Performance Queries on Semi-Markov Stochastic Petri Nets with an Extended Continuous Stochastic Logic

Jeremy T. Bradley Nicholas J. Dingle Peter G. Harrison William J. Knottenbelt
Department of Computing, Imperial College London
180 Queen's Gate, London SW7 2AZ, United Kingdom
Email: {jb,njd200,pgh,wjk}@doc.ic.ac.uk

Abstract

Semi-Markov Stochastic Petri Nets (SM-SPNs) are a high-level formalism for defining semi-Markov processes. We present an extended Continuous Stochastic Logic (eCSL) which provides an expressive way to articulate performance queries at the SM-SPN model level. eCSL supports queries involving steady-state, transient and passage time measures. We demonstrate this by formulating and answering eCSL queries on an SM-SPN model of a distributed voting system with up to 10^7 *states.*

1 Introduction

Formal logics for asking performance-related questions of stochastic systems provide a concise and rigorous way to pose such questions and allow for the composition of simple questions into more complex queries. One such logic is Continuous Stochastic Logic (CSL), which was originally presented in [4, 5] and applied practically to Markovian state spaces in [6, 7, 18]. CSL can express performance measures by selecting states and paths from a system that meet both steady-state and passage time quantile criteria. CSL has since been applied to Generalized Semi-Markov Processes [24] to specify performance properties on discrete event simulations, but its analytic application to semi-Markov chains is relatively recent [17].

In this paper, we present extended CSL (eCSL) which augments semi-Markov CSL with the ability to express a richer class of passage time quantities as well as measures based on transient distributions. Unlike basic CSL, which operates at a state-transition level, eCSL is designed to operate at the model level. Specifically it operates on semi-Markov stochastic Petri nets (SM-SPNs) [8], which are used here as a convenient high-level abstraction that maps onto an underlying semi-Markov process (SMP). Application of a temporal logic at the model level of a stochastic system has also been attempted for stochastic process algebras (aCSL [16]).

The concept of generating a semi-Markov process or Markov renewal process from a stochastic Petri net is not new. Indeed, since 1984 there have been many papers on non-Markovian stochastic Petri nets [11, 12, 13, 14, 20, 21]. EPSNs [11] were the first proposal for a non-Markovian stochastic Petri net formalism. Here, general distributions are allowed on transition firings and structural restrictions are imposed on the Petri net to ensure that either a Markov or semi-Markov process is derived. Classes of transitions help define the structural restrictions: a transition is *exclusive* if, whenever it is enabled, no other transition is enabled; a transition is *competitive* if it is non-exclusive and its firing both interrupts and disables all other enabled transitions; a transition is *concurrent* if it is non-exclusive and its firing does not disable all other enabled transitions.

It is established that an SMP can only be generated from an ESPN if transitions with general firing-time distributions (GEN) are constrained to be exclusive. Markovian transitions (with exponential firing times), on the other hand, can be both concurrently and competitively enabled.

When more than one GEN transition is enabled [20, 21], then issues of *scheduling policies* for residual pre-empted transition times need to be considered. In [12] three main scheduling policies for pre-empted transitions in Markov Regenerative SPNs are presented:

pre-emptive resume (*prs*) the original (firing-time) distribution sample is remembered and work done (time elapsed) is conserved for when the transition is next enabled

pre-emptive restart identical (*pri*) the distribution sample is remembered, but work done is lost and the transition firing delay starts from 0 when it is next enabled

pre-emptive restart different (*prd*) the original distribution sample is forgotten, work done is discarded, and when the transition is next enabled, the transition firing delay is resampled and starts from 0.

It is important to note that SM-SPNs do not try to tackle the issue of concurrently enabled GEN transitions in the most general case. If more that one GEN transition is enabled then a probabilistic choice is invoked to determine which will be fired. Pre-empted GEN transitions use a *prd* schedule if they become re-enabled. This approach is correctly described in [21] as not being a solution to the more complex issue of properly concurrently enabled GEN transitions, but is merely a way of specifying a different type of model – a semi-Markov model in fact where GEN transitions are forced to be exclusive. Where we do not have concurrently enabled GEN transitions, then proper concurrent and competitive transition behaviour is catered for with *prs* scheduling for pre-empted transitions.

We analyse SM-SPNs for passage and transient quantities in similar fashion to the Laplace domain solution technique in [14]. In order to make the calculations tractable for large state spaces (of the order of 10^7 states in this paper), we use a constant space representation of the Laplace transform functions based on the evaluation demands of the inversion algorithm [8]. We use either the Euler [1] method or the Laguerre [2] method with optimisations [15] to obtain our final passage time and transient distributions.

The rest of this paper is organised as follows. In Section 2, we briefly define the semi-Markov process and recap the definition of an SM-SPN. In Section 3, we discuss the current state of CSL and highlight those areas which we enhance in eCSL. eCSL itself is presented with example formulae in Section 4. In Section 5, we describe a distributed voting system and check eCSL specifications on it for models of up to 10^7 states.

2 Semi-Markov Stochastic Petri Nets

In this section, we recap the description of a semi-Markov process in terms of its underlying discrete-time Markov chain (DTMC) and a sojourn time distribution matrix. It is shown how SM-SPNs are defined over a semi-Markov state space and that SM-SPNs are a conservative extension of SPNs and GSPNs.

2.1 Semi-Markov Processes

Consider a Markov renewal process $\{(X_n, T_n) : n \geq 0\}$ where T_n is the time of the nth transition ($T_0 = 0$) and $X_n \in S$ is the state at the nth transition. Let the kernel of this process be:

$$R(n,i,j,t) = \mathbb{P}(X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_n = i) \tag{1}$$

for $i, j \in S$. The continuous time semi-Markov process (SMP), $\{Z(t), t \geq 0\}$, defined by the kernel R, is related to the Markov renewal process by:

$$Z(t) = X_{N(t)} \tag{2}$$

where $N(t) = \max\{n : T_n \leq t\}$, i.e. the number of state transitions that have taken place by time t. Thus $Z(t)$ represents the state of the system at time t. We consider time-homogeneous SMPs, in which $R(n,i,j,t)$ is independent of n, i.e. :

$$R(i,j,t) = p_{ij} H_{ij}(t) \tag{3}$$

where $p_{ij} = \mathbb{P}(X_1 = j \mid X_0 = i)$ for all n. p_{ij} is the state transition probability between states i and j and $H_{ij}(t) = \mathbb{P}(T_{n+1} - T_n \leq t \mid X_{n+1} = j, X_n = i)$ is the sojourn time distribution in state i when the next state is j.

Further information on the passage time and transient analysis of Markov and semi-Markov systems, specifically in the Laplace domain, can be found in [8, 10, 15].

2.2 SM-SPN Definition

Semi-Markov stochastic Petri nets [8] are extensions of GSPNs [3] which support arbitrary marking-dependent holding-time distributions and which generate an underlying semi-Markov process rather than a Markov process. As discussed already, it is not intended that they be a novel technique for dealing with concurrently-enabled GEN transitions. They are instead a useful high level vehicle for eCSL to operate over and for the demonstration of large semi-Markov model analysis.

Formally, an SM-SPN consists of a 4-tuple, $(PN, \mathcal{P}, \mathcal{W}, \mathcal{D})$, where:

- $PN = (P, T, I^-, I^+, M_0)$ is the underlying Place-Transition net. P is the set of places, T is the set of transitions, $I^{+/-}$ are the forward and backward incidence functions describing the connections between places and transitions and M_0 is the initial marking.
- $\mathcal{P} : T \times \mathcal{M} \to \mathbb{Z}^+$, denoted $p_t(m)$, is a marking-dependent priority function for a transition.
- $\mathcal{W} : T \times \mathcal{M} \to \mathbb{R}^+$, denoted $w_t(m)$, is a marking-dependent weight function for a transition, to allow implementation of probabilistic choice.

- $\mathcal{D} : T \times \mathcal{M} \to (\mathbb{R}^+ \to [0,1])$, denoted $d_t(m,r)$, is a marking-dependent cumulative distribution function for the firing time of a transition.

In the above, $\mathcal{M}$ is the set of all markings for a given net. Further, we define the following general net-enabling functions:

- $\mathcal{E}_N : \mathcal{M} \to P(T)$, a function that specifies net-enabled transitions from a given marking.
- $\mathcal{E}_P : \mathcal{M} \to P(T)$, a function that specifies priority-enabled transitions from a given marking.

The net-enabling function, $\mathcal{E}_N$, is defined in the usual way for standard Petri nets: if all preceding places have occupying tokens then a transition is net-enabled. Similarly, we define the more stringent priority-enabling function, $\mathcal{E}_P$. For a given marking, m, $\mathcal{E}_P(m)$ selects only those net-enabled transitions that have the highest priority, i.e. $\mathcal{E}_P(m) = \{t \in \mathcal{E}_N(m) : p_t(m) = \max\{p_{t'}(m) : t' \in \mathcal{E}_N(m)\}\}$. Now for a given priority-enabled transition, $t \in \mathcal{E}_P(m)$, the probability that it will be the one that actually fires after a delay sampled from its firing distribution, $d_t(m,r)$, is:

$$\mathbb{P}(t \in \mathcal{E}_P(m) \text{ fires}) = \frac{w_t(m)}{\sum_{t' \in \mathcal{E}_P(m)} w_{t'}(m)} \qquad (4)$$

Note that the choice of which priority-enabled transition is fired in any given marking is made by a probabilistic selection based on transition weights, and is not a race condition based on finding the minimum of samples extracted from firing-time distributions. This mechanism enables the underlying reachability graph of an SM-SPN to be mapped directly onto a semi-Markov chain.

2.3 Expressing SPNs and GSPNs as SM-SPNs

The marking-dependence of the weights and distributions allows us to translate SPNs and GSPNs into the SM-SPN paradigm in a straightforward manner. An SPN can be specified in the SM-SPN formalism in the following way. We let μ_t represent the exponential firing rate of a transition, t, in the SPN. Then $p_t(m) = 0$ for all t, m; $w_t(m) = \mu_t$ for all m; $d_t(m,r) = 1 - \exp(-\mu^\Sigma r)$ where $\mu^\Sigma = \sum_{t' \in \mathcal{E}_N(m)} \mu_{t'}$, i.e. the sum of the firing rates of the enabled transitions.

For GSPNs, the situation is very similar except that the immediate transitions have priority over the timed transitions. The translation to SM-SPN necessarily distinguishes between timed transitions ($t \in T_1$, having rate μ_t) and immediate transitions ($t \in T_2$, having a probabilistic weight c_t): $p_t(m) = 0$ if $t \in T_1$,1 if $t \in T_2$; $w_t(m) = \mu_t$ if $t \in T_1$, c_t if $t \in T_2$; $d_t(m,r) = 1 - \exp(-\mu^\Sigma r)$ if $t \in T_1$, $H(r;0)$ if $t \in T_2$, where $\mu^\Sigma = \delta_{t \in \mathcal{E}_P(m)} \sum_{t' \in \mathcal{E}_P(m)} \mu_{t'}$, $H(r;a)$ is the Heaviside function with step at time a, and δ_B is 1 if condition B is true and 0 otherwise. This gives us a meaningful combined exponential rate, μ^Σ, if only exponential transitions are priority enabled.

3 CSL

3.1 Technical Summary

In order to make detailed comparisons with CSL, and to understand fully the enhancements we introduce in eCSL, we first present a detailed summary of the standard CSL as used in [6, 7, 18, 17].

For our purposes, a semi-Markov CSL (similar to [17]) is defined directly over a semi-Markov state space, (S, P, H, A), where S is the set of states, P is the embedded probability transition matrix, H is the state holding time distribution matrix and A is a state labelling function. This labelling function attaches multiple labels to every state, and allows states to be identified by a more meaningful annotation than their integer position in the transition matrix.

A general CSL formula is defined as follows:

$$\Psi \overset{\text{def}}{=} \text{tt} \;\Big|\; a \;\Big|\; \Psi \wedge \Psi \;\Big|\; \neg\Psi \;\Big|\; \mathcal{S}_{\boldsymbol{\rho}}(\Psi) \;\Big|\; \mathcal{P}_{\boldsymbol{\rho}}(\psi) \qquad (5)$$

$$\psi \overset{\text{def}}{=} X\Psi \;\Big|\; \Psi \cup^{\boldsymbol{\tau}} \Psi \qquad (6)$$

$\mathcal{S}$ represents a steady-state condition and $\mathcal{P}$ represents a passage time condition on a set of paths defined by ψ. The values $\boldsymbol{\rho}$ and $\boldsymbol{\tau}$ represent ranges of allowed probabilities and times, respectively[1].

The semantics of the logic are expressed by stating the precise conditions under which a single state s satisfies each of the possible clauses of a Ψ-formula; as for other temporal logics this is written $s \models \Psi$.

The clause a is a label and a state s satisfies that label if $a \in A(s)$. Thus using the not and conjunctive clauses in combination with labelling allows whole sets of states to be defined with a Ψ-formula. The set of states specified in this manner is written $\text{Sat}(\Psi) = \{s \in S \mid s \models \Psi\}$.

Thus the steady-state clause $\mathcal{S}_{\boldsymbol{\rho}}(\Psi)$ defines a set of states $S_1 = \text{Sat}(\Psi)$ and is true if the sum of the long term probabilities of the states in S_1 lies in the range $\boldsymbol{\rho}$.

[1]In CSL, $\boldsymbol{\rho}$ and $\boldsymbol{\tau}$ are usually represented with $\leq x \equiv \{y : 0 < y \leq x\}$ where x needs to be rational for model checking to be decidable [4]; we use an arbitrary set here for notational simplicity.

3.2 Formal CSL semantics

The formal semantics of CSL are:

$$\begin{array}{ll} s \models \text{tt} & \text{for all } s \\ s \models a & \text{iff } a \in A(s) \\ s \models \Psi_1 \wedge \Psi_2 & \text{iff } s \models \Psi_1 \wedge s \models \Psi_2 \\ s \models \neg\Psi & \text{iff } s \not\models \Psi \\ s \models \mathcal{S}_{\boldsymbol{\rho}}(\Psi) & \text{iff } \Pi_{\vec{j}} \in \boldsymbol{\rho} \text{ where } \vec{j} = \mathrm{Sat}(\Psi) \\ s \models \mathcal{P}_{\boldsymbol{\rho}}(\psi) & \text{iff } \mathbb{P}(\sigma \in \mathrm{Path}(s) \mid \sigma \models \psi) \in \boldsymbol{\rho} \end{array} \qquad (7)$$

where $\mathrm{Path}(s)$ is the set of all paths starting from s. The quantity $\Pi_{\vec{j}}$ is the long term probability of being in any of the states in $\vec{j}$.

Further, a path σ satisfies a path formula, ψ, as follows:

$$\begin{array}{ll} \sigma \models X\Psi & \text{iff } \exists\sigma[1] \models \Psi \\ \sigma \models \Psi_1 \cup^{\boldsymbol{\tau}} \Psi_2 & \text{iff } \exists u \in \boldsymbol{\tau} \,. \\ & (\sigma@u \models \Psi_2 \wedge \forall u' < u, \sigma@u' \models \Psi_1) \end{array} \qquad (8)$$

where $\sigma[1]$ is a state immediately succeeding the start state of σ; $\sigma@t$ is the state that the system is in at time t on the path σ.

The X path operator is often referred to as the *next state operator* and is used to extract an aggregate DTMC probability for selecting a given set of successor states, $\mathrm{Sat}(\Psi)$.

Finally, the *time-bounded until* formula, $\Psi_1 \cup^{\boldsymbol{\tau}} \Psi_2$, specifies a set of paths starting in a single state s which satisfy Ψ_1 for the duration of the path and terminate when they satisfy Ψ_2, and this is further restricted to complete the passage in time $u \in \boldsymbol{\tau}$.

3.3 Opportunities for Enhancing CSL

There are three main issues regarding the specification of performance measures that arise from the definition of CSL:

1. Only a single start state can be specified for the time-bounded until formula with the existing formulation of CSL. Passage time specifications are more expressive when associated with many possible start states (as shown in [8, 15]). This is useful when asking performance questions of high level formalisms where the start and end conditions for a passage may not necessarily specify a unique state.

2. There is no ability to express performance conditions based on transient distributions.

3. Although compound formulae of steady-state and passage time constraints[2] are, technically speaking, allowed, the meaning of the derived formulae is somewhat obscure.

As an example of this last point, we consider $\mathcal{P}_{\boldsymbol{\rho}_1}(\mathcal{S}_{\boldsymbol{\rho}_2}(\Psi_1) \cup^{\boldsymbol{\tau}} \Psi_2)$ which would define a passage along a set paths that consists of states which satisfy $\mathcal{S}_{\boldsymbol{\rho}_2}(\Psi_1)$, and terminate satisfying Ψ_2. As long as $\mathrm{Sat}(\Psi_1)$ has a steady-state value in $\boldsymbol{\rho}_2$ (and the underlying process is irreducible), then all states will satisfy $\mathcal{S}_{\boldsymbol{\rho}_2}(\Psi_1)$, which therefore represents no constraint on the selected paths at all. Alternatively, if $\mathrm{Sat}(\Psi_1)$ does not have a steady-state value in $\boldsymbol{\rho}_2$, then only paths of length 0 may be selected.

Similarly abstruse would be an $\mathcal{S}$ formula which relied on the possible start states of a $\mathcal{P}$ formula, for example, $\mathcal{S}_{\boldsymbol{\rho}_1}(\mathcal{P}_{\boldsymbol{\rho}_2}(\Psi_1 \cup^{\boldsymbol{\tau}} \Psi_2))$: that is, calculating the long term state probability over the set of possible start states of the $\mathcal{P}$ portion.

It is here that we detect a slight conflict between the world of model-checking, which is concerned with finding a state or set of states which satisfy a temporal property, and the world of performance modelling, where the sets of states tend to be known entities and the unknown quantity is a probability or time value.

4 eCSL

We now present an extension to CSL, called eCSL, which can express a greater variety of performance related questions: for example, transient distribution-based properties and multiple start states for both passage and transient properties. Unlike standard CSL, which is applied directly to a labelled Markovian state space, eCSL operates on semi-Markov stochastic Petri nets, which can express any semi-Markov model or Markovian model with immediate transitions.

eCSL complements CSL, insofar as CSL concentrates on describing properties which are formally decidable, whereas eCSL focuses on providing performance queries of a more pragmatic nature.

For the reasons given in Section 3, we remove the possibility of specifying compound formulae in the manner of CSL. This highlights a pleasant abstraction within the new temporal logic: we now have one layer for specifying a set of states and a separate layer for specifying performance criteria.

[2]It is interesting to note that the steady-state clause, $\mathcal{S}$, was not originally part of the CSL syntax as laid down in Aziz et al [4] or in their later paper [5], so in that formulation only compound passage time formulae would have been permitted.

We also simplify the path formulae of the original CSL and instead specify paths by providing high-level rules that yield a set of start states, a set of terminating states and a set of excluded states though which a path cannot pass. Taking into account the fact that CSL could not specify multiple start states, this is equivalently expressive to the logical until formula, which provides a single start state, a set of end states and a set of states that the passage is restricted to. In eCSL, sets of states themselves are specified in terms of markings on the semi-Markov stochastic Petri net.

As we will see, these simplifications make for a formalism which maps more pleasantly onto both Petri nets and the underlying stochastic quantities. It also keeps simple the path formulae required to specify complex performance measures.

4.1 The Syntax of eCSL

We define the syntax of eCSL over SM-SPNs. An eCSL statement, Ψ, acting on an SM-SPN system with set of markings, M, is defined by:

$$\begin{aligned}\Psi &\overset{\text{def}}{=} \mathtt{tt} \;\big|\; \Psi \wedge \Psi \;\big|\; \neg\Psi \\ &\quad \big|\; \mathcal{S}_{\boldsymbol{\rho}}(\psi) \;\big|\; \mathcal{T}^{\boldsymbol{\tau}}_{\boldsymbol{\rho}}(\psi,\psi) \;\big|\; \mathcal{P}^{\boldsymbol{\tau}}_{\boldsymbol{\rho}}(\psi,\psi) \\ \psi &\overset{\text{def}}{=} \mathtt{tt} \;\big|\; p[N] \;\big|\; \psi \wedge \psi \;\big|\; \neg\psi \end{aligned} \tag{9}$$

Here we have deliberately separated out the state specification ψ-formulae from the performance specification Ψ-formulae. This avoids the conceptual problems associated with the compound performance properties that arise in CSL, while still being sufficiently expressive to allow for multiple simultaneous performance criteria to be specified. We define the function which operates on a ψ-formula and extracts the set of all states that satisfy it as $\mathrm{Sat}(\psi) = \{m \in M \mid m \models \psi\}$.

In the ψ specification, $N \in 2^{\mathbb{N}}$ and $p[N]$ is satisfied if the number of tokens on place p in some state m is in the set of allowed numbers of tokens N. As with CSL, $\boldsymbol{\rho} \in 2^{[0,1]}$ is a set of allowed probabilities and similarly $\boldsymbol{\tau} \in 2^{[0,\infty]}$ is a set of times.

$\mathcal{S}_{\boldsymbol{\rho}}(\psi)$ is true if the steady-state probability of being in the set of states defined by ψ lies in the set $\boldsymbol{\rho}$.

$\mathcal{T}^{\boldsymbol{\tau}}_{\boldsymbol{\rho}}(\psi_1,\psi_2)$ is satisfied by a set of start states if the probability of the system being in states $\mathrm{Sat}(\psi_1)$ at time t, while not having passed through states $\mathrm{Sat}(\psi_2)$, lies in $\boldsymbol{\rho}$ for all times $t \in \boldsymbol{\tau}$ (shown for an arbitrary transient distribution in Fig. 1).

Finally, $\mathcal{P}^{\boldsymbol{\tau}}_{\boldsymbol{\rho}}(\psi_1,\psi_2)$ is true for a set of start states if the random variable representing the passage time to target states

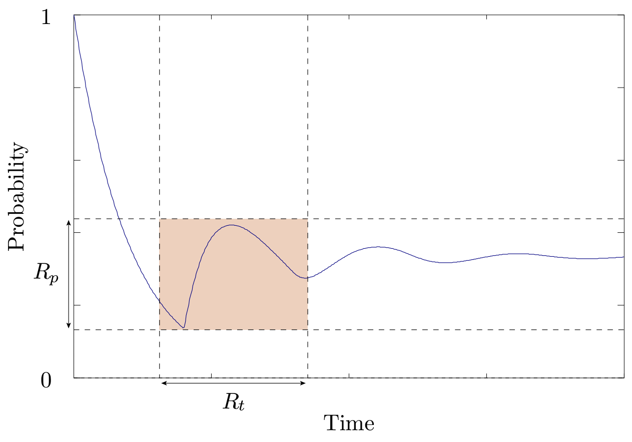


Fig. 1. An example of a transient constraint $\vec{m} \models \mathcal{T}^{R_t}_{R_p}(\Psi)$ which is satisfied by a transient distribution in the shaded area.

$\mathrm{Sat}(\psi_1)$, while not having traversed states in $\mathrm{Sat}(\psi_2)$, lies in the range of times $\boldsymbol{\tau}$ with probability $p \in \boldsymbol{\rho}$.

For a high-level modelling paradigm, we believe that specifying rules for sets of excluded states is simpler than having to specify explicitly all the permitted states for a path with state-by-state logical formulae, as used by standard CSL on conventionally labelled state spaces.

4.2 Examples of eCSL Formulae

As an example of how eCSL could be used to pose performance questions in practice, we consider the problem of finding the value of q that satisfies the formula:

$$\mathrm{Sat}(p_1[35] \wedge p_5[10]) \models \mathcal{P}^{[0,10)}_{\{q\}}(p_2[175], p_6[1]) \tag{10}$$

The question being asked is: what is the probability that a defined passage takes less than time 10? The passage time quantity is defined by the source states $p_1[35] \wedge p_5[10]$, by the target states $p_2[175]$ and taking into account the excluded states, $p_6[1]$. These expressions define sets of states, for instance $p_1[35] \wedge p_5[10]$ selects all the Petri net markings which have 35 tokens in p_1 and 10 tokens in p_5.

If we wish to define multiple performance requirements for a single set of start states on a system then we might ask:

$$\begin{aligned}&\mathrm{Sat}(p_1[35] \wedge p_5[10]) \models \\ &\qquad \left(\mathcal{P}^{[0,10)}_{(0.9,1]}(p_2[175], p_6[1]) \wedge \mathcal{P}^{[0,100)}_{(0.98,1]}(p_2[320], p_6[4])\right)\end{aligned} \tag{11}$$

This expresses the need to achieve a 90% quantile for a passage time within the first 10 time units of the passage starting and a 98% quantile, over a different passage, within 100 time units. In this way, multiple quality of service requirements may be succinctly expressed and verified with a single eCSL formula.

If distinct start states are required for each performance measure, we could compose them as follows:

$$\Big(\mathrm{Sat}(p_5[10]) \models \mathcal{P}^{[0,10)}_{(0.9,1]}(p_2[175], p_6[1])\Big) \wedge \\ \Big(\mathrm{Sat}(p_1[35]) \models \mathcal{T}^{[0,100)}_{(0.2,1]}(p_2[320], p_6[4])\Big) \quad (12)$$

4.3 Formal Stochastic Semantics of eCSL

In this section, we formally define the satisfiability formulae for eCSL over SM-SPNs. These are expressed in terms of a marking, m, of an SM-SPN where $m(p)$ is the number of tokens at place p in the marking. We test individual markings of the Petri net against every allowed combination of Ψ and ψ-expressions. As before, these are evaluated in terms of individual satisfiability questions, e.g. $m \models \psi_1$, which poses the question: does the single state m satisfy the formula ψ_1?

Formally, for the general ψ-expression:

$$\begin{array}{ll} m \models \text{tt} & \text{for all } m \\ m \models p[N] & \text{iff } m(p) \in N \\ m \models \psi_1 \wedge \psi_2 & \text{iff } m \models \psi_1 \wedge m \models \psi_2 \\ m \models \neg\psi & \text{iff } m \not\models \psi \end{array} \quad (13)$$

Importantly, the Ψ-formulae are satisfied by vectors of markings or states. This is so that a configuration of multiple start states can be defined and used to specify corresponding multiple performance properties. This overcomes a restriction inherent in CSL: specifically that of only being able to express performance properties with single start states.

$$\begin{array}{ll} \vec{m} \models \text{tt} & \text{for all } \vec{m} \\ \vec{m} \models \Psi_1 \wedge \Psi_2 & \text{iff } \vec{m} \models \Psi_1 \wedge \vec{m} \models \Psi_2 \\ \vec{m} \models \neg\Psi & \text{iff } \vec{m} \not\models \Psi \\ \vec{m} \models \mathcal{S}_{\boldsymbol{\rho}}(\psi) & \text{iff } \Pi_{\vec{j}} \in \boldsymbol{\rho} \text{ where } \vec{j} = \mathrm{Sat}(\psi) \\ \vec{m} \models \mathcal{T}^{\boldsymbol{\tau}}_{\boldsymbol{\rho}}(\psi_1, \psi_2) & \text{iff } \forall t \in \boldsymbol{\tau}, T^{\vec{k}}_{\vec{m}\vec{j}}(t) \in \boldsymbol{\rho} \text{ where} \\ & \quad \vec{j} = \mathrm{Sat}(\psi_1), \vec{k} = \mathrm{Sat}(\psi_2) \\ \vec{m} \models \mathcal{P}^{\boldsymbol{\tau}}_{\boldsymbol{\rho}}(\psi_1, \psi_2) & \text{iff } \mathbb{P}(P^{\vec{k}}_{\vec{m}\vec{j}} \in \boldsymbol{\tau}) \in \boldsymbol{\rho} \text{ where} \\ & \quad \vec{j} = \mathrm{Sat}(\psi_1), \vec{k} = \mathrm{Sat}(\psi_2) \end{array} \quad (14)$$

The steady-state operator $\Pi_{\vec{j}}$ represents the long term probability of being in the set of states $\vec{j}$ (independent of any start state, if the underlying system is irreducible).

For the transient operator, $\mathcal{T}$, we have a modified transient distribution function to take account of the excluded states in $\vec{k}$:

$$T^{\vec{k}}_{\vec{i}\vec{j}}(t) = \sum_{i \in \vec{i}} \alpha_i \mathbb{P}(Z(t) \in \vec{j} \mid Z(0) = i, \forall t' < t \,.\, Z(t') \notin \vec{k}) \quad (15)$$

The $\mathbb{P}(\cdot)$ term inside the summation describes the conditional probability that the SMP is in a state in $\vec{j}$ at time t given that it started from a state i and has never been through any state in $\vec{k}$. This probability is finally deconditioned over the set of all the possible start states in $\vec{i}$. $\tilde{\alpha}$ is taken to be a normalised steady-state vector, but there is no reason why it could not be generalised to an arbitrary initial weighting vector, specified by the user (although there is not currently syntactic support for this in eCSL).

Similarly for the passage time operator, $\mathcal{P}$, we can modify the passage time random variable to incorporate the excluded states vector $\vec{k}$:

$$P^{\vec{k}}_{\vec{i}\vec{j}} = \sum_{i \in \vec{i}} \alpha_i \inf\{u > 0 : Z(u) \in \vec{j} \\ \mid Z(0) = i, \forall u' < u \,.\, Z(u') \notin \vec{k}\} \quad (16)$$

Calculating the modified passage time probability, $\mathbb{P}(P^{\vec{k}}_{\vec{m}\vec{j}} \in \boldsymbol{\tau}) \in \boldsymbol{\rho}$, or transient probability, $T^{\vec{k}}_{\vec{i}\vec{j}}(t)$, quantities involves straightforward modification of the formulae of the standard passage time and transient formulae for an SMP [8]. All the excluded states in an exclusion set, $\vec{k}$, are removed from the embedded DTMC ($\forall i \in S, k \in \vec{k}$ let $p_{ik} = 0$ and $p_{ki} = 0$), while renormalising the probabilities as necessary, so that $\sum_j p_{ij} = 1$ for all i. The renormalising of the DTMC, after removal of the excluded states, reflects the conditional nature of Eq. (15) and Eq. (16).

4.4 Practical eCSL Implementation

In practice, the transient formula $\mathcal{T}^{\boldsymbol{\tau}}_{\boldsymbol{\rho}}(\psi_1, \psi_2)$ is approximated by the constraint that the discrete samples of the transient distribution, as extracted from the numerical Laplace transform inversion of $T^{\vec{k}*}_{\vec{i}\vec{j}}(s)$, all lie within the specified probability range, $\boldsymbol{\rho}$. The approximation arises because the function may have fluctuations outside this range, on a timescale shorter than the sampling frequency. If necessary the modeller can increase the sampling frequency on the transient function as appropriate.

It is not envisaged, as would be expected with a traditional temporal logic, that the set of start states satisfying an eCSL formula would be an unknown in an eCSL equation, e.g. find the start states $\vec{m}$ that satisfy the formula $\mathcal{T}^{(10,20)}_{(0.7,1]}(p_7[M/2], p_6[N])$ for some constants M and

Tab. 1. Complexity of different Ψ-formulae if the start-state set is known.

eCSL clause	Complexity
ψ	$O(N)$
$\mathcal{S}_{\boldsymbol{\rho}}(\psi)$	$O(N^2 \log N)$ to $O(N^3)$
$\mathcal{P}^{\boldsymbol{\mathcal{T}}}_{\boldsymbol{\rho}}(\psi_1, \psi_2)$	$\mathcal{G}(\psi_2)$
$\mathcal{T}^{\boldsymbol{\mathcal{T}}}_{\boldsymbol{\rho}}(\psi_1, \psi_2)$	$\lvert\mathrm{Sat}(\mathcal{T}^{\boldsymbol{\mathcal{T}}}_{\boldsymbol{\rho}}(\psi_1, \psi_2))\rvert \, \lvert\mathrm{Sat}(\psi_1)\rvert \, \mathcal{G}(\psi_2)$

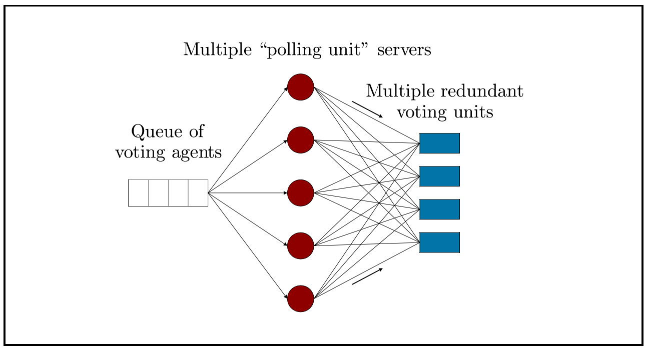
N. This is reasonable as the exclusion of compound performance formulae, for other reasons, means that the modeller is never using a performance measure to define a set of states to use in a second performance measure. So, although certainly possible, it is also not advisable as the search space for an unknown subset of start states with a given performance property could be $O(2^N)$ in the worst case.

Of more practical use to the performance modeller is the calculation of the time and probability constraints, $\boldsymbol{\rho}$ and $\boldsymbol{\tau}$, in the eCSL formulae, when start states and end states are defined. For this situation, we give some complexity results in Tab. 1. $\mathcal{G}(\psi)$ is the complexity of calculating the Laplace transform of $P^{\vec{k}}_{i\vec{j}}$, a single start-state passage time density measure, where ψ is the formula defining the excluded states, $\vec{k}$. With the constant space representation of transforms [8] and a dense state space, using SOR matrix inversion or similar, $\mathcal{G}(\psi)$ is $O(N'^3)$, where $N' = N - \lvert\mathrm{Sat}(\psi)\rvert$. Using iterative semi-Markov passage time calculation techniques [8] and assuming a sparse state space, in practice $\mathcal{G}$ goes more like $N'^2 \log(N')$. By using an appropriate normalised vector, $\tilde{\alpha}$, passage-time measures with multiple initial states can be simultaneously calculated at the same cost as for a single start-state passage (from [8]). Transient measures in SMPs are expensive to compute, as the complexity is governed by the product of the size of the sets of target and initial states in the formula, along with the standard passage time density calculation cost [22].

5 eCSL Performance Properties in Action

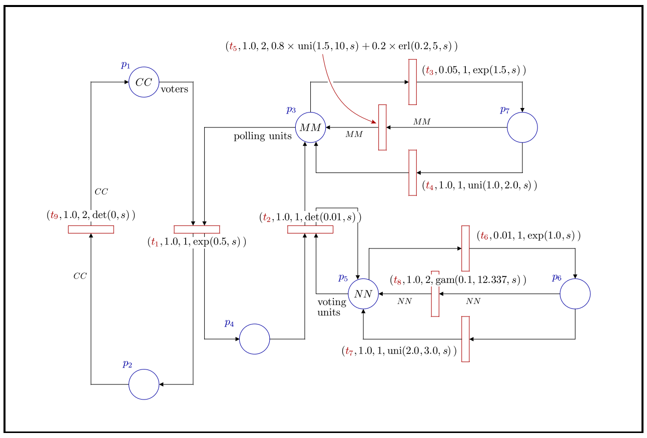
5.1 The Voting Example

Fig. 2 shows the distributed components of a voting system with breakdowns and repairs which we will use to generate a semi-Markov model. A voting agent queues to vote in the buffer; then, as a polling unit becomes free, the polling unit can receive the agent's vote and the agent can be marked as having voted. The polling unit contacts all the currently operational central voting units to register votes with all of them; this is done in order to prevent multiple vote fraud and to provide fault tolerance through redundancy. The polling unit then becomes available to receive another voting agent.



Multiple "polling unit" servers
Multiple redundant voting units
Queue of voting agents

Fig. 2. A queueing model showing the distinct distributed components of the voting system



$(t_5, 1.0, 2, 0.8 \times \mathrm{uni}(1.5, 10, s) + 0.2 \times \mathrm{erl}(0.2, 5, s))$
p_1
CC
voters
$(t_3, 0.05, 1, \exp(1.5, s))$
p_3
MM
polling units
MM
MM
p_7
$(t_4, 1.0, 1, \mathrm{uni}(1.0, 2.0, s))$
CC
$(t_9, 1.0, 2, \det(0, s))$
$(t_1, 1.0, 1, \exp(0.5, s))$
$(t_2, 1.0, 1, \det(0.01, s))$
$(t_6, 0.01, 1, \exp(1.0, s))$
CC
p_5
NN
voting units
$(t_8, 1.0, 2, \mathrm{gam}(0.1, 12.337, s))$
NN
NN
p_6
p_4
$(t_7, 1.0, 1, \mathrm{uni}(2.0, 3.0, s))$
p_2

Fig. 3. A semi-Markov stochastic Petri net of a voting system with breakdowns and repairs

The semi-Markov stochastic Petri net for this system is shown in Fig. 3. Voting agents vote asynchronously, moving from place p_1 to p_2 as they do so. A restricted number of polling units which receive their votes transit t_1 to place p_4. At t_2, the vote is registered with as many central voting units as are currently operational in p_5.

The system is considered to be in a failure mode if either all the polling units have failed and are in p_7 or all the central voting units have failed and are in p_6. If either of these complete failures occur, then with high priority a repair is performed which resets the failed units to a fully operational state. If some, but not all the polling or voting units fail, they attempt self-recovery. The system will continue to function as long as at least one polling unit and one voting unit remain operational.

This example is defined in full as a DNAmaca specifica-

```
\transition{t5}{
  \condition{p7 > MM-1}
  \action{
    next->p3 = p3 + MM;
    next->p7 = p7 - MM;
  }
  \weight{1.0}
  \priority{2}
  \sojourntimeLT{ return
       (0.8 * uniform(1.5,10,s)
       + 0.2 * erlang(0.001,5,s)); }
}
```

Fig. 4. Excerpt from specification of voting example, showing definition of transition t_5.

tion [19], an excerpt of which is shown in Fig. 4. This defines transition t_5, saying that it:

- is enabled when place p_7 has greater than $MM - 1$ tokens in it.
- removes MM tokens from place p_7 and adds MM tokens to place p_3, when fired.
- has a weight 1.0 (used to define probabilistic choice between transitions when two or more are enabled).
- has a priority of 2, which will enable it above other transitions which would otherwise be structurally enabled but have a lower priority.
- is given a firing distribution which, with probability 0.8, is a uniform distribution or, with probability 0.2, is an Erlang distribution. The Laplace transform, $g^*(s)$, for this firing-time distribution is of the form:

  $p \times \mathit{uniformLT}(a, b, s) + (1-p) \times \mathit{erlangLT}(\lambda, n, s)$

  where $\mathit{uniformLT}(a, b, s) = (e^{-as} - e^{-bs})/(s(b-a))$ and $\mathit{erlangLT}(\lambda, n, s) = (\lambda/(\lambda + s))^n$.

In general, an arbitrary Laplace transform function can be specified as a firing distribution using the `\sojourntimeLT{...}` pragma.

5.2 Analysis

We analyse the voting system, which is natively semi-Markov, using eCSL formulae. GSPNs and SPNs can be queried in exactly the same way, once they have been converted to SM-SPNs using the techniques of Section 2.3.

Tab. 2. Different system configurations used in the eCSL analysis

System				# States
	CC	MM	NN	
A	18	6	3	2061
B	60	25	4	106,540
C	175	45	5	1,140,050
D	300	80	10	10,991,400

Sys. A. Ψ_1**:** $\mathrm{Sat}(p_1[18] \wedge p_3[6] \wedge p_5[3]) \models (\mathcal{T}^{\{t\}}_{\{p\}}(p_2[5] \wedge p_3[6] \wedge p_5[3],\ \mathrm{ff}\ ) \wedge \mathcal{S}_{\{\pi\}}(p_2[5] \wedge p_3[6] \wedge p_5[3]))$
Find probability p that the system has processed exactly 5 voters after a reset at time t, and corresponding steady-state value π. A graph of p against t is presented in Fig. 5; note how the transient distribution tends towards its corresponding steady state value.

Sys. B. Ψ_2**:** $\mathrm{Sat}(p_1[60] \wedge p_3[25] \wedge p_5[4]) \models \mathcal{T}^{(50,83)}_{(0.0015,0.007)}(p_2[30] \wedge p_3[25] \wedge p_5[4],\ \mathrm{ff}\ )$
Does the system process exactly half its voters inside the bounds $t = 50$ and $t = 83$, within the probability range $(0.0015, 0.007)$; see Fig. 6.

Sys. C. Ψ_3**:** $\mathrm{Sat}(p_1[35] \wedge p_5[10]) \models \mathcal{P}^{(0,t]}_{\{p\}}(p_2[175], p_6[1])$
Find the probability p that the last 35 voters are processed within time t given that there were no central voting unit failures. The passage time CDF is shown in Fig. 7; we see, for example, that with probability 0.835, the last 35 voters are processed within 86 seconds.

Sys. D. Ψ_4**:** $\mathrm{Sat}(p_1[300] \wedge p_3[80] \wedge p_5[10]) \models \mathcal{P}^{(0,t]}_{\{p\}}(p_2[300],\ \mathrm{ff}\ )$
Find the probability p that all 300 voters are processed within time t. The passage time distribution from this 10.9 million state SMP is shown in Fig. 8.

6 Conclusion

We have described an extended Continuous Stochastic Logic (eCSL) which can be used to ask formal performance questions of semi-Markov models (as well as GSPNs and SPNs). eCSL enhances conventional CSL by supporting a wider range of performance questions, such as passage times with multiple source states and transient measures.

We also presented a semi-Markov Petri net formalism (SM-SPN) which we use to specify semi-Markov systems i.e.

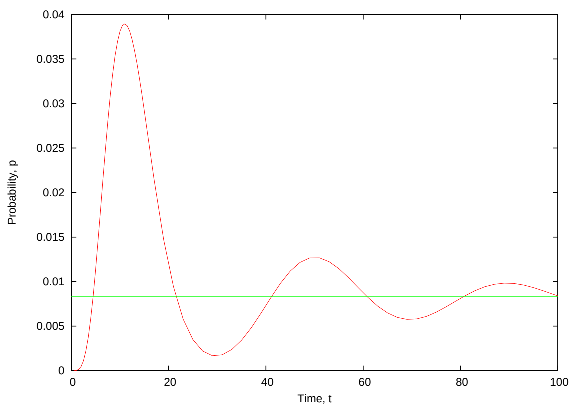


Fig. 5. Transient measure and steady-state measure Ψ_1 for system A: 2061 states

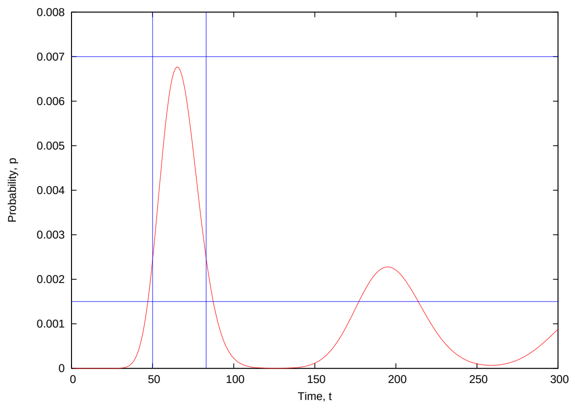


Fig. 6. Transient constraint Ψ_2 for system B: 106,540 states

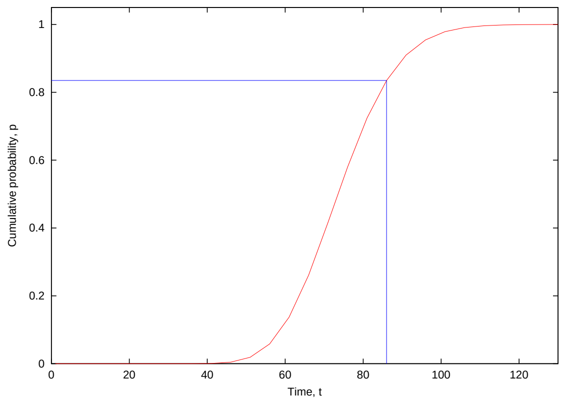


Fig. 7. Passage time quantile Ψ_3 for system C: 1,140,050 states

ones with more than just exponential and immediate transitions. eCSL was defined in terms of markings on an

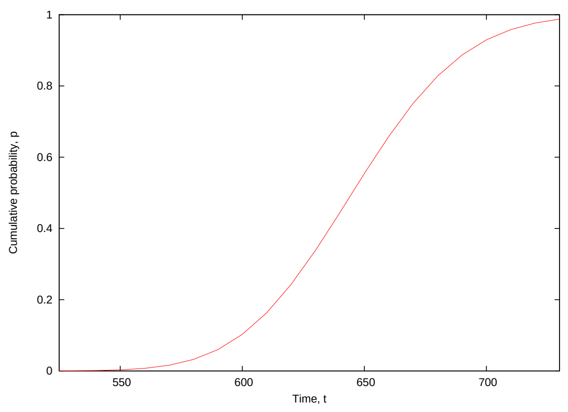


Fig. 8. Passage time quantile Ψ_4 for system D: 10,991,400 states

SM-SPN, giving the modeller a more pleasant abstraction with which to express performance questions than the actual state space would provide.

Finally, we analysed eCSL-derived performance queries on semi-Markov systems of up to 10.9 million states; this is a considerable advance on the previous state of the art for a semi-Markov system.

As future work, we would like to reduce the complexity of verifying the eCSL formulae. In particular, the transient constraint is currently computationally intensive and it may be possible to perform a transient distribution check in the same amount of time as required for a passage time calculation. Also by looking at some more general $\tilde{\alpha}$ initial distribution vectors, it may be possible to derive satisfiability sets for general eCSL expressions in polynomial time.

Acknowledgements

The authors would like to thank the Imperial College Parallel Computing Centre for the use of the Viking Beowulf Cluster. We are particularly grateful to Steve Newhouse and Keith Sephton for their help and technical advice. We would also like to thank Philippa Gardner, Tony Field, Jeff Magee and very diligent referees for helpful comments on many aspects of this paper.

References

[1] J. Abate and W. Whitt. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems*, 10(1):5–88, 1992.

[2] J. Abate and W. Whitt. Numerical inversion of Laplace transforms of probability distributions. *ORSA Journal on Computing*, 7(1):36–43, 1995.

[3] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.

[4] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous-time Markov chains. In *Computer-Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 269–276. Springer-Verlag, 1996.

[5] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.

[6] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *CAV'2000, Proceedings of the 12th International Conference on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 358–372, Chicago, July 2000. Springer-Verlag.

[7] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *CONCUR'99, Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–162. Springer-Verlag, 1999.

[8] J. T. Bradley, N. J. Dingle, P. G. Harrison, and W. J. Knottenbelt. Distributed computation of passage time quantiles and transient state distributions in large semi-Markov models. In *Performance Modelling, Evaluation and Optimization of Parallel and Distributed Systems*, Nice, April 2003. IEEE Computer Society Press.

[9] L. de Alfaro and S. Gilmore, editors. *PAPM-PROBMIV 2001, Proceedings of Process Algebra and Probabilistic Methods*, volume 2165 of *Lecture Notes in Computer Science*. Springer-Verlag, Aachen, September 2001.

[10] N. J. Dingle, P. G. Harrison, and W. J. Knottenbelt. Response time densities in Generalised Stochastic Petri Net models. In *Proceedings of the 3rd International Workshop on Software and Performance (WOSP'2002)*, pages 46–54, Rome, July 2002.

[11] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola. Extended stochastic Petri nets: Applications and analysis. In E. Gelenbe, editor, *Performance'84, Proceedings of 10th Intl. Symp. on Models of Computer System Performance*, pages 507–519. Elsevier, Paris, December 1984.

[12] R. M. Fricks, A. Puliafito, M. Telek, and K. S. Trivedi. Applications of non-Markovian stochastic Petri nets. *Performance Evaluation Review*, 26(2):15–27, 1998.

[13] R. German. *Performance Analysis of Communication Systems: Modelling with Non-Markovian Stochastic Petri Nets*. John Wiley & Sons, 2000.

[14] R. German, D. Logothetis, and K. S. Trivedi. Transient analysis of Markov regenerative stochastic Petri nets: A comparison of approaches. In *PNPM'95, Proceedings of Petri Nets and Performance Models*, pages 103–112, Durham, North Carolina, 1995.

[15] P. G. Harrison and W. J. Knottenbelt. Passage-time distributions in large Markov chains. In M. Martonosi and E. d. S. e Silva, editors, *Proceedings of ACM SIGMETRICS 2002*, pages 77–85, Marina Del Rey, USA, June 2002.

[16] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. Towards model checking stochastic process algebra. In *IFM 2000, Proceedings of 2nd International Conference on Integrated Formal Methods*, pages 420–439, November 2000.

[17] G. G. Infante López, H. Hermanns, and J.-P. Katoen. Beyond memoryless distributions: Model checking semi-Markov chains. In de Alfaro and Gilmore [9], pages 57–70.

[18] J.-P. Katoen, M. Kwiatkowska, G. Norman, and D. Parker. Faster and symbolic CTMC model checking. In de Alfaro and Gilmore [9], pages 23–38.

[19] W. J. Knottenbelt. Generalised Markovian analysis of timed transitions systems. MSc thesis, University of Cape Town, South Africa, July 1996.

[20] C. Lindemann. *Performance Modelling with Deterministic and Stochastic Petri Nets*. John Wiley & Sons, 1998.

[21] A. Puliafito, M. Scarpa, and K. S. Trivedi. Petri nets with k simultaneously enabled generally distributed timed transitions. *Performance Evaluation*, 32(1):1–34, 1998.

[22] R. Pyke. Markov renewal processes with finitely many states. *Annals of Mathematical Statistics*, 32(4):1243–1259, December 1961.

[23] C. M. Woodside and Y. Li. Performance Petri net analysis of communication protocol software by delay-equivalent aggregation. In *Proceedings of the 4th International Workshop on Petri nets and Performance Models*, pages 64–73, Melbourne, December 1991. IEEE Computer Society Press.

[24] H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In E. Brinksma and K. G. Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 223–235, Copenhagen, July 2002. Springer-Verlag.