CSL Model Checking of Deterministic and Stochastic Petri Nets

José M. Martínez and Boudewijn R. Haverkort

Faculty of Electrical Engineering, Mathematics and Computer Science.
University of Twente, the Netherlands.
{martinez,brh}@cs.utwente.nl

Abstract. Deterministic and Stochastic Petri Nets (DSPNs) are a widely used high-level formalism for modeling discrete-event systems where events may occur either without consuming time, after a deterministic time, or after an exponentially distributed time. The underlying process defined by DSPNs, under certain restrictions, corresponds to a class of Markov Regenerative Stochastic Processes (MRGP). In this paper, we investigate the use of CSL (Continuous Stochastic Logic) to express probabilistic properties, such a time-bounded until and time-bounded next, at the DSPN level. The verification of such properties requires the solution of the steady-state and transient probabilities of the underlying MRGP. We also address a number of semantic issues regarding the application of CSL on MRGP and provide numerical model checking algorithms for this logic. A prototype model checker, based on SPNica, is also described.

Keywords: DSPN, model checking, Markov regenerative process.

1 Introduction

Over the last 25 years, stochastic Petri nets of some form have been used widely for modelling and evaluation of all kinds of performance and dependability aspects of computer and communication systems. Where stochastic Petri nets (SPNs) only includes transitions with exponential delays [1], generalised stochastic Petri nets (GSPNs) [2] allowed for the use of transitions with either exponential delays (exponential transitions) or zero delays (immediate transitions). In the very similar stochastic activity networks (SANs) [3], the latter transitions are referred to as instantaneous. In the late 1980's the GSPN modelling framework was extened with the possibility to also use transitions with deterministic delays, leading to so-called deterministic and stochastic Petri nets (DSPNs) [4]. Later, this model was extended to allow for marking-dependent deterministic delays, as well as for any other than exponentially distributed delay (with deterministic delays being just a special case) [5, 6]. In order to keep the analysis practically feasible, also for larger models, in all of the DSPN-type approaches, the restriction applies that in any marking, at most one non-exponentially distributed transition can be enabled (although exceptions to this restriction do exist, cf. [7]). DSPN type models find their application in performance and dependability evaluation, e.g., there were deterministic time-outs are involved, or where deterministic arrivals patterns or job lengths are involved. Tool support for DSPNs is available through a variety of packages, most notably DSPNexpress [8], TimeNET [9], and SPNica [10].

Over the last 15 years, model checking has established itself as a technique for the validation or verification of system properties, based on a usually exhaustive state space search of a formally specified model. Originally, model checking techniques were used solely to verify qualitative system properties (like liveness or the absence of deadlock); a prominent logic to specify required system properties in this context has been CTL [11]. More recently, model checking techniques have been developed for models that include probabilities and non-determinism, as well as (stochastic) time, thereby using the specification logics pCTL [12] and CSL [13], respectively. When applying model checking techniques for models including stochastic time, these techniques can be applied for the evaluation of the performance and dependability of systems as well. In the context of model checking stochastic systems, the focus has primarily been on labeled continuous-time Markov chains (CTMCs) as base model [14]. We have seen this in particular with the logic CSL, which is now used widely for the specification of system performance and dependability properties for Markovian model, e.g., in a model checker like PRiSM [15], or in the tools GreatSPN [16] and APNN-toolbox [17].

For non-Markovian models, there have been less developments so far. As early as 2001, Infante López et al. reported on the use of CSL model checking of semi-Markov chains (SMCs) [18]. In that paper, the fixed-point equations describing the formal semantics of the logic are used as basis for the computational procedures to verify CSL properties. To do so, a system of Volterra integral equations has to be solved numerically, which turns out to be numerically less attractive (a similar finding as in the original paper by Baier et al. on model checking CSL for CTMCs [19]).

In this paper, we extend the use of CSL to the context of so-called Markov regenerative processes (MRGPs), a class of stochastic processes that arises from DSPNs (cf. Section 2). This class of process is more general than CTMCs. Considering the probability distributions involved, the process under study is less general than SMCs, but, if we consider the state dependencies between transitions, it is more general the SMCs. Given we are dealing with deterministic and exponential transitions only, we can derive more specific equations that describe the system evolution over time, hence, we can derive more efficient model checking procedures for models specified as DSPNs, something that has not been done previously.

This paper is further organised as follows. We present DSPNs in Section 2, and the logic CSL to be used for so-called labeled MRGPs in Section 3. We then focus on the most critical model checking algorithms for CSL on MRGPs in Section 4, being the evaluation of the steady-state operator, as well as the time-bounded next operator and the time-bounded until operator; this section forms the core of the current paper. Section 5 then present a concise overview of the model checking tool we have developed on the basis of SPNica, and Section 6 presents an application example. Section 7 concludes the paper.

2 DSPNs: Definition and Underlying Process

In this paper we assume the reader is familiar with Deterministic and Stochastic Petri Nets. Futhermore, we restrict our analysis to those DSPNs where at most one deterministic transition may be enabled in each marking.

The marking process underlying a DSPN, when there is at most one deterministic transition enabled at any time, corresponds to a particular class of Markov regenerative process [5, 20]. It is a particular class in the sense that transitions may only occur after a constant delay or an exponentially distributed time. In MRGP transitions with general firing time distributions are allowed provided that in each marking at most one transition of this kind may be enabled.

In addition to the firing time distributions and for univocally defining the stochastic process, an "execution policy" must be included in the description of the DSPN. In our research, we consider the firing "policy race with *enabled memory*" [21] (also know as *preemptive repeat different*), i.e., when a timed transition is preempted, its already carried out work is lost.

Example: An example of DSPN is shown in Figure 1(a), where a $M/D/1/K$ queueing system with server breakdowns is modeled. Basically, the behavior of the system is as follows: jobs arrive into the system buffer and are served immediately, assuming the server is working properly. If a failure occurs, the service over the current job is interrupted until the server recovers from the failure (both, failure and repair times, are exponentially distributed). The system is represented as follows: the places `free`, `buffer`, `operative`, and `failure` describe the number of free places in the buffer, number of jobs/customer in the buffer, if the serve is operative, and if the server failed, respectively. The transitions `arrival`, `service`, `fail`, and `restart`, describe the events arrival of a job/customer, service of a job/customer, failure of the server, and restart of the server, respectively. The transition `service` is deterministic, whereas all the others are exponentially distributed.□

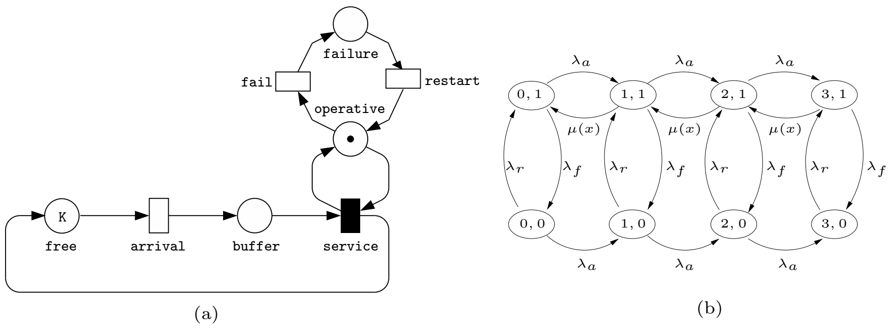


Fig. 1. (a) DSPN M/D/1/K with server breakdowns. (b) State transition diagram for DSPN in (a) with $\mathtt{K} = 3$.

The evolution of the DSPN can be described through its reachability graph. During the generation of the graph, vanishing markings (due to the firing of immediate transitions) are eliminated. Let T^D denote the set of all deterministic transitions. The tangible markings of a DSPN constitute the finite state space S of the underlying MRGP. Considering we restrict our analysis to have, at most, one deterministic transition enabled in each marking, S can be partitioned into the disjoint sets S^E, whose elements are the states in which only exponential transitions are enabled, and S^d (with $d \in T^D$), whose elements are the states in which the deterministic transition d is enabled.

In the analysis of DSPNs two branches of research can be identified: one based on the method of supplementary variables [22], and another one based on Markov regenerative theory [23]. Through out this paper we will use the first approach, due mainly to efficiency reasons (see [6] for a comparison).

Using the method of supplementary variables (see [24, 10, 25]), for any state $s \in S^d, d \in T^D$, together with the marking of the process we provide the information related to the age of the (enabled) deterministic transition. Defining $N(t) \in S, t \geq 0$, as the discrete-random variable giving the tangible marking at time t, the state of the underlying process (MRGP) of a DSPN can be described by the pair $\{(N(t), X(t)),\ t \geq 0\}$, where $X(t) \in [0, \infty)$ (if $N(t) \in S^d$) is the elapsed time of the deterministic transition d at time t. We assume that $X(t) = \infty$ when $N(t) \in S^E$.

Example cont.: Considering our running example, the state transition diagram corresponding to the DSPN depicted in Figure 1(a) with `K` $= 3$, is presented in Figure 1(b), where λ_a is the `arrival` rate (exponential), $\mu(x)$ is the `service` rate (deterministic) which depends on the time the deterministic transition has been enabled, λ_f is the `fail` rate (exponential), and λ_r is the `restart` rate (exponential). We assume the state identifier of the system is given by the number of tokens in place `buffer` and `operative`, respectively. $\square$

Given the state description above, the system dynamics can be stated through the followings probabilites:

- $\pi_i(t) = P\{N(t) = i\}$: the transient probability that at time t the marking is $i \in S$.
- $\pi_i(t, x)$: the age density function associated to marking $i \in S^d, d \in T^D$:

$$\pi_i(t, x) = \frac{P\{N(t) = i,\ x < X < x + dx\}}{dx}$$

- The relation between $\pi_n(t)$ and $\pi_n(t, x)$ is given by:

$$\pi_i(t) = \int_0^\infty \pi_i(t, x)\, dx\,,\ i \in S^D$$

Together with the probabilities defined above, and following the description given in [25], the structure of the underlying MRGP can be characterized by the square matrices $\mathbf{Q}$, $\bar{\mathbf{Q}}$ and $\mathbf{\Delta}$, each with dimension $|S|$, which are defined as follows:

- $\mathbf{Q} = [q_{i,j}]$, where $q_{i,j}, i \neq j$ is the rate of the exponential transition from state i to j which does not preempt a deterministic transition. $q_{i,i}$ is the negative sum of all rates of exponential transitions out of state i (including those that preempt a deterministic transition).
- $\bar{\mathbf{Q}} = [\bar{q}_{i,j}]$, where $\bar{q}_{i,j}, i \neq j$ is the rate of the exponential transition from i to j that preempts a deterministic transition.
- $\mathbf{\Delta} = [\delta_{i,j}]$, where $\delta_{i,j}$ is the probability a deterministic transition leads to state j, given that it fires in state i, i.e. $\delta_{i,j} = P\{N(t^+) = j \mid N(t^-) = i$, deterministic transition fires at time $t\}$.

3 CSL for DSPN

The logic CSL (for Continuous Stochastic Logic) [13, 14] is a continuous-time extension of PCTL and CTL [11, 12]. It comprises, in particular, both a probabilistic operator that can be used to express *path-based* properties, as well as a *steady-state* operator to express long-term properties. In the past, CSL has primarly been used to express properties for continuous-time Markov chains.

In the subsections that follow, we will present the syntax and semantics of CSL in the context of labeled MRGPs.

3.1 Labeled MRGPs

We now introduce the labeled Markov Regenerative Processes, analogously to the definition of labeled Markov chains in [14].

- Let AP denote a fixed, finite set of atomic propositions.
- A labeled MRGP $\mathcal{M} = (S, \mathbf{Q}, \bar{\mathbf{Q}}, \mathbf{\Delta}, L)$, with S a finite set of states, $\mathbf{Q}, \bar{\mathbf{Q}} : S \times S \to \mathbb{R}_{\geq 0}$ the rate matrices (considering non-preemption and preemption, respectively), $\mathbf{\Delta} : S \times S \to \{0, 1\}$ the branching probability matrix for deterministic transitions and $L : S \to 2^{AP}$ the labeling function.
- Given a labeled MRGP $\mathcal{M} = (S, \mathbf{Q}, \bar{\mathbf{Q}}, \mathbf{\Delta}, L)$, an infinite path σ is a sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots$, with $i \in \mathbb{N}$, $s_i \in S$, and $t_i \in \mathbb{R}_{\geq 0}$ such that either $\mathbf{Q}(s_i, s_{i+1}) > 0$, $\bar{\mathbf{Q}}(s_i, s_{i+1}) > 0$ or $\mathbf{\Delta}(s_i, s_{i+1}) > 0$, for all i. A finite path σ is a sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \ldots s_{l-1} \xrightarrow{t_{l-1}} s_l$ such that s_l is absorbing, and either $\mathbf{Q}(s_i, s_{i+1}) > 0$, $\bar{\mathbf{Q}}(s_i, s_{i+1}) > 0$ or $\mathbf{\Delta}(s_i, s_{i+1}) > 0$, for all $i < l$.

Furthermore, for $t \in \mathbb{R}_{\geq 0}$ and i the smallest index with $t \leq \sum_{j=0}^{i} t_j$, let $\sigma@t = \sigma[i]$ be the state in σ occupied at time t. And, for a path σ and $i \in \mathbb{N}$, we have $\sigma[i] = s_i$, the $(i+1)th$ state of σ, and $\delta(\sigma, i) = t_i$, the time spent in state s_i.

We must note that the labeling function for a MRGP underlying a DSPN, corresponds to the marking associated to a state, for example, considering the DSPN of Figure 1(a) with $\texttt{K} = 3$, the label of the state $(3, 1)$ is $\{\texttt{free} = 0, \texttt{buffer} = 3, \texttt{operative} = 1, \texttt{fail} = 0\}$.

3.2 CSL: Syntax and Semantics

Given an atomic proposition $a \in AP$, $\unlhd \in \{<, \leq, \geq, >\}$ and $p \in [0,1]$, we have the following *state-formulas*:

$$\phi ::= \mathtt{tt} \mid a \mid \neg\phi \mid \phi \vee \phi \mid \mathcal{S}_{\unlhd p}(\phi) \mid \mathcal{P}_{\unlhd p}(\varphi),$$

where $\mathcal{S}_{\unlhd p}(\phi)$ is understood as "the probability that ϕ holds in steady state is $\unlhd p$", and $\mathcal{P}_{\unlhd p}(\varphi)$ is understood as "the probability that a path fulfils φ is $\unlhd p$".

Furthermore, given a time point $t \in \mathbb{R}_{\geq 0}$, we have the following *path-formulas*:

$$\varphi ::= \mathcal{X}^{\leq t}\phi \mid \phi\, \mathcal{U}^{\leq t}\ \psi,$$

where $\mathcal{X}^{\leq t}\phi$ is understood as "the next state, which is to be reached before t fulfils ϕ", and $\phi\ U^{\leq t}\ \psi$ is understood as "ϕ holds along the path until ψ holds, at a time point before t".

Note that we have presented here the simplest time-bounded-until and -next formulas: we only allow for time-bounds of the form $I = [0, t)$. In general, CSL as defined in [14] allows for time-bounds of the form $I = [t_1, t_2]$, however, such time bounds lead to more involved numerical procedures for which we currently cannot provide satisfying results.

The state formulas are interpreted over the states of the MRGP underlying the DSPN. Let $Path(s)$ denote the set of all paths the process can evolve through when starting in s. Let $Sat(\phi) = \{s \in S \mid s \models \phi\}$. Then, the satsifaction relation $\models$ for CSL state formulas is defined by:

$$\begin{array}{ll} s \models \mathtt{tt}\ \ \forall s \in S\ \ , & s \models \phi_1 \vee \phi_2 \text{ iff } s \models \phi_1 \vee s \models \phi_2\ \ , \\ s \models a \quad \text{iff } a \in L(s)\ \ , & s \models \mathcal{S}_{\unlhd p}(\phi) \text{ iff } \sum_{s \in Sat(\phi)} \pi_s \unlhd p\ \ , \\ s \models \neg\phi \text{ iff } s \not\models \phi\ \ , & s \models \mathcal{P}_{\unlhd p}(\varphi) \text{ iff } Prob(s, \varphi) \unlhd p\ \ , \end{array}$$

where $Prob(s, \varphi)$ corresponds to the aggregated probability of all paths $\sigma \in Path(s)$ that satisfy φ, i.e., $Prob(s, \varphi) = P\{\sigma \in Path(s) \mid \sigma \models \varphi\}$ (see the satisfaction relation for path formulas below), and π_s is the steady-state probability of being in state s. The relation $\models$ for CSL path formulas is defined by:

$$\begin{array}{ll} \sigma \models X^{\leq t}\phi & \text{iff } \sigma[1] \text{ is defined and } \sigma[1] \models \phi \wedge \delta(\sigma, 0) \leq t\ \ , \\ \sigma \models \phi\, \mathcal{U}^{\leq t}\ \psi & \text{iff } \exists \tau \leq t.(\sigma@\tau \models \psi \wedge (\forall t' \in [0, \tau).\sigma@\tau \models \phi)). \end{array}$$

Example: Considering the DSPN in Figure 1(a), an example of CSL state formula would be:

$$\mathcal{P}_{>0.5}((\mathtt{operative} = \mathtt{1})\, \mathcal{U}^{<5}\ (\mathtt{buffer} = \mathtt{10} \wedge \mathtt{operative} = \mathtt{0}))\ .$$

This formula is satisfied for all the starting states of a path passing only through states corresponding to a marking where place $\mathtt{operative} = \mathtt{1}$, until a state corresponding to a marking where place $\mathtt{buffer} = \mathtt{10}$ and place $\mathtt{operative} = \mathtt{0}$ is reached, at a time point before 5 and with a probability greater than 0.5. □

4 Model Checking Algorithms

In this section we describe model checking algorithms to determine the satisfaction sets for the different probabilistic formulas defined in Section 3.

4.1 Steady-state operator: $\mathcal{S}_{\trianglelefteq p}(\phi)$

In order to obtain the set of all states satisfying the formula $\mathcal{S}_{\trianglelefteq p}(\phi)$, we need to compute the steady-state probabilities (or long-term proportion of time, if there are no limiting probabilities) for those states satisfying the state formula ϕ. We must consider two possibles cases leading to slightly different analysis methodologies: (1) the labeled process $\mathcal{M}$ is a strongly connected MRGP, and (2) the labeled process $\mathcal{M}$ is an MRGP composed of bottom strongly connected subsets of S.

If we have a strongly connected labeled process $\mathcal{M}$, we proceed by accumulating the steady-state probabilities, π_i, for those states $i \in Sat(\phi)$ and comparing their sum with $\trianglelefteq p$. Because the MRGP is strongly connected, the satisfaction set for this formula is either the whole state space S or the empty set ($\{\emptyset\}$).

On the other hand, in case the labeled process $\mathcal{M}$ is not strongly connected we must proceed to isolate the bottom strongly connected subsets of S, as in [14], using a graph algorithm [26]. Let S' denote a strongly connected subset of S and consider a state $s' \in S'$, then the probability $\pi(i, s')$ of being in $s' \in S'$ in the long-run, given we start in a state $i \in S$, equals

$$\pi(i,s') \;=\; \left(\sum_{j \in S'} \int_0^\infty \pi(i,j,\tau)\, d\tau \right) \cdot \pi_{s'}^{S'} \;,$$

where the first factor on the right-hand side corresponds to the probability of reaching a state $s' \in S'$ eventually ($\pi(i, j, x)$ is the transient probability of being in j at time x given we start in i), and the second factor on the right hand side ($\pi_{s'}^{S'}$) is the long-run probability of being in state s' considering the subset S' only.

The steady-state probabilities π_i with $i \in S$ can be calculated according to the procedure described in [8, 25] (see Appendix B).

4.2 Time-bounded Until operator: $\mathcal{P}_{\trianglelefteq p}(\phi_1\, \mathcal{U}^{\leq t}\, \phi_2)$

In order to obtain the set of states that satisfies the time-bounded until formula we need to calculate, $Prob(i, \phi_1 \mathcal{U}^{\leq t} \phi_2)$, i.e., the probability of visiting for the first time the set of states satisfying ϕ_2 within t time units, when we start in a state $i \in Sat(\phi_1)$ and transit through states satisfying ϕ_1 only. This probability can be calculated by following a similar strategy to that in [14] for the case of continuous time Markov chains, i.e., making absorbing those states that satisfy $(\neg\phi_1 \vee \phi_2)$ and considering the transient probability of being in states that satisfy ϕ_2 at time t, given we start in a state that satisfies ϕ_1.

Let $\mathcal{M}[\phi]$ denote the modified MRGP $\mathcal{M}$ with the subset of states that satisfy ϕ made absorbing. Using this notation, let $Prob^{\mathcal{M}}(i,\varphi)$ denote the aggregated probability of those paths starting in state i and satisfying φ considering the original MRGP $\mathcal{M}$ and let $Prob^{\mathcal{M}[\phi]}(i,\varphi)$ denote the aggregated probability of those paths starting in i and satisfying φ considering the modified MRGP $\mathcal{M}[\phi]$. Thus, the measure we are interested in can be evaluated as follows:

$$Prob^{\mathcal{M}}(i,\phi_1\mathcal{U}^{\leq t}\,\phi_2) = Prob^{\mathcal{M}[\neg\phi_1\vee\phi_2]}(i,\mathtt{tt}\,\mathcal{U}^{\leq t}\,\phi_2) = \sum_{j\models\phi_2}\pi^{\mathcal{M}[\neg\phi_1\vee\phi_2]}(i,j,t)\ , \tag{4.1}$$

where $\pi^{\mathcal{M}[\neg\phi_1\vee\phi_2]}(i,j,t)$ corresponds to the transient probability of being in state j at time $t>0$, given we start in state i at time $t=0$, and considering the modified MRGP $\mathcal{M}[\neg\phi_1\vee\phi_2]$.

To obtain the transient probability $\pi(i,j,t)$ in (4.1), we proceed by evaluating the state probability $\pi_j(t)$ with initial probability $\underline{\pi}(0)=\underline{1}_i$, where $\underline{1}_i$ is a vector with the entry corresponding to state i equal to 1 and all the other entries equal to 0, and using the methodology developed in [25, 27] (see Appendix A).

Regarding the procedure to obtain $\mathcal{M}[\neg\phi_1\vee\phi_2]$ from the original MRGP $\mathcal{M}$, we distinguish two cases based on whether there is a deterministic transition from the state that is made absorbing (i.e., either a state is in S^E or in $S^d, d\in T^D$):

– *If we make absorbing a state* $k\in S^E$, we only modify the matrix $\mathbf{Q}$ by filling in with zeros the row corresponding to state k. Let $\mathbf{Q}[k,:]$ denote the kth row of matrix $\mathbf{Q}$, and $\underline{0}$ a vector of size $|S|$ with all its entries equal to zero, then we have:

$$\mathbf{Q}[k,:]\leftarrow\underline{0}$$

– *If we make absorbing a state* $k\in S^d, d\in T^D$, the state k does no longer belong to the set S^d, but it belongs now to the set S^E, and all existing exponential transitions from any other state $j\in S^d, d\in T^D$, to state k, will now *preempt* the transition $d\in T^D$. Thus, we proceed by moving the state k from the set S^d to the set S^E, filling in with zeros the row corresponding to state k in the matrices $\mathbf{Q}$, $\bar{\mathbf{Q}}$ and $\mathbf{\Delta}$, and moving the entry (j,k) from matrix $\mathbf{Q}$ to the entry (j,k) in matrix $\bar{\mathbf{Q}}$, i.e.,

$$\begin{aligned}&(1)\ \mathbf{Q}[k,:]\leftarrow\underline{0},\ \bar{\mathbf{Q}}[k,:]\leftarrow\underline{0},\ \mathbf{\Delta}[k,:]\leftarrow\underline{0}\\&(2)\ \bar{\mathbf{Q}}[j,k]\leftarrow\mathbf{Q}[j,k]\\&(3)\ \mathbf{Q}[j,k]\leftarrow\underline{0}\end{aligned}$$

Example: To illustrate the modification of an MRGP in order to evaluate a time-bounded until operator, we consider the DSPN of Figure 1(a) with $\mathtt{K}=3$, its corresponding MRGP in Figure 1(b), and the property:

$$\mathcal{P}_{\trianglelefteq p}(\mathtt{tt}\,\mathcal{U}^{\leq t}\,\mathtt{buffer}=3)$$

Then, in order to check this property we need to make absorbing those states satisfying $(\neg\mathtt{tt}\vee\mathtt{buffer}=3)$, which correspond to the states $(3,1)$ and $(3,0)$. The resulting modified MRGP is shown in Figure 2, where the absorbing states

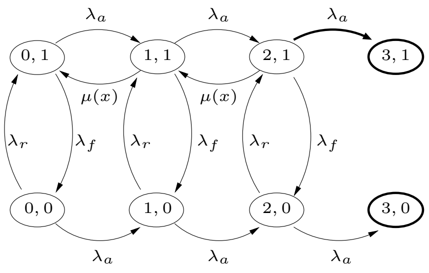


Fig. 2. Modified MRGP to evaluate $\mathcal{P}_{\trianglelefteq p}(\mathtt{tt}\, \mathcal{U}^{\leq t}\ \mathtt{buffer} = 3)$ in DSPN of Figure 1(a) with $\mathtt{K} = 3$.

are identified by a thicker ellipse, and where the transition from state $(2, 1)$ to state $(3, 1)$ (thicker arrow) now *preempts* the deterministic transition from state $(2, 1)$ to state $(1, 1)$. $\square$

It must be noted that, in the evaluation of $Prob^{\mathcal{M}}(i, \phi_1 \mathcal{U}^{\leq t}\, \phi_2)$, with $i \in S^d, d \in T^D$, we assume the firing time of the deterministic transition d enabled in state i has been reset to τ_d, allowing us to evaluate this transient probability with the same techniques, since we consider the corresponding initial marking of state i. This assumption imposes a constraint on the analysis of the time-bounded until operator; we consider only one case of the infinitely many possible scenarios for the elapsed time of the deterministic transition. Nevertheless, early experiments for models used through this paper have shown that the probability $Prob^{\mathcal{M}}(i, \phi_1 \mathcal{U}^{\leq t}\, \phi_2)$, with $i \in S^d, d \in T^D$ is contained within a region defined by two cases for the initial conditions of the deterministic firing time: τ_d and a value very close to 0 (just before transition d fires); no general guarantees can be given, however.

4.3 Time-bounded Next property: $\mathcal{P}_{\trianglelefteq p}(\mathcal{X}^{\leq t}\ \phi)$

Here we are interested in obtaining the set of markings that satisfies the property $\mathcal{P}_{\trianglelefteq p}(\mathcal{X}^{\leq t}\phi)$, considering ϕ as a state-formula.

In order to obtain such a set of states, we need to evaluate $Prob(i, \mathcal{X}^{\leq t}\ \phi)$ for a state i that has a transition to some element of the set that satisfies ϕ and compare it with $\trianglelefteq p$. In the case the outcome of the previous comparison is true, we can establish that state i satisfies $\mathcal{P}_{\trianglelefteq p}(\mathcal{X}^{\leq t}\phi)$. Given we are dealing with a non-Markovian process we need to carefully address the semantics for this operator.

In CSL model checking of CTMCs the evaluation of $Prob(i, \mathcal{X}^{\leq t}\ \phi)$ is quite straightforward (see [14]) due to the fact that the transition time between states is exponentially distributed, which means that no memory is involved when we have to decide where to go next, i.e.,

$$Prob(i, \mathcal{X}^{\leq t}\phi) \ = \ (1 - e^{-E(i)t}) \cdot \sum_{j \in Sat(\phi)} p_{i,j} \tag{4.2}$$

where $E(i)$ is the output transition rate from state i, and $p_{i,j}$ is the probability of reaching state j from state i.

In case of an MRGP, specifically an MRGP underlying a DSPN, the evaluation of the time-bounded next operator may depend on the time a deterministic transition has been enabled.

Hereafter, the analysis will be based on Markov renewal theory [23] (this approach has been used in [5, 25] for analyizing DSPNs). The main idea is to consider the MRGP at certain points where it is memoryless. Thus, the behavior of the MRGP can be described as follows: (1) When only exponential transitions are enabled, the stochastic behavior of the DSPN is sampled at the instant of firing of an exponential transition. (2) When a deterministic transition is exclusively enabled, the stochastic behavior of the DSPN is sampled at the instant of its firing. (3) If a deterministic transition is competitively enabled with some exponential transitions, the stochastic behavior is sampled when either the deterministic or the exponential transition fires. (4) If a deterministic transition is concurrently enabled with some exponential transitions, the stochastic behavior is sampled at the instant of firing the deterministic transition. The states of the renewal sequence formed by the sampling points described above, define the states of a discrete-time embedded Markov chain (EMC). Furthermore, if a deterministic transition, say t_k, is concurrently enabled with some exponential transitions, the stochastic behavior between two consecutive renewal points (which corresponds to the firing time of t_k) is specified by a continuous-time Markov chain, which is called the subordinated Markov chain (SMC) of the deterministic transition t_k.

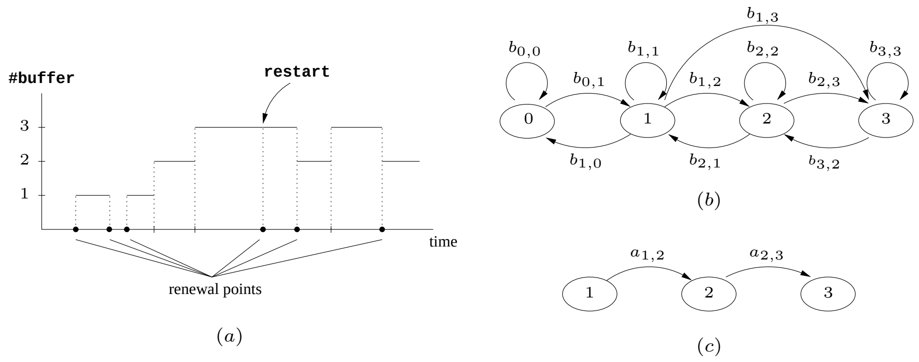


Fig. 3. MRGP underlying DSPN Figure 1(a) with `restart` transition made immediate, and `K` $= 3$: (a) a sample path, (b) embedded DTMC, and (c) subordinated CTMC.

To illustrate the previous concept, let us consider a slightly modified version of the DSPN of Figure 1(a), where the `restart` timed transition has been replaced by an immediate one. A sample path of the underlying process is depicted in Figure 3(a) where the renewal points (those that define the EMC) are indicated. The EMC corresponding to this DSPN is given in Figure 3(b), where $b_{i,j}$ is the transition probability from state i to j (where a state is defined as the number of tokens in place `buffer`). A renewal occurs either when there is an arrival to an empty system, there is a departure from the system (`service` transition fires), or there is a service restart (`restart` transition fires). The SMC, defined when the transition `service` is enabled, is depicted in Figure 3(c), where $a_{i,j}$ is the transition rate from state i to j. $\square$

Continuing with the analysis, we distinguish two cases for evaluating the time-bounded next operator in a MRGP, depending on whether there is a deterministic transition enabled in the current state, or not.

In case we are calculating $\mathit{Prob}(i, \mathcal{X}^{\leq t}\phi)$ for a state $i \in S^E$, i.e., when there is no deterministic transition enabled, the property can be evaluated using the expression given by (4.2).

On the other hand, if we are calculating $\mathit{Prob}(i, \mathcal{X}^{\leq t}\phi)$ for a state $i \in S^d, d \in T^D$, we must take into account how much time has passed since the deterministic transition got enabled.

Considering, hereafter, the case a deterministic transition $d \in T^D$ is concurrently enabled with some exponential transitions, we proceed by determining the SMC associated to d, together with the initial conditions for that SMC, in order to evaluate $\mathit{Prob}(i, \mathcal{X}^{\leq t}\phi)$ with $i \in S^d$. It is at this point where we add to the semantics of the time-bounded next operator: we assume the system has been operating for a while, i.e., *we consider a steady-state operation of the MRGP, in order to obtain the initial conditions for the SMC.* With this assumption we do *not* alter the meaning of the next operator, we only "displace" the analysis to stable conditions. In case of pure Markovian process, the analysis in transient or steady-state conditions is meaningless due to the memoryless property. Thus, the procedure for evaluating the time-bounded next operator can be summarized in the following steps: (1) determining the probability of being in a specific SMC, and (2) evaluating $\mathit{Prob}(i, \mathcal{X}^{\leq t}\phi)$, considering how much time the process has been in that SMC.

First, conditioning and deconditioning in the steady-state probability of being in a state j of the EMC (which also corresponds to the initial state in a SMC), we have:

$$\mathit{Prob}(i, \mathcal{X}^{\leq t}\phi) \;=\; \sum_{j \in S} \mathit{Prob}(i, \mathcal{X}^{\leq t}\phi \mid j)\ \mathbb{P}\{Z(\infty) = j\}\ , \quad i, j \in S^d$$

where $\mathbb{P}\{Z(\infty) = j\}$ is the steady-state probability of being in state j of the EMC Z defined by the MRGP underlying the DSPN, which can be obtained following the procedure described in [8], and $\mathit{Prob}(i, \mathcal{X}^{\leq t}\phi \mid j)$ is the probability a transition occurs within t time units and the next state visited from state i satisfies ϕ, given the SMC defined by the enabled deterministic transition $d \in T^D$ with initial state $j \in S^d$.

Now, conditioning and deconditioning on both the total time the deterministic transition has been enabled (i.e., time the process has been evolving within the SMC) and the fact that the last visited state of the SMC corresponds to state i, we have:

$$Prob(i, \mathcal{X}^{\leq t}\phi \mid j) = \int_0^{\tau_d} Prob(i, \mathcal{X}^{\leq t}\phi \mid j, i, x)\, \pi(i, x|j)\, dx\ , \quad i, j \in S^d, \quad (4.3)$$

where $Prob(i, \mathcal{X}^{\leq t}\phi \mid j, i, x)$ is the probability the next state from state i is in $Sat(\phi)$ within a time t, given the deterministic transition has been enabled for a time x and the last transition of the SMC was to state i, and $\pi(i, x|j)$ is the transient probability of being in state i at time x, given the initial state of the SMC is $j \in S^d, d \in T^D$.

The second factor inside the integral in (4.3), $\pi(i, x|j)$, can be calculated using randomization [28] as follows:

$$\pi(i, x|j) \ = \ \left(\sum_{n=0}^{\infty} e^{-q_d x} \frac{(q_d x)^n}{n!}\, \underline{\pi}_{0j} \mathbf{P}_d^n \right) \cdot \underline{1}_i\ , \quad (4.4)$$

where $\underline{\pi}_{0j}$ corresponds to the initial probability vector of the SMC defined by the transition $d \in T^D$ with all its entries equal to zero except that corresponding to state $j \in S^d$, q_d corresponds to the uniformization rate for the SMC, and $\mathbf{P}_d$ corresponds to the probability transition matrix of the uniformized process.

The first factor in the integral of (4.3), $Prob(i, \mathcal{X}^{\leq t}\phi \mid j, i, x)$, depends on the relation between x, τ_d (firing time of deterministic transition d) and t. Thus, we can distinguish two cases which are depicted in Figure 4, where the black filled rectangle represents the firing time of the deterministic transition and the hatched rectangle represents the time-bound for the next operator when the SMC is in state i:

(*a*) $x + t < \tau_d$: the time-bound of the next operator fits the interval $[0, \tau_d]$, i.e., there is no firing of the deterministic transition and $Prob(i, \mathcal{X}^{\leq t}\phi \mid j, i, x)$ may be evaluated considering only exponential transitions.

(*b*) $x + t \geq \tau_d$: only part of the time-bound of the next operator fits $[0, \tau_d]$, i.e., before the firing of d (τ_d), we consider only exponential transitions, but after the firing of d, the next state is defined by the branch probability associated with the deterministic transition.

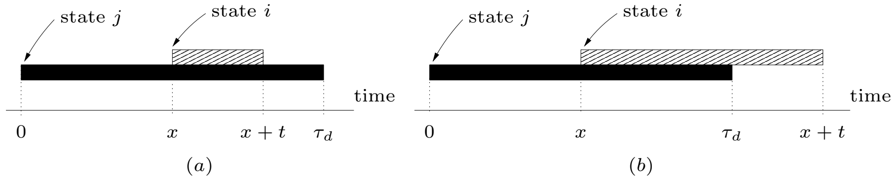


Fig. 4. Time-bounded Next: timing relation between time-bound and elapsed time of deterministic transition.

Thus, the expression for evaluating $Prob(i, \mathcal{X}^{\leq t}\phi \mid j, i, x)$ is given by:

$$Prob(i, \mathcal{X}^{\leq t}\phi \mid j, i, x) \;=\; \begin{cases} (1 - e^{-q_i t}) \cdot \sum\limits_{k \in Sat(\phi)} p_{i,k} \;, & x + t < \tau_d \\ \begin{array}{l} (1 - e^{-q_i(\tau_d - x)}) \cdot \sum\limits_{k \in Sat(\phi)} p_{i,k} \\ \quad + \; e^{-q_i(\tau_d - x)} \cdot \sum\limits_{k \in Sat(\phi)} r_{i,k} \end{array} \;, & x + t \geq \tau_d \;, \end{cases} \tag{4.5}$$

where q_i is the output rate from state i, $p_{i,k}$ is the probability of reaching state i when there are only exponential transitions involved (i.e., $p_{i,k} = q_{i,k}/q_i$, with $q_{i,k}$ = transition rate from i to k), and $r_{j,k}$ is the probability that the firing of the deterministic transition d leads to state $k \in Sat(\phi)$, given it fires in state $i \in S^d$. Substituing (4.4) and (4.5) in (4.3), we obtain:

$$\begin{aligned} Prob(i, \mathcal{X}^{\leq t}\phi \mid j) \;=\; & \int_0^{\tau_d - t} U(\cdot)\,(1 - e^{-q_i t}) \sum_{k \in Sat(\phi)} p_{i,k}\; dx \\ & + \int_{\tau_d - t}^{\tau_d} U(\cdot)\,(1 - e^{-q_i(\tau_d - x)}) \sum_{k \in Sat(\phi)} p_{i,k}\; dx \\ & + \int_{\tau_d - t}^{\tau_d} U(\cdot)\, e^{-q_i(\tau_d - x)} \sum_{k \in Sat(\phi)} r_{i,k}\; dx \;, \end{aligned} \tag{4.6}$$

with

$$U(\cdot) = \left(\sum_{n=0}^{\infty} e^{-q_d x} \frac{(q_d x)^n}{n!}\; \underline{\pi}_{0j} \mathbf{P}_d^n \right) \cdot \underline{1}_i$$

As can be seen, the evaluation of the time-bounded next operator is by far more involved than in the case of pure Markovian models, which discourages its use in practical applications. However, the analysis has been presented here to show that, in the case of DSPN, it is still possible to carry out some calculations via some assumptions in the behavior of the underlying process (i.e. long-term analysis). Appendix C shows an approximation in order to evaluate (4.6) numerically.

4.4 Time/Space complexity and accuracy

Considering only the case of the time-bounded until property, the algorithm complexity is strongly related to the evaluation of $Prob^{\mathcal{M}}(s, \phi_1\ \mathcal{U}^{\leq t}\ \phi_2)$. From the detailed developments given in [25] and [27], in the best case scenario, i.e., without initially enabled deterministic transitions, the asymptotic space complexity is given by $O(|S^E| + \sum_{d \in T^D} |S^d|(\tau_d/h) + \sum_{d \in T^D} |S^d|^2)$ and the asymptotic time complexity is $O(\sum_{d \in T^D} q^d t |S^d|^2 (\tau_d/h))$, where t is the time, q^d is the maximum absolute diagonal entry of $\mathbf{Q}^d$, and h is a fixed stepsize required for the discretization method which is used to numerically solve the differential equations (this is a parameter which gives good results for values less than 1% of the smallest deterministic firing time).

The accuracy in the numbers obtained relies on the numerical algorithms involved in the transient and steady-state evaluation. Transient analysis is quite sensible to the underlying algorithms due to the use of discretization in the numerical analysis, and a priori error bounds can not be given. This can be considered a drawback of the applicability of model checking techniques, although, to our best knowledge, it is the only practical way of dealing with this kind of analysis.

5 Tool support

To develop the algorithms involved in model checking DSPN we use the tool SPNica [10, 25, 29] for carrying out most of the numerical analysis.The framework used to implement/test the algorithms described in this paper is summarized in the block diagram of Figure 5, where the arrows reflect the relation between blocks. A description of each block in Figure 5 is as follows:

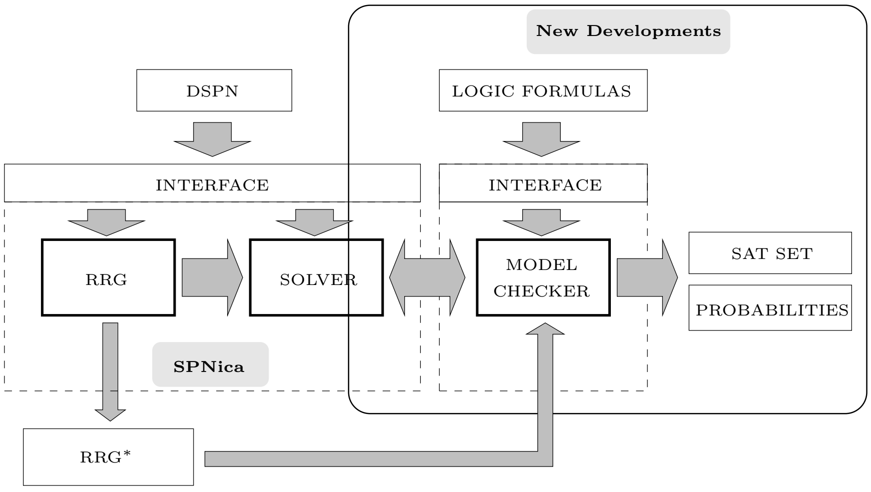


Fig. 5. Block Diagram of the Tool

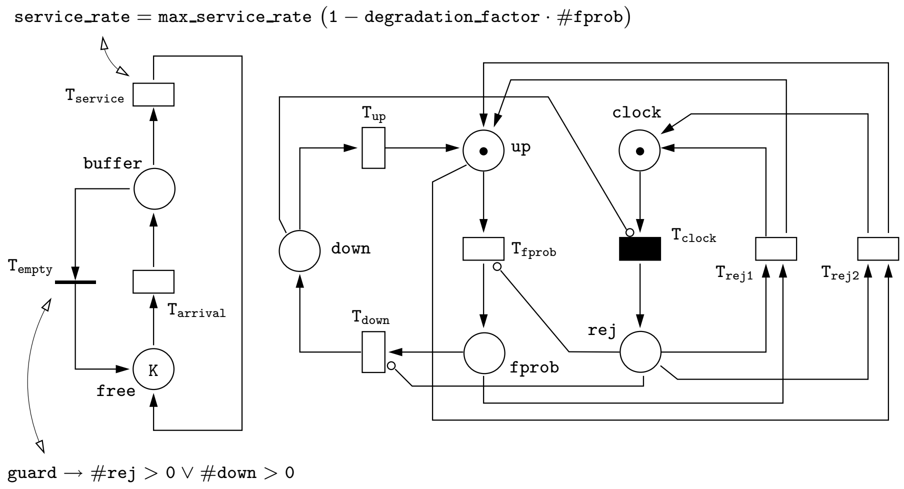
- **DSPN**: Specification of the model, for a detailed explanation of the syntax and semantics see [10] or [25].
- **INTERFACE**: Set of functions to allow access to the main algorithms either for the SPNica package or the model checker.

- **RRG**: Set of functions to generate the reduced reachability graph of a Petri net. The exploration of states is carried out through a breadth-first-search algorithm.
- **SOLVER**: Set of functions to evaluate probabilities and reward measures related to the model under study. The evaluation is carried out by, first, obtaining the reduced reachability graph, and then calculating the required measures, using either steady state or transient analysis algorithms.
- **RRG***: Reduced reachability graph. This is an output of the SPNica package that is used by our checker as an input to get information about the process under study.
- **LOGIC FORMULA**: Input to the model checker. The logic formulas are specified through a syntax built using the names of places specified in the DSPN model together with logical functions taken from Mathematica.
- **MODEL CHECKER**: Set of functions for evaluating the operators involved in the model checking procedure (i.e. steady-state operator, until operator and next operator). These functions interact with the SOLVER of SPNica in order to calculate those probabilities needed to determine the logical value of the expression being checked. The information of the process is acquired via the RRG.
- **SAT SET**: Set of states that satisfies the formula being checked.
- **PROBABILITIES**: Probabilities associated to each state for which the logic formula is evaluated. In case of a nested formula, the probabilities are related to the outer one.

6 Application Example

In Figure 6 we introduce a DSPN model of a software driven server with a preventive maintenance (rejuvenation) policy. The part of the DSPN on the left represents the M/M/1/K queueing model for the server; part of the DSPN on the right represents the failure/maintenance process.

The DSPN on the right of Figure 6 has been taken from [30], where the model was used to study software rejuvenation. In our case, it also models the degraded service suffered by the server due to aging of the software [31, 32]. A token in place `up` models a server fully available. The transition $\mathtt{T_{fprob}}$ models the aging of the software giving service. The firing of this transition causes the system to enter a failure-probable and performance-degraded state (place `fprob`). Transition $\mathtt{T_{down}}$ models a crash failure of the software. $\mathtt{T_{up}}$ models a software restart, and while this transition is enabled, all other activities are suspended. The deterministic transition $\mathtt{T_{clock}}$ models the forced maintenance (rejuvenation) period. Once it fires, a token is moved to place `rej` and the activity related with the maintenance starts. During this maintenance period all other activities related with the failure are suspended, which is modeled by the inhibitor arcs from `rej` to $\mathtt{T_{fprob}}$ and $\mathtt{T_{down}}$. After the forced maintenance period, the net is reinitialized with one token in place `up` and one token in place `clock`, and all the other places of this net empty.



service_rate = max_service_rate (1 − degradation_factor · #fprob)
T_service
buffer
T_empty
T_arrival
K
free
guard → #rej > 0 ∨ #down > 0
T_up
up
clock
down
T_fprob
T_clock
T_rej1
T_rej2
T_down
rej
fprob

Fig. 6. M/M/1/K system with preventive maintenance policy and service degradation

Regarding the server model (DSPN on the left in Figure 6), the free capacity of the server is modeled with the place free. The transition $\mathtt{T_{arrival}}$ models the arrival of jobs. The place buffer models the number of jobs enqueued in the server. The transition $\mathtt{T_{service}}$ represents the service event. The service rate of the server (service_rate) depends on the age of the system since the last rejuvenation (if there was any), which is modeled with a degradation factor related to the place fprob: if there is a token in place fprob, the service rate is lower by a certain factor (degradation_factor) in comparison to the service rate in a non-degraded system (max_service_rate). Either a total failure (firing of transition $\mathtt{T_{down}}$) or a forced maintenance (firing of transition $\mathtt{T_{clock}}$), empties the tokens of the place buffer, via the immediate transition $\mathtt{T_{empty}}$, which is enabled when the place rej or the place down contain a token (this is modeled with a guard associated with $\mathtt{T_{empty}}$).

The following table shows the values for the transitions rates for the DSPN model in Figure 6; it also shows the server parameters:

rejuvenation DSPN		server DSPN	
$\mathtt{T_{fprob}}$	1/15 (1/*days*)	$\mathtt{T_{arrival}}$	360 (1/*days*)
$\mathtt{T_{down}}$	1/30 (1/*days*)	max_service_rate	720 (1/*days*)
$\mathtt{T_{rej1}}$	72 (1/*days*)	degradation_factor	0.3
$\mathtt{T_{rej2}}$	72 (1/*days*)	max_capacity	10
$\mathtt{T_{up}}$	72 (1/*days*)		
$\mathtt{T_{clock}}$ (**determ.**)	1 to 24 (1/*days*)		

As a numerical example we consider two measures: (1) the transient probability of having a saturated buffer (buffer = max_capacity), given we start in the marking shown in Figure 6 (m_0), and (2) the transient probability of having a saturated buffer, given we start in the marking m_0 and no performance-degradation nor failure has occurred during the evaluation time ($\texttt{fprop} = 0 \wedge \texttt{down} = 0$). The CSL operator that allows us to express these measures is the time-bounded until. Assuming an evaluation time of 1 day, we have the following CSL expressions for the previous requirements:

(1) $\mathit{Prob}\big(m_0,\ \texttt{tt}\,\mathcal{U}^{\leq 1.0}\ (\texttt{buffer} = \texttt{max\_capacity})\big)$
(2) $\mathit{Prob}\big(m_0,\ (\texttt{fprob} = 0 \wedge \texttt{down} = 0)\,\mathcal{U}^{\leq 1.0}\ (\texttt{buffer} = \texttt{max\_capacity})\big)$

The resulting probabilities are depicted in Figure 7, considering different values for the firing time of $\texttt{T}_{\texttt{clock}}$: from 1 hour to 24 hours.

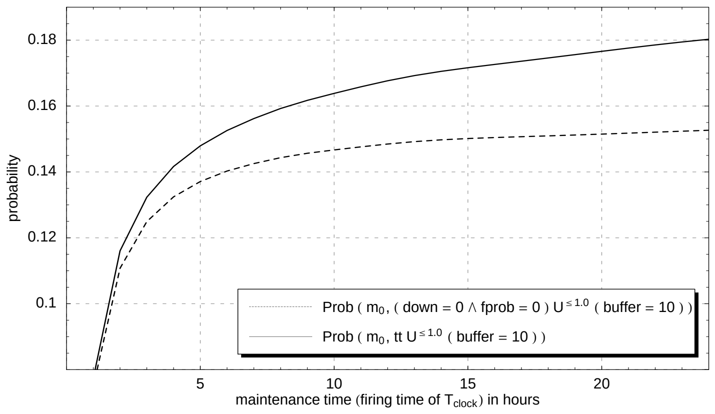


Fig. 7. Transient probability for different values of preventive maintenance (rejuvenation) time (firing time of T_{clock}), given the initial marking, m_0, in Figure 6.

It is clear from Figure 7, that the probability of having the buffer full of jobs increases when we increase the preventive maintenance time, the reason is that the firing of $\texttt{T}_{\texttt{clock}}$ empties the buffer, thus a smaller maintenance time implies more frequent buffer flushes, moving away the possibility of a buffer full of jobs.

The difference between the two resulting lines in Figure 7 is due to the different paths followed by the process during the analysis: the upper line, case (1), depicts the fact the server may suffer both, performance degradation and failure, leading to a condition of full buffer quicker than in case (2), lower line in Figure 7, where we consider the server does not fail or degrades its performance, i.e., the utilization factor of the server in case (1) is greater than in case (2).

This simple example shows us one of the strenght of the CSL model checking technique and its application in performance analysis [33]: it allows us to consider a single model, without any modification on it, in order to obtain the required measures, a task that, with pure analysis, would have needed some manual adaption of the model.

7 Concluding Remarks

In this paper we investigate the application of CSL model checking to DSPNs. The logic expressions are defined in terms of atomic properties which refer to the number of tokens in the places of the DSPN. The CSL model checking algorithms (originally intended for continuous-time Markov processes) are adapted to Markov regenerative processes, specifically those underlying a DSPN.

From a practical point of view, the applicability of model checking techniques in DSPNs falls behind the CTMC case. Verifying a CSL formula for DSPNs is more time/space consuming than with CTMCs due to the fact we are dealing with non-memoryless transitions. In this context, it has been shown how much involved the verification of the time-bounded next operator is, a procedure that is straightforward in the Markovian case.

Verification of the time-bounded until operator must include some assumptions for the initial conditions, in order to apply the transient evaluation algorithms developed for DSPNs: a deterministic transitions may be initially enabled, but its elapsed time must be assumed zero. This constraint requires further investigation. Early experiments have shown that the transient probability associated with this operator can be bounded for initial states where a deterministic transition is enabled, regardless of the time it has been enabled.

We have developed a prototype Mathematica based model checking tool for DSPNs, based on the package SPNica [10, 29]. An example is presented in order to show the use of the CSL extension to DSPNs.

References

1. Molloy, M.K.: Performance analysis using stochastic Petri nets. IEEE Transactions on Computers **31** (1982) 913–917
2. Marsan, M.A., Conte, G., Balbo, G.: A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. ACM Transactions on Computer Systems **2** (1984) 93–122
3. Meyer, J.F., Movaghar, A., Sanders, W.H.: Stochastic activity networks: Structure, behavior, and application. In: Proceedings of the International Workshop on Timed Petri Nets (PNPM'85), Torino, Italy, IEEE Computer Society Press (1985) 106–115
4. Marsan, M.A., Chiola, G.: On Petri nets with deterministic and exponentially distributed firing times. In: Proceedings of the 7th European Workshop on Applications and Theory of Petri Nets. Volume 266 of Lecture Notes in Computer Science., Springer Verlag (1986) 132–145
5. Choi, H., Kulkarni, V., Trivedi, K.S.: Markov regenerative stochastic Petri nets. Performance Evaluation **20** (1994) 337–357
6. German, R., Logothetis, D., Trivedi, K.S.: Transient analysis of Markov regenerative stochastic Petri nets: A comparison of approaches. In: Proceedings of the 6th International Workshop on Petri Nets and Performance Models (PNPM'95), IEEE Computer Society Press (1995) 103–112
7. Lindemann, C., Thümmler, A.: Transient analysis of deterministic and stochastic Petri nets with concurrent deterministic transitions. Performance Evaluation **36–37** (1999) 35–54

8. Lindemann, C.: Performance Modelling with Deterministic and Stochastic Petri Nets. John Wiley & Sons Ltd. (1998)
9. German, R., Kelling, C., Zimmermann, A., Hommel, G.: TimeNET: A toolkit for evaluating non-Markovian stochastic Petri nets. Performance Evaluation **24** (1995) 69–87
10. German, R.: Markov regenerative stochastic Petri nets with general execution policies: Supplementary variable analysis and a prototype tool. Performance Evaluation **39** (2000) 165–188
11. Emerson, E.A.: Temporal and modal logic. In Leeuwen, J.V., ed.: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B). Elsevier Science (1990) 995–1072
12. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing **6** (1994) 512–535
13. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.K.: Verifying continuous time Markov chains. In Alur, R., Henzinger, T.A., eds.: Proceedings of the 8th International Conference on Computer Aided Verification (CAV'96). Volume 1102., New Brunswick, NJ, USA, Springer Verlag (1996) 269–276
14. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE Transactions on Software Engineering **29** (2003) 524–541
15. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In Kemper, P., ed.: Proceedings Tools Session of Aachen 2001 International Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems. (2001) 7–12
16. D'Aprile, D., Donatelli, S., Sproston, J.: CSL model checking for the GreatSPN tool. In: Proceedings of the 19th International Symposium on Computer and Information Sciences (ISCIS 2004). Volume 3280 of Lecture Notes in Computer Science., Kemer-Antalya, Turkey, Springer Verlag (2004) 543–552
17. Buchholz, P., Katoen, J.P., Kemper, P., Tepper, C.: Model-checking large structured Markov chains. The Journal of Logic and Algebraic Programming **56** (2003) 69–97
18. Infante López, G.G., Hermanns, H., Katoen, J.P.: Beyond memoryless distributions: Model checking semi-Markov chains. In de Alfaro, L., Gilmore, S., eds.: PAPM-PROBMIV 2001. Volume 2165 of Lecture Notes in Computer Science., Springer-Verlag, Heidelberg (2001) 57–70
19. Baier, C., Katoen, J.P., Hermann, H.: Approximate symbolic model checking of continuous-time Markov chains (extended abstract). In Baeten, J.C.M., Mauw, S., eds.: CONCUR'99 - Concurrency Theory. LNCS 1664. Springer Verlag (1999) 146–161
20. Ciardo, G., German, R., Lindemann, C.: A characterization of the stochastic process underlying a stochastic Petri net. IEEE Transactions on Software Engineering **20** (1994) 506–515
21. Marsan, M.A., Balbo, G., Bobbio, A., Chiola, G., Conte, G., Cumani, A.: The effect of execution policies on the semantics and analysis of stochastic Petri nets. IEEE Transactions on Software Engineering **15** (1989) 832–846
22. Cox, D.R., Miller, H.D.: The Theory of Stochastic Processes. Chapman and Hall (1965)
23. Cinlar, E.: Introduction to Stochastic Processes. Prentice Hall (1975)
24. German, R.: New results for the analysis of deterministic and stochastic petri nets. In: Proceedings of IEEE Int. Performance and Dependability Symposium, Erlangen, Germany (1995) 114–123

25. German, R.: Performance Analysis of Communication Systems: Modeling with Non-Markovian Stochastic Petri Nets. John Wiley & Sons Ltd. (2000)
26. Nuutila, E., Soisalon-Soininen, E.: On finding the strongly connected components in a directed graph. Information Processing Letters (1993) 9–14
27. Heindl, A., German, R.: A fourth-order algorithm with automatic stepsize control for the transient analysis of DSPNs. IEEE Transactions on Software Engineering **25** (1999) 194–206
28. Gross, D., Miller, D.R.: The randomization technique as a modeling tool and solution procedure for transient Markov processes. Operations Research **32** (1984) 343–361
29. German, R.: SPNica. (ftp://ftp.wiley.co.uk/pub/books/german/)
30. Garg, S., Puliafito, A., Telek, M., Trivedi, K.S.: Analysis of software rejuvenation using Markov regenerative stochastic Petri nets. In: Proceedings of the 6th International Symposium on Software Reliability Engineering (ISSRE'95), IEEE Computer Society Press (1995)
31. Huang, Y., Kintala, C., Kolettis, N., Fulton, N.D.: Software rejuvenation: Analysis, module and applications. In: Proceedings of the 25th International Symposium on Fault-Tolerant Computing (FTCS'95), IEEE Computer Society Press (1995) 381–390
32. Garg, S., Puliafito, A., Telek, M., Trivedi, K.: Analysis of preventive maintenance in transactions based software systems. IEEE Transactions on Computers **47** (1998) 96–107
33. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model checking meets performance evaluation. ACM SIGMETRICS Performance Evaluation Review **32** (2005) 10–15

A Transient Analysis of MRGP

In this section we describe the transient behavior of the underlying MRGP of a DSPN via the method of supplementary variables [22], as developed in [25].

– If there is no deterministic transition enabled in state i, i.e., $i \in S^E$, the infinitesimal change in the probabilities for those states $i \in S^E$ is given by one of the following events: (1) firing of and exponential transition enabled in a state $j \neq i, j \in S^E$ that leads to the state i, (2) firing of a deterministic transition which leads to the state i, (3) firing of an exponential transition which preempts a deterministic transition and leads to the state i, and (4) firing of an exponential transition enabled in state i which leads to a state $j \neq i, j \in S$. These four events are represented in the same order, on the right-hand side of the following ordinary differential equation:

$$\begin{aligned} \frac{d}{dt}\pi_i(t) \;=\; & \sum_{\substack{j\neq i,\\ j\in S^E}} \pi_j(t)\, q_{j,i} + \sum_{d\in T^D} \sum_{j\in S^d} \pi_j(t,\tau_d)\, \delta_{j,i} \\ & + \sum_{d\in T^D} \sum_{j\in S^d} \pi_j(t)\, \bar{q}_{j,i} - \sum_{\substack{j\neq i,\\ j\in S}} \pi_i(t)\, q_{i,j} \;, \quad i \in S^E \end{aligned} \tag{A.1}$$

– If there is a deterministic transition $d \in T^d$ already enabled in state i, i.e., $i \in S^d$, the change of state probabilities for $i \in S^d$ is due to one of the following events: (1) firing of an exponential transition enabled in state $i \in S^d, d \in T^D$, that does not disable d, and (2) firing of an exponential transition enabled in state i which leads to a state $j \neq i, j \in S^d$. These events are represented on the right hand side of the following partial differential equation:

$$\frac{\partial}{\partial t}\pi_i(t,x) + \frac{\partial}{\partial x}\pi_i(t,x) \;=\; \sum_{\substack{j\neq i,\\ j\in S^d}} \pi_j(t,x)\, q_{j,i} - \sum_{\substack{j\neq i,\\ j\in S^d}} \pi_i(t,x)\, q_{i,j} \;, \tag{A.2}$$

where $0 < x < \tau_d\,,\ i \in S^d$. It must be noted that this system of partial differential equations can be reduced to a system of ordinary differential equations by the method of characteristics [24].

– *Initial conditions*: Given we are dealing with partial and ordinary differential equations, we must specify initial conditions

$$\pi_i(0) = \begin{cases} 1, & \text{if } i = s \\ 0, & \text{otherwise} \end{cases} , i \in S^E \cup S^D \;, \quad \pi_i(0,x) = \pi_{0_i} \;,\; i \in S^D \;, \tag{A.3}$$

where $x > 0$, π_{0_i} represents the initial state occupancy distribution.

– *Boundary conditions*: We have to consider those conditions that enable a deterministic transition $d \in T^d$. This may occur due either to: (1) firing of an exponential transition enabled in $j \in S^E$ leading to a state $i \in S^d$, (2) firing of a deterministic transition $c \in T^d$ leading to a state $i \in S^d$, or (3) firing of an exponential transition that preempts a deterministic transition leading to a state $i \in S^d$. These events are represented on the right-hand side of the following equation:

$$\pi_i(t,0) \;=\; \sum_{j \in S^E} \pi_j(t)\, q_{j,i} + \sum_{c \in T^D} \sum_{j \in S^c} \pi_j(t,\tau_c)\, \delta_{j,i} + \sum_{c \in T^D} \sum_{j \in S^c} \pi_j(t)\, \bar{q}_{j,i}\,, \quad (A.4)$$

where $i \in S^d$.

– We finally compute $\pi_i(t)$ through the following integral equation:

$$\pi_i(t) \;=\; \int_0^{\tau_d} \pi_i(t,x)\, dx\,, \quad i \in S^d, d \in T^D\,, \quad (A.5)$$

where, $q_{i,j}$, $\bar{q}_{i,j}$, and $\delta_{i,j}$, with $1 \le i,j \le |S|$, correspond to entries in the matrices defined in Section 2.

B Steady-State Analysis of MRGP

The set of equations that describe the long-run behavior, via the method of supplementary variable [25], can be derived from the transient state equations (A.1), (A.2), (A.4) and (A.5), having as a result:

– If there is no deterministic transition enabled in the state i, i.e., $i \in S^E$, we have from (A.1):

$$0 \;=\; \sum_{\substack{j \neq i \\ j \in S^E}} \pi_j\, q_{j,i} \;+\; \sum_{d \in T^D} \sum_{j \in S^d} \pi_j\, \delta_{j,i} \;+\; \sum_{d \in T^D} \sum_{j \in S^d} \pi_j\, \bar{q}_{j,i} \;-\; \sum_{\substack{j \neq i \\ j \in S}} \pi_i\, q_{i,j} \quad (B.1)$$

– If there is a deterministic transition enabled in the state i, i.e., $i \in S^d, d \in T^D$, we obtain from (A.2):

$$\frac{d}{dx}\pi_i(x) \;=\; \sum_{j \in S^d} \pi_i(x)\, q_{j,i} \quad (B.2)$$

– Boundary conditions for states $i \in S^d, d \in T^D$, are derived from (A.4):

$$\pi_i(0) \;=\; \sum_{j \in S^E} \pi_j\, q_{j,i} \;+\; \sum_{c \in T^D} \sum_{j \in S^c} \pi_j(\tau_c)\, \delta_{j,i} \;+\; \sum_{c \in T^D} \sum_{j \in S^c} \pi_j\, \bar{q}_{j,i} \quad (B.3)$$

– Finally, the state probability of $i \in S^d, d \in T^D$, can be obtained from (A.5):

$$\pi_i \;=\; \int_0^{\tau_d} \pi_i(x)\, dx \quad (B.4)$$

C Time-Bounded Next Property: complementary developments

In the equation (4.6), the integrals on the right side can be eliminated by using the following equivalence:

$$\int_0^t e^{-qx}\frac{(qx)^n}{n!}\,dx \;=\; \frac{1}{q}\left(1-\sum_{l=0}^{n} e^{-qt}\,\frac{(qt)^l}{l!}\right)\ ,$$

resulting, after some algebraic manipulation:

$$Prob(i, \mathcal{X}^{\leq t}\phi \mid j) \;=\; I_1 \;+\; I_2 \;+\; I_3\ ,$$

where I_1, I_2, and I_3 are given by

$$\begin{aligned} I_1 \;=\; & \sum_{k\in Sat(\phi)} p_{i,k}\ \underline{\pi}_{0j} \sum_{n=0}^{\infty} \mathbf{P}_d^n \cdot \underline{1}_i (1-e^{-q_i t})\ \frac{1}{q_d} \\ & \cdot \left(1-\sum_{l=0}^{n} e^{-q_d(\tau_d - t)}\,\frac{(q_d(\tau_d-t))^l}{l!}\right) \end{aligned} \tag{C.1}$$

$$\begin{aligned} I_2 \;=\; & \sum_{k\in Sat(\phi)} p_{i,k}\ \underline{\pi}_{0j} \sum_{n=0}^{\infty} \mathbf{P}_d^n \cdot \underline{1}_i \\ & \cdot \left[\frac{1}{q_d}\left(\sum_{l=0}^{n} e^{-q_d(\tau_d-t)}\,\frac{(q_d(\tau_d-t))^l}{l!} - \sum_{l=0}^{n} e^{-q_d\tau_d}\,\frac{(q_d\tau_d)^l}{l!}\right)\right. \\ & \quad - e^{q_i\tau_d}\,\frac{q_d^n}{(q_d-q_i)^{n+1}}\left(\sum_{l=0}^{n} e^{-(q_d-q_i)(\tau_d-t)}\,\frac{((q_d-q_i)(\tau_d-t))^l}{l!}\right. \\ & \quad \left.\left. - \sum_{l=0}^{n} e^{-(q_d-q_i)\tau_d}\,\frac{((q_d-q_i)\tau_d)^l}{l!}\right)\right] \\ = \;& \sum_{k\in Sat(\phi)} p_{i,k}\ \underline{\pi}_{0j} \sum_{n=0}^{\infty} \mathbf{P}_d^n \cdot \underline{1}_i \\ & \cdot \left[\frac{1}{q_d}\sum_{l=0}^{n} e^{-q_d\tau_d}\,\frac{q_d^l}{l!}\,\left(e^{q_d t}(\tau_d-t)^l - \tau_d^l\right)\right. \\ & \quad \left. - \frac{q_d^n}{(q_d-q_i)^{n+1}}\sum_{l=0}^{n} e^{-q_d\tau_d}\,\frac{(q_d-q_i)^l}{l!}\left(e^{(q_d-q_i)t}(\tau_d-t)^l - \tau_d^l\right)\right] \end{aligned} \tag{C.2}$$

$$\begin{aligned} I_3 &= \sum_{k \in Sat(\phi)} r_{j,k} \; \underline{\pi}_{0j} \; \sum_{n=0}^{\infty} \mathbf{P}_d^n \cdot \underline{1}_i \; e^{-q_i \tau_d} \frac{q_d^n}{(q_d - q_i)^{n+1}} \\ &\quad \cdot \left(\sum_{l=0}^{n} e^{-(q_d - q_i)(\tau_d - t)} \frac{((q_d - q_i)(\tau_d - t))^l}{l!} \right. \\ &\quad \left. - \sum_{l=0}^{n} e^{-(q_d - q_i)\tau_d} \frac{((q_d - q_i)\tau_d)^l}{l!} \right) \\ &= \sum_{k \in Sat(\phi)} r_{j,k} \; \underline{\pi}_{0j} \; \sum_{n=0}^{\infty} \mathbf{P}_d^n \cdot \underline{1}_i \; e^{-q_i \tau_d} \frac{q_d^n}{(q_d - q_i)^{n+1}} \\ &\quad \cdot \sum_{l=0}^{n} e^{-(q_d - q_i)\tau_d} \frac{(q_d - q_i)^l}{l!} \left(e^{(q_d - q_i)t} \left((\tau_d - t)^l - \tau_d^l \right) \right) \end{aligned} \tag{C.3}$$