Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 305 (2014) 67–83

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Extending Propositional Dynamic Logic for Petri Nets[1]

Bruno Lopes [a,2] Mario Benevides [b,3]
Edward Hermann Haeusler [a,4]

[a] *Departamento de Informática*
Pontifícia Universidade Católica do Rio de Janeiro

[b] *Programa de Engenharia de Sistemas e Computação*
Universidade Federal do Rio de Janeiro

Abstract

Propositional Dynamic Logic (PDL) is a multi-modal logic used for specifying and reasoning on sequential programs. Petri Nets is a widely used formalism to specify and analyze concurrent programs with a very intuitive graphical representation. Petri-PDL is an extension of PDL, whose programs are Petri Nets, which combines the advantages of both formalisms using a compositional and structural approach to deal with Petri Nets.
In this work we present an extension of Petri-PDL to deal with Stochastic Petri Nets, the $\mathcal{DS}_3$ logic. This system is an alternative to model performance evaluation in a compositional and structural approach. We discuss about its soundness, decidability and completeness regarding our semantics and present a proof of EXPTime-completeness of its SAT problem. A usage example is presented.

Keywords: Dynamic Logic, Petri Nets, Stochastic Petri Nets, program verification

1 Introduction

Random phenomena are ubiquitous to our everyday experience. Weather changing and equipment failures, mostly unpredictable, are familiar to anyone. Real-time and fault-tolerant computer systems have to consider these randomness phenomena and should be designed taking some environmental parameters into account. The designer of old multi-user backuping system considered the failure probability of the hard-disks logical (tracks and sectors) and physical (wr-heads) components to build a fault-tolerant driver able to provide a quality of service according to the

[1] This work was supported by the Brazilian research agencies CNPq and CAPES. The reviewers provided useful suggestions which led to an improved version of the first manuscript.

[2] Email: bvieira@inf.puc-rio.br

[3] Email: mario@cos.ufrj.br

[4] Email: hermann@inf.puc-rio.br

http://dx.doi.org/10.1016/j.entcs.2014.06.006

requirements. Of course this can be one of the simplest example in this field of modelling and performance evaluation [5]. This article describes our proposal of joining a logical formalism (Dynamic Modal Logic) and a Mathematical Modelling tool for describing Random phenomena (Stochastic Petri Nets) by allowing that a marked Stochastic Petri Net π and a property α describe a requirement $\langle\pi\rangle\alpha$ meaning that at least one expected behaviour of π satisfy α, or, $[\pi]\alpha$ meaning that every behaviour of π satisfy α. The verification of these properties can be viewed as a kind of qualitative evaluation of the probabilistic parameters determined for π at the designing phase.

We denote by $\mathcal{DS}_3$ the logic proposed here, where the ordinary Petri Net programs are replaced by Stochastic Petri Net programs. We discuss soundness, decidability and completeness problems for $\mathcal{DS}_3$ with respect to our semantics and we present a polynomial reduction of the two-person corridor tiling game to $\mathcal{DS}_3$ SAT, hence showing its EXPTime-completeness.

In section 2 we present the theoretical and conceptual background motivating our proposal and briefly compare it with other approaches. Section 3 presents the technical Petri Net background; section 4 presents the $\mathcal{DS}_3$ logic and discuss its soundness, decidability and completeness with respect to our semantics and section 5 presents the two-person corridor tiling game reduction. A usage example is presented in section 6 and the conclusions and further work are in section 7.

2 Theoretical Background

Stochastic process is a mathematical modelling tool largely used for describing phenomena of a probabilistic nature as a function of time as a mandatory parameter [22]. Taking for free the definition of a probability space and random variable [15], a stochastic process $\{Y(t) : t \in [0,\infty)\}$ is a family of random variables defined over the same probability space and taking values in the same state space. Thus, a stochastic process can be understood as a family of functions of time that raise *sample paths*, i.e. trajectories in the state space. General stochastic processes can be quite complex. Among them, those that have no memory of the trajectory to reach the present state [6], also called Markov processes, have been widely considered for modelling computational processes. Markov processes with a discrete state space are denominated Markov chains. If the time is continuous, the term Continuous Time Markov Chain (CTMC) is commonly used. CTMC are natural models of computing systems considered in environments subject to randomness (internal or external). CMTCs compete with Queueing networks as tools for modelling and performance evaluation. However, the later does not provide clean mechanisms to describe synchronization, blocking and forking (i.e. consumer splitting). On the other hand, Petri Nets are quite good on describing these last mentioned aspects of a system. The proposal of Stochastic Petri Nets (SPN) brought an equilibrium

[5] The subarea of computer science that studies and develops tools and methods to help modelling and evaluating computer systems subject to work taking randomness phenomena into account.

[6] Mathematically one require that $\Pr(Y(t) \leq y : Y(t_n) = y_n \ \ldots Y(t_0) = y_0\ ) = \Pr(Y(t) \leq y : Y(t_n) = y_n)$, for $t > t_n \cdots > t_1 > t_0$.

between the modelling and the performance evaluation phases of systems designing. In [24] it is proposed SPNs, in order to avoid the translation of queueing networks defined in the modelling phase into complex CTMCs for the evaluation phase. In section 3 the definition of SPN requires that transitions are enabled according an exponential probability distribution (in fact a negative exponential distribution). This requirement is essential in order to ensure that the Stochastic process naturally derived from an SPN is a CTMC [27].

Dynamic Logics [9,13] are often used to deal with program reasoning and Propositional Dynamic Logic (PDL) [9] is one of its most well-known variants. In PDL each program P corresponds to a modality $[P]$. The formula $[P]\alpha$ means that, after every possible running of P, α holds, considering that P stops. Thus, $\langle P\rangle\alpha$, a shortened form for $\neg[P]\neg\alpha$), indicates that the property holds after some possible running of P. Dynamic Logics provide a large amount of systems and tools and can be used in Model Checking [5,11,17].

Petri Nets are not only a widely used formalism to deal with concurrent programs but also have an intuitive graphical interpretation. Taking advantage of this, Petri-PDL [1,18] replaces the conventional PDL programs by Marked Petri Net programs. So if π is a program with markup s, then the formula $\langle s, \pi\rangle\varphi$ means that after the running of this program designed by its Petri Net which initial markup is s, φ will eventually be true (also possible a $\Box$-like modality replacing the tags by brackets).

As we have already discussed, ordinary Petri Nets are not able to express random phenomena in a precise way. For example a two processor system with a shared memory where each processor has a different clock cannot be modelled by an ordinary Petri Net. This scenario may be taken as a problem of real time, or as a problem of productiveness. The latter turns out to be a probabilistic problem that can be modelled by a CTMC or by an SPN in a more elegant way. Thus, using probability attached to the transitions in the Petri Net is a way to obtain SPNs. The Dynamic Logic derived from this extended Petri Nets is our proposal. Stochastic Petri Nets are, hence, obtained by associating an exponentially distributed random variable to each ordinary Petri Net transition [12,26]. This random variable will control the firing rate of its transition. A transition will only fire when it is enabled and its timing achieves zero. This formalism has been used to deal with non-linear time-modelling [3,14,21].

There are some other well-known stochastic approaches to PDL, but we believe that the fact that the probabilistic feature present in each of these formalism was added in a non-structured way. We say that a probabilistic formalism is more structure than other, whenever the first has cleaner Markovian structures than the other. In this sense, the system P-Pr(DL) [6,7] which has no finite axiomatization, do not allow boolean combination of propositional variables and is defined only for regular programs is the less structured. Pr(DL) [8] which has the same limitations as P-Pr(DL) and is undecidable compares to the former. The system PPDL [16] computes the probability of a proposition being true in some state but the program is replaced by a measurable function, that is, its stochastic component is not compositional. Finally, PPDL$>r$ [32] can only describe situations where

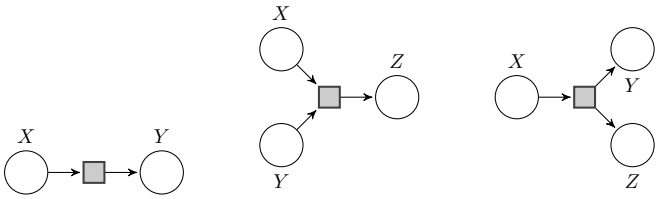
some probability is greater then a constant $r \in \mathbb{R}$ and PPDL> 0 [31] that can only describe situations where some probability is greater then zero, showing how the parameters of the modelling impose restriction in the queries, reverting completely the role of a model in a formal verification.

3 Background

In this section, we present the Petri Net systems used throughout this work and a brief review of Stochastic Petri Nets.

3.1 Petri Net convention system

Petri-PDL uses the Petri Net model defined by de Almeida and Haeusler [4]. In this model there are only three types of transitions which define all valid Petri Nets due to its compositions. These basic Petri Nets are as in figure 1.



(a) *Type* 1 : t_1 (b) *Type* 2 : t_2 (c) *Type* 3 : t_3

Fig. 1. Basic Petri Nets

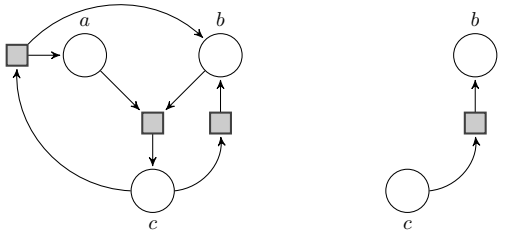
To compose more complex Petri Nets from these three basic kinds of composition, it is used a gluing procedure [4]. As an example, take the Petri Net of figure 2(a). It is a composition of the basic Petri Nets of figures 2(b), 2(c) and 2(d), where the same place names indicate that when gluing they will collapse.

3.2 Stochastic Petri Nets

A Stochastic Petri Net (SPN) [12,20,23,26] is a 5-tuple $\mathcal{P} = \langle P, T, L, M_0, \Lambda \rangle$, where P is a finite set of places, T is a finite set of transitions with $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$ and L is a function which defines directed edges between places and transitions and assigns a $w \in \mathbb{N}$, a multiplicative weight for the transition, as $L\colon (P \times T) \cup (T \times P) \to \mathbb{N}$ (in this work we assume $w = 1$ for all edges), M_0 is the initial markup and $\Lambda = \lambda_1, \lambda_2, \ldots, \lambda_n$ the firing rates of each transition.

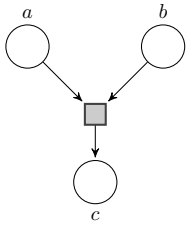
In an SPN the firing is determined by the markups and by the firing rate. To each transition $t_i \in T$ is associated a unique random variable with an exponential distribution with parameter $\lambda_i \in \Lambda$.

In the initial markup (M_0) each transition gets a firing delay through an occurrence of the random variable associated to it. Each firing delay is marking-dependent and the transition $t_i \in T$ firing rate at marking M_j is defined as $\lambda_i(M_j)$ and its average firing delay is $[\lambda_i(M_j)]^{-1}$. After a firing, each previously non-marking-enabled

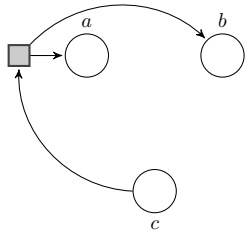


(a) Composed Petri Net

(b) Basic Petri Net Type 1



(c) Basic Petri Net Type 2



(d) Basic Petri Net Type 3

Fig. 2. Example of Petri Net composition with its basic Petri Nets

transition gets a new firing delay by sampling its associated random variable. A transition previously marking-enabled that keeps marking-enabled has its firing delay decreased in a constant speed. When a transition firing delay reaches zero, this transition fires.

We define the preset of $t \in T$, denoted by ${}^\bullet t$, as the set of all $s_k \in S$ that origins an edge to t. The postset of t, denoted by $t^\bullet$ is defined as the set of all $s_\ell \in S$ that t origins an edge to. We say that a transition t is enabled if, and only if, there is at least one token in each place $p \in {}^\bullet t$.

Given a markup M_j of a Petri Net, a transition t_i is enabled on M_j if and only if $\forall x \in {}^\bullet t_i, M_j(x) \geq 1$ and $\lambda_i(M_j) = \min(\lambda_1(M_j), \lambda_2(M_j), \ldots, \lambda_n(M_j))$, where ${}^\bullet t_i$ is the preset of t_i. A new markup generated by setting a transition which is enabled is defined in the same way as in a Marked Petri Net, i.e.

$$M_{j+1}(x) = \begin{cases} M_j(x) - 1 \ \forall x \in {}^\bullet t \setminus t^\bullet \\ M_j(x) + 1 \ \forall x \in t^\bullet \setminus {}^\bullet t \\ M_j(x) \qquad \text{other case} \end{cases} . \tag{1}$$

A new firing delay for a transition t_i for a markup M_j is defined as:

(i) if t_i fires then a new occurrence of the random variable associated with it is the new firing delay;

(ii) if t_i was disabled and has just been enabled then a new occurrence of the random variable associated with it is the new firing delay;

(iii) other case, the value of the firing delay of t_i must be decreased.

That is:

$$\lambda_i(M_{j+1}) \begin{cases} = \text{new}_e(\lambda_i) \text{ if } \begin{cases} \begin{cases} \forall x \in {}^\bullet t_i, M_j(x) \geq 1 \\ \lambda_i(M_j) \leq \min(\lambda_1(M_j), \ldots, \lambda_n(M_j)) \end{cases} \\ \text{or} \\ \begin{cases} \exists x \in {}^\bullet t_i, M_j(x) < 1 \\ \forall x \in {}^\bullet t_i, M_{j+1}(x) \geq 1 \end{cases} \end{cases} \\ < \lambda_i(M_j) \quad \text{other case} \end{cases} \tag{2}$$

where $\text{new}_e(\lambda)$ denotes a new occurrence of the random variable exponentially distributed with parameter λ associated to t_i.

The minimum of two random variables with parameters, respectively, λ_1 and λ_2, is a random variable with exponential distribution of parameter $\lambda_1 + \lambda_2$, the sojourn time in a marking M_j is a random variable exponentially distributed with mean

$$\left[\sum_{i:\forall k \in {}^\bullet t_i, M_j(k)>0} \lambda_i(M_j) \right]^{-1}. \tag{3}$$

As all random variables have an exponential distribution, then it is possible to compute the probability of an enabled transition t_i to have the minimum firing delay (i.e. the probability of t_i fires immediately) at a marking M_j:

$$\Pr(t_i \mid M_j) = \frac{\lambda_i(M_j)}{\sum_{k:\forall \ell \in {}^\bullet t_k, M_j(\ell)>0} \lambda_k(M_j)}. \tag{4}$$

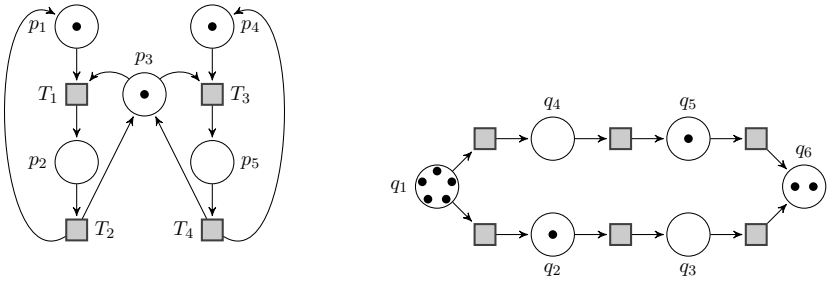
To illustrate the usage of Stochastic Petri Nets, we can model a two processes system that share a resource. Process 1 is I/O bound and process 2 is CPU bound, as in figure 3(a). The great difference in the amount of requests of input can be modelled by setting the Λ values (i.e. $\lambda_1 > \lambda_3$). Figure 3(b) presents a simple parallel system modelled in a SPN where the tokens denote processes. The Λ values determines if it would be faster in some of the ways. The probability of a process goes from q_1 to q_2 instead of to q_4 can be computed according to equation (4).

4 The $\mathcal{DS}_3$ logic

The language of $\mathcal{DS}_3$ is the same than the language of Petri-PDL. The difference is that the ordinary Petri Net program will be replaced by a Stochastic Petri Net program (more details on how to deal with its behaviour in the frame definition 4.4); it consists of

Propositional symbols: p, q..., where Φ is the set of all propositional symbols

Place names: e.g.: $a, b, c, d \ldots$



(a) A two processes system (b) A simple parallel system

Fig. 3. Stochastic Petri Net examples

Transition types: $T_1 : at_1b$, $T_2 : abt_2c$ and $T_3 : at_3bc$, each transition name has a unique type

Petri Net Composition symbol: $\odot$

Sequence of names: $S = \{\epsilon, s_1, s_2, \dots\}$, where ϵ is the empty sequence. We use the notation $s \prec s'$ to denote that all names occurring in s also occur in s'.

Definition 4.1 Programs:

We use π to denote a Stochastic Petri Net program and s a sequence of names (the markup of π).

Basic programs: $\pi_b ::= at_1b \mid at_2bc \mid abt_3c$ where t_i is of type $T_i, i = 1, 2, 3$

Stochastic Petri Net Programs: $\pi ::= s, \pi_b \mid \pi \odot \pi$

Definition 4.2 Formula

A $\mathcal{DS}_3$ formula is defined as: $\varphi = p \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle s, \pi\rangle\varphi$.

We use the standard abbreviations $\bot \equiv \neg\top$, $\varphi \vee \phi \equiv \neg(\neg\varphi \wedge \neg\phi)$, $\varphi \rightarrow \phi \equiv \neg(\varphi \wedge \neg\phi)$ and $[s, \pi]\varphi \equiv \neg\langle s, \pi\rangle\neg\varphi$, and π is a Stochastic Petri Net program with markup s.

The firing of a transition in $\mathcal{DS}_3$ is defined according to the firing function in definition 4.3.

Definition 4.3 We define the *firing* function $f : S \times \pi_b \rightarrow S$ as follows

- $f(s, at_1b) = \begin{cases} s_1bs_2, \ if \ s = s_1as_2 \\ \epsilon, \ if \ a \not\prec s \end{cases}$
- $f(s, at_3bc) = \begin{cases} s_1s_2bc, \ if \ s = s_1as_2 \\ \epsilon, \ if \ a \not\prec s \end{cases}$
- $f(s, abt_2c) = \begin{cases} s_1cs_2s_3, \ if \ s = s_1as_2bs_3 \\ \epsilon, \ if \ a, b \not\prec s \end{cases}$
- $f(\epsilon, \eta) = \epsilon$, for all petri nets programs η.

Definition 4.4 $\mathcal{DS}_3$ Frame

A frame for $\mathcal{DS}_3$ is a 5-tuple $\mathcal{F}_3 = \langle W, R_\pi, M, \Pi, \Lambda, \delta\rangle$ where

- W is a non-empty set of states

- $M\colon W \to S$
- Π is a finite Stochastic Petri Net such that for any program π used in a modality, $\pi \in \Pi$ (i.e. π is a subnet of Π)
- $\Lambda(\pi) = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$ is the sequence of $\mathbb{R}^+$ values denoting the fire rate of each transition of $\pi_1 \odot \pi_2 \odot \cdots \odot \pi_n = \pi \in \Pi$
- $\delta(w, \pi) = \langle d_1, d_2, \ldots, d_n \rangle$ is the sequence of firing delays of the program $\pi \in \Pi$ in the world $w \in W$ respectively for each program $\pi_1 \odot \pi_2 \odot \cdots \odot \pi_n = \pi$, satisfying the following conditions (let $s = M(w)$ and $r = M(v)$)
  - if $wR_{\pi_b}v$, $f(r, \pi_b) = \epsilon$ then $\delta(w, \pi_b) = \delta(v, \pi_b)$
  - if $f(s, \pi_b) = \epsilon$, $f(r, \pi_b) \neq \epsilon$ and $wR_{\pi_b}v$, $\delta(v, \pi_b)$ is an occurrence of a random variable of exponential distribution with parameter $\Lambda(\pi_b)$; by the inversion theorem, $\delta(v, \pi_b) = \frac{\ln(1-u)}{-\Lambda(\pi_b)}$ where u is an occurrence of a uniform random variable
  - if $f(s, \pi_b) \neq \epsilon$, $f(r, \pi_b) \neq \epsilon$ and $wR_{\pi_b}v$, $\delta(v, \pi_b) < \delta(w, \pi_b)$
- R_α is a binary relation over W, for each basic program $\alpha \in \pi_b$, satisfying the following conditions (let $s = M(w)$)
  - if $f(s, \alpha) \neq \epsilon$ and $\delta(w, \alpha) = \min(\delta(w, \Pi))$, $wR_\alpha v$ iff $f(s, \alpha) \prec M(v)$
  - if $f(s, \alpha) = \epsilon$ or $\delta(w, \alpha) \neq \min(\delta(w, \Pi))$, $wR_\alpha v$ iff $w = v$
- we inductively define a binary relation R_η, for each Petri Net program $\eta = \eta_1 \odot \eta_2 \odot \cdots \odot \eta_n$, as

$$R_\eta = \{(w, v) \mid \exists \eta_i, \exists u \text{ such that } s_i \prec M(u) \text{ and } wR_{\eta_i}u \text{ and } \delta(w, \eta_i) = \min(\delta(w, \Pi)) \text{ and } uR_\eta v\}$$

  where $s_i = f(s, \eta_i)$, for all $1 \leq i \leq n$.

Lemma 4.5 *Reflexivity over empty occurrences*

For any Petri Net program π, $f(\epsilon, \pi) = \epsilon$, $R_{\epsilon,\pi}$ is reflexive.

Proof. This proof is straightforward from definitions 4.3 and 4.4. □

Definition 4.6 $\mathcal{DS}_3$ Model

A model for $\mathcal{DS}_3$ is a pair $\mathcal{M} = \langle \mathcal{F}_3, \mathbf{V} \rangle$, where $\mathcal{F}_3$ is a $\mathcal{DS}_3$ frame and $\mathbf{V}$ is a valuation function $\mathbf{V}\colon \Phi \to 2^W$.

Definition 4.7 Semantic notion of $\mathcal{DS}_3$

Let $\mathcal{M}_3$ be a model for $\mathcal{DS}_3$. The notion of satisfaction of a formula φ in $\mathcal{M}_3$ at a state w, denoted by $\mathcal{M}_3, w \Vdash \varphi$ is inductively defined as follows.

- $\mathcal{M}_3, w \Vdash p$ iff $w \in \mathbf{V}(p)$
- $\mathcal{M}_3, w \Vdash \top$ always
- $\mathcal{M}_3, w \Vdash \neg\varphi$ iff $\mathcal{M}_3, w \nVdash \varphi$
- $\mathcal{M}_3, w \Vdash \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}_3, w \Vdash \varphi_1$ and $\mathcal{M}_3, w \Vdash \varphi_2$
- $\mathcal{M}_3, w \Vdash \langle s, \eta \rangle \varphi$ if there exists $v \in W$, $wR_\eta v$ and $\Pr(\mathcal{M}_3, v \Vdash \langle s, \eta_b \rangle \varphi \mid \delta(v, \Pi)) > 0$ where η_b is some basic program of η (i.e. after the running of η beginning in the world w, if there is a world v accessible from w by R_η, as in definition 4.4,

where η stops and φ holds)

If φ is satisfied in all states of $\mathcal{M}_3$ then φ is valid in $\mathcal{M}_3$, denoted by $\mathcal{M}_3 \Vdash \varphi$; and if φ is valid in any model then φ is valid, denoted by $\Vdash \varphi$.

Lemma 4.8 *Truth Probability of a Modality*
The probability of $\mathcal{M}_3,w \Vdash \langle s,\pi_b\rangle\varphi$ *is (let* $s = M(w)$*)*

$$\Pr(\mathcal{M}_3,w \Vdash \langle s,\pi_b\rangle\varphi \mid \delta(w,\Pi)) = \frac{\delta(w,\pi_b)}{\sum_{\pi_b\in\Pi: f(s,\pi_b)\neq\epsilon}\delta(w,\pi_b)}.$$

Proof. This proof is straightforward from relation (4) and definition 4.4. □

4.1 Axiomatic System

We consider the following set of axioms and rules, where p and q are proposition symbols, φ and ψ are formulas, $\eta = \eta_1 \odot \eta_2 \odot \cdots \odot \eta_n$ is a Petri Net program and π is a Marked Petri Net program.

(PL) Enough propositional logic tautologies

(K) $[s,\pi](p \to q) \to ([s,\pi]p \to [s,\pi]q)$

(Du) $[s,\pi]p \leftrightarrow \neg\langle s,\pi\rangle\neg p$

(PC$_3$) $\langle s,\eta\rangle\varphi \leftrightarrow \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$, where $s_i = f(s,\eta_i)$, for all $1 \leq i \leq n$.

(R$_{3_\epsilon}$) $\langle s,\eta\rangle\varphi \leftrightarrow \varphi$, if $f(s,\eta) = \epsilon$

(Sub) If $\Vdash \varphi$, then $\Vdash \varphi^\sigma$, where σ uniformly substitutes proposition symbols by arbitrary formulas.

(MP) If $\Vdash \varphi$ and $\Vdash \varphi \to \psi$, then $\Vdash \psi$.

(Gen) If $\Vdash \varphi$, then $\Vdash [s,\pi]\varphi$.

4.2 Soundness

The axioms **(PL)**, **(K)** and **(Du)** and the rules **(Sub)**, **(MP)** and **(Gen)** are standard in the modal logic literature.

Lemma 4.9 *Validity of* $\mathcal{DS}_3$ *axioms*

(i) $\Vdash$ **PC$_3$**

(ii) $\Vdash$ **R**$_{\epsilon_3}$

Proof.

(i) $\Vdash$ **PC$_3$**:
Suppose that there is a world w from a model $\mathcal{M}_3 = \langle W', R_\eta, M, \Pi, \Lambda, \delta, \mathbf{V}\rangle$ where PC$_3$ is false. For PC$_3$ to be false in w, there are two cases:
(a) Suppose $\mathcal{M}_3,w \Vdash \langle s,\eta\rangle\varphi$ (1) and
$\mathcal{M}_3,w \nVdash \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$ (2).

(1) iff there is a world v such that $wR_\eta v$ and $\Pr(\mathcal{M}_3,v \Vdash \langle s,\eta_b\rangle\varphi \mid \delta(v,\Pi)) > 0$ (3).
By definition 4.4 $R_\eta = \{(w,v) \mid \exists\eta_i, \exists u$ such that $s_i \prec M(u)$ and $wR_{\eta_i}u$ and $\delta(w,\eta_i) = \min(\delta(w,\Pi))$ and $uR_\eta v\}$,
from (3) $\mathcal{M}_3,u \Vdash \langle s_i,\eta\rangle\varphi$ and $\mathcal{M}_3,w \Vdash \langle s,\eta_i\rangle\langle s_i,\eta\rangle\varphi$, which implies that $\Pr(\mathcal{M}_3,w \Vdash \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi \mid \delta(w,\Pi)) > 0$ (4).
From (4) $\mathcal{M}_3,w \Vdash \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$, which contradicts (2).

(b) Suppose $\mathcal{M}_3,w \Vdash \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$ (2), iff for some i $(1 \le i \le n)$, $M,w \Vdash \langle s,\eta_i\rangle\langle s_i,\eta\rangle\varphi$ iff
there is a u such that $wR_{\eta_i}u$, $\Pr(\mathcal{M}_3,w \Vdash \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi \mid \delta(w,\Pi)) > 0$ (3)
iff there is a v such that $uR_\eta v$ and $\mathcal{M}_3,v \Vdash \varphi$ (4).
By definition 4.4, (3) and (4) we have $wR_\eta v$, $\Pr(\mathcal{M}_3,w \Vdash \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi \mid \delta(w,\Pi)) > 0$ and $\mathcal{M}_3,v \Vdash \varphi$. Thus, $\mathcal{M}_3,w \Vdash \langle s,\eta\rangle\varphi$.

So, PC_3 is valid.

(ii) $\Vdash \mathbf{R_{\epsilon_3}}$**:**
Suppose that there is a world w from a model $\mathcal{M}_3 = \langle W', R_\eta, M, \Pi, \Lambda, \delta, \mathbf{V}\rangle$ where R_ϵ is false. For R_{ϵ_3} be false in w, there are two cases:

(a) Suppose $\mathcal{M}_3,w \Vdash \langle\epsilon,\eta\rangle\varphi$ (1) and
$\mathcal{M}_3,w \nVdash \varphi$ (2)
(1) iff there is a v such that $wR_{\epsilon,\eta}$ and $\Pr(\mathcal{M}_3, w \Vdash \langle\epsilon,\eta\rangle\varphi \mid \delta(w,\Pi)) > 0$.
As $f(\epsilon,\eta) = \epsilon$, by lemma 4.5, $w = v$, $wR_\eta w$ and $\mathcal{M}_3, w \Vdash \varphi$, which contradicts (2).

(b) Suppose $\mathcal{M}_3,w \nVdash \langle\epsilon,\eta\rangle\varphi$ (1) and
$\mathcal{M}_3,w \Vdash \varphi$ (2). (1) iff $\Pr(\mathcal{M}_3,w \Vdash \langle\epsilon,\eta\rangle\varphi \mid \delta(w,\Pi)) = 0$.
As $f(\epsilon,\eta) = \epsilon$, by lemma 4.5, $wR_\eta w$ and, by definition 4.4, $M_3,w \nVdash \varphi$, which contradicts (2).

So, R_{ϵ_3} is valid. □

4.3 Completeness

As pointed out by the work of A. Mazurkiewicz [28,29], logics that deal with Petri Nets are usually incomplete, due to the possibility of a place always increasing its token amount (up to countable infinite). In order to being able to get decidability and completeness results we will restrict ourselves to a subset of Petri Nets we call *normalised* Petri Nets. Basically, a normalised Petri Net is a Petri Net composed as in section 3.1 which does not accumulate an infinity amount of tokens. From now on, we will consider only normalised Petri Nets.

The completeness proof for $\mathcal{DS}_3$ is done in the same way as in Blackburn et. al. [2], Harel et. al. [13] and Goldblatt [10].

Theorem 4.10 *The $\mathcal{DS}_3$ logic is complete for normalised SPN programs*

Proof. (Sketch)
The first step is to define the Fischer-Ladner closure (FL), where $FL(\varphi)$ denotes the smallest set containing φ which is closed under sub formulae.

Then, given a $\mathcal{DS}_3$ formula φ and a $\mathcal{DS}_3$ model $\mathcal{K}_3 = \langle W, R_\eta, M, \Pi, \Lambda, \delta, \mathbf{V}\rangle$, we define a new model $\mathcal{K}_3^\varphi = \langle W^\varphi, R_\eta^\varphi, M^\varphi, \Pi^\varphi, \Lambda^\varphi, \delta^\varphi, \mathbf{V}^\varphi\rangle$, the filtration of $\mathcal{K}_3$ by $FL(\varphi)$, as follows.

The relation $\equiv$ over the worlds of $\mathcal{K}_3$ is defined as

$$u \equiv v \leftrightarrow \forall \phi \in FL(\varphi), \Pr(\mathcal{K}_3, u \Vdash \phi \mid \delta(u, \Pi)) = \Pr(\mathcal{K}_3, v \Vdash \phi \mid \delta(v, \Pi))$$

and the relation R_η^φ is defined as

$$[u] R_\eta^\varphi [v] \leftrightarrow (\exists u' \in [u] \wedge \exists v' \in [v] \wedge u' R_\eta v').$$

(a) $[u] = \{v \mid v \equiv u\}$
(b) $W^\varphi = \{[u] \mid u \in W\}$
(c) $[u] \in \mathbf{V}^\varphi(p)$ iff $u \in \mathbf{V}(p)$
(d) $M^\varphi([u]) = \langle s_1, s_2, \dots\rangle$ where for all $j \geq 1, v_j \in [u]$ iff $M(v_j) = s_j$
(e) $\Pi^\varphi = \Pi$
(f) $\Lambda^\varphi = \Lambda$
(g) $\delta^\varphi([u], \pi) = \langle d_1, d_2, \dots, d_n\rangle$ where $\pi = \pi_1 \odot \pi_2 \odot \cdots \odot \pi_n$ and $d_i = \int_0^i h(\delta(u, \Pi))du$ where h is the function that decreases the firing delays, according to the stability process [12].

We show that the number of worlds (states) in a filtered model is finite, hence $\mathcal{DS}_3$ is decidable. Taking a Canonical Model of $\mathcal{DS}_3$ (a model where the set of worlds is the set of all maximal consistent sets of formulae) for a language $\mathcal{L}$, $\mathcal{C}_3^\mathcal{L} = \langle W^\mathcal{L}, R_\pi^\mathcal{L}, M^\mathcal{L}, \Pi^\mathcal{L}, \Lambda^\mathcal{L}, \delta^\mathcal{L}, \mathbf{V}^\mathcal{L}\rangle$, we prove that $[s, \pi]\varphi \in u$ iff in all v such that $u R_\pi^\mathcal{L} v$, $\varphi \in v$. Then we prove that for any $w \in W^\mathcal{L}$, $w \Vdash \varphi$ iff $\varphi \in w$. Hence we may show that if $\Vdash \varphi$ then $\vdash \varphi$. The complete proof is available at http://www.tecmf.inf.puc-rio.br/BrunoLopes/Proofs. □

Corollary 4.11 *Petri-PDL completeness*

As Petri-PDL is subsumed by the $\mathcal{DS}_3$ logic in the case where all transitions have the same firing rate, then Petri-PDL is also complete for normalised Petri Nets.

5 $\mathcal{DS}_3$ Satisfiability complexity

In this section we present a polynomial reduction of a well-know EXPTime-complete problem to $\mathcal{DS}_3$ satisfiability (SAT) problem: the two-person corridor tiling game. The $\mathcal{DS}_3$ SAT problem concerns in determine if there is an interpretation that satisfies a $\mathcal{DS}_3$ formula.

Lemma 5.1 *The satisfiability of $\mathcal{DS}_3$ is EXPTime-hard.*

Proof. The proof goes by reducing the two-person corridor tiling problem to the $\mathcal{DS}_3$ SAT problem, following the methodology of Blackburn et al. [2].

In the two-person corridor tiling game, two players (Eloise and Abelard) must place square tiles in a grid so that colours match (each tile side may have a different color). The players begin with a finite amount of tiles (colours randomly defined) and the beginning of the grid has a special colour (say white) and there is a special tile for Eloise that if placed on column 1 then Eloise wins. When the game begins Eloise should put a tile in the column 0; in his turn, Abelard must place a tile in the following position on the grid. After the end of the row (for an instance n of the game, the game has n columns), the player must place a tile in the next row, column 0. If no player is able to make a valid move or there are no tiles then Abelard wins.

Given an instance $\mathcal{T} = (n, \{T_0, \ldots, T_{s+1}\})$ of the two-person corridor tiling game where n is the width of the corridor and T_i are the tiling types, we will construct a formula φ^τ such that

(i) If Eloise has a winning strategy, φ^τ is satisfiable at the root of some game tree for $\mathcal{T}$ (viewed as a regular $\mathcal{DS}_3$ model such that for a formula of size n the size of the model will be a^n where $a > 1$).

(ii) If φ^τ is satisfiable, then Eloise has a winning strategy in the game $\mathcal{T}$.

(iii) The formula φ^τ can be computed in time polynomial in n and s.

The formula φ^τ describes the game and states necessary and sufficient conditions for Eloise to win. To construct φ^τ, we will use the following proposition letters:

$\mathbf{t_0}, \ldots, \mathbf{t_{s+1}}$ to represent the tiles, where t_0 is white;

$\mathbf{p_1}, \ldots, \mathbf{p_n}$ to indicate where the tile must be placed in the current round;

$\mathbf{c_i(t)}, \mathbf{0 \leq i \leq n+1}, \forall \mathbf{t} \in \{\mathbf{t_0}, \ldots, \mathbf{t_{s+1}}\}$ to indicate the type t of previously placed tile in column i;

$\mathbf{w}$ to indicate that the current position is a winning position for Eloise.

The General schema of a Petri Net (η) that models the game is in figure 4, where there is one transition similar to $\mathrm{R}^\star$ for each row r such that $1 < r < n$ to denote that a new row has begun. The sequence s denotes the initial markup of the Petri Net, that is, one token in Row_1 and the tokens needed to denote the initial set of pieces of Eloise and Abelard. Each place is described bellow.

EC Eloise can play

EH Eloise has piece

EP Eloise plays

AC Abelard can play

AH Abelard has piece

AP Abelard plays

Col$_1$,...,Col$_n$ Each column of the game

Row$_1$,...,Row$_n$ Each row of the game

Then, the beginning of the game is described as $e \wedge p_1 \wedge c_0(\text{white}) \wedge c_1(t_{I_1}) \wedge \cdots \wedge c_n(t_{I_n}) \wedge c_{n+1}(\text{white})$, where $I_i, 1 \leq i \leq n$ denotes the initial tiles.

The set of formulas that rules the game is below.

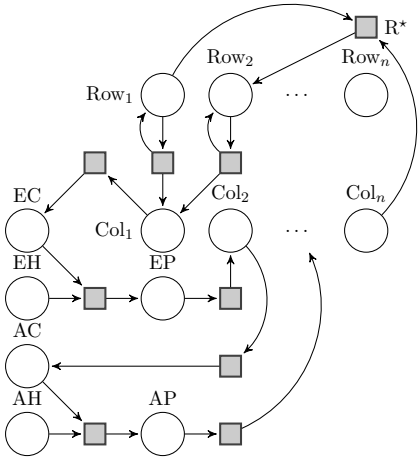


Fig. 4. Schema of a Petri Net for tiling game

(i) If Eloise does not have a piece to play at that moment, it is the turn of Abelard: $(([s,\eta]\phi \leftrightarrow \phi) \wedge \neg([s',\eta]\phi \leftrightarrow \phi) \rightarrow [s',\eta]\phi$, where s' is sequence who differs from s only by inverting EC and AC, and EH and AH;

(ii) If Abelard does not have a piece to play at that moment, it is the turn of Eloise: $(([s',\eta]\phi \leftrightarrow \phi) \wedge \neg([s,\eta]\phi \leftrightarrow \phi) \rightarrow [s',\eta]\phi$ (from now on all formulas have omitted a disjunction with a formula ρ such that ρ differs from the corresponding formula only for changing s to s' in this enumeration of rules);

(iii) The referee has already placed white tiles in the columns 0 and $n+1$: $[s,\eta]$ $col_0(\text{white}) \wedge col_{n+1}(\text{white})$;

(iv) The players must respect tiles colors: $C(t',t,t'') \leftrightarrow \text{right}(t') = \text{left}(T)$ and $\text{down}(T') = \text{up}(T'')$ (e.g. C holds if the tile t can be placed on the right of t' and above t'', where t, t' and t'' are the propositions correspondents to T, T' and T'');

(v) Ensures tile matching left and downwards: $[s,\eta]((p_i \wedge c_{i-1}(t') \wedge c(t'')) \rightarrow [s',\eta] \bigvee\{c_i(t) \mid C(t',t,t'')\})$, where $0 \leq i \leq n$ and, by convention, $\bigvee \emptyset = \bot$

(vi) Ensures matching of tiles placed on column n with white corridor: $[s,\eta](p_n \rightarrow [s',\eta] \bigvee\{c_n(t) \mid \text{right}(T) = \text{white}\})$

(vii) The first position is a winning position for Eloise: w.

As Eloise has a winning strategy, then: $[s,\eta](w \rightarrow (c_1(t_{s+1}) \vee ([s',\eta]\neg w) \vee ([s',\eta]w)))$.

To ensure that the game is finite (e.g. if the game has no end, Abelard wins), the game is limited to $N = n^{s+2}$ steps with no repetition, so: $[s,\eta](\text{counter} = N) \rightarrow [s',\eta]\neg w$.

Then, φ^τ is the conjunction of all these formulas.

If Eloise has a winning strategy then there is a game tree such that φ^τ is satisfiable at its root seen as a $\mathcal{DS}_3$ model. So if Eloise has a winning strategy she can win in at most N steps. For a $\mathcal{DS}_3$ model $\mathcal{M}$ corresponding to this at-most-N

steps strategy it is straightforward to check φ^τ satisfiability at the root of $\mathcal{M}$.

Otherwise, if $\mathcal{M}, v \Vdash \varphi^\tau$, then Eloise has a winning strategy, encoded in $\mathcal{M}$, in the game T. As w is satisfied in v, Eloise can keep moving through winning positions that she is always able to choose. Hence if counter $= N$ (e.g. the counter has reached), $[s, \eta]\neg w$ is satisfied, so there are no more winning positions, but as $c_1(t_{s+1})$ is satisfied, so the winning tile was placed in the first step and Eloise has already won. The detailed list of formulas that are satisfied or not in each case is available at http://www.tecmf.inf.puc-rio.br/BrunoLopes/Proofs.

As it is possible to encode any $m \geq 2$ in $O(\log m + 1)$ binary digits, N can be encoded in $O(\log n^{s+2})$, which corresponds to $(s+2)\log n \leq (s+2)n$. Then, it is polynomial in s and n. So, the two-person corridor tiling problem is polynomially reducible to the $\mathcal{DS}_3$ SAT problem. Hence, $\mathcal{DS}_3$ satisfiability is EXPTime-hard. □

Theorem 5.2 *$\mathcal{DS}_3$ satisfiability problem is EXPTime-complete.*

Proof. In a given play of the two-person corridor tiling game it is possible to keep track of the current assignments that have appeared. For n tiles there are at most $2n$ rounds Therefore there is an Alternating Turing Machine operating in polynomial space that determines whether Eloise or Abelard wins this given instance of the game. Since any polynomial time implementation on an Alternating Turing Machine can be done in ordinary Turing Machine using polynomial space, hence at most exponential time [30]. This shows that the upper bound is EXPTime. Concerning the lower bound, we know that tiling is EXPTime-complete [2] and we provided a polynomial reduction from tiling to $\mathcal{DS}_3$ SAT in lemma 5.1. So $\mathcal{DS}_3$ SAT problem is in EXPTime and then it is EXPTime-complete. □

Corollary 5.3 *Petri-PDL satisfiability problem is EXPTime-complete.*

Proof. As lemma 5.1 and theorem 5.2 use the same firing rate for all transitions, they are valid to Petri-PDL. So Petri-PDL SAT problem is EXPTime-complete. □

6 Usage example

As a usage example take a Kanban system [25], a *Just-In-Time* based flow control method. The SPN designed in figure 5 represents a "cards" (the K tokens of place BB) flow of resources control with failure for a Kanban cell (a processing unit that may communicate with others). The place IB denotes the Input Buffer where the resources are stored (already with a card) before processed. If everything is OK (i.e. the place OK has a token) and the processing system is not busy (i.e. there is a token in place Id) then the resource is processed (the token goes to place B) and thereafter the resource goes to the Output Buffer (the place OB). The firing of F means that some failure occurred. When R fires it means that the system was repaired. The failure rate and the time to process the resources are controlled by the parameters of the random variables associated with the respective transitions.

Modelling this scenario in a $\mathcal{DS}_3$ model $\mathcal{M} = \langle W, R_\pi, M, \Pi, \Lambda, \delta\rangle$, we have the formula $\langle(s), Kt_1IB \odot IB, Idt_2B \odot B, OKt_2l \odot lt_3OK, x \odot xt_3Id, OB \odot OBt_1BB \odot$

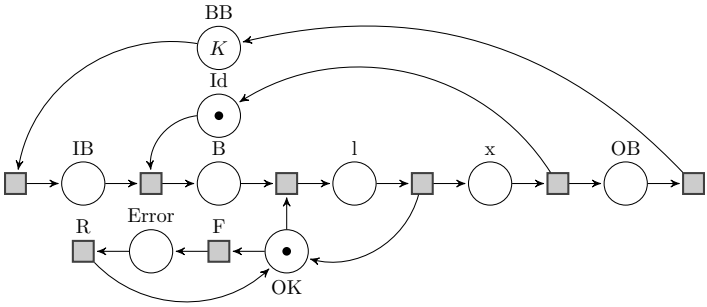


Fig. 5. A Kanban cell with failure

$OKt_1\text{Error}\odot\text{Error}t_1Ok\rangle\varphi$ where s is a sequence of names composed by K repetitions of "BB" and "OK" and φ is some property that holds after the running of this SPN. Verify if this formula holds in a world w of a model $\mathcal{M}$ (i.e. some transition may fires) is equivalent to compute the probability of some basic program fires is greater then zero, which reduces to the equation in lemma 4.8. To verify if it is possible process two resources in parallel, we see that after some of them begin to process (i.e. a token in place B), Id will not be in the sequence of names, so other resources can not begin their process unless a transition that restates a token to Id fires.

Verify if from a world $w \in W$ it is possible that some resource begins its processing is equivalent to compute if $\Pr(\mathcal{M},w \Vdash \langle r, IB,Idt_2B \odot B\rangle\top \mid \delta(w, Kt_1IB \odot IB,Idt_2B\odot B,OKt_2l\odot lt_3OK,x\odot xt_3Id,OB\odot OBt_1BB\odot OKt_1\text{Error}\odot\text{Error}t_1Ok)) > 0$ where $r = M(w)$. Using lemma 4.8 it is equivalent to verify if

$$\frac{\delta(Idt_2B \odot B)}{\sum_{\pi_b\in\Pi: f(r,\pi_b)\neq\epsilon} \delta(w,\pi_b)} > 0$$

where $\Pi = Kt_1IB \odot IB,Idt_2B \odot B,OKt_2l \odot lt_3OK,x \odot xt_3Id,OB \odot OBt_1BB \odot OKt_1\text{Error} \odot \text{Error}t_1Ok$ and π_b is a basic transition of Π.

7 Conclusions and further work

This work extends Petri-PDL, a Dynamic Logic conceived to reasoning about Marked Petri Nets, to a novel Dynamic Logic tailored to reasoning about Marked Stochastic Petri Nets, not only increasing its expressiveness but also presenting a modular and compositional approach to probabilistic modal logic. The system proposed aims to be an alternative to model performance evaluation.

We present a PDL which the programs are Marked Stochastic Petri Nets. Unlike previous approaches, which translate Petri Nets into Dynamic Logic, in our's we have Stochastic Petri Nets encoded as programs of PDL yielding a new Dynamic Logic tailored to reasoning about Petri Nets in a more natural way.

We present an axiomatization to our logic and prove its soundness and completeness. Finally, we establish the decidability and finite model property and EXPTime-completeness of its SAT problem and provide usage examples of our approach.

Using $\mathcal{DS}_3$ it is possible to take advantage of systems that generate SPN automatically from UML diagrams [19], used in software specification, to verify properties. The behaviour of the system can also be translated to a CTMC.

Further work include prospecting meaningful case studies to apply in concrete situations, propose a Natural Deduction and a Resolution systems for $\mathcal{DS}_3$ and investigate Model Checking and Automatic Theorem Prover to $\mathcal{DS}_3$.

References

[1] Mario Benevides, Edward Haeusler, and Bruno Lopes. Propositional Dynamic Logic for Petri Nets. In *Annals of the 6th Workshop on Logical and Semantic Frameworks, with Applications*, 2011.

[2] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Theoretical Tracts in Computer Science. Cambridge University Press, 2001.

[3] J. L. Coleman, W. Henderson, and P. G. Taylor. Product form equilibrium distributions and a convolution algorithm for Stochastic Petri Nets. *Performance Evaluation*, 26(3):159–180, 1996.

[4] E. S. de Almeida and E. H. Haeusler. Proving properties in ordinary Petri Nets using LoRes logical language. *Petri Net Newsletter*, 57:23–36, 1999.

[5] Giuseppe De Giacomo and Fabio Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, 160:2000, 1998.

[6] Yishai A. Feldman. A decidable Propositional Probabilistic Dynamic Logic. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, pages 298–309. ACM, 1983.

[7] Yishai A. Feldman. A decidable Propositional Dynamic Logic with explicit probabilities. *Information and Control*, 63(1-2):11–38, 1984.

[8] Yishai A. Feldman and David Harel. A Probabilistic Dynamic Logic. *Journal of Computer and System Sciences*, 28(2):193–215, 1984.

[9] Michael J. Fischer and Richard E. Ladner. Propositional Dynamic Logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[10] R. Goldblatt. Parallel action: Concurrent Dynamic Logic with independent modalities. *Studia Logica*, 51:551–558, 1992.

[11] Stefan Göller and Markus Lohrey. Infinite state model-checking of Propositional Dynamic Logics. In Zoltán Ésik, editor, *Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 349–364. Springer Berlin Heidelberg, 2006.

[12] P. J. Haas. *Stochastic Petri Nets: modelling, stability, simulation*. Springer, 2002.

[13] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. Foundations of Computing Series. MIT Press, 2000.

[14] W. Henderson, D. Lucic, and P. G. Taylor. A net level performance analysis of Stochastic Petri Nets. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 31(2):1446–8735, 2009.

[15] Andrey N. Kolmogorov. *Foundations of the Theory of Probability*. Chelsea Publishing, 1956.

[16] Dexter Kozen. A probabilistic PDL. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, pages 291–297. ACM, 1983.

[17] Martin Lange. Model checking Propositional Dynamic Logic with all extras. *Journal of Applied Logic*, 4(1):39–49, 2006.

[18] Bruno Lopes, Mario Benevides, and Edward Hermann Haeusler. Propositional dynamic logic for Petri Nets. *Logic Journal of the IGPL*, 2014.

[19] Juan Pablo López-Grao, José Merseguer, and Javier Campos. From UML Activity Diagrams to Stochastic Petri Nets: Application to software performance engineering. *SIGSOFT Software Engineering Notes*, 29(1):25–36, 2004.

[20] Douglas Lyon. Using Stochastic Petri Nets for real-time nth-order stochastic composition. *Computer Music Journal*, 19(4):13–22, 1995.

[21] Andrea Marin, Simonetta Balsamo, and Peter G. Harrison. Analysis of Stochastic Petri Nets with signals. *Performance Evaluation*, 69(11):551–572, 2012.

[22] M. Ajmone Marsan. Stochastic Petri Nets: An elementary introduction. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pages 1–29. Springer Berlin Heidelberg, 1990.

[23] M. Ajmone Marsan. Stochastic Petri Nets: an elementary introduction. In *Advances in Petri Nets 1989*, volume 429 of *Lecture Notes in Computer Science*, pages 1–29. Springer Berlin Heidelberg, 1990.

[24] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of Generalised Stochastic Petri Nets for the analysis of multiprocessor systems. *ACM Transactions On Computer Systems*, 2(1), 1984.

[25] M. Ajmone Marsan, Gianfranco Balbo, Gianni Conte, Susanna Donatelli, and Giuliana Franceschinis. *Modelling with Generalised Stochastic Petri Nets*. Wiley, 1995.

[26] M. Ajmone Marsan and G. Chiola. On Petri Nets with deterministic and exponentially distributed firing times. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1987*, volume 266 of *Lecture Notes in Computer Science*, pages 132–145. Springer Berlin Heidelberg, 1987.

[27] M. Ajmone Marsan and Giovanni Chiola. On Petri Nets with deterministic and exponentially distributed firing times. In *Advances in Petri Nets 1987*, pages 132–145. Springer Berlin Heidelberg, 1987.

[28] Antoni Mazurkiewicz. Trace theory. In W. Brauer., W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 278–324. Springer, 1987.

[29] Antoni Mazurkiewicz. Basic notions of trace theory. In J. W. Bakker, W.-P. Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 285–363. Springer, 1989.

[30] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[31] M. Tiomkin and J.A. Makowsky. Decidability of finite probabilistic Propositional Dynamic Logics. *Information and Computation*, 94(2):180–203, 1991.

[32] M.L. Tiomkin and J.A. Makowsky. Propositional Dynamic Logic with local assignment. *Theoretical Computer Science*, 36:71–87, 1985.