DCPN is established in this Section (and is labelled by §6). The arrow labelled by [D] is established by Davis (1984). The arrow from PDP to GSHP, labelled by [B], was shown in Bujorianu* et al. *(2003). The arrows between DCPN and PDP and between SDCPN and GSHP (labelled by §4) are established in Section 4.*

7 Conclusions

This paper extended the power hierarchy of dependability models developed by Malhotra and Trivedi (1994) and Muppala *et al.* (2000) to include Piecewise Deterministic Markov Processes (PDP), Generalised Stochastic Hybrid Processes (GSHP), Dynamically Coloured Petri Nets (DCPN) and Stochastically and Dynamically Coloured Petri Net (SDCPN). The report explained the existence of into-mappings between PDP and DCPN and between GSHP and SDCPN, yielding that they have similar modelling power, and has shown that DCPN have more modelling power than Deterministic and Stochastic Petri Nets (DSPN).

PDPs are known as the largest class of continuous-time Markov processes not involving diffusions. Dynamically Coloured Petri Nets are defined to make ample use of these PDP properties and have shown to be very useful in developing PDP models for complex practical problems. This usefulness has been explicitly used for accident risk assessment modelling application to Air Traffic Management (e.g. Blom *et al.*, 2001).

8 References

Bujorianu, M.L., Lygeros, J., Glover, W., Pola, G. (2003). A stochastic hybrid system modelling framework, Hybridge report D1.2, 29th May 2003, Univ. of Cambridge and Univ. of L'Aquila.

Blom, H.A.P., Bakker, G.J., Blanker, P.J.G., Daams, J., Everdij, M.H.C., Klompstra, M.B. (2001). Accident risk assessment for advanced ATM, In: *Air Transportation Systems Engineering*, AIAA, Eds. G.L. Donohue, A.G. Zellweger, AIAA, pp. 463-480.

Blom, H.A.P., Bakker, G.J., Everdij, M.H.C, Park, M.N.J. van der (2003). Stochastic Analysis background of accident risk assessment for Air Traffic Mangement, Hybridge report D2.2, 31th March 2003, NLR.

Champagnat, R., Esteban, P., Pingard, H., Valette, R. (1998), Modeling and simulation of a hybrid system through PR/TR PN-DAE model, *Proc. of the 3rd Int. Conf. on Automation of Mixed Processes*, Reims, France, March 1998, pp. 131-137.

Cassandras, C.G., Lafortune, S. (1999), Introduction to Discrete Event Systems. Kluwer Academic Publishers.

David, R., Alla, H. (1994). Petri Nets for the modeling of dynamic systems — A survey, *Automatica*, Vol. 30, No. 2, pp. 175-202.

Davis, M.H.A. (1984). Piecewise Deterministic Markov Processes: a general class of non-diffusion stochastic models, *Journal Royal Statistical Soc. (B)*, Vol. 46, pp. 353-388.

Davis, M.H.A. (1993). Markov models and optimization, Chapman & Hall.

Demongodin, I., Koussoulas, N.T. (1998). Differential Petri Nets: Representing continuous systems in a discrete-event world, *IEEE Transactions on Automatic Control*, Vol. 43, No. 4.

Everdij, M.H.C., Blom, H.A.P., Klompstra, M.B. (1997). Dynamically Coloured Petri Nets for Air Traffic Management safety purposes, *Proc. 8th IFAC Symposium on Transportation Systems, Chania, Greece*, pp. 184-189.

Everdij, M.H.C., Blom, H.A.P. (2000). Piecewise Deterministic Markov Processes represented by Dynamically Coloured Petri Nets, submitted to Stochastics, 2000, published in February 2005.

Everdij, M.H.C., Blom, H.A.P. (2003). Petri-Nets and Hybrid-State Markov Processes in a Power-Hierarchy of Dependability Models, *Proc. IFAC Conference on Analysis and Design of Hybrid Systems*, Saint-Malo, Brittany, France, 16-18 June 2003, pp. 355-360.

Giua, A., Usai, E. (1996). High-level Hybrid Petri Nets: a definition, *Proc. 35th Conference on Decision and Control, Kobe, Japan*, pp. 148-150.

Haas, P.J. (2002). Stochastic Petri Nets, Modelling, Stability, Simulation, Springer-Verlag, New York.

Jensen, K. (1992). Coloured Petri Nets: Basic concepts, analysis methods and practical use, Volume 1, Springer-Verlag.

Le Bail, J., Alla, H., David, R. (1991). Hybrid Petri Nets, *European Control Conference, Grenoble, France*, pp. 1472-1477.

Malhotra, M., Trivedi, K.S. (1994). Power-hierarchy of dependability-model types, *IEEE Transactions on Reliability*, Vol. R-43, No. 3, pp. 493-502.

Muppala, J.K., Fricks, R.M., Trivedi, K.S. (2000). Techniques for system dependability evaluation, *In: Computational probability*, W. Grasman (ed.), pp 445-480, Kluwer Academix Publishers, The Netherlands.

Trivedi, K.S., Kulkarni, V.G. (1993). FSPNs: Fluid Stochastic Petri Nets, *Lecture notes in Computer Science*, Vol. 691, M. Ajmone Marsan (ed.) *Proc. 14th Int. Conference on Applications and theory of Petri Nets*, pp. 24-31, Springer Verlag, Heidelberg.

Yang, Y.Y., Linkens, D.A., Banks, S.P. (1995). Modelling of hybrid systems based on Extended Coloured Petri Nets, *Hybrid Systems II*, P. Antsaklis *et al.* (eds.), pp. 509-528, Springer.

Appendices

A Formal definition of Dynamically Coloured Petri Nets

This appendix presents a formal definition of Dynamically Coloured Petri Net. As much as possible, the notation introduced by Jensen (1992) for Coloured Petri Net is used.

Definition:
A Dynamically Coloured Petri Net (DCPN) is an 11-tuple *DCPN* = ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{V}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$, $\mathcal{I}$), together with some rules. Below, first the structure of the components in the tuple is given, next the DCPN evolution through time is explained, finally, the DCPN generated process is outlined.

DCPN elements:

1. $\mathcal{P}$ is a finite set of places.
2. $\mathcal{T}$ 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 $\mathcal{T}_G \cap \mathcal{T}_D = \mathcal{T}_D \cap \mathcal{T}_I = \mathcal{T}_I \cap \mathcal{T}_G = \emptyset$.
3. $\mathcal{A}$ 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 $\mathcal{A}_O \cap \mathcal{A}_E = \mathcal{A}_E \cap \mathcal{A}_I = \mathcal{A}_I \cap \mathcal{A}_O = \emptyset$.
4. $\mathcal{N} : \mathcal{A} \rightarrow \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$ is a node function which maps each arc A in $\mathcal{A}$ to a pair of ordered nodes $\mathcal{N}(A)$. 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:
   - $A(T) = \{A \in \mathcal{A} \mid T(A) = T\}$ denotes the set of arcs connected to transition T, with $A(T) = A_{in}(T) \cup A_{out}(T)$, where
   - $A_{in}(T) = \{A \in A(T) \mid \mathcal{N}(A) = (P(A), T)\}$ is the set of input arcs of T and
   - $A_{out}(T) = \{A \in A(T) \mid \mathcal{N}(A) = (T, P(A))\}$ is the set of output arcs of T.

   Moreover,
   - $A_{in,O}(T) = A_{in}(T) \cap \mathcal{A}_O$ 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
   - $P(A(T))$ is the set of places connected to T by the set of arcs $A(T)$.

   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.

5. $\mathcal{S}$ is a finite set of colour types. Each colour type is to be written in the form $\mathbb{R}^n$, with n a natural number.
6. $\mathcal{C} : \mathcal{P} \to \mathcal{S}$ is a colour function which maps each place $P \in \mathcal{P}$ to a specific colour type in $\mathcal{S}$.
7. $\mathcal{I} : \mathcal{P} \to \mathcal{C}(\mathcal{P})_{ms}$ is an initialisation function, where $\mathcal{C}(P)_{ms}$ for $P \in \mathcal{P}$ denotes the set of all multisets over $\mathcal{C}(P)$. It defines the initial marking of the net, i.e., for each place it specifies the number of tokens (possibly zero) initially in it, together with the colours they have, and their ordering per place.
8. $\mathcal{V}$ is set of a token colour functions. For each place $P \in \mathcal{P}$ it contains a locally Lipschitz continuous function $\mathcal{V}_P : \mathcal{C}(P) \to \mathcal{C}(P)$.
9. $\mathcal{G}$ is a set of transition guards. For each $T \in \mathcal{T}_G$, it contains a transition guard $\mathcal{G}_T : \mathcal{C}(P(A_{in,OE}(T))) \to \{\text{True, False}\}$. $\mathcal{G}_T(c_t)$ evaluates to True when c_t enters ∂G_T for the first time, where G_T is an open subset in $\mathcal{C}(P(A_{in,OE}(T)))$.
   Here, if $P(A_{in,OE}(T))$ contains more than one place, e.g., $P(A_{in,OE}(T)) = \{P_i, \ldots, P_j\}$, then $\mathcal{C}(P(A_{in,OE}(T)))$ is defined by $\mathcal{C}(P_i) \times \cdots \times \mathcal{C}(P_j)$.
10. $\mathcal{D}$ is a set of transition delays. For each $T \in \mathcal{T}_D$, it contains a transition delay $\mathcal{D}_T : \mathcal{C}(P(A_{in,OE}(T))) \to \mathbb{R}_0^+$, which, if evaluated from stopping time τ on, follows $\mathcal{D}_T(c_t) = \inf\{t \mid e^{-\int_\tau^t \delta_T(c_s)ds} \leq u\}$, where $\delta_T : \mathcal{C}(P(A_{in,OE}(T))) \to \mathbb{R}_0^+$ is integrable and u is a random number drawn from $U[0,1]$ at τ.
11. $\mathcal{F}$ is a set of firing measures. For each $T \in \mathcal{T}$ it specifies a probability measure $\mathcal{F}_T$ which maps $\mathcal{C}(P(A_{in,OE}(T)))$ into the set of probability measures on $\{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))$.

DCPN execution:

The execution of a DCPN provides a series of increasing stopping times, $\tau_0 < \tau_i < \tau_{i+1}$, 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. This number of tokens and the colours of these tokens are generated as follows:

Each token residing in place P has a colour of type $\mathcal{C}(P)$. If a token in place P has colour c at time τ, and if it remains in that place up to time $t > \tau$, then the colour c_t at time t equals the unique solution of the differential equation $\dot{c}_t = \mathcal{V}_P(c_t)$ with initial condition $c_\tau = c$.

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; denote $\tau_1^{pre} = \inf\{t \mid T \text{ is pre-enabled at time } t\}$. Consider one token per ordinary and enabling arc in the input places of T and write $c_t \in \mathcal{C}(P(A_{in,OE}(T)))$, $t \geq \tau_1$, as the column

vector containing the colours of these tokens; c_t may change through time according to its corresponding token colour functions. If this vector is not unique (for example, one input place contains several tokens per arc), all possible such vectors are executed in parallel.

A transition T is enabled if it is pre-enabled and a second condition holds true. For $T \in \mathcal{T}_I$, the second condition automatically holds true. For $T \in \mathcal{T}_G$, the second condition holds true when $\mathcal{G}_T(c_t)$ = True. For $T \in \mathcal{T}_I$, the second condition holds true $\mathcal{D}_T(c_t)$ units after τ_1^{pre}. Guard or delay evaluation of a transition T stops when T is not pre-enabled anymore, and is restarted when it is.

For the evaluation of $\mathcal{G}_T(c_t)$ and $\mathcal{D}_T(c_t)$, use is made of a Hilbert cube $\Omega = \prod_{i=1}^{\infty} Y_i$, with Y_i a copy of $Y = [0, 1]$, which provides the canonical space for a countable sequence of independent random variables U_1, U_2, ..., each having a uniform $[0, 1]$ distribution, defined by $U_i(\omega) = \omega_i$ for elements $\omega = (\omega_1, \omega_2, \ldots) \in \Omega$. This Hilbert cube is used as follows: Suppose T is a transition that is pre-enabled at time τ and has vector of input colours c_t at time $t \geq \tau$, with $c_\tau = c$.

- If T is a delay transition then consider the survivor function $H_c^{\mathcal{D}_T}(t-\tau) \underline{\triangle} \exp\left\{-\int_\tau^t \delta_T(c_s)ds\right\}$ and define the function $\psi_1^T(u, c)$ by

$$\psi_1^T(u,c) \underline{\triangle} \begin{cases} \inf\{t : H_c^{\mathcal{D}_T}(t-\tau) \leq u\} \\ +\infty \text{ if the above set is empty} \end{cases}$$

  then transition T is enabled at time $\psi_1^T(U_1(\omega), c)$.

- If T is a guard transition then consider the survivor function $H_c^{\mathcal{G}_T}(t-\tau) \underline{\triangle} I_{(t-\tau<t_*(c))}$, where I is an indicator function and $t_*(c)$ denotes the time until the first boundary hit after $t = \tau$, which is given by $t_*(c) \underline{\triangle} \inf\{t - \tau > 0 | c_t \in \partial G_T\}$. Next, define the function $\psi_1^T(u, c)$ by

$$\psi_1^T(u,c) \underline{\triangle} \begin{cases} \inf\{t : H_c^{\mathcal{G}_T}(t-\tau) \leq u\} \\ +\infty \text{ if the above set is empty} \end{cases}$$

  then transition T is enabled at time $\psi_1^T(U_1(\omega), c)$.

In case of ambiguities, the following rules apply:

R_0 The firing of an immediate transition has priority over the firing of a guard or a delay transition.

R_1 If one transition becomes enabled by two or more disjoint sets of input tokens at exactly the same time, then it will fire these sets of tokens independently, at the same time.

R_2 If one transition becomes enabled by two or more non-disjoint sets of input tokens at exactly the same time, then the set that is fired is selected randomly.

R_3 If two or more transitions become enabled at exactly the same time by disjoint sets of input tokens, then they will fire at the same time.

R_4 If two or more transitions become enabled at exactly the same time by non-disjoint sets of input tokens, then the transition that will fire is selected randomly.

Here, two sets of input tokens are disjoint if they have no tokens in common that are reserved by ordinary arcs, i.e., they may have tokens in common that are reserved by enabling arcs.

If T is enabled, suppose this occurs at time τ_1, it removes one token per arc in $A_{in,O}(T)$ from each of its input places. At this time τ_1, T produces zero or one token along each output arc: If c_{τ_1} is the vector of colours of tokens that enabled T and (f, a_{τ_1}) is a sample from $\mathcal{F}_T(\cdot; c_{\tau_1})$, then vector f specifies along which of the output arcs of T a token is produced (f holds a one at the corresponding vector components and a zero at the arcs along which no token is produced) and a_{τ_1} specifies the colours of the produced tokens. The colours of the new tokens have sample paths that start at time τ_1.

For drawing the sample from $\mathcal{F}_T(\cdot; c_{\tau_1})$, use is again made of the Hilbert cube Ω: Let $\psi_2^T : [0,1] \times \mathcal{C}(P(A_{in,OE}(T))) \to \{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))$ be a measurable function such that $l\{u : \psi_2^T(u,c) \in B\} = \mathcal{F}_T(B,c)$ for B in the Borel set of $\{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))$. Then a sample from $\mathcal{F}_T(\cdot; c_{\tau_1})$ is given by $\psi_2^T(U_2(\omega), c_{\tau_1})$, if c_{τ_1} is the vector of input colours that enabled T.

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 function produces zero or one token along each output arc).

DCPN stochastic process:

The DCPN generates a stochastic process which is uniquely defined as follows: The process state at time t is defined by the numbers of tokens in each place, and the colours of these tokens. Provided there is a unique ordering of DCPN places, and a unique ordering of tokens within a place, this characterisation is unique, except at time instants when one or more transitions fire. To make this characterisation of DCPN process state unique, it is defined as follows:

- At times t when no transition fires, the number of tokens in each place is uniquely

characterised by the vector $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ of length $|\mathcal{P}|$, where $v_{i,t}$ 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 time instants when one or more transitions fire, uniqueness of $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ is assured as follows: Suppose that τ is such time instant at which one transition or a sequence of transitions fires. Next, assume without loss of generality, that this sequence of transitions is $\{T_1, T_2, \ldots, T_m\}$ and that time is running again after T_m (note that T_1 must be a guard or a delay transition, and T_2 through T_m must be immediate transitions). Then the number of tokens in each place at time t is defined as that vector $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ that occurs after T_m has fired. This construction also ensures that the process $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ has limits from the left and is continuous from the right, i.e., it satisfies the càdlàg property.

- If $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ is the distribution of the tokens among the places of the DCPN at time t, which is uniquely defined above, then the associated colours of these tokens are uniquely gathered in a vector as follows: This vector first contains all colours of tokens in place P_1, next 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). Since $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ satisfies the càdlàg property, the corresponding vector of token colours does too. An additional case occurs, however, when $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ jumps to the same value again, so that only the process associated with the vector of token colours makes a jump at time τ. In that case, let the process associated with the vector of token colours be defined according to the timing construction as described for $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ above (i.e. at time τ, the process associated with the vector of token colours is defined as that vector of token colours that occurs after the last transition has fired in the sequence of transitions that fire at time τ).

With this, the DCPN definition is complete.

B Proof of Theorem 1

This appendix shows that for an arbitrary Piecewise Deterministic Markov Process there exists a Dynamically Coloured Petri Net, the execution of which generates a stochastic process indistinguishable from the PDP, by providing an appropriate into-mapping from PDP into the set of DCPNs. Notice that we do not claim the into-mapping to be unique; there may be other DCPN instantiations describing the same PDP.

Consider an arbitrary PDP $\{x_t, \theta_t\}$ described by the PDP elements $\{\mathbf{K}, d(\theta), x_0, \theta_0, \partial E_\theta, g_\theta, \lambda, Q\}$.

First, the DCPN elements $\{\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F}\}$ and the rules R_0 – R_4 are characterised in terms of the PDP elements $\{\mathbf{K}, d(\theta), x_0, \theta_0, \partial E_\theta, g_\theta, \lambda, Q\}$ as follows:

$\mathcal{P} = \{P_\theta; \theta \in \mathbf{K}\}$.

$\mathcal{T} = \mathcal{T}_G \cup \mathcal{T}_D \cup \mathcal{T}_I$, with $\mathcal{T}_I = \emptyset$, $\mathcal{T}_G = \{\mathcal{T}_\theta^G; \theta \in \mathbf{K}\}$, $\mathcal{T}_D = \{\mathcal{T}_\theta^D; \theta \in \mathbf{K}\}$.

$\mathcal{A} = \mathcal{A}_O \cup \mathcal{A}_E \cup \mathcal{A}_I$, with $|\mathcal{A}_I| = 0$, $|\mathcal{A}_E| = 0$, and $|\mathcal{A}_O| = 2|\mathbf{K}| + 2|\mathbf{K}|^2$.

$\mathcal{N}$**:** The node function maps each arc in $\mathcal{A} = \mathcal{A}_O$ to a pair of nodes. These connected pairs of nodes are: $\{(P_\theta, T_\theta^G); \theta \in \mathbf{K}\} \cup \{(P_\theta, T_\theta^D); \theta \in \mathbf{K}\} \cup$
$\{(T_\theta^G, P_\vartheta); \theta, \vartheta \in \mathbf{K}\} \cup (T_\theta^D, P_\vartheta); \theta, \vartheta \in \mathbf{K}\}$.

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

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

$\mathcal{I}$**:** Place P_{θ_0} contains one token with colour x_0. All other places initially contain zero tokens.

$\mathcal{V}$**:** For all $\theta \in \mathbf{K}$, $\mathcal{V}_{P_\theta}(\cdot) = g_\theta(\cdot)$.

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

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

$\mathcal{F}$**:** If x denotes the colour of the token removed from place P_θ, $(\theta \in \mathbf{K})$, at the transition firing, then for all $\vartheta' \in \mathbf{K}$, $x' \in E_{\vartheta'}$: $\mathcal{F}_{T_\theta^G}(e', x'; x) = Q(\vartheta', x'; \theta, x)$, where e' is the vector of length $|\mathbf{K}|$ containing a one at the component corresponding with arc $(T_\theta^G, P_{\vartheta'})$ and zeros elsewhere. For all $\theta \in \mathbf{K}$, $\mathcal{F}_{T_\theta^D} = \mathcal{F}_{T_\theta^G}$.

R_0 **–** R_4**:** Since there are no immediate transitions in the constructed DCPN instantiation, rule R_0 holds true. Since there is only one token in the constructed DCPN instantiation, R_1 – R_3 also hold true. Rule R_4 is in effect when for particular θ, transitions T_θ^G and T_θ^D become enabled at exactly the same time. Since λ is integrable, the probability that this occurs is zero, yielding that R_4 holds with probability one. However, if this event should occur, then due to the fact that the firing measures for the guard transition and the delay transition are equal, rule R_4 holds true.

This shows that for any PDP we are able to construct a DCPN instantiation. Next, we have to show that the DCPN execution delivers the 'same' cadlag stochastic process as the PDP process.

In the DCPN instantiation constructed only one token resides. The possible places for this token are $\{P_\vartheta; \vartheta \in \mathbf{K}\}$. Figure 6 shows the situation at some time τ_{k-1}, when the PDP is given by $(\theta_{\tau_{k-1}}, x_{\tau_{k-1}})$. The token resides in place P_{ϑ_i}, which models that $\theta_{\tau_{k-1}} = \vartheta_i$. This token has colour $x_{\tau_{k-1}}$. The colour of the token up to and at the time of the next jump is determined in two steps:

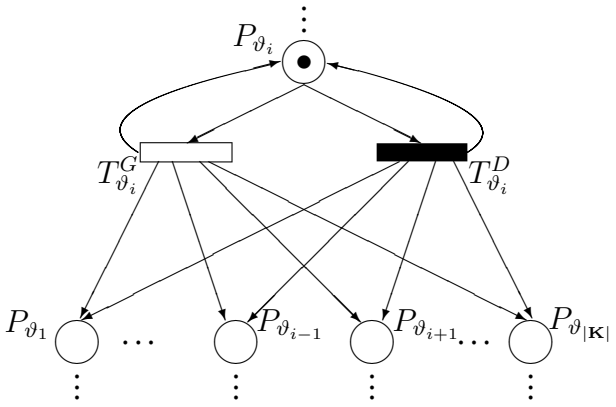

Fig. 6 Part of a Dynamically Coloured Petri Net representing a Piecewise Deterministic Markov Process.

Step 1: While the token is residing in place P_{ϑ_i}, its colour x_t changes according to the flow $\phi_{\vartheta_i, x_{\tau_{k-1}}}$, i.e., $x_t = \phi_{\vartheta_i, x_{\tau_{k-1}}}(t - \tau_{k-1})$. Transitions $T^G_{\vartheta_i}$ and $T^D_{\vartheta_i}$ are both pre-enabled and compete for this token which resides in their common input place P_{ϑ_i}. Transition $T^G_{\vartheta_i}$ models the boundary hitting generating a mode switch, while transition $T^D_{\vartheta_i}$ models the Poisson process generating a mode switch. The transition that is enabled first, determines the kind of switch occurring. The time at which this happens is denoted by τ_k.

Step 2: Next, with one of the transitions enabled, its firing measure is evaluated. This firing measure is such, that if a sample ζ_k from transition measure $Q(\cdot; \vartheta_i, \phi_{\vartheta_i, x_{\tau_{k-1}}}(\tau_k - \tau_{k-1}))$, would appear to be $\zeta_k = (\vartheta_j, x)$, then the enabled transition would produce one token with colour $x_{\tau_k} = x$ for place P_{ϑ_j}. The other places get no token.

After this, the process starts again in the same way from the new state on.

Pathwise equivalence of PDP and DCPN processes

With this construction, the PDP and DCPN processes generate the same sequence of stopping times. For pathwise equivalence from stopping time to stopping time both processes need to

use the same countable sequence of independent random variables U_1, U_2, ..., each having uniform $[0,1]$ distribution, defined by $U_i(\omega) = \omega_i$ for elements $\omega = (\omega_1, \omega_2, \ldots)$ of the Hilbert cube $\Omega = \prod_{i=1}^{\infty} Y_i$, with Y_i a copy of $Y = [0,1]$, to generate all random variables in both the PDP process and the DCPN process. This can be easily accomplished by drawing samples for the PDP and the associated DCPN at the same times and for the equivalent purposes, i.e.

- The first sample in each cycle is drawn to generate a stopping time associated with the Poisson point process generating a point, which is equivalent to the time the Delay function enables a transition;
- The second sample in each cycle is drawn to generate the value of the hybrid state after the jump, which is equivalent to the place of the token after the transition firing, and its colour.

Remark

The DCPN instantiation defined above has many places, and only one token. An interesting problem would be to find another into-mapping, in which the DCPN instantiation has fewer places and more tokens. Addressing this problem falls outside the scope of this paper.

C Proof of Theorem 2

Under some conditions, for each Dynamically Coloured Petri Net there exists a probabilistically equivalent Piecewise Deterministic Markov Process. In this appendix this is shown by providing an into-mapping from DCPN into the set of PDPs. Subsequently it is shown that this mapping is unique.

For an arbitrary DCPN that satisfies conditions D_1 – D_4, we first construct a PDP that is probabilistically equivalent to the DCPN process. As a preparatory step, the given DCPN is enlarged as follows: for each guard transition and each place from which that guard transition may be enabled, copy the corresponding places and transitions, including guard and firing functions, and revise the firing functions of the input transitions to these places, such that the new firings ensure that the corresponding guards may be reached from one side only. This step is illustrated with an example:

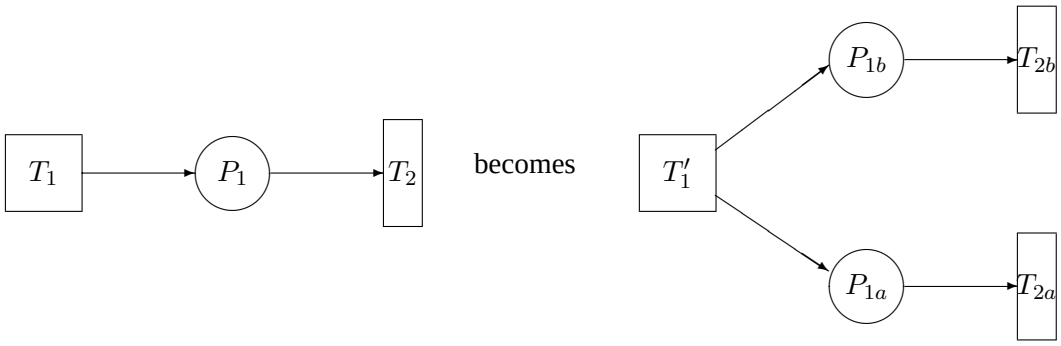

Fig. 7 Example transformation to model DCPN enlargement

In the picture on the left above, transition T_1 (which may be of any type) may fire tokens to place P_1, while transition T_2 is a guard transition that uses these tokens as input. In this example, assume that $\mathcal{C}(P_1) = I\!R$ and that $\partial G_{T_2} = 3$. This means, transition T_2 is enabled if the colour of the token in place P_1 reaches value 3. This value may be reached from above or from below, depending on whether the initial colour of the token in P_1 is larger or smaller than 3, respectively.

In the picture on the right, place P_1 and transition T_2 have been copied. Transitions T_{2a} and T_{2b} get the same guard as T_2, but transition T_1' gets a new firing function with respect to T_1: it is similar to the one of T_1, but it delivers a token to place P_{1a} if the colour of this new token is smaller than 3, and it delivers a token to place P_{1b} if its colour is larger than 3. This way, the guard of transition T_{2a} is always reached from below, i.e., its input colours are smaller than 3. The guard of transition T_{2b} is always reached from above, i.e., its input colours are larger than

3.

Let this enlarged DCPN be described by the tuple $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ and satisfy the rules R_0 – R_4, and assume that the conditions D_1 – D_4 are satisfied. In order to represent this DCPN by a PDP, all PDP elements $\mathbf{K}$, θ_t, x_t, $d(\theta)$, ξ_0, g_θ, ∂E_θ, λ, Q and the PDP conditions $C_1 - C_4$ are characterised in terms of this DCPN:

K: The domain $\mathbf{K}$ for the mode process $\{\theta_t\}$ can be found from the reachability graph (RG) of the DCPN graph. The nodes in the RG are vectors $V = (v_1, \ldots, v_{|\mathcal{P}|})$, where v_i equals the number of tokens in place P_i, $i = 1, \ldots, |\mathcal{P}|$, where these places are uniquely ordered. The RG is constructed from DCPN components $\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$ and $\mathcal{I}$. The first node V_0 is found from $\mathcal{I}$, which provides the numbers of tokens initially in each of the places[1]. From then on, the RG is constructed as follows: If it is possible to move in one jump from token distribution V_0 to, say, either one of distributions $V^1, \ldots, V^k$ unequal to V_0, then arrows are drawn from V_0 to (new) nodes $V^1, \ldots, V^k$. Each of $V^1, \ldots, V^k$ is treated in the same way. Each arrow is labelled by the (set of) transition(s) fired at the jump. If a node V^j can be directly reached from V^i by different (sets of) transitions firing, then multiple arrows are drawn from V^i to V^j, each labelled by another (set) of transition(s). Multiple arrows are also drawn if V^j can be directly reached from V^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, exclusive the nodes from which an immediate transition is effective, form the discrete domain $\mathbf{K}$ of the PDP. To emphasise them in the RG picture, these nodes are given in *italics*. 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 countable, $\mathbf{K}$ is a countable set, which satisfies the PDP conditions.

As an example, consider the following DCPN graph, which initially has two tokens in place P_1 and one in P_4, such that $V_0 = (2, 0, 0, 1, 0)$. This vector forms the first node of the reachability graph.

[1] Notice that $\mathbf{K}$ has to be constructed for all $\mathcal{I}$ by following the proposed procedure such that is applies for each possible instantiation of the initial token distribution.

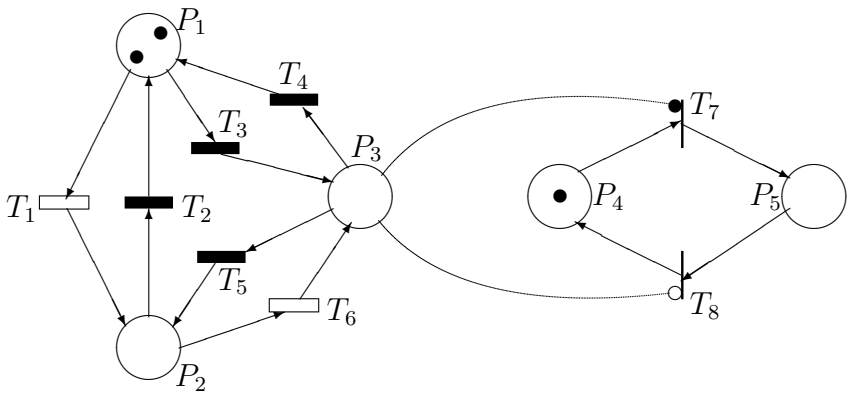

Fig. 8 Example DCPN graph to explain reachability graph

Both T_1 and T_3 are pre-enabled. They both have two tokens per incoming arc in their input place, hence for both transitions, two vectors of input colours are evaluated in parallel. If T_1 becomes enabled for one of these input tokens, it removes the corresponding token from P_1 and produces a token for P_2 (we assume that all firing measures are such, that each transition will fire a token when enabled, i.e., $\mathcal{F}_T(0,\cdot;\cdot) = 0$), so the new token distribution is $(1,1,0,1,0)$. Therefore, in the reachability graph two arcs labelled by T_1 are drawn from $(2,0,0,1,0)$ to the new node $(1,1,0,1,0)$; this duplication of arcs characterises that T_1 has evaluated two vectors of input tokens in parallel. It may also happen that from $(2,0,0,1,0)$, the guard transition T_1 is enabled by its two input tokens at exactly the same time. Due to Rule R_1 it then fires these two tokens at exactly the same time, resulting in node $(0,2,0,1,0)$. Therefore, an additional arc labelled $T_1 + T_1$ is drawn from $(2,0,0,1,0)$ to $(0,2,0,1,0)$. If, from $(2,0,0,1,0)$, transition T_3 fires before T_1 does, the token distribution becomes $(1,0,1,1,0)$. Subsequently, the immediate transition T_7 is enabled; its firing leads to $(1,0,1,0,1)$. Since $(1,0,1,1,0)$ enables an immediate transition it is drawn in italics and is excluded from $\mathbf{K}$. Unlike the case for T_1, there is no arc drawn from $(2,0,0,1,0)$ labelled by $T_3 + T_3$, since T_3 is a delay transition, hence the probability that it is enabled by both its input tokens at the same time is zero.

The resulting reachability graph for this example is given in Figure 9. So, for this example, $\mathbf{K} = \{(2,0,0,1,0),(1,1,0,1,0),(1,0,1,0,1),(0,2,0,1,0),(0,1,1,0,1),(0,0,2,0,1)\}$.

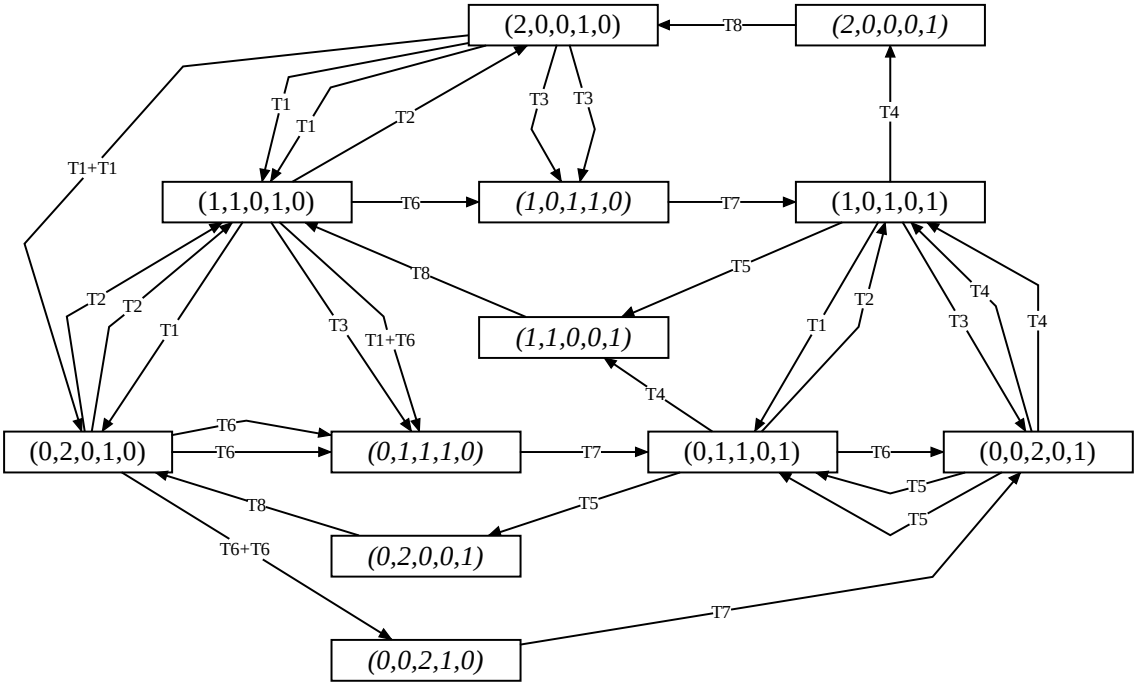

Fig. 9 Example reachability graph

$d(\theta)$**:** The colour of a token in a place P is an element of $\mathcal{C}(P) = I\!R^{n(P)}$, therefore $d(\theta) = \sum_{i=1}^{|\mathcal{P}|} \theta_i \times n(P_i)$, with $\theta = (\theta_1, \ldots, \theta_{|\mathcal{P}|}) \in \mathbf{K}$, with $\{1, \ldots, |\mathcal{P}|\}$ referring to the unique ordering of places adopted for the DCPN.

g_θ**:** For $x = \text{Col}\{x^1, \ldots, x^{|\mathcal{P}|}\}$, with $x^i \in I\!R^{\theta_i \times n(P_i)}$, and with $\{1, \ldots, |\mathcal{P}\}$ referring to the unique ordering of places adopted for the DCPN, g_θ is defined by $g_\theta(x) = \text{Col}\{g_\theta^1(x^1), \ldots, g_\theta^{|\mathcal{P}|}(x^{|\mathcal{P}|})\}$, where for $x^i = \text{Col}\{x^{i1}, \ldots, x^{i\theta_i}\}$, with $x^{ij} \in I\!R^{n(P_i)}$ for all $j \in \{1, \ldots, \theta_i\}$: $g_\theta^i(x^i) = \text{Col}\{\mathcal{V}_{P_i}(x^{i1}), \ldots, \mathcal{V}_{P_i}(x^{i\theta_i})\}$. Here, $j \in \{1, \ldots, \theta_i\}$ refers to the unique ordering of tokens within their place defined for DCPN (see Appendix A). Since, for all P_i, $\mathcal{V}_{P_i}$ is locally Lipschitz continuous, g_θ is also locally Lipschitz continuous.

∂E_θ**:** The boundary ∂E_θ of subset E_θ is determined from the transition guards corresponding with the set of transitions in $\mathcal{T}_G$ that, under token distribution θ, are pre-enabled (this set is uniquely determined). Without loss of generality, suppose this set of transitions is $T_1, \ldots, T_m$. Suppose $\{P^{i1}, \ldots, P^{ir_i}\}$ are the input places of T_i that are connected to T_i by means of ordinary or enabling arcs. Define $d_i = \sum_{j=1}^{r_i} n(P^{ij})$, then $\partial E_\theta = \partial G'_{T_1} \cup \ldots \cup \partial G'_{T_m}$, where $G'_{T_i} = [G_{T_i} \times I\!R^{d(\theta)-d_i}] \in I\!R^{d(\theta)}$. Here $[\cdot]$ denotes a special ordering of all vector elements: Vector elements corresponding with tokens in place P_a are ordered before vector elements corresponding with tokens in place P_b if $b > a$, according to the unique ordering of places adopted for the DCPN; vector elements corresponding with tokens within one place are ordered according to the

unique ordering of tokens within their place defined for DCPN (see Appendix A).

λ: The jump rate $\lambda(\theta, \cdot)$ is determined from the transition delays corresponding with the set of transitions in $\mathcal{T}_D$ that, under token distribution θ, are pre-enabled (this set is uniquely determined). Without loss of generality, suppose this set of transitions is $T_1, \ldots, T_m$. Then $\lambda(\theta, \cdot) = \sum_{i=1}^{m} \delta_{T_i}(\cdot)$. This equality is due to the fact that the combined arrival process of individual Poisson processes is again Poisson, with an arrival rate equal to the sum of all individual arrival rates. Since δ_T is integrable for all $T \in \mathcal{T}_D$, λ is also integrable.

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 $R_0 - R_4$. 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 $R_0 - R_4$ 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 state (θ, x) before the jump and the set of transitions that will fire in the jump. Because of its complexity, this characterisation is given in Appendix D, but an outline is given next:

Main challenge in the characterisation of Q is the following: In some situations one does not know for certain which transitions will fire in a jump, even if one knows the state (θ, x) before the jump and knows that a jump will occur from (θ, x) to (θ', x'). Hence, in these situations it is not known with certainty which firing functions one should combine in order to construct $Q(\theta', x'; \theta, x)$ from DCPN elements. However, one does know the following:

- Given θ, one knows which transitions are pre-enabled; this can be read off the reachability graph (i.e. gather the labels of all arrows leaving node θ).
- Given that $\theta \in \mathbf{K}$, no immediate transitions are enabled in θ.
- The probability that a guard transition and a delay transition are enabled at exactly the same time is zero.
- The probability that two delay transitions are enabled at exactly the same time is zero.
- There is a possibility that two or more guard transitions are enabled at exactly the same time. It may even occur (due to rule R_1) that one single guard transition fires twice at the same time.

Hence, the steps to be followed to construct $Q(\theta', x'; \theta, x)$, for any (θ', x', θ, x) are:

1. Determine (using the reachability graph) which transitions are pre-enabled in θ.
2. Consider the guard transitions in this set of pre-enabled transitions and determine which of these are enabled. For a transition T, this is done by considering its vector

of input colours (which is part of x) and checking whether this vector has entered the boundary ∂G_T. If the set of enabled guard transitions is not empty, then use rules $R_1 - R_4$ to find out which of these transitions will actually fire with which probability.
If this set of enabled guard transitions is empty, then one pre-enabled delay transition must be enabled. Use $\mathcal{D}$ to determine for each pre-enabled delay transition the probability with which it will actually fire.

3. Determine which transition firings can actually lead to discrete process state θ' in one jump. This set can be found by identifying in the reachability graph all arrows directly from node θ to θ' and all directed paths from node θ to θ' that pass only nodes that enable immediate transitions (i.e. that pass only nodes in italics).
4. Finally, $Q(\theta', x'; \theta, x)$ is constructed from the firing functions, by conditioning on these arrows and paths from θ to θ'.

$\xi_0 = (\theta_0, x_0)$**:** This can be constructed from $\mathcal{I}$, the DCPN initial marking, which provides the places the tokens are initially in and the colours these tokens have. Hence, $\theta_0 = (v_{1,0}, \ldots, v_{|\mathcal{P}|,0})$, where $v_{i,0}$ denotes the initial number of tokens in place P_i, with the places ordered according to the unique ordering adopted for DCPN, and $x_0 \in I\!R^{d(\theta_0)}$ is a vector containing the colours of these tokens. Within a place the colours of the tokens are ordered according to the specification in $\mathcal{I}$. With this, and due to condition D_4 (which prevents different token distributions to be applicable at the initial time), the constructed ξ_0 is uniquely defined.

C_1**:** This condition (no explosions) follows from assumption D_1.

C_2**:** This condition (λ is integrable) follows from the fact that δ_T is integrable for all $T \in \mathcal{T}_D$.

C_3**:** This condition (Q measurable and $Q(\{\xi\}; \xi) = 0$) follows from the assumption that $\mathcal{F}$ is continuous and from assumption D_2.

C_4**:** This condition ($I\!E N_t < \infty$) follows from assumption D_3.

This shows that for any DCPN satisfying conditions D_1 – D_4, we are able to construct unique PDP elements, and thus a unique PDP.

Finally, we show that the PDP process $\{\theta_t, x_t\}$ is probabilistically equivalent to the process generated by the DCPN:

With the mapping from DCPN elements into PDP elements, it is easily shown that the PDP process $\{\theta_t, x_t\}$ is probabilistically equivalent to the process generated by the DCPN characterised in Appendix A: at each time t the process $\{\theta_t\}$ is probabilistically equivalent to the process $(v_{1,t}, \ldots, v_{|\mathcal{P}|,t})$ and the process $\{x_t\}$ is probabilistically equivalent to the process

associated with the vector of token colours. This is shown by observing that the initial PDP state (θ_0, x_0) is probabilistically equivalent to the initial DCPN state through the mapping constructed above. Moreover, also by the unique mapping of DCPN elements into PDP elements, at each time instant after the initial time, the PDP state is probabilistically equivalent to the DCPN state: At times t when no jump occurs, the PDP process evolves according to g_θ and the DCPN process evolves according to $\mathcal{V}$. Through the mapping between g_θ and $\mathcal{V}$ developed above, these evolutions provide probabilistically equivalent processes. At times when a jump occurs, the PDP process makes a jump generated by Q, while the DCPN process makes a jump generated by $\mathcal{F}$. Through the mapping between Q and $\mathcal{F}$ developed above, these jumps provide probabilistically equivalent processes.

D Characterisation of Q in terms of DCPN elements

In this appendix, Q is characterised in terms of DCPN, as part of the characterisation in Appendix C of PDP in terms of DCPN.

For each $\theta \in \mathbf{K}$, $x \in E_\theta$, $\theta' \in \mathbf{K}$ and $A \subset E_{\theta'}$, the value of $Q(\theta', A; \theta, x)$ is a measure for 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). Measure $Q(\theta', A; \theta, x)$ is characterised in terms of the DCPN by the reachability graph (RG) (see Appendix C), elements $\mathcal{D}$, $\mathcal{G}$ and Rules $R_0 - R_4$ and the set $\mathcal{F}$, as below. This is done in four steps:

1. Determine which transitions are pre-enabled in (θ, x).
2. Determine for each pre-enabled transition the probability with which it is enabled in (θ, x).
3. Determine for each pre-enabled transition whether its firing can possibly lead to discrete state θ'.
4. Use the results of the previous two steps and the set of firing functions 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 at exactly the same time), etc. Therefore the arrows leaving θ may be characterised by these labels. Denote the multi-set of arrows, characterised by these labels, by $\mathcal{B}_\theta$. 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 $B \in \mathcal{B}_\theta$ for an element B of $\mathcal{B}_\theta$ (e.g. $B = T_1$ represents an arrow with T_1 as label), and notation $T \in B$ for a transition T in label B (e.g. as in $B = T + T_1$).

Step 2: Determine for each pre-enabled transition 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 $\mathcal{B}_\theta$. In the following, we determine, for all $B \in \mathcal{B}_\theta$, the probability $p_B(\theta, x)$ that all transitions in label B 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 unique since we consider transitions with multiple vectors of input colours separately in the multi-set $\mathcal{B}_\theta$.
- Consider the multi-set $\mathcal{B}_\theta^G = \{B \in \mathcal{B}_\theta | \forall T \in B : T \in \mathcal{T}_G \text{ and } c_T^x \in \partial G_T\}$.

- If $\mathcal{B}_\theta^G \neq \emptyset$ then this set contains all transitions that are enabled in (θ, x). Rules $R_1 - R_4$ are used (R_0 is not applicable) to determine for each $B \in \mathcal{B}_\theta^G$ the probability with which the transitions in label B will actually fire:
  - Rules R_1 and R_3 are used as follows: if B is such that there exists $B' \in \mathcal{B}_\theta^G$ such that the transitions in B form a real subset of the set of transitions in B', then $p_B(\theta, x) = 0$. The set of thus eliminated labels B is denoted by $\mathcal{B}_\theta^{R_{1,3}}$.
  - Rules R_2 and R_4 are used as follows: If the multi-set $\mathcal{B}_\theta^G - \mathcal{B}_\theta^{R_{1,3}}$ contains m elements, then each of these labels gets a probability $p_B(\theta, x) = 1/m$.
- If $\mathcal{B}_\theta^G = \emptyset$ then only Delay transitions can be enabled in (θ, x). Consider the multi-set $\mathcal{B}_\theta^D = \{B \in \mathcal{B}_\theta | \forall T \in B : T \in \mathcal{T}_D\}$. Each $B \in \mathcal{B}_\theta^D$ consists of one delay transition, with $p_B(\theta, x) = \dfrac{\delta_B(c_B^x)}{\sum\limits_{T \in \mathcal{B}_\theta^D} \delta_T(c_T^x)}$.

Step 3: Determine for each pre-enabled transition 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 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, x) to a state in (θ', A) is not possible.

Step 4: Use the results of the previous two steps and the set of firing functions 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 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 $\mathcal{L}_{\theta\theta'}$. Examples of elements of $\mathcal{L}_{\theta\theta'}$ 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 $L \in \mathcal{L}_{\theta\theta'}$ along which the jump is made. Under the condition that a jump occurs:

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

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

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

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

First, assume that $\mathcal{L}_{\theta\theta'}$ does not contain immediate transitions. This yields: each $L \in \mathcal{L}_{\theta\theta'}$ either contains one or more guard transitions, or one delay transition (other combinations occur with zero probability). In particular, $\mathcal{L}_{\theta\theta'}$ is a subset of $\mathcal{B}_\theta$ defined earlier. Then $p_{L|\theta,x}(L \mid \theta, x)$ is determined by $p_{L|\theta,x}(L \mid \theta, x) = \dfrac{p_L(\theta, x)}{\sum_{B \in \mathcal{L}_{\theta\theta'}} p_B(\theta, x)}$, with $p_B(\theta, x)$ defined earlier.

Next, consider the situations where $\text{RG}_{\theta\theta'}$ may also contain nodes that enable immediate transitions. If L is of the form $L = T_j \circ T_k$, with T_j an immediate transition, then $p_{L|\theta,x}(L \mid \theta, x) = p_{T_k|\theta,x}(T_k \mid \theta, x)$, with the right-hand-side constructed as above for the case without immediate transitions. The same value $p_{T_k|\theta,x}(T_k \mid \theta, x)$ follows for cases like $L = T_m \circ T_j \circ T_k$, with T_j and T_m immediate transitions. However, if the firing of T_k enables more than one immediate transition, then the value of $p_{T_k|\theta,x}(T_k \mid \theta, x)$ is equally divided among the corresponding paths. This means, for example, that if there are $L_1 = T_j \circ T_k$ and $L_2 = T_m \circ T_k$ then $p_{L_1|\theta,x}(L_1 \mid \theta, x) = p_{L_2|\theta,x}(L_2 \mid \theta, x) = \dfrac{1}{2} p_{T_k|\theta,x}(T_k \mid \theta, x)$.

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

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

For probability $p_{\theta', x' | \theta, x, L}(\theta', A \mid \theta, x, L)$, first notice that both (θ, x) and (θ', x') represent states of the complete DCPN, while the firing of L 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 L of transitions fires.

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

In all other cases:

- Assume L 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 sample 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:

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

  where $\mathbf{I}_{(\theta',A;e',c',c_T^x)}$ 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 L consists of several transitions $T_1, \ldots, T_m$ 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 $T_1, \ldots, T_m$:

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

  where $\mathbf{I}_{(\theta',A;e'_1,c'_1,c_{T_1}^x,\ldots,e'_k,c'_k,c_{T_k}^x)}$ 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 L is of the form $L = T_j \circ T_k$, with T_j an immediate transition, then the result is:

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

  where $\mathbf{I}_{(\theta',A;e'_j,c'_j,e'_k,c'_k,c_T^x)}$ 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 $L = T_m \circ T_j \circ T_k$, with T_j and T_m immediate transitions, the firing functions of this sequence of transitions are multiplied in a similar way as above.

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

E Acronyms and Symbols

Acronyms used

CTMC	Continuous Time Markov Chain
DCPN	Dynamically Coloured Petri Net
DSPN	Deterministic and Stochastic Petri Net
ECPN	Extended Coloured Petri Net
FSPN	Fluid Stochastic Petri Net
FT	Fault Tree
FTRE	Fault Tree with Replaced Events
GSHP	Generalised Stochastic Hybrid Process
GSPN	Generalised Stochastic Petri Net
HLHPN	High-Level Hybrid Petri Net
PDP	Piecewise Deterministic Markov Process
PN	Petri Net
RBD	Reliability Block Diagram
RG	Reachability Graph
RRG	Reduced Reachability Graph
SDCPN	Stochastically and Dynamically Coloured Petri Net

Symbols used

$\cup_\theta$	:	Unity
$\cap_\theta$	:	Intersection
$\|\cdot\|$	:	Number of elements in a set
$\mathbb{R}^n$	:	n-dimensional real numbers
$\mathbb{N}$	:	Natural numbers
∂E	:	Boundary of open subset E
$R_0 - R_4$	:	Rules
$D_1 - D_3$	:	Conditions

t	:	Time
τ, τ_k	:	Stopping times
Δ	:	Time
$t_*(\theta, x)$	:	Time until first boundary hit
$t_\infty(\theta, x)$	:	Explosion time of flow $\phi_{\theta,x}(\cdot)$
σ_k, ζ_k	:	Samples from probability distributions
N_t	:	Number of jumps until time t

x_t	:	Continuous state
θ_t	:	Discrete state
ξ_t	:	Hybrid state
E_θ	:	Open subset of $\mathbb{R}^{d(\theta)}$
E	:	Disjoint unity of all subsets E_θ
$\mathcal{E}$	:	Borel-measurable subsets of E
Γ^*	:	Reachable boundary of E
$\mathbf{K}$	:	Countable domain for process $\{\theta_t\}$
$d(\cdot)$	:	Function that maps $\mathbf{K}$ into $\mathbb{N}$
$g_\theta(\cdot)$	:	Lipschitz continuous function
ϕ_{θ,x_0}	:	Flow
$\lambda(\theta_t, x_t)$	:	Rate of Poisson point process
$G_\xi(\cdot)$	:	Survivor function
I_A	:	Indicator function
$Q(\cdot;\xi)$	:	Transition measure

$\mathcal{P}$	:	Set of places
$\mathcal{P}(A)$	:	The place that is connected to arc A
$\mathcal{T}$	:	Set of transitions
$\mathcal{T}_G$	:	Set of guard transitions
$\mathcal{T}_D$	:	Set of delay transitions
$\mathcal{T}_I$	:	Set of immediate transitions
$\mathcal{T}(A)$	:	The transition that is connected to arc A
$\mathcal{A}$	:	Set of arcs
$\mathcal{A}_O$	:	Set of ordinary arcs
$\mathcal{A}_E$	:	Set of enabling arcs
$\mathcal{A}_I$	:	Set of inhibitor arcs
$A(T)$	:	Set of arcs connected to transition T
$A_{in}(T)$	:	Set of input arcs of transition T
$A_{in,O}(T)$	:	Set of ordinary input arcs of transition T
$A_{in,OE}(T)$	:	Set of input arcs of transition T that are either ordinary or enabling
$A_{out}(T)$	:	Set of output arcs of transition T

$\mathcal{N}$	:	Node function
$\mathcal{S}$	:	Set of colour types
$\mathcal{C}$	:	Colour function
$\mathcal{I}$	:	Initial marking
$\mathcal{V}$	:	Set of token colour functions
$\mathcal{V}_P$	:	Token colour function for place P
$\mathcal{G}$	:	Set of transition guards
$\mathcal{G}_T$	:	Transition guard for transition T
$\mathcal{D}$	:	Set of transition delays
$\mathcal{D}_T$	:	Transition delay for transition T
$\mathcal{F}$	:	Set of firing measures
$\mathcal{F}_T$	:	Firing measure for transition T
P, P_i	:	Place
$P(A(T))$	:	Set of places connected to T by the set of arcs $A(T)$
T, T_i	:	Transition
T_i^G	:	Guard transition
T_i^D	:	Delay transition
A, A_i	:	Arc
$\mathcal{C}(P)_{ms}$	:	Set of all multisets over $\mathcal{C}(P)$
c, c_t	:	Colour of token or vector of colours
δ_T	:	Rate of transition delay
z_t	:	Vector containing position and velocity of aircraft
v_t	:	Velocity of aircraft
u	:	Random number
$p_{B\|C}$	:	Conditional probability density function
α_ℓ	:	Auxiliary variable
f	:	Vector of zeros and ones
ϑ_i, m_i	:	Element of $\mathbf{K}$
$v_{i,t}$, v_i	:	Number of tokens in place P_i
V, V^i	:	Nodes of reachability graph
$x_{i,j,t}$	:	Colour of jth token in place P_i at time t
$\mathcal{L}$	:	Set of paths characterised by labels
L	:	Set of transitions, element of $\mathcal{L}$