Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2006-681

Piecewise deterministic Markov processes represented by dynamically coloured Petri nets

M.H.C. Everdij and H.A.P. Blom

This report contains a paper that appeared in Stochastics, Vol 77, Issue 1, February, 2005.

Customer:	National Aerospace Laboratory NLR
Working Plan number:	2005 AT.1.A
Owner:	National Aerospace Laboratory NLR
Division:	Air Transport
Distribution:	Unlimited
Classification title:	Unclassified
	September 2006

Approved by:

Author	Reviewer	Managing department
ME '06	Anonymous peer reviewers	22/12/06

Piecewise deterministic Markov processes represented by dynamically coloured Petri nets

MARIKEN H.C. EVERDIJ and HENK A.P. BLOM*

National Aerospace Laboratory NLR, P.O. Box 90502, 1006 BM, Amsterdam, The Netherlands

(Received 3 May 2001; in final form 12 October 2004)

Piecewise deterministic Markov processes (PDPs) are known as the largest class of Markov processes virtually describing all continuous-time processes not involving diffusions. For PDPs, a substantial amount of powerful analysis and control results are available. As such, PDPs are attractive for use in modelling complex distributed systems. However, the specification of an appropriate PDP model for complex distributed systems that exist in practice is far from trivial. This difficulty already applies for the specification of a continuous-time Markov chain (CTMC). For a compositional specification of a CTMC model, Petri Nets have proven to be extremely useful. In order to realise a similar situation for PDP, this paper develops a novel type of Petri Net, named dynamically coloured Petri Net (DCPN), and proves that there exist into-mappings between PDPs and DCPNs.†

Keywords: Piecewise deterministic Markov processes; High-level Petri Nets; Hybrid systems; Modelling; Poisson process; Discrete event systems

AMS 1991 Classification: 60G07 (General theory of processes); 93E03 (Stochastic systems general)

1. Introduction

1.1 Aim of the paper

Mark Davis [1] has introduced piecewise deterministic Markov processes (PDPs) as a general class of continuous-time Markov processes which include both discrete and continuous processes, except diffusion. Suppose $\mathbf{K}$ is a countable set, d is a mapping of $\mathbf{K}$ in the natural numbers, moreover E_k, with $k \in \mathbf{K}$, is an open connected subset of $\mathbb{R}^{d(k)}$ with boundary ∂E_k, whereas $E = \{(k, x); k \in \mathbf{K}, x \in E_k\}$. Then a PDP $\{\xi_t\}$ with ξ_t assuming values in E, consists of two components: a discrete valued component $\{\theta_t\}$, $\theta_t \in \mathbf{K}$, and a continuous valued component $\{x_t\}$, $x_t \in E_k$, see [1,2]. At discrete times, $\{\theta_t\}$ may jump from one value to another value which is selected according to some probabilistic relation. Between jumps, the continuous valued component is a solution of a θ_t-dependent differential equation $\dot{x}_t = g_{\theta_t}(x_t)$. At discrete moments in time, $\{x_t\}$ may jump according

*Corresponding author. E-mail: blom@nlr.nl

†Part of this research has been performed within the project HYBRIDGE of the European Commission, contract number IST-2001-32460

Stochastics: An International Journal of Probability and Stochastic Processes
ISSN 1744-2508 print/ISSN 1744-2516 online © 2005 Taylor & Francis Group Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/17442500512331341040

to some probability measure Q, which makes it only piecewise continuous. The PDP state is given by $\xi_t = \mathrm{Col}\{\theta_t, x_t\}$, and is called a *hybrid state*. A jump in $\{x_t\}$ and/or $\{\theta_t\}$ occurs either when a doubly stochastic Poisson process generates a point with rate $\lambda(\theta_t, x_t)$ or when $\{x_t\}$ hits the boundary ∂E_θ of a predefined area. PDPs are defined such that their sample paths are right-continuous and have left-hand-side limits, often abbreviated as càdlàg, which is an acronym for the French "continu à droite, limites à gauche", see e.g. [3]. PDPs form a powerful and useful class of processes that have strong support in stochastic analysis and control. In addition to this, PDP's relationship to hybrid automata is well known [4].

Petri nets [5], and their many extensions, see e.g. [6] for a good overview, have proven to be extremely useful in developing models for various complex practical applications. This usefulness is especially due to their compositional specification power [5], which allows to divide a complex operation into entities, to develop a submodel for each such entity, and next to combine the submodels in a constructive way. The aim of the paper is to introduce a novel class of hybrid Petri nets and to show that there exist into-mappings between this novel class of hybrid Petri nets and the class of PDPs. The existence of such into-mappings allows combining the compositional specification power of Petri nets with the stochastic analysis and control power of PDPs.

The idea of establishing into-mappings between Petri nets and stochastic processes in order to combine advantages of both classes is well developed for finite state processes. In particular, Malhotra and Trivedi [7] and Muppala *et al.* [8] developed a hierarchy of various dependability models based on their modelling power. At the top of this hierarchy are continuous-time Markov chains (CTMC) on the one hand, and generalised stochastic Petri Nets (GSPN) on the other hand. GSPN have already been well established for developing CTMC for complex practical discrete-valued applications [7]. As shown by Davis [1], CTMC form a particular subclass of PDPs. Hence, this paper extends the power hierarchy with PDPs and with PDP-related Petri nets.

1.2 Basic Petri net

A Petri net graph is a directed bi-partite graph with two types of nodes: *places* and *transitions*, coupled by *arcs* (arrows). The transitions (rectangles) generally model actions, the places (circles) generally represent possible pre or post conditions for these actions. A basic Petri net is a Petri net graph with one or more places containing one or more *tokens* (dots). A token residing in a place models a pre condition being current. In general, multiple tokens are allowed in a Petri net, even inside one place, such that compound pre conditions can be modelled without explosion of the size of a Petri net graph. In its simplest form, the execution of a Petri net is as follows: a transition is said to be *enabled* if each of its input places contains a token. When this occurs, the transition *fires*: it removes the tokens from its input places and produces tokens for its output places. In addition to an ordinary arc, in literature it is common practice to also use in a basic Petri net an *enabling arc* from a place to a transition (the working of the connected transition is similar as for ordinary arc, but upon its firing, it does not remove the token from its input place) or an *inhibitor arc* from a place to a transition (the connected transition becomes *disabled* when its input place contains a token). See figure 1 for a very simple Petri net example with two places and two transitions. In this figure, place P_1 is current and transition T_2 is enabled. After T_2 has fired, place P_2 is current.

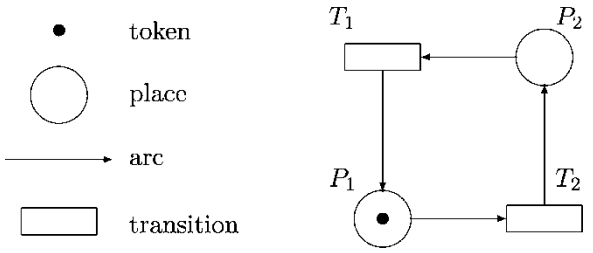

Figure 1. A simple Petri net and its building stones. Place P_1 is current and transition T_2 is enabled.

A basic Petri net as defined above can represent some of the elements of a PDP only: If we draw a place P_θ for each element θ of $\mathbf{K}$, and couple these places with transitions and arcs, then the resulting Petri net gives the structure of discrete PDP states, where the transitions model possible mode switches. However, an important modelling capability still missing is to model the continuous-valued process component $\{x_t\}$ of a PDP and events depending of this $\{x_t\}$.

1.3 PDP-driven extensions of Petri nets

In order to represent PDP by means of Petri nets, we have to extend the basic Petri net definition with novel elements until we can prove the existence of a one-to-one mapping between PDP and the resulting Petri net extension. Necessary elements to be added are:

- Element to model a continuous-valued process component. We solve this by introducing a colour function $\mathcal{C}$ which maps each place P_θ into finite set $\mathcal{S}$ of Euclidean spaces, and attach to each token in place P_θ a colour process $c_t \in \mathcal{C}(P_\theta)$. This specific extension of a basic Petri net with Euclidean valued colour elements has been introduced in [9]. The term colour, referring to a value or an identification attached to a token, was introduced in [10], who considered finite colour sets only.
- Element to change a colour dynamically with time. We solve this by coupling to each place P_θ the PDP function g_θ and let the process c_t evolve according to $\dot{c}_t = g_\theta(c_t)$ as long as the token resides in place P_θ. This specific extension over a basic Petri net has originally been introduced both in [11] and [12].
- Element to model Boundary hitting process. We solve this by coupling a boundary to some of the transitions, which we name Guard transitions. If an input place for such Guard transition is P_θ, then the transition is associated with a boundary ∂E_θ. If place P_θ has a token from time τ onwards, with this token having a (dynamically evolving) colour c_t at time $t > \tau$, then the transition will be enabled (and remove the token from P_θ) at the time when the token colour c_t hits ∂E_θ. This specific extension over a basic Petri net has originally been introduced both in [11] and [12]. The concept of guards was introduced by [10], for token colours that do not dynamically evolve while they reside in their place.
- Element to model Poisson type of jumps. We solve this by coupling random delays to some of the transitions, which we name delay transitions. If an input place for such delay transition is P_θ, then the transition is associated with the enabling rate $\lambda(\theta, \cdot)$. If place P_θ has a token from time τ onwards, with this token having a (dynamically evolving) colour

c_t at time $t > \tau$, then the transition will be enabled (and remove the token from P_θ) at the time when a Poisson point process with enabling rate $\lambda(\theta, c_t)$ generates a point. This type of extension, but with Poisson-type of enabling rates that are not colour-dependent, is well known, see e.g. [13]. The specific extension of colour-dependent enabling rates has originally been introduced in [11].

- Element to model probability measure. We solve this by coupling a probability measure to each transition, and naming this measure a firing measure. When the transition is enabled (i.e. its delay has passed or its boundary has been hit by the colour of the input token), then this firing measure determines the colour of the output token, based on the colour of the input token. This specific extension over a basic Petri net has originally been introduced both in [11] and [12].

The Petri net, extended with these elements, is named *dynamically coloured Petri net* (DCPN), referring to the notion that its tokens have values (i.e. are coloured) that change through time (i.e. dynamically).

In addition to defining DCPN, the key contribution of this paper is to prove the existence of one-to-one mappings between DCPN and PDP. The mapping of a PDP into DCPN is rather straightforward. The mapping of a DCPN into a PDP is however, more demanding. The two main challenges of the latter mapping are: (a) in a DCPN, a sequence of immediate transitions may fire at a single moment in time, while for PDP at each moment in time only one jump may occur; (b) in a DCPN, there is a non-fixed number of tokens which evolve individually, while a PDP is represented by a single hybrid state. For the mapping of DCPN into PDP, problem (a) is overcome by defining a pathwise unique càdlàg stochastic process that is generated by a DCPN. Problem (b) is overcome by constructing the reachability graph of the DCPN considered, and using the resulting nodes as the basis for the discrete set of PDP.

The development of this DCPN idea started in [11] and was continued in [14]. The current paper realises a significant improvement over these earlier results.

1.4 Hybrid Petri nets in literature

This section discusses how other hybrid kinds of main Petri nets extensions from literature relate to PDPs.

One interesting Petri Net extension is named hybrid Petri net [15], which is a generalisation of continuous Petri net [16]. Besides places that may contain discrete tokens, the hybrid Petri net has places that may contain a non-negative real-valued amount of tokens. Transitions connected to these places consume continuous amounts of these tokens at certain quantities per time unit and next produce these amounts for other continuous places. The state of the hybrid Petri net at each time instant t is written as a vector giving for each place its marking, i.e. the amount of tokens it contains. The marking of a place at a later time instant is equal to the marking at time t, minus the amount of token consumed by output transitions of the place, plus the amount produced by input transitions of the place. Since a change of marking for one place automatically incorporates a change of marking for another place, and since all amounts of tokens should be non-negative, into-mappings between PDPs and hybrid Petri nets are far from obvious.

Related to hybrid Petri net is fluid stochastic Petri net [17], which also moves fluid tokens between continuous places and discrete tokens between discrete places. The transition times

for the discrete tokens are exponentially distributed (possibly depending on the fluid level) such that the discrete Petri net part gives rise to a continuous time Markov process. The discrete token marking influences the continuous flow rate between the continuous places. In the continuous net, conservation of token mass is to be preserved. Hence, into-mappings between PDPs and fluid stochastic Petri nets are generally not feasible.

In extended coloured Petri net (ECPN), as introduced in [9], the token colours are real-valued vectors that may follow the solution path of a difference equation. The token colour is updated in an external loop around its residence place by an additional updating transition. The state of the ECPN is given by the current colours of all tokens and the places they reside in. PDPs might be represented by ECPN but only in an approximate way: the mode values might be mapped to the places and the drift process might be mapped to the colours of the tokens; however, the continuous-time drift process component has to be approximated by means of a discrete-time difference equation. This makes that a boundary hitting can never be exactly timed. Moreover, the necessary presence of updating transitions may result in a very large Petri net graph when modelling a complex process.

High-level hybrid Petri net (HLHPN) has been introduced in [18] as a further elaboration of hybrid Petri net. In an HLHPN there are two kinds of places: the usual places for discrete tokens and a new type of places storing real-valued tokens which follow the solution path of a differential equation. A token switch between discrete places may generate a jump in the value of the real valued vector. Advantage of this extension with respect to hybrid Petri net is that continuous valued processes can be modelled that do not need to have a positive value. Advantage with respect to ECPN is the avoidance of the time discretisation. Similarly as with ECPN, HLHPNs use updating transitions which still may result in a very large Petri net graph when modelling a complex process.

Related to HLHPN are differential Petri nets [19], which have discrete places (having a non-negative integer marking), differential places (having a real valued marking, which can also be negative), discrete transitions and differential transitions. A differential transition is enabled if its discrete input places contain a number of tokens that is larger than or equal to the weight of the corresponding arcs. An enabled differential transition that fires yields a change of marking equal to the speed of the transition, times the weight of the corresponding arc. This speed may be a constant, a linear combination, or a non-linear function of the markings connected to the transition (the speed may also be negative). The facts that time is discretised and that the markings are one-dimensional, may result in a very large Petri net graph when modelling a complex process. Moreover, mode switches are enabled by integer bounds only.

Differential predicate transition Petri nets (DPT Petri Nets) [12] associate variables to each token, associate a differential equation to each place and associate an enabling function and a junction function to each transition. The token-associated variables follow the solution of the place-associated differential equation, and the transition-associated enabling function triggers the firing of the transition according to the value of the input tokens of these transitions. The transition-associated junction function defines the value of the output tokens of the transition at the firing. DPT Petri nets do not support transitions which represent the Poisson type of jumps of PDP.

This overview leads to the conclusion that there are many valuable Petri net extensions available in literature that are related to PDPs; however, for none of them into-mappings with PDPs are known, and difficulties are foreseen trying to develop such into-mappings.

1.5 Organisation of the paper

The organisation of the paper is as follows. Section 2 briefly describes PDPs. Section 3 defines DCPNs. Section 4 shows that each PDP can be represented by a DCPN process. Section 5 shows that each DCPN process can be represented by a PDP. Section 6 presents a DCPN model for a simple aircraft evolution example and its mapping to a PDP. Section 7 draws conclusions.

2. Piecewise deterministic Markov processes

2.1 PDP brief explanation

A piecewise deterministic Markov process $\{\xi_t\}$, with $\xi_t = (\theta_t, x_t)$, is defined as follows (see Davis [2]): For each θ in its countable domain $\mathbf{K}$, let E_θ be an open connected subset† of $\mathbb{R}^{d(\theta)}$, and 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 $g_\theta : \mathbb{R}^{d(\theta)} \rightarrow \mathbb{R}^{d(\theta)}$ 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 $E_{\theta_\tau}, t > \tau$.

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.

2.2 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:

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

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 $t_*(\theta_0, x_0) \triangleq \inf\{t - \tau_0 > 0 | \phi_{\theta_0, x_0}(t - \tau_0) \in \partial E_{\theta_0}\}$. The first factor

†Note that [2] extends E_θ to $E'_\theta = E_\theta \cup \partial_1 E_\theta$, with E_θ an open subset of $\mathbf{R}^{d(\theta)}$ and $\partial_1 E_\theta$ those points on the boundary of E_θ from which E_θ can be reached by the flow ϕ, but which cannot be reached from the interior of E_θ

in Expression (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 Expression (1) comes from the Poisson process: this second factor ensures 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 Section 23 of [2]), 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 $\psi_2 : [0, 1] \times (E \cup \Gamma^*) \rightarrow E$, 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

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

2.3 PDP conditions

Following Section 24.8 of [2], 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. $|\phi_{\theta,x}(t)| \rightarrow \infty$ 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 $E = \cup_\theta E_\theta$, $\lambda : E \rightarrow \mathbb{R}^+$ is a measurable function such that for all $\xi \in E$, there is $\epsilon(\xi) > 0$ such that $t \rightarrow \lambda(\theta, \phi_{\theta,x}(t))$ 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, ε), with ε the Borel-measurable subsets of E, while for each fixed $A \in \varepsilon$, the map $\xi \rightarrow Q(A; \xi)$ 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 $t \in \mathbb{R}^+$, $\mathbb{E}N_t < \infty$. This means, there will be a finite number of jumps in finite time.

3. Dynamically coloured Petri net (DCPN)

This section presents a definition of DCPN. As much as possible, the notation introduced by Jensen [10] for coloured Petri net is used.

DEFINITION. A 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 R_0–R_4. Below, first the structure of the elements in the tuple is given, next the DCPN evolution through time is explained, finally, the DCPN generated process is outlined.

3.1 DCPN elements

The DCPN elements are defined as follows:

1. $\mathcal{P}$ is a finite set of places. In a graphical notation, places are denoted by circles: Place: ◯
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 $\mathcal{T}_G \cap \mathcal{T}_D = \mathcal{T}_D \cap \mathcal{T}_I = \mathcal{T}_I \cap \mathcal{T}_G = \emptyset$. Notations are:
   Guard transition: ▭ Delay transition: ▬ Immediate transition: ━
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 $\mathcal{A}_O \cap \mathcal{A}_E = \mathcal{A}_E \cap \mathcal{A}_I = \mathcal{A}_I \cap \mathcal{A}_O = \emptyset$. Notations are:
   Ordinary arc: ⟶ Enabling arc: ——● Inhibitor arc: ——○
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:

   - $A(T) = \{A \in \mathcal{A} | T(A) = T\}$ denotes the set of arcs connected to transition T, with $A(T) = A_{\text{in}}(T) \cup A_{\text{out}}(T)$, where
   - $A_{\text{in}}(T) = \{A \in A(T) | \mathcal{N}(A) = (P(A), T)\}$ is the set of input arcs of T and
   - $A_{\text{out}}(T) = \{A \in A(T) | \mathcal{N}(A) = (T, P(A))\}$ is the set of output arcs of T. Moreover,
   - $A_{\text{in},O}(T) = A_{\text{in}}(T) \cap \mathcal{A}_O$ is the set of ordinary input arcs of T,
   - $A_{\text{in},OE}(T) = A_{\text{in}}(T) \cap \{\mathcal{A}_E \cup \mathcal{A}_O\}$ 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, $\{A_i \in \mathcal{A}_I | \exists A \in \mathcal{A}, A \neq A_i : \mathcal{N}(A) = \mathcal{N}(A_i)\} = \emptyset$, 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 and with $\mathbb{R}^0 = \emptyset$.
6. $\mathcal{C} : \mathcal{P} \rightarrow \mathcal{S}$ is a colour function which maps each place $P \in \mathcal{P}$ to a specific colour type in $\mathcal{S}$.
7. $\mathcal{I} : \mathcal{P} \rightarrow \mathcal{C}(\mathcal{P})_{\text{ms}}$ is an initialisation function, where $\mathcal{C}(P)_{\text{ms}}$ 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 $P \in \mathcal{P}$ for which $\mathcal{C}(P) \neq \mathbb{R}^0$, it contains a locally Lipschitz continuous function $\mathcal{V}_P : \mathcal{C}(P) \rightarrow \mathcal{C}(P)$.
9. $\mathcal{G}$ is a set of transition guards. For each $T \in \mathcal{T}_G$, it contains a transition guard $\mathcal{G}_T : \mathcal{C}(P(A_{\text{in},OE}(T))) \rightarrow \{\text{True}, \text{False}\}$. $\mathcal{G}_T(c)$ evaluates to True if c is in the boundary ∂G_T of an open subset G_T in $\mathcal{C}(P(A_{\text{in},OE}(T)))$. Here, if $P(A_{\text{in},OE}(T))$ contains more than one place, e.g. $P(A_{\text{in},OE}(T)) = \{P_i, \ldots, P_j\}$, then $\mathcal{C}(P(A_{\text{in},OE}(T)))$ is defined by $\mathcal{C}(P_i) \times \ldots \times \mathcal{C}(P_j)$. If $\mathcal{C}(P(A_{\text{in},OE}(T))) = \mathbb{R}^0$ then $\partial G_T = \emptyset$ and the guard will always evaluate to False.
10. $\mathcal{D}$ is a set of transition enabling rate functions. For each $T \in \mathcal{T}_D$, it contains an integrable transition enabling rate function $\delta_T : \mathcal{C}(P(A_{\text{in},OE}(T))) \rightarrow \mathbb{R}_0^+$, which, if T is evaluated from stopping time τ on, specifies a delay time equal to $\mathcal{D}_T(\tau) = \inf\{t | e^{-\int_\tau^t \delta_T(c_s)\mathrm{d}s} \leq u\}$, where u is a random number drawn from $U[0,1]$ at τ. If $\mathcal{C}(P(A_{\text{in},OE}(T))) = \mathbb{R}^0$ then δ_T is a constant function.
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 $\mathcal{C}(P(A_{\text{in},OE}(T)))$ into the set of probability measures on $\{0,1\}^{|A_{\text{out}}(T)|} \times \mathcal{C}(P(A_{\text{out}}(T)))$.

3.2 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 $\tau_1^{\text{pre}} = \inf\{t | T \text{ is pre-enabled at time } t\}$. Consider one token per ordinary and enabling arc in the input places of T and write $c_t \in \mathcal{C}(P(A_{\text{in},OE}(T)))$, $t \geq \tau_1^{\text{pre}}$, 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 requirement holds true. For $T \in \mathcal{T}_I$, the second requirement automatically holds true. For $T \in \mathcal{T}_G$, the second requirement holds true when $\mathcal{G}_T(c_t) = \text{True}$. For $T \in \mathcal{T}_D$, the second requirement holds true $\mathcal{D}_T(\tau_1^{\text{pre}})$ 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 $\mathcal{D}_T(\tau_1^{\text{pre}})$, 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 applies as follows: Suppose T is a delay transition that is pre-enabled at time τ and has vector of input colours c_t at time $t \geq \tau$. Then transition T is enabled at random time $\inf\{t : \exp\{-\int_\tau^t \delta_T(c_s)\mathrm{d}s\} \leq U_i\}$, with $\inf\{\} = +\infty$.

In case of competing enablings, 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 $A_{\text{in},O}(T)$ 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 $\mathcal{F}_T(\cdot; c_{\tau_1})$, again use is made of the Hilbert cube Ω: Let $\psi_2^T : [0, 1] \times \mathcal{C}(P(A_{\text{in},OE}(T))) \rightarrow \{0, 1\}^{|A_{\text{out}}(T)|} \times \mathcal{C}(P(A_{\text{out}}(T)))$ 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 $\{0, 1\}^{|A_{\text{out}}(T)|} \times \mathcal{C}(P(A_{\text{out}}(T)))$. 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 measure produces zero or one token along each output arc).

3.3 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 number 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 time 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.

4. Piecewise deterministic Markov processes into dynamically coloured Petri nets

This section shows that each piecewise deterministic Markov process can be represented by a DCPN, by providing a pathwise equivalent into-mapping from PDP into the set of DCPN processes.

THEOREM 1. For any arbitrary PDP with a finite domain $\mathbf{K}$ there exists P-almost surely a pathwise equivalent process generated by a DCPN $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ satisfying R_0 through R_4.

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

First, we construct a DCPN, the 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 of which are characterised in terms of the PDP elements $\{\mathbf{K}, \mathrm{d}(\theta), x_0, \theta_0, \partial E_\theta, g_\theta, \lambda, Q\}$ as follows:

- $\mathcal{P} = \{P_\theta; \theta \in \mathbf{K}\}$. Hence, for each $\theta \in \mathbf{K}$ there is one place P_θ.
- $\mathcal{T} = \mathcal{T}_G \cup \mathcal{T}_D \cup \mathcal{T}_I$, with $\mathcal{T}_I = \emptyset$, $\mathcal{T}_G = \{T^G_\theta; \theta \in \mathbf{K}\}$, $\mathcal{T}_D = \{T^D_\theta; \theta \in \mathbf{K}\}$. Hence, for each place P_θ there is one guard transition T^G_θ and one delay transition T^D_θ.
- $\mathcal{A} = \mathcal{A}_O \cup \mathcal{A}_E \cup \mathcal{A}_I$, with $|\mathcal{A}_I| = 0$, $|\mathcal{A}_E| = 0$, and $|\mathcal{A}_O| = 2|\mathbf{K}| + 2|\mathbf{K}|^2$.
- $\mathcal{N}$: The node function maps each arc in $\mathcal{A} = \mathcal{A}_O$ to a pair of nodes. These connected pairs of nodes are: $\{(P_\theta, T^G_\theta); \theta \in \mathbf{K}\} \cup \{(P_\theta, T^D_\theta); \theta \in \mathbf{K}\} \cup \{(T^G_\theta, P_\vartheta); \theta, \vartheta \in \mathbf{K}\} \cup (T^D_\theta, P_\vartheta); \theta, \vartheta \in \mathbf{K}\}$. Hence, each place P_θ has two outgoing arcs: one to guard transition T^G_θ and one to delay transition T^D_θ. Each transition has $|\mathbf{K}|$ outgoing arcs: one arc to each place in $\mathcal{P}$.
- $\mathcal{S} = \{\mathbb{R}^{\mathrm{d}(\theta)}; \theta \in \mathbf{K}\}$.
- $\mathcal{C}$: For all $\theta \in \mathbf{K}, \mathcal{C}(P_\theta) = \mathbb{R}^{\mathrm{d}(\theta)}$.
- $\mathcal{I}$: 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 $\theta \in \mathbf{K}$, $\partial G_{T^G_\theta} = \partial E_\theta$.
- $\mathcal{D}$: For all $\theta \in \mathbf{K}$, $\delta_{T^D_\theta}(\cdot) = \lambda(\theta, \cdot)$. Moreover, for the evaluation of the DCPN survivor functions, the same Hilbert cube applies as the one applied by the PDP.
- $\mathcal{F}$: If x denotes the colour of the token removed from place $P_\theta, (\theta \in \mathbf{K})$, at the transition firing, then for all $\vartheta' \in \mathbf{K}, x' \in E_{\vartheta'} : \mathcal{F}_{T^G_\theta}(e', x'; x) = Q(\vartheta', x'; \theta, x)$, where e' is the vector of length $|\mathbf{K}|$ containing a one at the component corresponding with arc $(T^G_\theta, P_{\vartheta'})$ and zeros elsewhere. For all $\theta \in \mathbf{K}, \mathcal{F}_{T^D_\theta} = \mathcal{F}_{T^G_\theta}$. Moreover, for the evaluation of the DCPN firing, the same Hilbert cube applies as the one applied by the PDP.
- 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 holds true. Rule R_4 is in effect when for particular θ, transitions T^G_θ and T^D_θ 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, the application of rule R_4 has no effect on the path of the DCPN process.

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, initially there is one token in place P_{θ_0}. Because each transition firing removes one token and produces one token, the number of tokens does not change for $t > 0$. Hence, for $t > 0$ there is one token and the possible places for this single token are $\{P_\vartheta; \vartheta \in \mathbf{K}\}$. Figure 2 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

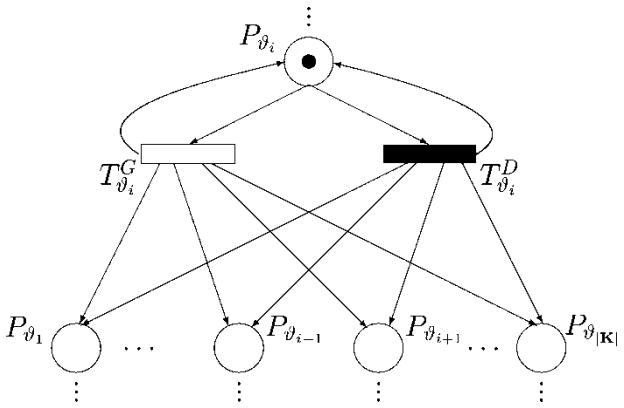

Figure 2. Part of a DCPN representing a PDP.

at the time of the next jump is evaluated according to two steps that are similar to those of PDP:

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. For this, use is made of a random sample from the Hilbert cube. 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: With one, or more (has probability zero), of the transitions enabled at time τ_k, its firing measure is evaluated. For this, use is made of a random sample from the Hilbert cube. The 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 above two steps are repeated in the same way from the new state on. The pathwise equivalence of the PDP and DCPN processes can be shown from the first stopping time to the next stopping time, and so on. From stopping time to stopping time both processes use the same independent realisations of the 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. Hence, from stopping time to stopping time, the PDP and the associated DCPN process have equivalent paths and equivalent stopping times. □

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.

5. Dynamically coloured Petri nets into piecewise deterministic Markov processes

Under some conditions, each DCPN can be represented by a piecewise deterministic Markov process. In this section this is shown by providing an into-mapping from DCPN into the set of PDPs.

THEOREM 2. For each stochastic process generated by a DCPN $(\mathcal{P},\mathcal{T},\mathcal{A},\mathcal{N},\mathcal{S},\mathcal{C},\mathcal{I},\mathcal{V},\mathcal{G},\mathcal{D},\mathcal{F})$ satisfying R_0 through R_4 there exists a unique probabilistically equivalent PDP 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, and for $t \rightarrow \infty$ the number of tokens remains finite.
- D_4 The initial marking is such, that no immediate transition is initially enabled.

Proof. 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 guards and firing measures, and revise the firing measures of the input transitions to these places, such that the new firings ensure that the corresponding guard transitions may be reached from one side only. This step is illustrated with an example in figure 3, where on the left 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 $\mathcal{C}(P_1) = \mathbb{R}$ 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. Infigure 3 on the right, place P_1 and transition T_2 have been copied. Transitions T_{2a} and T_{2b} get the same guard

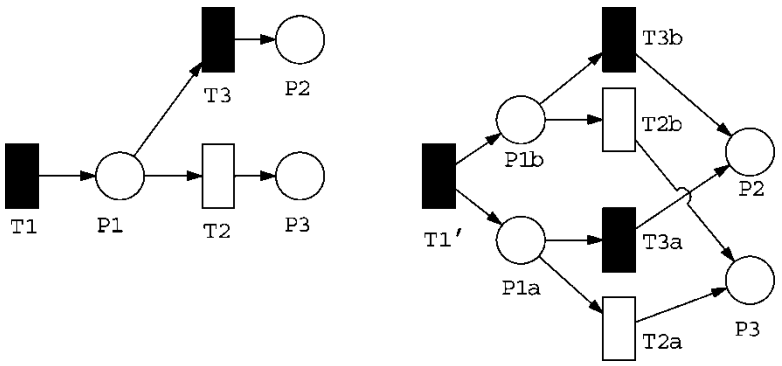

Figure 3. Example transformation to model DCPN enlargement. The original is on the left and the enlarged is on the right.

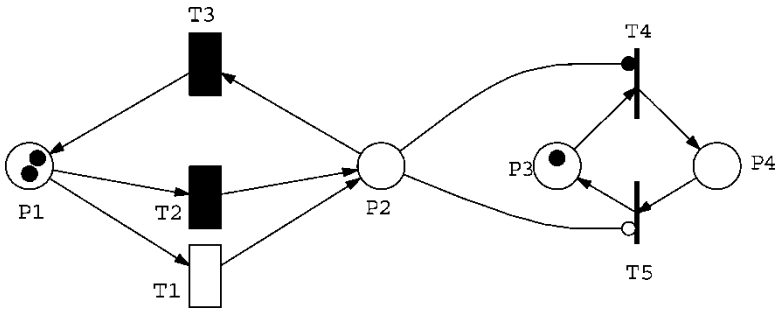

Figure 4. Example DCPN to explain reachability graph.

as T_2, but transition $T_1{}'$ gets a new firing measure 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. The second output transition T_3 of place P_1 also needs to be copied, but the output place of these copies can remain the same as before.

Proof continued: Let the 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 $\mathbf{K}, \mathrm{d}(\theta), \xi_0, g_\theta, \partial E_\theta, \lambda, Q$ and the PDP conditions C_1–C_4 are characterised in terms of this DCPN:

- **K**: The domain **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 $P_\mathrm{i}, i = 1, \ldots, |\mathcal{P}|$, 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†. 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 $V^1, \ldots, V^k$ notequal 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 RG, *exclusive* the nodes from which an immediate transition is enabled, form the discrete domain **K** of the PDP. To emphasise these nodes from which an

†Note that **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

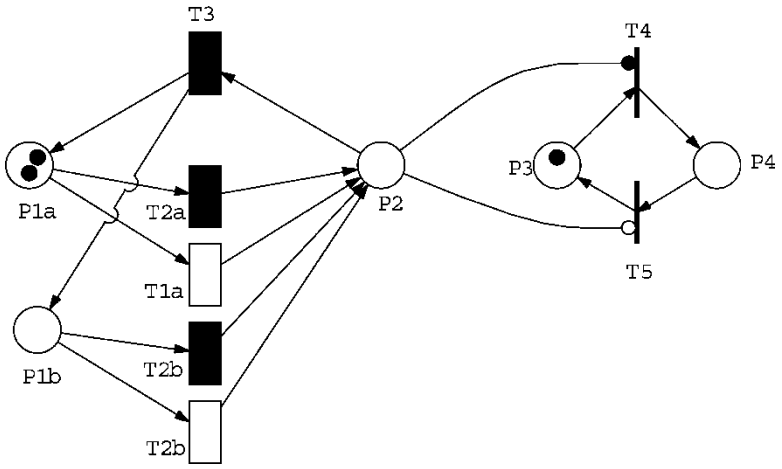

Figure 5. Enlarged DCPN of example in figure 4 .

immediate transition is enabled in the RG picture, they 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, **K** is a countable set, which satisfies the PDP conditions.

As an example, consider the following DCPN graph (figure 4), which is first enlarged as explained above (figure 5). The reachability graph (RG) for this example is given in figure 6. The enlarged graph initially has two tokens in place P_{1a} and one in P_3, and the unique ordering of places is (P_{1a}, P_{1b}, P_2, P_3, P_4) such that $V_0 = (2, 0, 0, 1, 0)$. This vector forms the first node of the RG.

Both T_{1a} and T_{2a} are pre-enabled. They both have two tokens per incoming arc in their input place, hence for both the transitions, two vectors of input colours are evaluated in parallel. If T_{1a} becomes enabled for one of these input tokens, it removes the corresponding token from P_{1a} 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 (1,0,1,1,0). Therefore, in the RG two arcs labelled by T_{1a} are drawn from (2,0,0,1,0) to the new node (1,0,1,1,0); this duplication of arcs characterises that T_{1a} has evaluated two vectors of input tokens in parallel. The same reasoning holds for transition T_{2a}: two arcs are drawn from (2,0,0,1,0) to (1,0,1,1,0). It may also happen that from (2,0,0,1,0), the guard transition T_{1a} 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 (0,0,2,1,0). Therefore, an additional arc labelled $T_{1a} + T_{1a}$ is drawn from (2,0,0,1,0) to (0,0,2,1,0). Unlike the case for T_{1a}, there is no arc drawn from (2,0,0,1,0) labelled by $T_{2a} + T_{2a}$, since T_{2a} is a delay transition, hence the probability that it is enabled by both its input tokens at the same time is zero. Now consider node (0,0,2,1,0). From this token distribution the immediate transition T_4 is enabled; its firing leads to (1,0,1,0,1). Since node (1,0,1,1,0) enables an immediate transition it is drawn in italics and is excluded from **K**. The resulting RG for this example is given in figure 6. So, for this example, $\mathbf{K} = \{(2, 0, 0, 1, 0), (0, 0, 2, 0, 1), (1, 0, 1, 0, 1), (0, 1, 1, 0, 1), (1, 1, 0, 1, 0), (0, 2, 0, 1, 0)\}$.

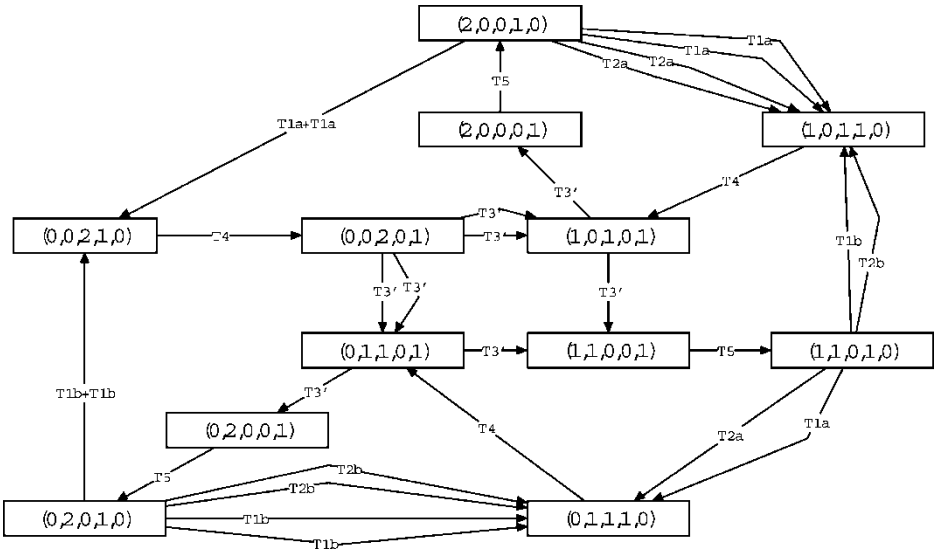

Figure 6. Reachability graph for the DCPN in figure 5.

Continuation of proof:

- $\mathrm{d}(\theta)$: The colour of a token in a place P is an element of $\mathcal{C}(P) = \mathbb{R}^{n(P)}$, therefore, $\mathrm{d}(\theta) = \sum_{i=1}^{|\mathcal{P}|} \theta_i \times n(P_i)$, 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 $x = \mathrm{Col}\{x^1, \ldots, x^{|\mathcal{P}|}\}$, with $x^i \in \mathbb{R}^{\theta_i \times n(P_i)}$, and with $\{1, \ldots, |\mathcal{P}|\}$ referring to the unique ordering of places adopted for the DCPN, g_θ is defined by $g_\theta(x) = \mathrm{Col}\{g_\theta^1(x^1), \ldots, g_\theta^{|\mathcal{P}|}(x^{|\mathcal{P}|})\}$, where for $x^i = \mathrm{Col}\{x^{i1}, \ldots, x^{i\theta_i}\}$, with $x^{ij} \in \mathbb{R}^{n(P_i)}$ for all $j \in \{1, \ldots, \theta_i\} : g_\theta^i(x^i) = \mathrm{Col}\{\mathcal{V}_{P_i}(x^{i1}), \ldots, \mathcal{V}_{P_i}(x^{i\theta_i})\}$. Here, $j \in \{1, \ldots, \theta_i\}$ refers to the unique ordering of tokens within their place defined for DCPN (see Section 3). Since, for all $P_i, \mathcal{V}_{P_i}$ is locally Lipschitz continuous, g_θ is also locally Lipschitz continuous.
- ∂E_θ : For each token distribution 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,...,T_m (note that this set may contain one transition multiple times, if multiple tokens are evaluated in parallel). 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 $G'_{T_i} = [G_{T_i} \times \mathbb{R}^{d(\theta)-d_i}] \in \mathbb{R}^{d(\theta)}$. 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 Section 3). If the set of pre-enabled guard transitions is empty, then $\partial E_\theta = \emptyset$.
- λ: For each token distribution θ, 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. If the set of pre-enabled delay transitions is empty, then $\lambda(\theta,\cdot)=0$.

- Q: For each $\theta\in\mathbf{K}, x\in E_\theta, \theta'\in\mathbf{K}$ and $x'\in E_{\theta'}, Q(\theta',x';\theta,x)$ is characterised by the RG, the sets $\mathcal{D},\mathcal{G}$ and $\mathcal{F}$ and the rules R_0–R_4. The RG 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, the characterisation of Q is given in the Appendix, an outline of which 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'). In these situations it is not trivial which firings measures 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 RG (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.

  Based on the above, we propose the following steps to construct $Q(\theta', x'; \theta, x)$, for any (θ', x', θ, x):
  1. Determine (using the RG) 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 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. If the set of enabled guard transitions is not empty, then rules R_1–R_4 determine the probability which of these transitions 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 RG 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 measures, 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 $x_0 \in \mathbb{R}^{d(\theta_0)}$ 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 ($\mathbb{E}N_t < \infty$) 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 Section 3: 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. □

6. Example DCPN and mapping to PDP

This section gives a simple example DCPN model and its mapping to PDP of the evolution of an aircraft. First, Subsection 6.1 explains how a DCPN that models a complex operation is generally constructed in three steps. In order to illustrate these steps, Subsection 6.2 presents a simple example of the evolution of one aircraft. Subsection 6.3 gives a DCPN that models this aircraft evolution and Subsection 6.4 explains the mapping of this DCPN example in a PDP.

6.1 DCPN construction and verification process

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 which contains a fixed number of tokens. 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, such that these interactions do not change the number of tokens per LPN.
3. In the third phase, one verifies whether the conditions D_1–D_4 under which a mapping to PDP is guaranteed to exist have been fulfilled. Because of the modularity and fixed number of tokens per LPN, these conditions can easily be verified per LPN, and subsequently per interaction between LPNs.

The additional advantage of this phased approach is that the total DCPN can be verified simultaneously by multiple domain experts. For example, a LPN model for a navigation system can be verified by a navigational system expert; a LPN model for a pilot can be verified by a human factors expert; interactions can be verified by a pilot.

6.2 Aircraft evolution example

This subsection presents a simple aircraft evolution example. The next subsections present a DCPN model and a mapping to PDP 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 aircraft evolves in *nominal* mode and 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 aircraft evolves in *non-nominal* mode and 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 has *landed*, i.e. its vertical position and velocity is equal to zero. Once landed, the aircraft is assumed not to depart anymore, hence the rate of change of its position and velocity equals zero. In order to model this aircraft evolution example mathematically, one could define three discrete valued processes $\{\kappa_t^1\}, \{\kappa_t^2\}, \{\kappa_t^3\}$, and an $\mathbb{R}^6$-valued process $\{x_t\}$:

- $\{\kappa_t^1\}$ represents the aircraft evolution mode assuming values in {*nominal*, *non-nominal*, *landed*};
- $\{\kappa_t^2\}$ represents the navigation mode assuming values in {*working*, *not-working*};
- $\{\kappa_t^3\}$ represents the engine mode assuming values in {*working*, *not-working*};
- $\{x_t\}$ represents the 3D position and velocity of the aircraft

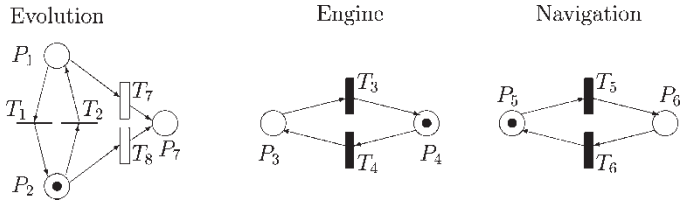

Figure 7. LPNs for the aircraft operations example. Place P_1 models Evolution Nominal, P_2 models evolution Non-nominal, P_3 models Engine system Not working, P_4 models Engine system Working, P_5 models Navigation system Not working, P_6 models Navigation system Working. P_7 models aircraft has landed.

Unfortunately, the process $\{\kappa_t, x_t\}$, with $\kappa_t = \text{Col}\{\kappa_t^1, \kappa_t^2, \kappa_t^3\}$, is not a PDP, since some κ_t combinations lead to immediate jumps, which is not allowed for PDP. This simple aircraft evolution example illustrates the kind of difficulty encountered when one wants to model a realistic problem directly as a PDP.

6.3 DCPN model for the aircraft evolution example

This subsection gives a DCPN instantiation that models the aircraft evolution example of the previous subsection. In order to illustrate the three-phased approach of Subsection 6.1, we first give the LPN graphs that have been identified in the first phase of the modelling. The entities identified are: aircraft evolution, navigation system, and engine system. This gives us three LPNs. The resulting graphs are given in figure 7.

The interactions between the Engine and Navigation LPN and the Evolution LPN 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 in figure 8. Notice that transition T_1 has to be replaced by two transitions T_{1a} and T_{1b} in order to allow both the engine and the navigation LPNs to influence transition T_1 separately from each other.

The graph in figure 8 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.

- $\mathcal{S}$: Two colour types are defined; $\mathcal{S} = \{\mathbb{R}^0, \mathbb{R}^6\}$.
- $\mathcal{C} : \mathcal{C}(P_1) = \mathcal{C}(P_2) = \mathcal{C}(P_7) = \mathbb{R}^6$, hence $n(P_1) = n(P_2) = n(P_7) = 6$. The first three colour components 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, $\mathcal{C}(P_i) = \mathbb{R}^0 = \emptyset$ hence $n(P_i) = 0$.
- $\mathcal{I}$: Place P_1 initially has a token with colour $z_0 = (x_0, v_0)'$, with $x_0 \in \mathbb{R}^2 \times (0, \infty)$ and $v_0 \in \mathbb{R}^3 \backslash \text{Col}\{0, 0, 0\}$. Places P_4 and P_6 initially each have a token without 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 there is no token colour function.
- $\mathcal{G}$: Transitions T_7 and T_8 have a guard that is defined by $\partial G_{T_7} = \partial G_{T_8} = \mathbb{R}^2 \times \{0\} \times \mathbb{R}^2 \times \{0\}$.
- $\mathcal{D}$: The enabling 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.

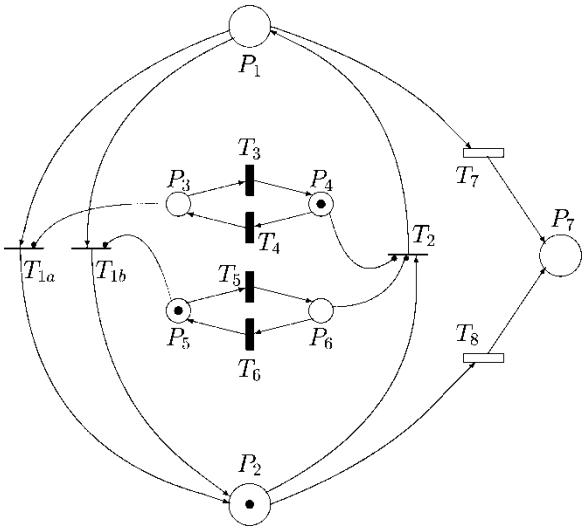

Figure 8. LPNs of figure 7 integrated into one DCPN.

- $\mathcal{F}$: Each transition has a unique output place, to which it fires a token with a colour (if applicable) equal to the colour of the token removed, i.e. for all T, $\mathcal{F}_T(1,\cdot;\cdot) = 1$.

6.4 Mapping to PDP

In this subsection, the DCPN aircraft evolution example is mapped to a PDP, following the construction in the proof of Theorem 2. Since the boundaries of the guard transitions T_7 and T_8 (i.e. $\partial G_{T_7} = \partial G_{T_8} = \mathbb{R}^2 \times \{0\} \times \mathbb{R}^2 \times \{0\}$) are always reached from one side only, there is no need to first enlarge the DCPN for these guard transitions (see Section 5).

The DCPN of figure 8 has seven places hence the RG has elements that are vectors of length 7. Since there is always one token in the set of places $\{P_1, P_2, P_7\}$, one token in $\{P_3, P_4\}$ and one token in $\{P_5, P_6\}$, the RG has $3 \times 2 \times 2 = 12$ nodes, see figure 9. However, four

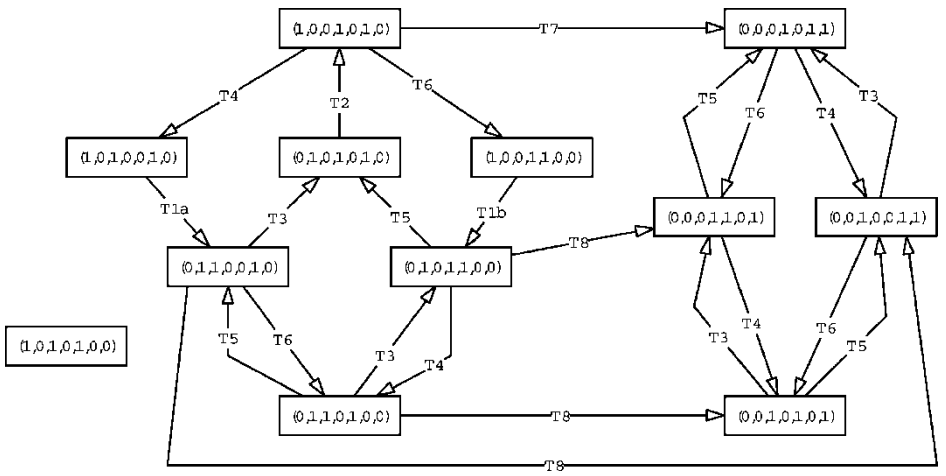

Figure 9. Reachability graph for the DCPN of Figure 8.

Table 1. Discrete modes in **K**.

Node	*Engine*	*Navigation*	*Evolution*
$m_1 = (1,0,0,1,0,1,0)$	Working	Working	Nominal
$m_2 = (0,1,1,0,0,1,0)$	Not working	Working	Non-nominal
$m_3 = (0,1,1,0,1,0,0)$	Not working	Not working	Non-nominal
$m_4 = (0,1,0,1,1,0,0)$	Working	Not working	Non-nominal
$m_5 = (0,0,0,1,0,1,1)$	Working	Working	Landed
$m_6 = (0,0,1,0,0,1,1)$	Not Working	Working	Landed
$m_7 = (0,0,1,0,1,0,1)$	Not working	Not working	Landed
$m_8 = (0,0,0,1,1,0,1)$	Working	Not working	Landed

nodes are excluded from **K**: nodes (1,0,1,0,0,1,0), (0,1,0,1,0,1,0) and (1,0,0,1,1,0,0) enable immediate transitions, and node (1,0,1,0,1,0,0) cannot be reached since it requires the enabling of a delay transition that is competing with an immediate transition, while due to DCPN rule R_0, an immediate transition always gets priority. Therefore, **K** consists of the remaining 8 nodes $\{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8\}$, which are specified in table 1.

Following Section 5, for each $\theta = (\theta_1, \ldots, \theta_7) \in \mathbf{K}$, the value of $\mathrm{d}(\theta)$ equals $\mathrm{d}(\theta) = \sum_{i=1}^{|\mathcal{P}|} \theta_i \times n(P_i)$. Since there is always one token in the set of places $\{P_1, P_2, P_7\}$, hence $\theta_1 + \theta_2 + \theta_7 = 1$, and since $n(P_1) = n(P_2) = n(P_7) = 6$ and $n(P_3) = n(P_4) = n(P_5) = n(P_6) = 0$, we find for all θ that $\mathrm{d}(\theta) = 6$.

Since initially there is a token in places P_1, P_4 and P_6, the initial mode θ_0 equals $\theta_0 = m_1 = (1,0,0,1,0,1,0)$. The PDP initial continuous state value equals the vector containing the initial colours of all initial tokens. Since the initial colour of the token in place P_1 equals z_0, and the tokens in places P_4 and P_6 have no colour, the PDP initial continuous state value equals z_0.

Following Section 5, with $\theta = (\theta_1, \ldots, \theta_7) \in \mathbf{K}$, for $x = \mathrm{Col}\{x^1, \ldots, x^7\}$, with $x^i \in \mathbb{R}^{\theta_i \times n(P_i)}$, the function g_θ is defined by $g_\theta(x) = \mathrm{Col}\{g_\theta^1(x^1), \ldots, g_\theta^7(x^7)\}$, where for $x^i = \mathrm{Col}\{x^{i1}, \ldots, x^{i\theta_i}\}$, with $x^{ij} \in \mathbb{R}^{n(P_i)}$ for all $j \in \{1, \ldots, \theta_i\} : g_\theta^i(x^i)$ satisfies $g_\theta^i(x^i) = \mathrm{Col}\{\mathcal{V}_{P_i}(x^{i1}), \ldots, \mathcal{V}_{P_i}(x^{i\theta_i})\}$. Since there is at most one token in each place, θ_i is either zero or one, hence either $x^i = \emptyset$ or $x^i = x^{i1}$. Since there is no token colour function for places $\{P_3, P_4, P_5, P_6\}$ and there is only one token in $\{P_1, P_2, P_7\}$, $g_\theta(x) = \mathcal{V}_1$ for $\theta = m_1$, $g_\theta(x) = \mathcal{V}_2$ for $\theta \in \{m_2, m_3, m_4\}$, and $g_\theta(x) = 0$ otherwise, see table 2.

The boundary ∂E_θ is determined from the transitions guards that, under token distribution θ, are enabled. This yields: for $\theta = m_1$, $\partial E_\theta = \partial G_{T_7} = \mathbb{R}^2 \times \{0\} \times \mathbb{R}^2 \times \{0\}$; for $\theta \in \{m_2, m_3, m_4\}$, $E_\theta = \partial G_{T_8} = \mathbb{R}^2 \times \{0\} \times \mathbb{R}^2 \times \{0\}$; for $\theta \in \{m_5, m_6, m_7, m_8\}$, $\partial E_\theta = \emptyset$.

Table 2. Example PDP components $g_\theta(\cdot)$ and λ as a function of $\theta \in \mathbf{K}$.

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 3. 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,	$Q(m_7, z; m_8, z) = \frac{\delta_4}{\delta_4+\delta_5}$,	$Q(m_5, z; m_8, z) = \frac{\delta_5}{\delta_4+\delta_5}$

The jump rate $\lambda(\theta, \cdot)$ is determined from the enabling rates corresponding with the set of delay transitions in $\mathcal{T}_D$ that, under token distribution θ, are pre-enabled. At each time, always two delay transitions are pre-enabled: either T_3 or T_4 and either T_5 or T_6. Hence $\lambda(\theta, \cdot) = \sum_{i=j,k} \delta_{T_i}(\cdot)$ if T_j and T_k are pre-enabled. See table 2 for the resulting λ's.

The probability measure Q is determined by the RG, the sets $\mathcal{D}$, $\mathcal{G}$ and $\mathcal{F}$ and the rules R_0–R_4. In table 3, $Q(\zeta; \xi) = p$ 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.

From a mathematical perspective, the PDP model has clear advantages. However, the PDP model does not show the compositional structure of the DCPN. Because of this, the DCPN model of Subsection 6.3 is simpler to comprehend and to verify against the aircraft evolution example description of Subsection 6.2. These complementary advantages from both perspectives tend to increase with the complexity of the operation considered.

7. Conclusions

Piecewise deterministic Markov processes (PDPs) can be used to describe virtually all complex continuous-time stochastic processes not involving diffusions. However, for complex practical problems it is often difficult to develop a PDP model, and have it verified both by mathematical and by multiple operational domain experts. This paper has introduced a novel Petri net, which is named dynamically coloured Petri net (DCPN) and has shown that under some mild conditions, any DCPN generated process can be mapped into a probabilistically equivalent PDP. Moreover, it is shown that any PDP with a finite discrete state domain can be mapped into a pathwise equivalent process which is generated by a DCPN. A consequence of both results is that there exist into-mappings between PDPs and DCPN processes. The development of a DCPN model for complex practical problems has similar compositional and modular specification advantages as basic Petri nets have over automata [5].

The key result of this paper is that this is the first time that proof of the existence of into-mappings between PDPs and Petri nets has been established. This significantly extends the modelling power hierarchy of [7,8] in terms of Petri nets and Markov processes. For other hybrid Petri nets [9,12,15–19] such into-mappings are not known and difficulties are foreseen in developing them. Due to the existence of these into-mappings, PDP theoretical results like stochastic analysis, stability and control theory, also apply to DCPN stochastic processes. The mapping of DCPN into PDP implies that any specific DCPN stochastic process can be

analysed as if it is a PDP, often without the need to first apply the transformation into a PDP as we did for the aircraft evolution example in Section 6. Because of this, for accident risk modelling in air traffic management, in [20] DCPNs are adopted for their compositional specification power and for their PDP inherited stochastic analysis power.

Acknowledgements

The authors of this paper would like to thank an anonymous reviewer and Arjan Van der Schaft of Twente University for providing their useful comments and suggestions for improvement.

References

[1] Davis, M.H.A., 1984, Piecewise deterministic Markov processes: a general class of non-diffusion stochastic models. *Journal Royal Statistical Society (B)*, **46**, 353–388.
[2] Davis, M.H.A., 1993, *Markov Models and Optimization* (Chapman & Hall).
[3] Protter, P., 1990, *Stochastic Integration and Differential Equations, a New Approach* (Springer-Verlag).
[4] Branicky, M.S., Studies in hybrid systems: modelling, analysis and control. Ph.D. thesis, Cambridge, MAMIT.
[5] Cassandras, C.G. and Lafortune, S., 1999, *Introduction to Discrete Event Systems* (Kluwer Academic Publishers).
[6] David, R. and Alla, H., 1994, Petri Nets for the modeling of dynamic systems—A survey. *Automatica*, **30**(2), 175–202.
[7] Malhotra, M. and Trivedi, K.S., 1994, Power-hierarchy of dependability-model types. *IEEE Transactions on Reliability*, **R-43**(3), 493–502.
[8] Muppala, J.K., Fricks, R.M. and Trivedi, K.S., 2000, Techniques for system dependability evaluation. In: W. Grasman (Ed.) *Computational Probability*, pp. 445–480 (The Netherlands: Kluwer Academix Publishers).
[9] Yang, Y.Y., Linkens, D.A., Banks, S.P., 1995, Modelling of hybrid systems based on Extended Coloured Petri Nets. In: P. Antsaklis *et al.* (Eds) *Hybrid Systems II* pp. 509–528 (Springer),.
[10] Jensen, K., 1992, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Vol. 1 (Springer-Verlag).
[11] Everdij, M.H.C., Blom, H.A.P. and Klompstra, M.B., 1997, Dynamically Coloured Petri Nets for Air Traffic Management safety purposes, *Proceedings 8th IFAC Symposium on Transportation Systems*, Chania, Greece, 184–189.
[12] Champagnat, R., *et al.,* 1998, Modelling and simulation of a hybrid system through Pr/Tr PN-DAE Model. *Proc of the 3th International Conference on Automation of Mixed Processes (ADPM'98)* (Reims).
[13] Ajmone-Marsan, M., Balbo, G., Chiola, G., Conte, G., Donatelli, S. and Franceschinis, G., 1991, An introduction to generalized stochastic Petri nets, . *Microelectronics and Reliability*, Vol. **31**, No. 4.
[14] Everdij, M.H.C. and 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* (Brittany, France: Saint-Malo), 16–18 June, pp. 355–360.
[15] Le Bail, J., Alla, H. and David, R., 1991, Hybrid Petri Nets, *European Control Conference, Grenoble, France*, 1472–1477
[16] David, R. and Alla, H., 1987, Continuous Petri Nets, 8th European workshop on application and theory of Petri Nets, Saragosse, 275–294
[17] Trivedi, K.S. and Kulkarni, V.G., 1993, FSPNs: Fluid stochastic Petri nets, lecture notes in computer science. In: M. Ajmone Marsan (Ed.) *Proceedings 14th International Conference on Applications and theory of Petri Nets*, Vol. 691, pp. 24–31 (Heidelberg: Springer Verlag).
[18] Giua, A. and Usai, E., 1996, High-level hybrid Petri nets: a definition, *Proceedings 35th Conference on Decision and Control, Kobe, Japan*, 148–150
[19] Demongodin, I. and Koussoulas, N.T., 1998, Differential Petri nets: representing continuous systems in a discrete-event world. *IEEE Transactions on Automatic Control*, **43**(4).
[20] 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, (1998). In: G.L. Donohue and A.G. Zellweger (Eds) *Air Transportation Systems Engineering*, AIAA, 463–480

Appendix: Characterisation of Q in terms of DCPN

In this appendix, Q is characterised in terms of DCPN, as part of the characterisation in Section 5 of PDP in terms of DCPN.

For each $\theta \in \mathbf{K}$, $x \in E_\theta$, $\theta' \in \boldsymbol{K}$ 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 RG (see Section 5), 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 measures to characterise Q.

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

Consider all arrows in the RG leaving node θ. These arrows are labelled by names of transitions which are pre-enabled in θ, for example T_1 (if T_1 is pre-enabled in θ), $T_1 + T_2$ (if T_1 and T_2 are both pre-enabled and there is a non-zero probability that they fire 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 $\mathcal{B}_\theta^G = \{B \in \mathcal{B}_\theta | \forall T \in B : T \in_G \text{ and } c_T^x \in \partial G_T\}$.
- 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 $p_B(\theta, x) = \frac{\delta_B(c_B^x)}{\sum_{T \in \mathcal{B}_\theta^D} \delta_T(c_T^x)}$.

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 **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 $RG_{\theta\theta'}$. Then, if θ and θ' are not connected in $RG_{\theta\theta'}$ 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 measures to characterise Q

From the previous step we have

- $Q(\theta', A; \theta, x) = 0$ if θ and θ' are not connected in $RG_{\theta\theta'}$ by at least one path.

If θ and θ' are connected then in $RG_{\theta\theta'}$ 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:

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

where $p_{\theta', x' | \theta, x, L}(\theta', A | \theta, x, L)$ 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, $p_{L|\theta,x}(L|\theta, x)$ 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 $p_{L|\theta,x}(L|\theta, x)$ is characterised for each $L \in \mathcal{L}_{\theta\theta'}$. Next, $p_{\theta', x' | \theta, x, L}(\theta', A | \theta, x, L)$ is characterised for each $L \in \mathcal{L}_{\theta\theta'}$.

Characterisation of $p_{L|\theta,x}(L|\theta, x)$ 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 $p_{L|\theta,x}(L|\theta, x)$ is determined by $p_{L|\theta,x}(L|\theta, x) = p_L(\theta, x) / \sum_{B \in \mathcal{L}_{\theta\theta'}} p_B(\theta, x)$, with $p_B(\theta, x)$ defined earlier.

Next, consider the situations where $RG_{\theta\theta'}$ 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 $p_{L|\theta,x}(L|\theta,x) = p_{T_k|\theta,x}(T_k|\theta,x)$, with the right-hand-side constructed as above for the case without immediate transitions. The same value $p_{T_k|\theta,x}(T_k|\theta,x)$ 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 $p_{T_k|\theta,x}(T_k|\theta,x)$ 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 $p_{L_1|\theta,x}(L_1|\theta,x) = p_{L_2|\theta,x}(L_2|\theta,x) = (1/2)p_{T_k|\theta,x}(T_k|\theta,x)$.

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

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

For probability $p_{\theta',x'|\theta,x,L}(\theta',A|\theta,x,L)$, 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.

- $p_{\theta',x'|\theta,x,L}(\theta',A|\theta,x,L) = 0$ 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:

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

  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 $T_1,\ldots, T_m$:

$$p_{\theta',x'|\theta,x,L}(\theta',A|\theta,x,L) = \sum_{c_{T_1}^x,\ldots,c_{T_k}^x} \int_{(e_1',c_1'),\ldots,(e_k',c_k')} \mathcal{F}_{T_1}(e_1',c_1';c_{T_1}^x) \times \cdots \times \mathcal{F}_{T_k}(e_k',c_k';c_{T_k}^x)$$

$$\times \mathbf{I}_{(\theta',A;e_1',c_1',c_{T_1}^x,\ldots,e_k',c_k',c_{T_k}^x)},$$

  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:

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

$$\times \mathbf{I}_{(\theta',A;e'_j,c'_j,e'_k,c'_k,c^x_T)},$$

  where $\mathbf{I}_{(\theta',A;e'_j,c'_j,e'_k,c'_k,c^x_T)}$ denotes indicator function for the event that the removal of $c^x_{T_k}$ 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 measures 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.