

AUTOMATED GENERATION AND ANALYSIS OF MARKOV REWARD MODELS USING STOCHASTIC REWARD NETS

GIANFRANCO CIARDO*, ALEX BLAKEMORE†,
PHILIP F. CHIMENTO, JR.‡, JOGESH K. MUPPALA* AND
KISHOR S. TRIVEDI§

Abstract. Markov and Markov reward models are widely used for the performance and reliability analysis of computer and communication systems. Models of real systems often contain thousands or even millions of states. We propose the use of Stochastic Reward Nets (SRNs) for the automatic generation of these large Markov reward models. SRNs do allow the concise specification of practical performance, reliability and performability models.

An added advantage of using SRNs lies in the possibility of analyzing the (time-independent) logical behavior of the modeled system. This helps both the validation of the system (is the right system being built?) and of the model (does the model correctly represent the system?).

We discuss the methods to convert SRNs into Markov reward processes automatically. We review the solution techniques for the steady state and transient analysis of SRNs and Markov reward processes. We also discuss methods for the sensitivity analysis of SRNs.

1. Introduction. Reliability block diagrams and fault trees are commonly used for system reliability and availability analysis [61]. These model types allow a concise description of the system under study and can be evaluated efficiently, but they cannot easily represent dependencies occurring in real systems [59]. Markov models, on the other hand, are capable of capturing various kinds of dependencies that occur in reliability/availability models [8,24,32].

Task precedence graphs [39,42,58] can be used for the performance analysis of concurrent programs with unlimited system resources. Product-form queueing networks [40,41] can be used to model contention for resources. The product-form assumptions are not satisfied, however, when behavior such as concurrency within a job, synchronization, and server failures is considered. Once again, Markov models do provide a framework to address all these concerns [21].

The common solution for modeling reliability/availability or performance would then appear to be the use of Markov models, but one major drawback of Markov models is the largeness of their state spaces. Stochastic Petri nets (SPNs) can be used to generate a large underlying Markov

* Software Productivity Consortium, Herndon, VA 22070.

† Dept. of Computer Science, University of Maryland, College Park, MD 20742.

‡ IBM Corporation, Research Triangle Park, NC 27709.

§ Dept. of Electrical Engineering, Duke University, Durham, NC 27706. This work was supported in part by the National Science Foundation under Grant CCR-9108114 and by the Naval Surface Weapons Center under the ONR Grant N00014-91-5-4162.

model automatically starting from a concise description.

Traditionally, performance analysis assumes a fault-free system. Separately, reliability and availability analysis is carried out to study system behavior in the presence of component faults, disregarding the performance levels of different configurations. Several types of interactions and corresponding tradeoffs have prompted researchers to consider combined evaluation of performance and reliability/availability [44,70]. Most work on the combined evaluation is based on the extension of Markov models to Markov reward models [30], where a reward is attached to each state of the Markov model.

Markov reward models have the potential to reflect concurrency, contention, fault-tolerance, and degradable performance [63]; they can be used to obtain not only program/system performance and system reliability/availability measures [14], but also combined measures of performance and reliability/availability [6,13,49,70]. Since the Markov chain is generated from a concise SPN model, it is necessary to express the reward structure in terms of SPN entities. In other words, the SPN becomes a “Stochastic Reward Net (SRN)” which can be automatically transformed into a Markov reward model.

Steady-state analysis of SRNs is often adequate to study the performance of a system, but time-dependent behavior is sometimes of greater interest: instantaneous availability, interval availability, and reliability (for a fault-tolerant system); response time distribution of a program (for performance evaluation of software); computational availability (for a degradable system) are examples where transient analysis is required. Except for a few instances [10,18,7,60], transient analysis of SPN models has not received much attention in the past.

Analytical models are often used during the design phase of a system, when exact values for all the input parameters may not be known yet. *Parametric sensitivity analysis* is performed so as to determine the parameters to which the model is sensitive and the degree of sensitivity. Frank [23] suggests the use of sensitivity functions as a method of estimating parametric sensitivities. Sensitivity functions are the derivatives of state probabilities with respect to the parameters. We will discuss this method of sensitivity analysis for SRNs.

The SRNs described in this paper, allow the concise specification of Markov reward models, the computation of instantaneous, cumulative, and time-averaged measures both in steady state or in the transient. The derivatives of these measures with respect to input parameters can be obtained as well. Efficient and numerically stable algorithms employing sparse matrix techniques are used to solve the underlying CTMC.

In Section 2, we introduce the SRN formalism. Logical and temporal analysis of SRNs are discussed in Sections 3 and 4, respectively, while Section 5 examines numerical issues connected to the solution of SRNs. Section 6 presents some concluding remarks.

2. From Petri nets to stochastic reward nets. We introduce the SRN formalism incrementally. First, we define the (untimed) PNs in Section 2.1, which describes the basic PN formalism, and in Section 2.2, which contains a set of extensions. Then, we define the stochastic PN (SPN) formalism by adding timing and probabilistic information to the PNs, in Section 2.3. Finally, we define the SRN formalism by imposing a reward structure upon the stochastic process described by a SPN, in Section 2.4.

2.1. Petri nets. PNs were introduced by C. A. Petri in 1962 [52]. A PN is a bipartite directed graph with two classes of nodes, places and transitions. Arcs may only connect places to transitions or transitions to places. Graphically, places are represented by circles or ovals, while transitions are represented by rectangles or bars. An arc from place p to transition t is an *input arc* of t and an *output arc* of p . We also say that p is an *input place* of t and t is an *output transition* of p . Analogous definitions hold for an arc from t to p . Then, $\cdot t$, $t \cdot$, $\cdot p$, and $p \cdot$ denote the set of input and output places of transition t , and the set of input and output transitions of place p , respectively.

A PN graph may be marked by placing tokens, drawn as black circles, in places. If P is the set of places, a *marking* is a multiset, $\mu \in \mathbb{N}^{|P|}$, describing the number of tokens in each place. A marking represents the state of the model at a particular instant. This concept is central to PNs. The notation $\#(p, \mu)$ is used to indicate the number of tokens in place p in marking μ . If the marking is clear from the context, the notation $\#(p)$ can be used.

A transition is *enabled* when each of its input places contains at least one token. An enabled transition may *fire* by removing a token from each of its input places and depositing a token in each of its output places. Firing a transition is an atomic event. While timing becomes an issue only in the timed or stochastic PNs, sequencing is important even in the basic PNs. If two transitions are enabled in a PN, they cannot be fired “at the same time”: a choice must be made concerning which one to fire first, the other can only fire after that, if it is still enabled. This convention greatly simplifies the analysis of PNs.

PNs are dynamic in that they describe how the modeled system evolves over time. A PN evolves by firing enabled transitions to change the marking. Figure 2.1 illustrates a simple PN before and after the firing of transition t_1 , which is enabled because its sole input place, p_1 , contains at least one token. Transition t_2 is disabled because, while p_1 contains a token, p_2 is empty and there is an input arc from p_2 to t_2 . When firing transition t_1 , a token is removed from place p_1 and, at the same time, a token is deposited into each of p_2 and p_3 . In the new marking, both t_1 and t_2 are enabled.

Let μ_i and μ_j be markings and t be a transition. We use the notation $\mu_i \xrightarrow{t}$ to denote that t is enabled in μ_i . If its firing in μ_i changes the marking

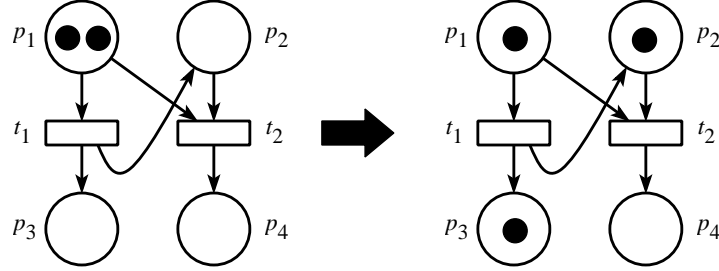


FIG. 2.1. A PN before and after the firing of transition t_1 .

to μ_j , we write $\mu_i \xrightarrow{t} \mu_j$. A sequence of transitions s is a *firing sequence* if the transitions may be legally fired in the order given by the sequence from some marking. We then use the notation $\mu_i \xrightarrow{s} \mu_j$ to mean that the firing sequence s , when started in marking μ_i , terminates in marking μ_j . The notation $\mu_i \xrightarrow{*} \mu_j$ indicates that it is possible to reach μ_j from μ_i with some firing sequence.

The system behavior depends not only upon the structure of the PN graph, but also upon the *initial marking*, μ_0 . Changing the initial marking can completely change the behavior of the PN. We define a PN to include the initial marking in its specification, and will use the term PN graph to refer to a net with an unspecified initial marking.

DEFINITION 2.1. A PN is a 5-tuple $\mathbf{A} = \{P, T, D^-, D^+, \mu_0\}$ where:

- $P = \{p_1, \dots, p_{|P|}\}$ is a finite set of places. Each place contains a non-negative number of tokens.
- $T = \{t_1, \dots, t_{|T|}\}$ is a finite set of transitions ($P \cap T = \emptyset$).
- $D^- \in \{0, 1\}^{|P \times T|}$, $D^+ \in \{0, 1\}^{|P \times T|}$ describe the input and output arcs. There is an input arc from place p to transition t iff $D_{p,t}^- = 1$. Analogously, there is an output arc from transition t to place p iff $D_{p,t}^+ = 1$.
- μ_0 is the initial marking.

A marking μ is *reachable* if $\mu_0 \xrightarrow{*} \mu$. The set of all reachable markings is termed the *reachability set*:

$$\mathcal{S} = \{\mu \mid \mu_0 \xrightarrow{*} \mu\}$$

The evolution of a PN can be completely described by its *reachability graph* $(\mathcal{S}, \mathcal{A})$. Each marking in the reachability set is a node in the reachability graph, while the arcs \mathcal{A} describe the possible marking-to-marking transitions. If $\mu_i \in \mathcal{S}$ and $\mu_i \xrightarrow{t} \mu_j$, then $\mu_i \xrightarrow{t} \mu_j \in \mathcal{A}$. A single PN transition typically corresponds to many arcs in the reachability graph and a relatively simple PN may give rise to a large or even infinite reachability graph. Each arc in \mathcal{A} defines not only the source and destination marking, but also the PN transition causing the change of marking. If more than

one transition can cause this change, \mathcal{A} contains multiple arcs with same source and destination (see Section 3).

Places often represent *local* conditions or states, while transitions represent possible events or activities. According to this interpretation, input arcs describe the conditions necessary for an event to occur, while input and output arcs together describe the effect that an event has upon the system, that is, the changes the event causes in the system state.

It is possible to determine the conditions required for a transition to fire and the subsequent effect solely from the arcs connected to that transition. Likewise, the events (transitions) which depend upon or may affect a condition (place) can be determined solely from the arcs connected to the place. Thus, a PN graph describes the system behavior in terms of local conditions and events, while the reachability graph describes the possible evolution of the *global* system state. This is why PNs have also been called condition/event nets.

PNs excel at modeling parallel systems, in spite of the apparently severe restriction that no two transitions fire simultaneously. When transitions are simultaneously enabled, the associated activities are proceeding in parallel. The first transition to fire simply represents the first activity to complete. Transition enabling corresponds to the *commencement* of an activity, while transition firing corresponds to the *completion* of an activity. It is quite possible that firing a transition will cause a previously enabled transition to become disabled. According to this interpretation, the interrupted activity was aborted before completing and had no lasting effect. Since PN semantics do not state which of two simultaneously enabled transitions must fire first, PN analysis must examine every possible ordering - thus reintroducing the notion of parallelism and non-determinism. The interpretation of the semantics of simultaneously enabled transitions arises again in the context of SPN models, where the choices for semantics and their implications are more complex.

The firing and enabling rules and the basic PN primitives can be combined to model many types of behavior. PNs are especially well suited for describing behavior that arises naturally in complex systems, such as sequencing, concurrency, synchronization, and conflict (see Figure 2.2).

2.2. PN extensions. Many extensions to PNs have been proposed both to increase the class of problems that can be represented (modeling power) and the practical ability to represent common behavior (modeling convenience). Unfortunately, increasing the modeling power decreases the decision power (the set of properties which may be analyzed). Consequently, proposed extensions to the basic PN formalism must be evaluated in terms of their effect upon modeling and decision power. Peterson gives an excellent overview of these tradeoffs [51].

Extensions which affect only modeling convenience may be adopted without losing analytical ability. Such extensions can be removed by trans-

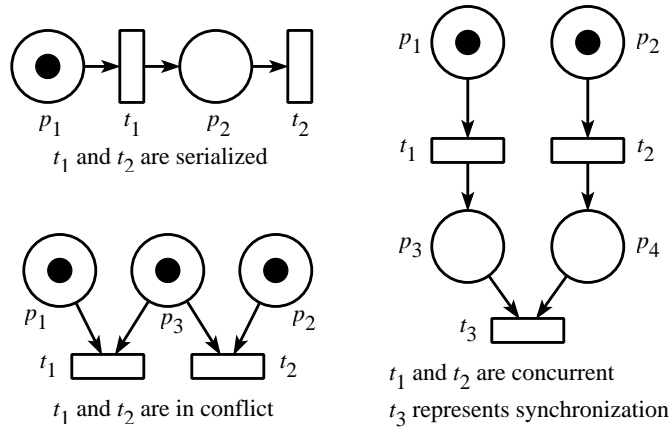


FIG. 2.2. Types of behavior that can be represented by PNs.

forming an extended PN into an equivalent PN satisfying Definition 2.1. Theoretically, such extensions may be regarded as simply a convenient shorthand. In practice, they may drastically improve the ability to apply PNs to real problems.

One such extension has been so widely adopted that it is now considered part of the *standard PN* definition, *arc multiplicity*. In the presence of multiple arcs, the enabling and firing rules of Definition 2.1 must be revised. A transition is enabled iff there are at least as many tokens in each input place as the multiplicity of the corresponding input arc. Similarly, firing a transition removes and deposits as many tokens as the multiplicities of the input and output arcs, respectively. The set of input/output arcs is generalized to a multiset. The entries in matrices D^+ and D^- are non-negative integers. Multiple arcs are usually drawn as a single arc annotated with an expression denoting the multiplicity. The addition of multiple arcs does not change the modeling or decision power of the PN formalism.

Another frequently encountered extension is the *inhibitor arc*. Inhibitor arcs connect a place to a transition and are drawn with a small circle instead of the arrowhead. An inhibitor arc from place p to transition t disables t in any marking where p is not empty. Multiple inhibitor arcs can also be defined, in which case t is disabled whenever p contains at least as many tokens as the multiplicity of the inhibitor arc. Inhibitor arcs provide the ability to test whether a place is empty, which is not possible using standard PNs. This ability to perform *zero testing* extends the modeling power of PNs, and reduces their decision power, to equal that of Turing machines.

Inhibitor arcs are the most common “Turing extension”, but several others have been introduced. In theory, once any Turing extension has been adopted, any other extension is redundant. In practice, each of the

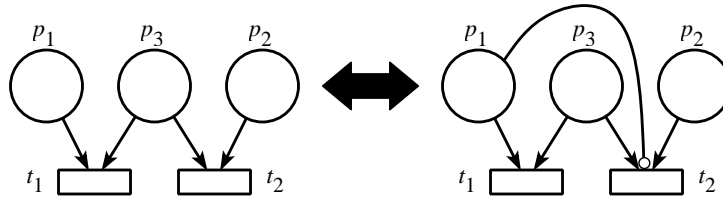


FIG. 2.3. *Equivalence of priorities and inhibitor arcs ($t_1 > t_2$).*

extensions described below is particularly well suited to representing some type of behavior.

Often, an activity must have precedence over another when they both require the same resource. Inhibitor arcs can be used to represent such constraints, but they may clutter the model. It is much more convenient to incorporate *transition priorities* directly into the formalism (see Figure 2.3). Traditionally, PN priorities have been defined by assigning an integer priority level to each transition, and adding the constraint that a transition may be enabled only if no higher priority transition is enabled. The extended PN definition that follows generalizes the notion of priorities by requiring only a partial order among transitions. This added flexibility provides a simple way to model the situation where $t_1 > t_2$, $t_3 > t_4$, but t_1 has no priority relation with respect to t_3 and t_4 .

We insist that transition priorities be defined by a static relation – the relative priority of any two transitions cannot depend upon the marking of the PN. This restriction allows more efficient analysis techniques, as well as a simpler, less error prone model specification. However, there are situations where a transition needs to be disabled depending upon *marking dependent* conditions. It is always possible to define a marking dependent constraint to prevent transition enabling using the primitives discussed so far, but the construction may be cumbersome. A better alternative is to define a marking dependent predicate as an additional enabling criterion for each transition. We associate a *guard* with each transition; the transition is enabled only if the guard is satisfied. Guards are also known as enabling functions or, if the sense of the logic is reversed, as inhibiting functions.

The convenience of priorities and guards comes at a price. Standard PNs can capture the entire system behavior graphically, but, when priorities and guards are used, this ability is partially lost. Our choice is to include priorities and guards in the PN definition, letting the modeler determine the proper balance between graphical expressiveness and conciseness.

Finally, we allow the multiplicity of an arc to depend on the marking of the PN. *Marking-dependent arc multiplicities* belong to the same class of extensions as inhibitor arcs, priorities and guards, but unlike the others, they provide a convenient way to describe activities where the number

of tokens needed to be transferred (or to enable the transition) depends upon the system state. A common use for marking dependent arcs is to allow a transition to flush all the tokens from a place with a single firing. Marking dependent arc multiplicities allow simpler and more compact PNs than would be otherwise possible in many situations. When exhaustive state space exploration techniques are employed, their use can dramatically reduce the state space. However, they preclude the structural analysis techniques described in Section 3.4.

Other extensions to PNs, which we do not include in our definition, are debit arcs and anti-tokens [65], colored tokens [35], and predicate-transition nets [56] (the last two fall in the category of *high-level nets*).

In the special case of PNs with finite reachability sets, all the Turing extensions provide only convenience and do not extend the class of systems that may be represented beyond that of finite state machines. This is the case of interest to us.

DEFINITION 2.2. An extended PN is a 8-tuple $\mathbf{A} = \{P, T, D^-, D^+, D^o, \epsilon, >, \mu_0\}$ where:

- P , T , and μ_0 are defined as in Definition 2.1.
- $\forall p \in P, \forall t \in T, D_{p,t}^- : \mathbb{N}^{|P|} \rightarrow \mathbb{N}, D_{p,t}^+ : \mathbb{N}^{|P|} \rightarrow \mathbb{N},$ and $D_{p,t}^o : \mathbb{N}^{|P|} \rightarrow \mathbb{N}$ are the marking-dependent multiplicities of the input arc from p to t , the output arc from t to p , and the inhibitor arc from p to t , respectively. If an arc multiplicity evaluates to zero in a marking, the arc is ignored (does not have any effect) in that marking.

We say that a transition $t \in T$ is arc-enabled in marking μ iff

$$\forall p \in P, D_{p,t}^-(\mu) \leq \#(p, \mu) \wedge (D_{p,t}^o(\mu) > \#(p, \mu) \vee D_{p,t}^o(\mu) = 0)$$

When transition t fires in marking μ_i , the new marking μ_j satisfies the following firing rule:

$$\forall p \in P, \#(p, \mu_j) = \#(p, \mu_i) - D_{p,t}^-(\mu_i) + D_{p,t}^+(\mu_i)$$

Note that both $D_{p,t}^-$ and $D_{p,t}^+$ are evaluated on μ_i *before* firing t .

- $\forall t \in T, g_t : \mathbb{N}^{|P|} \rightarrow \{true, false\}$ is the guard for transition t . If $g_t(\mu) = false$, t is disabled in μ .
- $>$ is a transitive and irreflexive relation imposing a priority among transitions. In a marking μ , t_1 is marking-enabled iff it is arc-enabled, $g_{t_1}(\mu) = true$, and no other transition t_2 exists such that $t_2 > t_1$, t_2 is arc-enabled, and $g_{t_2}(\mu) = true$.

2.3. Stochastic Petri nets. In the previous section, we adopted a particular PN formalism which can, in theory, describe any discrete-state behavior, being equivalent to Turing machines. By associating stochastic and timing information to it, we obtain the SPN formalism.

In general terms, we are interested in imposing a timing upon the flow of tokens. We do this by associating a random variable called *firing time* with transitions, representing the time that must elapse from the instant the transition is enabled to the instant it actually fires in isolation, that is, assuming that no other transition firing affects it. An alternative approach associates a waiting time with places [62], so that a token arriving into a place can enable a transition only after this waiting time is elapsed. The two approaches can model the same class of systems, but the latter has the disadvantage that the marking of the (untimed) PN does not represent the entire state of the model, since it is further necessary to distinguish whether each token in each place is “waiting” or “ready”.

Much more fundamental are the choices about the *execution policy*, the *memory policy*, and the probabilistic distributions available to specify the timing (we follow the terminology used in [2], but see also [7]).

The execution policy specifies which transition will fire among those enabled in a marking. We adopt the *race policy*, where the transition whose firing time elapses first is the one that will fire. An alternative is the *preselection policy*, where the next transition to fire in a given marking is chosen among the enabled transitions using a probability distribution independent of their firing times.

If the race policy is adopted, a memory policy must be specified to determine the effect of the change of marking on the (remaining) firing time of the transitions that did not fire. With the *resampling policy*, each change of marking causes the other enabled transitions to resample a new firing time from their distribution. With the *enabling memory policy*, transitions which remain enabled after the change of marking retain the firing time initially sampled, reduced by the time they have been enabled without interruption; transitions which become disabled instead “lose the work done” and must resample a new firing time once they become enabled again. Finally, with the *age policy*, the time spent by a transition while enabled is never lost, even when it becomes subsequently disabled; a new sample is drawn from the firing time distribution only after the previous sample elapsed, causing the transition to fire.

These policies correspond to different types of behavior. In the same SPN, different policies might be needed for different transitions or even for the same transition in different markings.

According to the type of distribution allowed, the SPN formalism may correspond to a wide range of well-known stochastic processes. For example, assuming an enabling memory policy:

- If firing times are geometrically or exponentially distributed, a discrete or continuous-time Markov chain (DTMC or CTMC) is obtained, respectively [46].
- If firing times can have general distributions, a generalized semi-Markov process (GSMP) is obtained [28].
- Under certain conditions restricting the type of distribution that

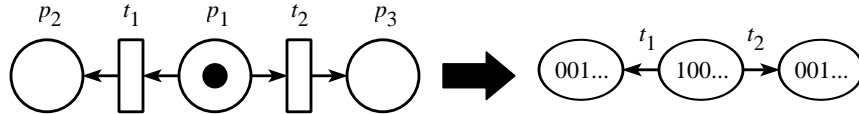


FIG. 2.4. Influence of timing over the structural behavior of the SPN.

concurrently enabled transitions can have, a semi-Markov process (SMP) is obtained [7,12].

Consider now a SPN with race policy where some or all of the transitions have a firing time distribution with discrete impulses, such as the constant distribution. In this case, the race policy is insufficient to specify the entire behavior of the SPN, since it does not specify which transition should be fired when several are enabled and their firing time elapses at the same instant (this event has probability zero if the distributions are continuous). The introduction of *firing probabilities*, usually specified as *weights* to be normalized, resolves these conflicts [3].

It is often desirable to ensure that the reachability graphs for the SPN and underlying PN models are equivalent. When this holds, all the PN analysis results of Section 3 can be applied without change. Unfortunately, timing may affect the logical behavior of the SPN as well. In this case, the reachability graph of the SPN has arcs or even nodes missing with respect to that of the underlying PN. For example, in Figure 2.4, transitions t_1 and t_2 are both enabled in the underlying PN. If their firing times are constants equal 1 and 2, respectively, t_2 cannot fire in that marking, since the token will be removed by t_1 , which has a shorter firing time.

If all the distributions involved are continuous with support $[0, \infty)$, the reachability graphs of the PN and the SPN are indeed equivalent. This is one of the reasons why the firing times are often restricted to have an exponential distribution. Even more important in this case, though, is the fact that the SPN can be automatically transformed into a CTMC [46].

In the generalized stochastic Petri nets (GSPNs) [3], the transitions are allowed to either have an exponential distribution (*timed transitions*) or fire in zero time (*immediate transitions*). This destroys the equivalence between the SPN and the PN reachability graphs, since the firing time of immediate transitions is always smaller than that of timed transitions. The equivalence of the reachability graphs is easily restored, for example, by assigning a higher priority to the immediate transitions. Also the GSPNs can be automatically transformed into CTMCs, provided that the underlying process is regular, that is, the number of transition firings in a finite interval of time is finite with probability one. The presence of an infinite firing sequence containing only immediate transitions could violate this assumption. For finite reachability sets, this can only happen if a *vanishing loop* exists [12]. This case is of little practical interest and we do not consider

it further.

Our definition of SPNs differs from that of GSPNs in several key aspects. From a structural point of view, both formalisms are equivalent to Turing machines. Our SPNs, though, provide guards, marking-dependent arc multiplicities, a more general approach to the specification of priorities, and the ability to decide in a marking-dependent fashion whether the firing time of a transition is exponentially distributed or zero, often resulting in more compact nets.

DEFINITION 2.3. A SPN is a 10-tuple $\mathbf{A} = \{P, T, D^-, D^+, D^0, e, >, \mu_0, \lambda, w\}$ where:

- $P, T, D^-, D^+, D^0, \mu_0, e$, and $>$, are defined as in Definition 2.2. They describe an extended PN.
- $\forall t \in T, \lambda_t : \mathbb{N}^{|P|} \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is the rate of the exponential distribution for the firing time of transition t . If $\lambda_t(\mu) = \infty$, the firing time of t in μ is zero. This is a generalization of the GSPN definition, where transitions are *a priori* classified as timed or immediate. The definition of vanishing and tangible marking, though, is still applicable: a marking μ is said to be *vanishing* if there is a marking-enabled transition t in μ such that $\lambda_t(\mu) = \infty$; μ is said to be *tangible* otherwise.

We indicate with \mathcal{T} and \mathcal{V} the tangible and vanishing portion of the reachability set, respectively: $\mathcal{T} \cup \mathcal{V} = \mathcal{S}$.

We additionally impose the interpretation that, in a vanishing marking μ , all transitions t with $\lambda_t(\mu) < \infty$ are implicitly inhibited. Hence, a transition t in a marking μ is enabled in the usual sense and can fire iff it is marking-enabled and either μ is tangible or $\lambda_t(\mu) = \infty$.

- $\forall t \in T, w_t : \mathbb{N}^{|P|} \rightarrow \mathbb{R}^+$ describes the weight assigned to the firing of enabled transition t , whenever its rate λ_t evaluates to ∞ . The probability of firing transition t enabled in a vanishing marking μ is:

$$\frac{w_t(\mu)}{\sum_{t_i: \mu \xrightarrow{t_i}} w_{t_i}(\mu)}$$

If a marking-dependent weight specification is not needed, the definition of w can be reduced to

$$\forall t \in T, w_t \in \mathbb{R}^+$$

Both λ_t and w_t are partial functions. λ_t need only be defined in markings where transition t is marking-enabled, while w_t need only be defined when t is enabled and λ_t is infinite.

According to our definition, a SPN identifies a trivariate discrete-time stochastic process: $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]})\}, n \in \mathbb{N}\}$. For $n = 0$, we use the con-

vention $t^{[0]} = NULL$ and $\theta^{[0]} = 0$, that is, the SPN is in marking μ_0 with probability one at time zero. For $n > 0$, $t^{[n]} \in T$ is the n -th transition to fire; its firing leads to $\mu^{[n]}$, the n -th marking encountered, and $\theta^{[n]}$ is the time at which it fires. Hence, $\mu^{[n-1]} \xrightarrow{t^{[n]}} \mu^{[n]}$ and $\theta^{[n+1]} - \theta^{[n]} \geq 0$ is the sojourn time for the corresponding visit in marking $\mu^{[n]}$. This process is a SMP where the sojourn times are either exponentially distributed, for tangible markings, or zero, for vanishing markings. It is also possible to define a continuous-time process describing the marking at time θ , $\{\mu(\theta), \theta \geq 0\}$, which is completely determined given $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbf{N}\}$:

$$\mu(\theta) = \mu_{\max\{n:\theta^{[n]} \leq \theta\}}$$

This process considers only the evolution with respect to the tangible markings, that is, $Pr\{\mu(\theta) \in \mathcal{V}\} = 0$. $\{\mu(\theta), \theta \geq 0\}$ is a CTMC [3].

Marking dependent transition firing rates [3] can succinctly describe a complex behavior. For example, traditional queueing network models describe the effect of competition for a shared resource in terms of the queueing and service disciplines [5]. Common queueing disciplines include first-come-first-served (FCFS), processor sharing (PS), and last-come-first-served-preemptive-resume (LCFSPR). Common service disciplines include single (SS), multiple (MS), and infinite (IS) number of servers. In the SPN formalism, tokens (customers) are indistinguishable (single class), and the firing (service) times are exponentially distributed, hence the queueing discipline is not an issue. Any service discipline, though, can be represented in a SPN, using the appropriate firing rates.

Consider the SPN in Figure 2.5, where transition t and place p represent a service center and its associated queue, respectively. The IS discipline is obtained by defining the firing rate of t be defined as a constant $x \in \mathbb{R}^+$ times the number of tokens in the input place:

$$\lambda_t(\mu) = \#(p, \mu)x$$

The SS discipline corresponds to a constant firing rate:

$$\lambda_t(\mu) = x$$

A MS discipline with K servers is obtained by defining

$$\lambda_t(\mu) = \min\{\#(p, \mu), K\}x$$

In some models, several transitions may share a single resource. For example, transitions may represent the internal behavior of software processes. Logically, the processes may be independent, yet they do affect one another by competing for a shared resource such as the physical computer system. The firing rates for the competing transitions can be modified to take this into account. Consider Figure 2.6, where t_1 and t_2 share a resource (this is not apparent from the PN graph). In the first marking,

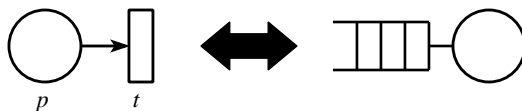


FIG. 2.5. Representing a single queue with a SPN.

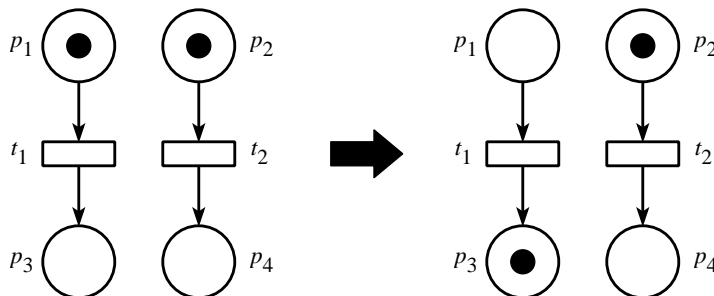


FIG. 2.6. Representing shared resources with SPNs.

both t_1 and t_2 are enabled and competing for the shared resource, so their rates should be half of what they would be if they each had exclusive use of the resource. If t_1 fires, t_2 has exclusive use of the resource, so its firing rate is no longer degraded. Assuming that each token in p_1 and p_2 attempts to use the resource, the firing rates could be defined as

$$\lambda_{t_1}(\mu) = \frac{x_1 \#(p_1, \mu)}{\#(p_1, \mu) + \#(p_2, \mu)}$$

$$\lambda_{t_2}(\mu) = \frac{x_2 \#(p_2, \mu)}{\#(p_1, \mu) + \#(p_2, \mu)}$$

where x_1 and x_2 are the rates at which tokens would be processed by t_1 and t_2 in isolation, respectively.

Several extensions are possible. There could be s transitions t_1, \dots, t_s sharing a marking-dependent number $K(\mu)$ of identical resources (this could be used to model failure and repair of servers). The number of requests issued to the resource by each transition t_i could be a more complex function than just the number of tokens in its input place, $f_i(\mu)$. The overhead of sharing resources could be non-negligible, requiring a marking-dependent function $0 < o(\mu) \leq 1$ to describe it. The firing rates could then be defined as:

$$\lambda_{t_i}(\mu) = x_i f_i(\mu) \frac{\min \left\{ K(\mu), \sum_{i=1}^s f_i(\mu) \right\}}{\sum_{i=1}^s f_i(\mu)} o(\mu)$$

Finally, if the resource must be made unavailable in certain states, even if there are customers ready to use it, a guard can be associated with the appropriate transitions.

2.4. Proposed Formalism. The SPN formalism underlying our SRN definition can model the same class of systems as the GSPN formalism: the power of a Turing machine is available to describe the structural behavior, while exponential or constant zero distributions are available to describe the timing of activities. The SPNs in Definition 2.3, though, are much more flexible because they have a richer set of constructs which often result in a more compact model for a given problem.

The treatment of the measures in the SRN is even more important: the SRN formalism considers the measure specification as an integral part of the model. Underlying an SRN is an independent semi-Markov reward process with reward rates associated to the markings and reward impulses associated to the transitions between markings. Our definition of SRN explicitly includes parameters (inputs) and the specification of multiple measures (outputs). A SRN with m inputs and n outputs defines a function from \mathbb{R}^m to \mathbb{R}^n .

DEFINITION 2.4. A (non-parametric) SRN is an 11-tuple $\mathbf{A} = \{P, T, D^-, D^+, D^0, e, >, \mu_0, \lambda, w, M\}$ where:

- $P, T, D^-, D^+, D^0, \mu_0, e, >, \lambda,$ and w are defined as in Definition 2.3. They describe a SPN.
- $M = \{(\rho_1, r_1, \psi_1), \dots, (\rho_{|M|}, r_{|M|}, \psi_{|M|})\}$ is a finite set of measures, each specifying the computation of a single real value. A measure $(\rho, r, \psi) \in M$ has three components. The first and second components specify a reward structure over the underlying stochastic process $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$.

$$\rho : \mathbb{N}^{|P|} \rightarrow \mathbb{R}$$

is a reward rate: $\forall \mu \in \mathcal{S}, \rho(\mu)$ is the rate at which reward is accumulated when the marking is μ .

$$\forall t \in T, r_t : \mathbb{N}^{|P|} \rightarrow \mathbb{R}$$

is a reward impulse: $\forall \mu \in \mathcal{S}, r_t(\mu)$ is the instantaneous reward gained when firing transition t in marking μ . Often, a marking-dependent reward impulse specification is not needed, so the definition of r can be reduced to

$$\forall t \in T, r_t \in \mathbb{R}$$

The reward structure specified by ρ and r over $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$ completely determines the sequence $\{Y^{[n]}, n \in \mathbb{N}\}$, where

$$Y^{[n]} = \int_0^{\theta^{[n]}} \rho(\mu(u)) du + \sum_{i=1}^n r_{t^{[i]}}(\mu^{[i-1]})$$

represents the reward accumulated until time $\theta^{[n]}$ and $Y^{[0]} = 0$.

We can also define a continuous process $\{Y(\theta), \theta \geq 0\}$, describing the reward accumulated by the SRN up to an arbitrary time θ :

$$Y(\theta) = \int_0^\theta \rho(\mu(u))du + \sum_{n=1}^{\max\{n:\theta^{[n]}\leq\theta\}} r_{t^{[n]}}(\mu^{[n-1]})$$

The third component of a measure specification, ψ , is a function that computes a single real value from the stochastic process $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}, Y^{[n]}), n \in \mathbb{N}\}$ or, more simply, from $\{Y(\theta), \theta \geq 0\}$.

The generality of this definition is best illustrated by showing the wide range of measures the triplet (ρ, r, ψ) can capture (in a given SRN, some of these measures might be infinite):

- Expected number of transition firings up to time θ : this is simply $E[Y(\theta)]$ when all reward rates are zero and all reward impulses are one.
- Expected time-averaged reward up to time θ :

$$E\left[\frac{Y(\theta)}{\theta}\right]$$

- Expected instantaneous reward rate at time θ :

$$\lim_{\delta \rightarrow 0} E\left[\frac{Y(\theta + \delta) - Y(\theta)}{\delta}\right]$$

- Expected accumulated reward rate up to steady state:

$$E[Y] = \lim_{\theta \rightarrow \infty} E[Y(\theta)]$$

- Mean time to absorption: this is a particular case of the previous measure, obtained by setting the reward rate of transient and absorbing markings to one and zero, respectively, and all reward impulses to zero.
- Expected instantaneous reward rate in steady state:

$$\lim_{\theta \rightarrow \infty} \lim_{\delta \rightarrow 0} E\left[\frac{Y(\theta + \delta) - Y(\theta)}{\delta}\right]$$

which can also be expressed as the expected time-average reward up to steady state:

$$\lim_{\theta \rightarrow \infty} E\left[\frac{Y(\theta)}{\theta}\right]$$

- Supremum reward rate in any reachable marking:

$$\sup_{n \geq 0} \left\{ \rho(\mu) : Pr(\mu^{[n]} = \mu) > 0 \right\}$$

A parametric SRN is obtained allowing each component of an SRN to depend on a set of parameters $\beta = (\beta_1, \dots, \beta_m) \in \mathbb{R}^m$:

$$\mathbf{A}(\beta) = \{P(\beta), T(\beta), D^-(\beta), D^+(\beta), D^o(\beta), e(\beta), >(\beta), \mu_0(\beta), \lambda(\beta), w(\beta), M(\beta)\}$$

Once the parameters β are fixed, a simple (non-parametric) SRN is obtained.

A fundamental capability captured by our parametrization is that of specifying the initial marking not as a single marking, but as a probability vector defined over a set of markings. This is often required in transient analysis, if the initial state of the system is uncertain. If $\mu_0 = \mu_i$, $1 \leq i \leq K$ with probability $\gamma_i(0)$, the following construction will suffice:

1. Add a place p_0 .
2. Add transitions t_0^i , $1 \leq i \leq K$, with rate $\lambda_{t_0^i} = \infty$ and weight $w_{t_0^i} = \gamma_i(0)$.
3. For each new transition t_0^i , set $D_{p_0, t_0^i}^- = 1$.
4. For each new transition t_0^i and for each original place p_j , set $D_{p_j, t_0^i}^+ = \#(p_j, \mu_i)$.
5. Set μ_0 to have one token in p_0 , no tokens elsewhere.

The modified SRN will initiate its activity by removing the token from p_0 and adding the appropriate tokens to create one of the possible initial markings, with the corresponding probability.

While this is not a practical method if the number of possible initial markings K is large, it shows that the parametrization of the initial marking probability falls into our framework.

In practice, more efficient *ad hoc* approaches for the specification of the initial probability vector might be employed by solution packages. For example, SPNP [17] uses the following approach:

1. Build the reachability set starting from a single initial marking μ_0 specified by the user.
2. Evaluate a user-specified marking-dependent non-negative expression f in each marking μ of the reachability set.
3. Assign initial probability $\gamma_i(0) = f(\mu_i) / \sum_{j \in \mathcal{S}} f(\mu_j)$ to marking μ_i .

3. Logical analysis. A PN, a marking, or a reachability graph may exhibit several logical properties. It is important to determine whether these properties hold since they correspond to properties in the system being modeled and they may also affect the choice of analysis technique.

A place p is *k-bounded* iff the place contains at most k tokens in any reachable marking:

$$\forall \mu \in \mathcal{S}, \#(p, \mu) \leq k$$

In particular, a place is *safe* iff it is 1-bounded. A PN is *k-bounded (safe)* iff each of its places is *k-bounded (safe)*.

A PN is *conservative* iff the total number of tokens remains constant in each reachable marking. A PN is *weighted conservative* iff there exists a set of positive integer weights such that the weighted sum of the number of tokens in each place remains constant in every reachable marking:

$$\exists c_1, \dots, c_{|P|} \in \mathbb{N}^+, \forall \mu_i, \mu_j \in \mathcal{S}, \sum_{p \in P} c_p \#(p, \mu_i) = \sum_{p \in P} c_p \#(p, \mu_j)$$

A PN which exhibits any of the above properties (boundedness or conservation) has a finite reachability set.

A transition t is said to be *live* iff for each reachable marking, there exists some firing sequence that leads to a marking where the transition is enabled:

$$\forall \mu_i \in \mathcal{S}, \exists \mu_j : \mu_i \xrightarrow{*} \mu_j \wedge \mu_j \xrightarrow{t}$$

Actually, there are several degrees of liveness, the above definition being the most strict [51]. A PN is *live* iff each of its transitions is live. Liveness is an important property when studying potential deadlocks in a system.

A PN is *reversible* iff the initial marking can be reached from every reachable marking:

$$\forall \mu_i \in \mathcal{S}, \mu_i \xrightarrow{*} \mu_0$$

This definition may be relaxed to require only that there exist some marking μ_j that is reachable from every reachable marking [50]:

$$\exists \mu_j : \forall \mu_i \in \mathcal{S}, \mu_i \xrightarrow{*} \mu_j$$

The definition of reversibility commonly applied to PNs is unrelated to the concept of time-reversibility used to describe a class of Markov chains that admit a recursive solution [38].

Two transitions t_i and t_j are *mutually exclusive* iff they are never simultaneously enabled

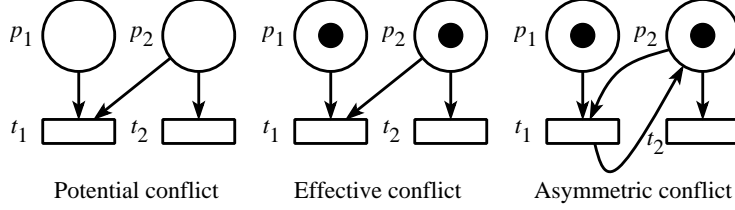
$$\neg(\exists \mu \in \mathcal{S} : \mu \xrightarrow{t_i} \wedge \mu \xrightarrow{t_j})$$

A PN is *pure* if it does not contain a transition whose firing has no effect on some marking

$$\neg(\exists t \in T, \exists \mu \in \mathcal{S} : \mu \xrightarrow{t} \mu)$$

A PN is *simple* if it does not contain two distinct transitions t_i and t_j whose firing cause the same effect in some marking¹

¹ For standard PNs with constant multiplicity arcs, simplicity is used to ensure that no redundant transitions or places exist. In our definition, instead, we are only concerned with redundant transitions.

FIG. 3.1. *Different types of conflict.*

$$\neg(\exists t_i, t_j \in T, t_i \neq t_j, \exists \mu_1, \mu_2 \in \mathcal{S} : \mu_1 \xrightarrow{t_i} \mu_2 \wedge \mu_1 \xrightarrow{t_j} \mu_2)$$

Two transitions t_1 and t_2 are in *potential conflict* iff they share some input place: $\cdot t_1 \cap \cdot t_2 \neq \emptyset$. Potential conflict depends only on the structure of the PN graph. Two transitions t_1 and t_2 are instead in *effective conflict* in some marking μ iff they are both enabled in μ and firing one disables the other. Conflict is not necessarily symmetric. In the last PN in Figure 3.1, transition t_2 is in conflict with t_1 , but t_1 is not in conflict with t_2 .

Two transitions t_1 and t_2 are *concurrent* in some marking μ iff they are both enabled in μ and neither is in effective conflict with the other.

$$\mu \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \mu_2 \wedge \mu \xrightarrow{t_2} \mu_3 \xrightarrow{t_1} \mu_4$$

If two transitions are concurrent and their input and output arcs have constant multiplicity, firing them in either order leads to the same marking. Let t_1 and t_2 be concurrent in marking μ as above, then

$$\mu \xrightarrow{t_1, t_2} \mu_3 \wedge \mu \xrightarrow{t_2, t_1} \mu_3$$

Combining concurrency and conflict may lead to a situation known as *confusion*. Confusion occurs when the manner and order in which separate conflicts are resolved affects the final outcome. In the presence of confusion, conflicts that involve distinct sets of transitions or even that occur at different times are not independent. For example, in Figure 3.2 there are two conflicts to resolve: the conflict between t_1 and t_2 and the conflict between t_3 and t_4 . Assume that t_2 is chosen over t_1 and that t_3 is chosen over t_4 . If t_3 fires first then t_5 becomes enabled and may fire removing the conflict between t_1 and t_2 altogether. If, on the other hand, t_2 fires before t_3 then the conflict between t_3 and t_4 is unaffected, and t_3 may fire. In this case, the final marking depends not only on how the conflicts are resolved, but in which order.

A more precise definition of confusion requires the notion of a *conflict set*. The conflict set of transition t enabled in marking μ is the maximal

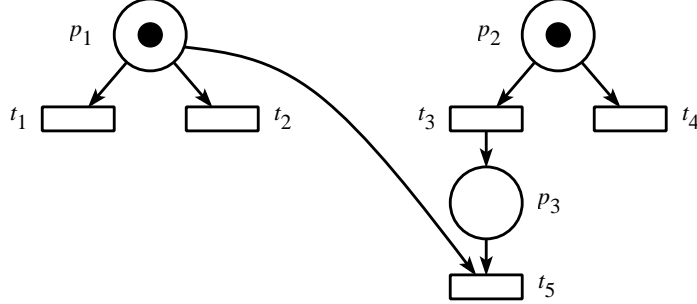


FIG. 3.2. A confused PN.

set of transitions which are in conflict with t .

$$cfl(t, \mu) = \{t_i \in T : \exists \mu_j \in S, \mu \xrightarrow{t_i} \mu_j \wedge \neg(\mu_j \xrightarrow{t})\}$$

Let t_1 and t_2 be simultaneously enabled in μ , then (μ, t_1, t_2) is a confusion iff $cfl(t_1, \mu) \neq cfl(t_2, \mu)$ where $\mu \xrightarrow{t_1} \mu_j$ and $t_2 \notin cfl(t_1, \mu)$. A PN is *confused* in marking μ iff there is a confusion in μ [57]. That is, a PN is confused when firing a transition changes the conflict set of a concurrent transition.

Confusion may accurately represent the behavior of the system being modeled or be artificially introduced by the interleaving semantics required for PN transition firings. Confusion becomes particularly critical to detect among the immediate transitions in a GSPN or SRN. It is disconcerting for model results to depend upon the interleaving of asynchronous *instantaneous* events that are not in conflict. In [4], a test for finding potential confusion among immediate transitions is presented.

3.1. Marking classification. The standard state classification schemes for Markov chains apply to SRN markings as well [69]. A marking is *recurrent* iff, upon exit from that marking, it will be reentered with probability one. Otherwise a marking is *transient*. We may further classify recurrent markings as *positive recurrent* and *null recurrent*, according to whether the mean time to return to the marking is finite or infinite, respectively.

We only consider the case of finite reachability sets, where no null recurrent markings exist and the determination whether a state is transient or recurrent depends solely upon the structure of the reachability graph, and not upon the transition firing rates:

$$\mu_i \text{ is transient} \Leftrightarrow \exists \mu_j \in \mathcal{S} : \mu_i \xrightarrow{*} \mu_j \wedge \neg(\mu_j \xrightarrow{*} \mu_i)$$

Every SRN with a finite reachability set will eventually enter a marking from where no transient state is reachable.

An SRN will instead return to a recurrent marking an arbitrary number of times after it has reached it at least once. However, a recurrent marking may not be reached at all in a particular evolution of the SRN, because there may be multiple sets of states that act as traps, such that once the SRN enters one state in the set, it may thereafter only enter states belonging to that same set. A set of mutually reachable recurrent states is called a *recurrent class*.

In general, a reachability graph contains a possibly empty set of transient markings and one or more sets of recurrent markings. If there is more than one recurrent class, then the initial marking must be transient, since there is at least one firing sequence that leads from the initial marking, possibly through some intermediate transient markings, to each recurrent class. Obviously, the recurrent markings are of special interest when studying the steady state behavior of the SRN, but transient markings can affect the probability of which recurrent class the SRN eventually enters. When there is a single recurrent class, transient markings are irrelevant to the study of the steady state behavior of the SRN.

For finite reachability graphs, the first step in determining the recurrent classes is to identify the strongly connected components of the reachability graph [1]. Each strongly connected component which cannot be exited constitutes a recurrent class. In particular, a single marking enabling no transition, or an *absorbing marking*, is a recurrent class by itself.

Finally, when discussing SRN markings, it is important to note whether they are vanishing or tangible as defined in Section 2.3. This depends solely upon the rates of the enabled transitions in that marking and is completely independent of any structural properties of the reachability graph. In particular a marking may be either tangible or vanishing and also be either transient or recurrent. Vanishing or tangible refers to the time an SRN remains in a marking per visit. Recurrent or transient refers to whether the SRN returns to a marking a finite or an infinite number of times.

3.2. Classes of PNs. It is natural to consider the effect of restricting or extending the definition of PNs upon the class of systems that may be modeled and the feasibility or efficiency of various types of analysis (see Figure 3.3).

A *marked graph* is a PN where every place has one input arc and one output arc. When drawing a marked graph, it is customary to draw only the transitions and to substitute each place and its input and output arcs with a single arc, so that the place is only implicitly represented and tokens reside on the arcs of the graph. Transitions in a marked graph are typically shown as circles.

Marked graphs can represent synchronization and concurrency but not conflict. In return for the restricted modeling power, marked graphs can be analyzed more efficiently than standard PNs. As a consequence of the definition of a marked graph, the steady-state throughput is the same for all

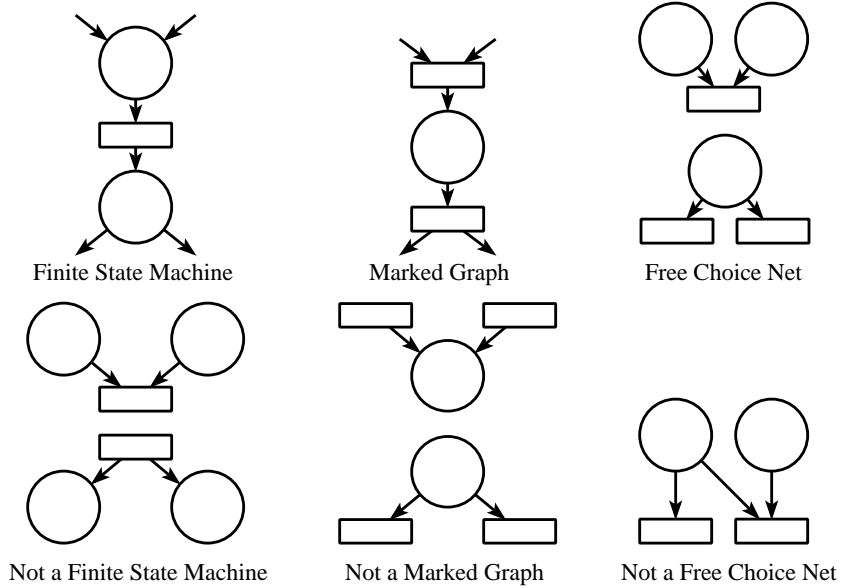


FIG. 3.3. *Classes of PNs.*

transitions (defined as the expected number of times a transition fires per unit time), independent of the firing time distributions. Thus it is possible to speak of the throughput of a marked graph. Though restrictive compared to standard PNs, marked graphs are an extension of PERT networks and task precedence graphs, which are simply safe acyclic marked graphs.

A finite state machine (FSM) is a PN where every transition has exactly one input arc and exactly one output arc. Similarly to marked graphs, it is customary to draw only the places since there is an implicit transition along every arc.

A PN is said to be *free choice* iff each transition that has more than one input place is the only output transition for each of its input places. The free choice restriction implies that each conflict may be resolved in isolation. Free choice PNs extend the class of systems that can be represented by marked graphs. They allow a limited type of conflict that can be resolved based solely on local information. Free choice PNs cannot represent confusion. Balbo *et al.* generalized the definition of free choice based on the concept of extended conflict sets [4].

Standard PNs can represent many forms of conflict, including those that are disallowed in free choice PNs, but cannot represent zero testing. As discussed in Section 2.2, adding inhibitor arcs increases the modeling power of PNs to equal that of Turing machines.

```

1)  $\mathcal{S} = \{\mu_0\}$ 
2)  $\mathcal{A} = \emptyset$ 
3)  $\mathcal{S}^{new} = \{\mu_0\}$ 
4) while  $\mathcal{S}^{new} \neq \emptyset$  do
5)   choose a marking  $\mu$  from  $\mathcal{S}^{new}$ 
6)    $\mathcal{S}^{new} = \mathcal{S}^{new} \setminus \{\mu\}$ 
7)   foreach  $t \in T$  do
8)     if  $t$  enabled in  $\mu$ 
9)        $\mu^{new} = \mu + D_{\bullet,t}^+(\mu) - D_{\bullet,t}^-(\mu)$ 
10)      if  $\mu^{new} \notin \mathcal{S}$  then
11)         $\mathcal{S}^{new} = \mathcal{S}^{new} \cup \{\mu^{new}\}$ 
12)         $\mathcal{S} = \mathcal{S} \cup \{\mu^{new}\}$ 
13)         $\mathcal{A} = \mathcal{A} \cup \{(\mu \xrightarrow{t} \mu^{new})\}$ 

```

FIG. 3.4. Algorithm to generate the reachability graph $(\mathcal{S}, \mathcal{A})$.

3.3. Reachability graph analysis. A standard technique for studying PNs is to construct and explore the reachability graph. Obviously, this approach requires the reachability graph to be finite.² It is possible to simultaneously generate the reachability set \mathcal{S} and the reachability graph $(\mathcal{S}, \mathcal{A})$ using a simple generate-and-test algorithm. This is accomplished by visiting each reachable marking, starting from the initial marking. The algorithm is outlined in Figure 3.4.

If the SRN is not simple, \mathcal{A} contains multiple arcs from μ_i to μ_j , labeled with different transitions. Furthermore, if the SRN is not pure, self-arcs $\mu_i \xrightarrow{t} \mu_i$ are also present.

When the reachability set is finite, all questions about a PN are decidable and can be answered by exploring the reachability graph. Unfortunately, the reachability set is extremely large in most practical problems, so tractability, rather than decidability, is often the concern when using reachability graph analysis.

When the size of the reachability graph prohibits exhaustive analysis, a subset of the reachability graph may be explored using simulation. Most properties cannot be determined with certainty using simulation (such as the absence of reachable markings satisfying a given condition), yet it is possible to gain some insight into the behavior of the system and to obtain timing information, using Monte Carlo simulation. Alternatively, it may be possible to avoid generating the reachability graph altogether, using the techniques illustrated in the next section.

² When the reachability graph is infinite, it is possible to construct a finite representation of it known as the *reachability tree*, [51,56]. Inspection of the reachability tree can answer only a subset of the questions that can be answered by inspecting the reachability graph.

3.4. Structural analysis. This section presents results applicable only to PNs with constant arc multiplicities. It is important to distinguish whether a property refers to the set of all reachable markings, as with mutual exclusion, or to a particular marking, as with conflict.

In some cases, it is possible to determine that a property is preserved in all reachable markings solely from information present in the PN graph. For example, in the standard PN formalism, two transitions that do not share any input places can never be in effective conflict. Such transitions are said to be *structurally* conflict-free. In [4], structural tests for logical properties such as conflict, mutual exclusion, and confusion are presented. Such tests are weaker than those based upon exploration of all reachable markings, but they are much more efficient.

For example, transitions t_1 and t_2 in the first PN in Figure 3.1 share an input place p_2 , so they are in potential conflict. If the initial marking is such that p_1 and p_2 cannot both have a token at the same time, then the two transitions are never in effective conflict.

In contrast to reachability graph analysis, structural analysis techniques consider only the structure of the PN graph and examine the set of reachable markings only implicitly. Structural analysis is normally quite efficient, requiring a number of operations that is polynomial in the number of places and transitions.

Invariant analysis considers only the PN incidence matrix $(D^+ - D^-)$ which describes the “net marking change” when firing each transition, but not the conditions required for a transition to be enabled.

An s -invariant is a non-negative integer solution to the equation

$$(3.1) \quad x(D^+ - D^-) = 0$$

The weighted sum $\sum_{p \in P} x_p \#(p)$ has the same value in any reachable marking. The s -invariant x *covers* a place if $x_p > 0$.

A t -invariant is a solution to the equation

$$(3.2) \quad (D^+ - D^-)y^T = 0$$

(where the superscript T indicates transposition). If, $\forall t \in T$, a firing sequence s contains y_t occurrences of transition t , then s leads any marking back to itself. The existence of a t -invariant does not imply that any such *legal* firing sequence can actually occur.

If a and b are two s -invariants and $\alpha \in \mathbb{N}^+$, then so are αa and $a + b$. The analogous property holds for t -invariants. Hence, we are normally interested in a *minimal set of invariants*, from which all other invariants are obtained as linear combinations.

Invariant analysis has limitations. The set of invariants is independent of the initial marking, while the behavior of the PN can be substantially affected by it. Also, the incidence matrix does not capture the situation where a place p is both an input and an output for a transition t , since the

entries $D_{p,t}^-$ and $D_{p,t}^+$ cancel each other in this case. This “control place”, though, could have an important effect on the PN behavior. Nor can invariant analysis make use of the presence of inhibitor arcs, transition priorities, or guards, as they do not effect the incidence matrix. The invariants given by solutions to Equations 3.1 and 3.2 are still valid in these cases, but there are other conditions which might hold in the PN and are not captured by the invariants.

Marking invariants simply use the initial marking to fix the constant implicit in each s -invariant. While an s -invariant states that the weighted sum of the tokens in some set of places remains constant, the corresponding marking invariant determines the value of this constant from the initial marking.

Invariants may be used for several purposes. For example, if there exists an s -invariant that covers every place in a PN, then the PN is structurally bounded, it has a finite reachability graph for any initial marking. Consider a GSPN where every transition is defined *a priori* to be either timed or immediate for all markings, as in [3]. If no t -invariant contains only immediate transitions, then the state space does not contain any vanishing loops, regardless of the initial marking. This property may be exploited to warn the user of potential vanishing loops or to use a more general (but less efficient) solution method only when needed.

4. Temporal analysis. This section shows how to perform the temporal analysis of a SRN, starting from its reachability graph $(\mathcal{S}, \mathcal{A})$. The steps we provide are those used by automatic tools such as SPNP [17]. For the type of numerical solution we discuss, we assume that the measures of interest are of the form given by the examples in Section 2.4. These measures are computed by solving the SMP $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$ first, to compute the following quantities (see also Table 4.1):

- The expected amount of time spent in each tangible marking i during the interval $[0, \theta]$ or up to steady state:

$$\sigma(\theta) = [\sigma_i(\theta)] = \left[\int_0^\theta Pr\{\mu(u) = i\} du \right]$$

$$\sigma = [\sigma_i] = \left[\lim_{\theta \rightarrow \infty} \int_0^\theta Pr\{\mu(u) = i\} du \right]$$

σ_i is of interest only if i is a transient marking, since $\sigma_i = \infty$ if i is recurrent. σ is often studied in connection with SRNs having absorbing markings, hence it is usually indicated as the expected time spent in each marking “up to absorption”.

- The probability of being in each tangible marking i at a given time θ or in steady state:

$$\pi(\theta) = [\pi_i(\theta)] = [Pr\{\mu(\theta) = i\}]$$

	Cumulative		Instantaneous	
	Transient	Steady state	Transient	Steady state
Marking i	$\sigma_i(\theta)$	σ_i	$\pi_i(\theta)$	π_i
Transition $i \rightarrow j$	$N_{i,j}(\theta)$	$N_{i,j}$	$\Phi_{i,j}(\theta)$	$\Phi_{i,j}$

TABLE 4.1

Eight quantities obtainable from the study of $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$.

$$\pi = [\pi_i] = \left[\lim_{\theta \rightarrow \infty} Pr\{\mu(\theta) = i\} \right]$$

- The expected number of marking-to-marking transitions from i to j , indicated with $i \rightarrow j$, up to time θ or up to steady state:

$$N(\theta) = [N_{i,j}(\theta)] = \left[E \left[\sum_{n=1}^{\max\{n:\theta^{[n]} \leq \theta\}} Pr\{\mu^{[n]} = i \wedge i \xrightarrow{t^{[n]}} j\} \right] \right]$$

$$N = [N_{i,j}] = \left[E \left[\sum_{n \geq 1} Pr\{\mu^{[n]} = i \wedge i \xrightarrow{t^{[n]}} j\} \right] \right]$$

$N_{i,j}$ is of interest only if i is transient.

- The expected frequency of marking-to-marking transitions $i \rightarrow j$ at time θ or in steady state:

$$\Phi(\theta) = [\Phi_{i,j}(\theta)] = \left[\lim_{\delta \rightarrow 0} E \left[\frac{N_{i,j}(\theta + \delta) - N_{i,j}(\theta)}{\delta} \right] \right]$$

$$\Phi = [\Phi_{i,j}] = \left[\lim_{\theta \rightarrow \infty} \lim_{\delta \rightarrow 0} E \left[\frac{N_{i,j}(\theta + \delta) - N_{i,j}(\theta)}{\delta} \right] \right]$$

The SRN measures are then obtained as a linear combination of the above quantities using the specified reward rates or impulses as the multiplying factors. $\sigma(\theta)$, σ , $\pi(\theta)$, and π are related to sojourns in the SRN markings, so they are multiplied by the corresponding reward rates: the multiplying factor associated to marking i is $\rho(i)$. $N(\theta)$, N , $\Phi(\theta)$, and Φ are instead related to transitions between markings of the SRN, so they are multiplied by the corresponding reward impulses: the multiplying factor associated to $i \rightarrow j$ is $r_t(i)$ if t is the only transition such that $i \xrightarrow{t} j$. In general, though, the SRN might not be simple, so i and j might not uniquely identify t . We must then define the reward impulse associated to the marking-to-marking

transition $i \rightarrow j$, based on the reward impulses associated to transition firings:

$$r_{i,j} = \sum_{t \in T: i \xrightarrow{t} j} r_t(i) Pr\{t^{[n]} = t | \mu^{[n-1]} = i \wedge \mu^{[n]} = j\} = \begin{cases} \frac{\sum_{t \in T: i \xrightarrow{t} j} r_t(i) \lambda_t(i)}{\sum_{t \in T: i \xrightarrow{t}} \lambda_t(i)} & \text{if } i \in \mathcal{T} \\ \frac{\sum_{t \in T: i \xrightarrow{t} j} r_t(i) w_t(i)}{\sum_{t \in T: i \xrightarrow{t}} w_t(i)} & \text{if } i \in \mathcal{V} \end{cases}$$

The expected instantaneous reward at time θ is then obtained as

$$\lim_{\delta \rightarrow 0} E \left[\frac{Y(\theta + \delta) - Y(\theta)}{\delta} \right] = \sum_{i \in \mathcal{S}} \rho(i) \pi_i(\theta) + \sum_{i,j \in \mathcal{S}} r_{i,j} \Phi_{i,j}(\theta)$$

and the expected reward up to time θ as

$$E[Y(\theta)] = \sum_{i \in \mathcal{S}} \rho(i) \sigma_i(\theta) + \sum_{i,j \in \mathcal{S}} r_{i,j} N_{i,j}(\theta)$$

Time-averaged measures are obtained by dividing $E[Y(\theta)]$ by θ . The analogous steady state measures are obtained by using the steady state quantities π , σ , N , and Φ instead of their transient counterparts.

The reachability graph $(\mathcal{S}, \mathcal{A})$ implicitly describes $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$ but, in practice, the following quantities are used.

- $h = [h_i] = [E[\theta^{[n+1]} - \theta^{[n]} | \mu^{[n]} = i]]$, a vector describing the expected holding time in each marking:

$$h_i = \begin{cases} \left(\sum_{t \in T: i \xrightarrow{t}} \lambda_t(i) \right)^{-1} & \text{if } i \in \mathcal{T} \\ 0 & \text{if } i \in \mathcal{V} \end{cases}$$

- $\Pi = [\Pi_{i,j}] = [Pr\{\mu^{[n+1]} = j | \mu^{[n]} = i\}]$, a matrix describing the probability of marking-to-marking transitions. Π is stochastic, cor-

responding to the embedded DTMC process $\{\mu^{[n]}, n \in \mathbb{N}\}$.

$$\Pi_{i,j} = \begin{cases} \sum_{t \in \mathcal{T}: i \xrightarrow{t} j} \lambda_t(i) h_i & \text{if } i \in \mathcal{T} \\ \sum_{t \in \mathcal{T}: i \xrightarrow{t} j} w_t(i) \left(\sum_{t \in \mathcal{T}: i \xrightarrow{t}} w_t(i) \right)^{-1} & \text{if } i \in \mathcal{V} \end{cases}$$

- $\gamma(0) = [\gamma_i(0)] = [Pr\{\mu_0 = i\}]$, a vector describing the probability of being initially in each marking. The embedded DTMC process $\{\mu^{[n]}, n \in \mathbb{N}\}$ has the same initial probability vector. As discussed in Section 2.4, the initial distribution could be captured by the SRN, but it is more efficient and common to assume that $\gamma(0)$ is explicitly provided as input.

4.1. Preservation or elimination of the vanishing markings.

So far, we have considered the entire state space \mathcal{S} of the SMP. In practice, the reward rates associated to the vanishing markings do not contribute in any way to the value of a measure, since $\sigma(\theta)$, σ_i , $\pi(\theta)$, and π_i are zero if i is a vanishing marking. A common approach to the solution of the SMP requires the elimination of the vanishing markings. If h , Π , and $\gamma(0)$ are partitioned according to the type of marking (vanishing or tangible) into:

$$h = [0 \mid h_{\mathcal{T}}] \quad \Pi = \left[\begin{array}{c|c} \Pi_{\mathcal{V},\mathcal{V}} & \Pi_{\mathcal{V},\mathcal{T}} \\ \hline \Pi_{\mathcal{T},\mathcal{V}} & \Pi_{\mathcal{T},\mathcal{T}} \end{array} \right] \quad \gamma(0) = [\gamma_{\mathcal{V}}(0) \mid \gamma_{\mathcal{T}}(0)]$$

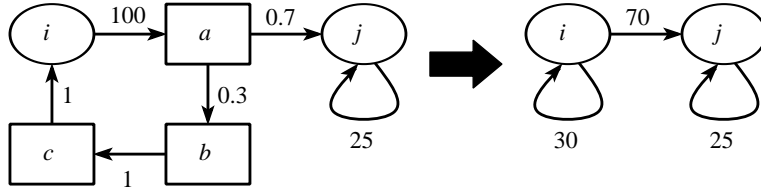
it is possible to define a new process that considers only the tangible-marking-to-tangible-marking transitions. This new process is described by $h_{\mathcal{T}}$, Π^* , and $\pi(0)$, given by:

$$\Pi^* = \Pi_{\mathcal{T},\mathcal{T}} + \Pi_{\mathcal{T},\mathcal{V}} (I - \Pi_{\mathcal{V},\mathcal{V}})^{-1} \Pi_{\mathcal{V},\mathcal{T}}$$

$$\pi(0) = \gamma_{\mathcal{T}}(0) + \gamma_{\mathcal{V}}(0) (I - \Pi_{\mathcal{V},\mathcal{V}})^{-1} \Pi_{\mathcal{V},\mathcal{T}}$$

Π^* might have nonzero diagonal entries either because $\Pi_{\mathcal{T},\mathcal{T}}$ does (if the PN is not pure) or because the reachability graph contains paths $i \xrightarrow{s} i$ where $i \in \mathcal{T}$ but all the intermediate markings are vanishing. Figure 4.1 depicts these situations on a portion of a reachability graph before and after this elimination (tangible and vanishing markings are represented as ovals and rectangles, respectively).

Since the sojourn time in the tangible markings is exponentially distributed, the resulting process is the CTMC $\{\mu(\theta), \theta \geq 0\}$ already mentioned in Section 2.3 [3] and it is more commonly described by the infinites-

FIG. 4.1. *Origin of diagonal entries in the transition rate matrix.*

imal generator matrix Q plus the initial probability vector $\pi(0)$, where

$$Q_{i,j} = \begin{cases} \frac{\Pi_{i,j}^*(1 - \Pi_{i,i}^*)}{h_i} & \text{if } i \neq j \\ -\frac{1 - \Pi_{i,i}^*}{h_i} & \text{if } i = j \end{cases}$$

The factor $(1 - \Pi_{i,i}^*)$ is needed to account for the nonzero diagonal entries in Π . In other words, the expected sojourn time in tangible marking i is h_i for the SMP $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$, but $h_i/(1 - \Pi_{i,i}^*)$ for the CTMC $\{\mu(\theta), \theta \geq 0\}$, which does not consider transitions $i \rightarrow i$ as interruptions of the sojourn in marking i .

The elimination of the vanishing markings reduces the number of states in the process to be studied, but it also perturbs the structure of the reachability graph, which describes the marking-to-marking transitions. For example, if $r_{a,b} \neq 0$ in Figure 4.1, the arc $a \rightarrow b$ must be taken into account even if a and b have been eliminated.

We can then say that the temporal analysis of a SRN is concerned with computing $\sigma(\theta)$, σ , $\pi(\theta)$, π , $N(\theta)$, N , $\Phi(\theta)$, and Φ given h , Π , and $\gamma(0)$, or Q and $\pi(0)$. We call the first approach *preservation*, as opposed to the second one requiring the *elimination* of the vanishing markings.

4.2. Instantaneous steady state analysis. For steady state analysis, it is possible to use either preservation or elimination. As shown in [15], both approaches have advantages and disadvantages and the best choice among the two depends on the problem being solved. This discussion assumes that the state space contains a single recurrent class, see Section 4.4 for the general case.

4.2.1. Preservation. To compute the steady state instantaneous probabilities π and marking-to-marking transition frequencies Φ with preservation:

- Compute the steady state probability vector for the underlying DTMC:

$$(4.1) \quad \gamma = \gamma\Pi \quad \text{subject to} \quad \sum_{i \in \mathcal{S}} \gamma_i = 1$$

- Then:

$$\pi_i = \frac{\gamma_i h_i}{\sum_{k \in \mathcal{S}} \gamma_k h_k} = \frac{\gamma_i h_i}{\sum_{k \in \mathcal{T}} \gamma_k h_k}$$

- And:

$$\Phi_{i,j} = \Pi_{i,j} \phi_i = \Pi_{i,j} \frac{\gamma_i}{\sum_{k \in \mathcal{S}} \gamma_k h_k} = \frac{\gamma_i}{\sum_{k \in \mathcal{T}} \gamma_k h_k}$$

where $\phi = [\phi_i] = [\phi_{\mathcal{T}} | \phi_{\mathcal{V}}]$ represents the frequency with which each marking i is entered, or exited, in steady state.

4.2.2. Elimination. To compute the steady state instantaneous probabilities π and marking-to-marking transition frequencies Φ with elimination:

- Solve:

$$(4.2) \quad \pi Q = 0 \quad \text{subject to} \quad \sum_{i \in \mathcal{T}} \pi_i = 1$$

- If i is a tangible marking, $\phi_i = \pi_i/h_i$, hence:

$$\Phi_{i,j} = \Pi_{i,j} \phi_i = \Pi_{i,j} \frac{\pi_i}{h_i}$$

We compute ϕ_i as π_i/h_i and not as $-\pi_i/Q_{i,i} = \pi_i(1 - \Pi_{i,i}^*)/h_i$, so that ϕ_i represents the frequency at which tangible marking i is entered in steady state *before* the elimination of the vanishing markings. This ensures that ϕ_i has the same value independently of whether it is compute using preservation or elimination.

- For the vanishing markings, though, $\phi_{\mathcal{V}}$ must be obtained by solving the following equation:

$$(4.3) \quad \phi_{\mathcal{V}}[I - \Pi_{\mathcal{V},\mathcal{V}}] = \phi_{\mathcal{T}} \Pi_{\mathcal{T},\mathcal{V}}$$

Once $\phi_{\mathcal{V}}$ is obtained, $\Phi_{i,j}$ for $i \in \mathcal{V}$ is computed as

$$\Phi_{i,j} = \Pi_{i,j} \phi_i$$

4.3. Cumulative steady state analysis. As with instantaneous steady state analysis, the study of the cumulative behavior up to steady state can be performed using either preservation or elimination.

4.3.1. Preservation. To compute the cumulative sojourn times (for the transient markings) σ and marking-to-marking transitions (from the transient markings) N up to steady state with preservation:

- Define $\Pi^{(0,0)}$ and $\gamma^{(0)}(0)$ to be the restrictions of Π and $\gamma(0)$ to the transient markings of the underlying DTMC, respectively.
- Define $n = [n_i] = [n_{\mathcal{T}} | n_{\mathcal{V}}]$ to be the expected number of visits to each transient marking up to steady state for the underlying DTMC. n is computed by solving

$$(4.4) \quad n(I - \Pi^{(0,0)}) = \gamma^{(0)}(0)$$

- Then:

$$\sigma_i = n_i h_i$$

- And:

$$N_{i,j} = \Pi_{i,j} n_i$$

4.3.2. Elimination. To compute the cumulative sojourn times (for the transient markings) σ and marking-to-marking transitions (from the transient markings) N up to steady state with elimination:

- Define $Q^{(0,0)}$ and $\pi^{(0)}(0)$ to be the restrictions of Q and $\pi(0)$ to the transient markings, respectively.
- Solve:

$$(4.5) \quad \sigma Q^{(0,0)} = -\pi^{(0)}(0)$$

- If i is a tangible marking, $n_i = \sigma_i / h_i$, hence:

$$N_{i,j} = \Pi_{i,j} n_i = \Pi_{i,j} \frac{\sigma_i}{h_i}$$

- For the vanishing markings, though, $n_{\mathcal{V}}$ must be obtained by solving the following equation:

$$(4.6) \quad n_{\mathcal{V}}[I - \Pi_{\mathcal{V},\mathcal{V}}^{(0,0)}] = n_{\mathcal{T}} \Pi_{\mathcal{T},\mathcal{V}}^{(0,0)}$$

where $\Pi_{\mathcal{V},\mathcal{V}}^{(0,0)}$ and $\Pi_{\mathcal{T},\mathcal{V}}^{(0,0)}$ are the restrictions of $\Pi_{\mathcal{V},\mathcal{V}}$ and $\Pi_{\mathcal{T},\mathcal{V}}$ to the transient markings, respectively. Once $n_{\mathcal{V}}$ is obtained, $N_{i,j}$ is computed as

$$N_{i,j} = \Pi_{i,j} n_i$$

4.4. Multiple recurrent classes. Equation 4.1 admits a unique solution γ iff the state space \mathcal{S} of the corresponding stochastic process contains a single recurrent class. This section describes how to approach the solution when multiple recurrent classes exist. Exactly the same discussion would apply to Equation 4.2 and π .

If m recurrent classes exist, the initial marking μ_0 , and possibly other markings as well, are transient. We can partition \mathcal{S} into $\mathcal{S}^{(0)}, \mathcal{S}^{(1)}, \dots, \mathcal{S}^{(m)}$,

corresponding to the transient markings and to the m recurrent classes, respectively. Accordingly, Π and γ can be partitioned as

$$\Pi = \left[\begin{array}{c|c|c|c|c} \Pi^{(0,0)} & \Pi^{(0,1)} & \Pi^{(0,2)} & \dots & \Pi^{(0,m)} \\ \hline 0 & \Pi^{(1,1)} & 0 & \dots & 0 \\ \hline \dots & \dots & \Pi^{(2,2)} & \dots & 0 \\ \hline 0 & 0 & \dots & \dots & \Pi^{(m,m)} \end{array} \right] \quad \text{and} \quad \gamma = \left[\gamma^{(0)} \mid \gamma^{(1)} \mid \dots \mid \gamma^{(m)} \right]$$

The solution of each equation

$$\gamma^{(i)} \Pi^{(i,i)} = \gamma^{(i)} \quad \text{subject to} \quad \sum_{k \in \mathcal{S}^{(i)}} \gamma_k^{(i)} = 1$$

for $i = 1, \dots, m$, is unique, but

$$\forall c_1, \dots, c_m \in \mathbb{R}^+, \sum_{i=1}^m c_i = 1, \gamma = \left[0 \mid c_1 \gamma^{(1)} \mid \dots \mid c_m \gamma^{(m)} \right]$$

is a solution to Equation 4.1.

To obtain the correct value for γ , the constants c_i must be set to the probability of entering recurrent class $\mathcal{S}^{(i)}$ given the initial probability vector $\gamma(0)$. This is obtained by first computing the vector n , as described in Section 4.3. Then,

$$c_i = n \Pi^{(0,i)} \mathbf{1}^T$$

where $\mathbf{1}$ is a vector of the appropriate dimension with all entries equal one.

This approach is particularly efficient since the identification of the recurrent classes has complexity $O(|\mathcal{A}|)$. Furthermore, it has the additional advantage of requiring the manipulation of smaller matrices.

4.5. Transient Analysis. For transient analysis, elimination is the method of choice. To compute the transient instantaneous probabilities $\pi(\theta)$, solve the Kolmogorov ordinary differential equation

$$(4.7) \quad \dot{\pi}(\theta) = \pi(\theta)Q \quad \text{with initial condition} \quad \pi(0)$$

where $\dot{\pi}(\theta)$ is the derivative of π with respect to θ .

To compute the cumulative sojourn times $\sigma(\theta)$, solve the Kolmogorov differential equation:

$$(4.8) \quad \dot{\sigma}(\theta) = \sigma(\theta)Q + \pi(0), \quad \text{with initial condition} \quad \sigma(0) = 0,$$

which is obtained by integrating Equation 4.7 with respect to θ .

The computation of $\Phi(\theta)$ and $N(\theta)$ is exactly analogous to the steady state case and is omitted.

4.6. Sensitivity analysis. Analytical models are often used to evaluate alternatives during the design of a system, such as: (1) If the interconnections among the components are changed, how is the system throughput affected? (2) What effect does the increase in the number of buses have on their utilization? (3) How is the response time affected if the speed of the buses is increased?

Furthermore, exact values for the parameters used in the specification of the system may be unknown. One way of overcoming this problem is to solve the model for different values of the parameters and comparing the solutions obtained. If the model is highly sensitive to variation in a parameter, greater effort should be placed in estimating its exact value, and, possibly, the system should be overengineered, to allow for errors in the estimation process. The importance of performing a parametric or *sensitivity* analysis has been stressed in [9,29,64].

Sensitivity analysis of SRN models can be performed at various levels: (1) Changes in the structure of the SRN; (2) Changes in the initial number of tokens in a place; (3) Changes in an independent parameter β which is used in the definition of the rate or probability of one or more transitions. Levels (1), (2) and (3) directly correspond to the three examples mentioned above. For (1) and (2), the structure of the underlying reachability graph is altered, hence, the entire analysis process must be performed for each value of β . For (3), this approach is also possible, but it is not required, since the variation in β only affects some of the entries in h , Π , and $\gamma(0)$, with preservation, or Q and $\pi(0)$, with elimination.

4.6.1. Parametric sensitivity analysis. Assume that a parameter β appears in the specification of one or more firing rates, firing probabilities, or reward functions. Define x' to be derivative of x with respect to β . Given an output measure, $\Psi(\beta)$, we can compute its derivative $\Psi'(\beta)$ and then use the linear approximation

$$\Psi(\beta + \epsilon) \approx \Psi(\beta) + \Psi'(\beta)\epsilon$$

to obtain the value of Ψ for reasonably small variations ϵ in the parameter β (for simplicity, we omit to write explicitly the dependency on β in the remainder of this section).

If the measure is the expected instantaneous reward in steady state,

$$\Psi = \sum_{i \in \mathcal{S}} \rho(i)\pi_i + \sum_{i,j \in \mathcal{S}} r_{i,j}\Phi_{i,j}$$

and

$$\Psi' = \sum_{i \in \mathcal{S}} (\rho'(i)\pi_i + \rho(i)\pi'_i) + \sum_{i,j \in \mathcal{S}} (r'_{i,j}\Phi_{i,j} + r_{i,j}\Phi'_{i,j})$$

To compute Ψ' , $\rho(i)$, π_i , $r_{i,j}$, and $\Phi_{i,j}$ and the corresponding derivatives $\rho'(i)$, π'_i , $r'_{i,j}$, and $\Phi'_{i,j}$ are needed. If $\rho(i)$ and $r_{i,j}$ are given analytically,

the computation of their derivatives is trivial, so we focus on π'_i and $\Phi'_{i,j}$. To compute π' and Φ' with preservation:

- Compute γ first, then solve for γ' in

$$(4.9) \quad \gamma'(I - \Pi) = \gamma\Pi' \quad \text{subject to} \quad \sum_{i \in \mathcal{S}} \gamma'_i = 0$$

This equation is obtained by taking the derivative of Equation 4.1 with respect to β .

- Then:

$$\pi'_i = \frac{\gamma'_i h_i + \gamma_i h'_i}{\sum_{k \in \mathcal{T}} \gamma_k h_k} - \frac{\gamma_i h_i \sum_{k \in \mathcal{T}} (\gamma'_k h_k + \gamma_k h'_k)}{\left(\sum_{k \in \mathcal{T}} \gamma_k h_k \right)^2}$$

- And:

$$\Phi'_{i,j} = \frac{\gamma'_i}{\sum_{k \in \mathcal{T}} \gamma_k h_k} - \frac{\gamma_i \sum_{k \in \mathcal{T}} (\gamma'_k h_k + \gamma_k h'_k)}{\left(\sum_{k \in \mathcal{T}} \gamma_k h_k \right)^2}$$

To compute π' and Φ' with elimination:

- Compute π first, then solve for π' in

$$(4.10) \quad \pi'Q = -\pi Q' \quad \text{subject to} \quad \sum_{i \in \mathcal{T}} \pi'_i = 0$$

This equation is obtained by taking the derivative of Equation 4.2 with respect to β . While Π' is trivially obtained given that the functional dependency of the entries of Π on β is known, the computation of Q' is more complex. First, obtain $\Pi^{*'}$, the derivative of Π^* with respect to β :

$$\Pi^{*'} = \Pi'_{\mathcal{T},\mathcal{T}} + \Pi'_{\mathcal{T},\mathcal{V}}X + \Pi_{\mathcal{T},\mathcal{V}}X'$$

where X and X' are computed by solving, respectively,

$$[I - \Pi_{\mathcal{V},\mathcal{V}}]X = \Pi_{\mathcal{V},\mathcal{T}}$$

and

$$[I - \Pi_{\mathcal{V},\mathcal{V}}]X' = \Pi'_{\mathcal{V},\mathcal{T}} + \Pi'_{\mathcal{V},\mathcal{V}}X$$

Then, $Q'_{i,j}$ is given by,

$$Q'_{i,j} = \begin{cases} \frac{\Pi'^*_{i,j}(1 - \Pi^*_{i,i}) - \Pi^*_{i,j}\Pi'^*_{i,i}}{h_i} - \frac{\Pi^*_{i,j}(1 - \Pi^*_{i,i})h'_i}{h_i^2} & \text{if } i \neq j \\ \frac{\Pi'^*_{i,i}}{h_i} + \frac{(1 - \Pi^*_{i,i})h'_i}{h_i^2} & \text{if } i = j \end{cases}$$

- If i is a tangible marking,

$$\Phi_{i,j} = \Pi'_{i,j} \frac{\pi_i}{h_i} + \Pi_{i,j} \left(\frac{\pi'_i}{h_i} - \frac{\pi_i h'_i}{h_i^2} \right)$$

- For the vanishing markings, $\phi'_{\mathcal{V}}$ must be obtained by solving the following equation:

$$(4.11) \quad \phi'_{\mathcal{V}}[I - \Pi_{\mathcal{V},\mathcal{V}}] = \phi'_{\mathcal{T}} \Pi_{\mathcal{T},\mathcal{V}} + \phi_{\mathcal{T}} \Pi'_{\mathcal{T},\mathcal{V}} + \Pi'_{\mathcal{V},\mathcal{V}} \phi_{\mathcal{V}}$$

Once $\phi'_{\mathcal{V}}$ is obtained, $\Phi'_{i,j}$ is computed as

$$\Phi'_{i,j} = \Pi'_{i,j} \phi_i + \Pi_{i,j} \phi'_i$$

If Ψ is the expected accumulated reward up to steady state, σ'_i and $N'_{i,j}$ are needed. To compute σ'_i and $N'_{i,j}$ with preservation:

- Solve for n' using the equation:

$$(4.12) \quad n' \left(I - \Pi^{(0,0)} \right) = n \Pi^{(0,0)'}$$

- Then:

$$\sigma'_i = n'_i h_i + n_i h'_i$$

- And:

$$N_{i,j} = \Pi'_{i,j} n_i + \Pi_{i,j} n'_i$$

To compute σ'_i and $N'_{i,j}$ with elimination:

- Solve:

$$(4.13) \quad \sigma' Q^{(0,0)} = -\sigma Q^{(0,0)'}$$

where $Q^{(0,0)'}$ is the restriction of Q' to the transient markings.

- If i is a tangible marking,

$$N_{i,j} = \Pi'_{i,j} \frac{\sigma_i}{h_i} + \Pi_{i,j} \left(\frac{\sigma'_i}{h_i} - \frac{\sigma_i h'_i}{h_i^2} \right)$$

- For the vanishing markings, $n'_{\mathcal{V}}$ must be obtained by solving the following equation:

$$(4.14) \quad n'_{\mathcal{V}}[I - \Pi_{\mathcal{V},\mathcal{V}}] = n'_{\mathcal{T}}\Pi_{\mathcal{T},\mathcal{V}} + n_{\mathcal{T}}\Pi'_{\mathcal{T},\mathcal{V}} + n_{\mathcal{V}}\Pi'_{\mathcal{V},\mathcal{V}}$$

Once $n'_{\mathcal{V}}$ is obtained, $N'_{i,j}$ is computed as

$$N'_{i,j} = \Pi'_{i,j}n_i + \Pi_{i,j}n'_i$$

If Ψ is the instantaneous or accumulated reward at time θ , $\pi'(\theta)$ and $\Phi'(\theta)$, or $\sigma'(\theta)$ and $N(\theta)$ must be computed, respectively. To compute $\pi'(\theta)$ and $\sigma(\theta)$, solve the ordinary differential equations

$$\dot{\pi}'(\theta) = \pi'(\theta)Q + \pi(\theta)Q' \quad \text{with initial condition} \quad \pi'(0) = 0$$

and

$$\dot{\sigma}'(\theta) = \sigma'(\theta)Q + \sigma(\theta)Q' \quad \sigma'(0) = 0,$$

obtained by taking the derivative of Equations 4.7 and 4.8 with respect to β [29,47].

The computation of $\Phi'(\theta)$ and $N'(\theta)$ is exactly analogous to the steady state case and is omitted.

5. Numerical solution. This section describes numerical techniques that can be used to solve the equations encountered in Section 4 when the state space is finite.

5.1. Steady state analysis. From the perspective of linear algebra, Equations 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.9, 4.10, 4.11, 4.12, 4.13, and 4.14 can be expressed in the form

$$xA = b$$

where either $b = 0$ and A is a Q -matrix, that is, an ergodic matrix with non-negative off-diagonal entries and row-sum equal zero, or $b \neq 0$ and A is a submatrix of a Q -matrix, obtained by eliminating the rows and columns corresponding to a subset of the states. For example, Equation 4.1, $\gamma = \gamma\Pi$, can be rewritten as $\gamma(\Pi - I) = 0$, and $(\Pi - I)$ is a Q -matrix.

If the system is singular ($b = 0$), the problem can be stated as finding the left eigenvector $x \neq 0$ associated with the eigenvalue 0 of A . Since the CTMC is ergodic, there is exactly one redundant equation in the system, the Null Space of Q has dimension one, and we can uniquely determine x using one additional constraint, given by $x\mathbf{1}^T = 1$ in our case, since x is then a probability vector. Methods for computing eigenvectors, such as QR or non-symmetric Lanczos, can be quite expensive. In our particular case, though, we can use iterative methods to solve this linear equation, which can produce savings both in storage and in arithmetic operations. The

same iterative methods can be used when $b \neq 0$, so we do not distinguish between these two possibilities.

Following Stewart and Goyal [64], we find that SOR (Successive Over-Relaxation) is a convenient and effective method of solving $xA = b$. One major advantage of an iterative method is that it computes the solution x in place, without modifying A : in theory, only A and x need to be stored. By storing A in compressed-row format [53], only $O(\eta)$ storage is required for A , where η is the number of non-zero entries in A . In practical problems, η is $O(n)$ where n is the number of states, rather than $O(n^2)$, the size of the full Q matrix.

SOR is a well-known method with good convergence characteristics [64,66]. Stewart and Goyal recommend estimating the optimal relaxation parameter ω from the differences between successive iterates. We use a variation of their method guiding the parameter ω directly by the magnitude of the difference vectors.

The iteration matrix for SOR is derived by splitting A into three components [19]:

$$A = (L + I + U)D$$

where L and U are strictly upper triangular and lower triangular, respectively.

The SOR iteration is [19]

$$x^{(k+1)} = x^{(k)}[(1 - \omega)I - \omega D^{-1}L][I + \omega D^{-1}U]^{-1} + b\omega D^{-1}[I + \omega D^{-1}U]^{-1}$$

where $x^{(k)}$ is the k -th iterate for x and the SOR iteration matrix is:

$$M_\omega = [(1 - \omega)I - \omega D^{-1}L][I + \omega D^{-1}U]^{-1}$$

If $b = 0$, the SOR iteration for the homogeneous system of equations is simply

$$x^{(k+1)} = x^{(k)}M_\omega$$

and $x^{(0)}$ should be chosen as recommended in [64], to ensure that the initial vector is not deficient in the direction of the solution vector. Also, as recommended in [64], we renormalize $x^{(k)}$ if the vector becomes too large or too small.

If $b \neq 0$, the choice of the initial vector is less critical, and no renormalization must be performed. In addition, convergence is guaranteed in this case, because in our applications, when $b \neq 0$ the matrix A is always strictly diagonally dominant. Convergence is not guaranteed for the homogeneous case.

Each iteration, we track the size of the difference vector $\delta^{(i)} = x^{(i)} - x^{(i-1)}$, $i \geq 1$. Specifically, we use the max vector norm to compute, for each

iteration,

$$\frac{\|\delta^{(i)}\|_\infty}{\|x^{(i)}\|_\infty} = \frac{\|x^{(i)} - x^{(i-1)}\|_\infty}{\|x^{(i)}\|_\infty}$$

The solution vector can be of any length, so we normalize the norm of $\delta^{(i)}$ using the norm of $x^{(i)}$.

The algorithm starts with $\omega = 1$, and progresses by performing 10 iterations before changing to a new ω . When $\omega = 1$, $M_\omega = -D^{-1}L(I + D^{-1}U)^{-1}$, hence SOR reduces to Gauss-Seidel for the first ten iterations. Let k be the index of the ω 's: that is

$$k = \left\lfloor \frac{i}{10} \right\rfloor$$

For every ω_k , compute Δ_k as

$$\Delta_k = \sum_{j=i}^{i+9} \frac{\|\delta^{(j)}\|_\infty}{\|x^{(j)}\|_\infty}$$

As we are accumulating the norms of the difference vectors, we normalize them by the norm of the current iterate. Every 30 iterations, the algorithm checks the progress of the computation in relation to the values of Δ_k . Let the index of this check be

$$\ell = \left\lfloor \frac{i}{30} \right\rfloor$$

At each check, the values of ω_k and ϵ_ℓ , used to increase or decrease ω as the computation proceeds, are updated ($\epsilon = 0.1$ initially). After every 10 iterations, unless we are checking the progress of the computation, we increase ω_k by the current ϵ_ℓ . That is,

$$\omega_{k+1} = \omega_k + \epsilon_\ell$$

Every 30^{th} iteration, we adjust the value of ω_k and ϵ_ℓ depending upon the values of Δ_{k-2} , Δ_{k-1} , and Δ_k . The adjustments are described in Table 5.1.

Intuitively, we are causing the SOR routine to oscillate around the optimal value of ω as measured by the sum of difference norms between successive iterates. The key idea is that the ω producing the smallest sum of difference vector norms will be close to optimal. This heuristic appears to work well in practice. These quantities are related to an estimate of the subdominant eigenvalue of the iteration matrix [64, Equation (10)]. In our procedure, we attempt to smooth the difference vectors (which will oscillate) by summing their norms over 10 iterations. When ϵ_ℓ is small enough, say around 10^{-7} , we assume that the algorithm has found a value

Condition	Action	Explanation
$\Delta_{k-2} \leq \Delta_{k-1} \leq \Delta_k$	$\omega_{k+1} = \omega_k - 4\epsilon_\ell$ $\epsilon_{\ell+1} = 0.97\epsilon_\ell$	δ^i is increasing, we have overshoot ω . Decrease ω so that it is smaller than ω_{k-2} and approach that point more slowly.
$\Delta_k \leq \Delta_{k-2} \leq \Delta_{k-1}$	$\omega_{k+1} = \omega_k - \epsilon_\ell$ $\epsilon_{\ell+1} = \epsilon_\ell$	The last set of 10 iterations produced the smallest accumulated difference norms. Back up to ω_{k-1} and perform another set of 30 iterations.
$\Delta_{k-2} \leq \Delta_k \leq \Delta_{k-1}$	$\omega_{k+1} = \omega_k - 3\epsilon_\ell$ $\epsilon_{\ell+1} = 0.89\epsilon_\ell$	The smallest sum of accumulated difference norms was at the beginning of the set. Back up to before ω_{k-2} and approach more slowly.
$\Delta_{k-1} \leq \Delta_k \leq \Delta_{k-2}$ or $\Delta_{k-1} \leq \Delta_{k-2} \leq \Delta_k$	$\epsilon_{\ell+1} = 0.5\epsilon_\ell$ $\omega_{k+1} = \omega_k - 3\epsilon_\ell$	In both of these cases, Δ_{k-1} is the smallest sum of difference norms. Decrease ϵ_ℓ by half, set $\omega_{k+1} = \omega_{k-1} - 1/2\epsilon_\ell$ and approach ω_{k-1} more slowly from below.
$\Delta_k \leq \Delta_{k-1} \leq \Delta_{k-2}$	$\omega_{k+1} = \omega_k - \epsilon_\ell$ $\epsilon_{\ell+1} = 0.97\epsilon_\ell$	The sum of difference norms is getting progressively smaller. Back up to ω_{k-1} and perform another set of 30 iterations.

TABLE 5.1

Adjustments to ϵ and ω for the SOR method.

close to the optimal value for ω and stop adjusting it. In the best case, this requires about 20 adjustments to ϵ_ℓ (600 iterations) and in the worst case about 500 adjustments to ϵ_ℓ (15,000 iterations). If, at any point, we exceed an upper bound on the number of iterations allowed, we terminate the adjustment of ϵ_ℓ and switch to Gauss-Seidel iteration. In practice, we have found this simple procedure to be very effective.

5.2. Instantaneous transient analysis. Several methods for the transient solution of a CTMC are available. Fully symbolic solution using Laplace transforms is possible only for CTMCs having a small number of states or a very regular structure [69]. Semi-symbolic solution of a CTMC in terms of time θ can be obtained via algebraic methods [67]. However, this algorithm has complexity $O(|T|^3)$, needs full storage for Q , and can be numerically unstable.

Thus, we resort to purely numerical solution techniques. We can write the general solution of Equation 4.7 as:

$$(5.1) \quad \pi(\theta) = \pi(0)e^{Q\theta}$$

where the *matrix exponential* $e^{Q\theta}$ is given by the Taylor series [45]

$$e^{Q\theta} = \sum_{i=0}^{\infty} \frac{(Q\theta)^i}{i!}$$

Direct evaluation of the matrix exponential is subject to severe round-off problems since Q contains both positive and negative entries.

We can use solution methods for linear differential equations like Runge-Kutta to solve Equation 4.7 directly. Jensen’s method [34] (also called Uniformization or Randomization by various authors [26,27,37,54]) is yet another numerical method based on infinite series summation. The above two methods have complexity $O(\eta q\theta)$ where

$$q = \max_{i \in \mathcal{T}} \{ |Q_{i,i}| \}$$

The computation requirements increase with q and θ , hence $q\theta$ has been identified as an index of the *stiffness* of a CTMC [54]. An implicit ODE method called *TR-BDF2* [54] is insensitive to the value of $q\theta$. An implicit Runge-Kutta method for stiff problems is described in [43].

5.2.1. Jensen’s method. This section presents Jensen’s method, which has many desirable properties, and a modification of it, which improves its performance for stiff problems. A significant advantage of this modification over implicit ODE methods is that it does not suffer from large overhead for non-stiff problems and, at the same time, can yield good accuracy.

The transient state probabilities of the CTMC are computed using Jensen’s method as:

$$(5.2) \quad \pi(\theta) = \sum_{i=0}^{\infty} \hat{\gamma}(i) e^{-q\theta} \frac{(q\theta)^i}{i!}$$

where $\hat{\gamma}(i)$ is the state probability vector of the underlying DTMC $\hat{Q} = Q/q + I$ at step i and is computed iteratively as

$$(5.3) \quad \hat{\gamma}(i) = \hat{\gamma}(i-1)\hat{Q} \quad \text{starting from} \quad \hat{\gamma}(0) = \pi(0)$$

In practice, the summation in Equation 5.2 can and must be carried out only up to a finite number of terms k called the *right truncation point*. Furthermore, as $q\theta$ increases, the Poisson distribution thins on the left as well and the terms in the summation for small values of i become insignificant. Thus it is advisable to start the summation at a value $l > 0$, called the *left truncation point* [20,54]. Equation 5.2 reduces to

$$(5.4) \quad \pi(\theta) \approx \sum_{i=l}^k \hat{\gamma}(i) e^{-q\theta} \frac{(q\theta)^i}{i!}$$

Given a truncation error tolerance requirement ϵ , we can precompute the number of terms of the series needed to satisfy this tolerance as

$$l = \max \left\{ j \in \mathbb{N} : \sum_{i=0}^{j-1} e^{-q\theta} \frac{(q\theta)^i}{i!} \leq \frac{\epsilon}{2} \right\} \quad \text{and} \quad k = \min \left\{ j \in \mathbb{N} : 1 - \sum_{i=0}^j e^{-q\theta} \frac{(q\theta)^i}{i!} \leq \frac{\epsilon}{2} \right\}$$

Since this method involves only additions and multiplications and no subtractions, it is not subject to severe roundoff errors. One of the main problems, though, is its $O(\eta q\theta)$ complexity [54]. The number of terms needed for Jensen's method between the left and the right truncation point is $O(\sqrt{q\theta})$. However, it is necessary to obtain the DTMC state probability vector at l , the left truncation point, and l is $O(q\theta)$. Thus we need to compute $O(\eta q\theta)$ matrix-vector multiplications. Instead of using successive matrix-vector multiplications to compute $\hat{\gamma}$, we could use the matrix squaring method and change the complexity from $O(\eta q\theta)$ to $O(|\mathcal{T}|^3 \log(q\theta))$ [54]. However, this method results in fill-in (reducing sparsity) and is not feasible for CTMC with large state spaces. When $q\theta$ is large, computing the Poisson probabilities, especially near the tails of the distribution, may result in underflow problems [22]. This also causes round-off problems since the number of floating point operations needed is large.

We address some of the problems caused by large values of $q\theta$ from the practical point of view, by modifying Equation 5.2. This modification is based on recognizing when the underlying DTMC has reached steady state and rewriting the equations to avoid further computations. The total computation time becomes then proportional to the sub-dominant eigenvalue of the DTMC matrix rather than to $q\theta$. Thus, stiffness as seen by the modified algorithm is the same as that of using the *power method* [64] to compute the steady state solution. In our experience with a variety of problems, we have found significant improvement when using this modification.

We begin by observing that Equation 5.3, used to compute the probability vectors for the underlying DTMC, also represents the iteration equation of the power method used for computing the steady state solution of a CTMC. If the convergence of the power method is guaranteed, we can terminate the iteration in Equation 5.3 upon attaining steady state and obtain considerable computational savings. To ensure convergence in Equation 5.3, we require that

$$q > \max_{i \in \mathcal{T}} \{ |Q_{i,i}| \}$$

since this guarantees that the DTMC described by \hat{Q} is aperiodic [25].

Assume that, observing the sequence $\hat{\gamma}(i)$, we establish that convergence has been achieved at the S -th iteration. Three cases arise: $S > k$, $l < S \leq k$, and $S \leq l$.

- $S > k$: steady state detection does not take place and $\pi(\theta)$ is computed using Equation 5.4.

- $l < S \leq k$: by substituting $\hat{\gamma}(i)$ with $\hat{\gamma}(S)$ for $i > S$, we can rewrite Equation 5.4 setting the right truncation point k to ∞ :

$$\begin{aligned}
 \pi(\theta) &\approx \sum_{i=l}^{\infty} \hat{\gamma}(i) e^{-q\theta} \frac{(q\theta)^i}{i!} \\
 &= \sum_{i=l}^S \hat{\gamma}(i) e^{-q\theta} \frac{(q\theta)^i}{i!} + \hat{\gamma}(S) \sum_{i=S+1}^{\infty} e^{-q\theta} \frac{(q\theta)^i}{i!} \\
 &= \sum_{i=l}^S \hat{\gamma}(i) e^{-q\theta} \frac{(q\theta)^i}{i!} + \hat{\gamma}(S) \left(1 - \sum_{i=0}^S e^{-q\theta} \frac{(q\theta)^i}{i!} \right)
 \end{aligned}$$

- $S \leq l$: the DTMC reaches steady state before the left truncation point. In this case, no additional computation is necessary and $\pi(\theta)$ is set to $\hat{\gamma}(S)$.

For stiff problems, the number of terms needed to meet the truncation error tolerance requirements can be large, but substantial computational savings result if the DTMC steady state is detected. In our experience, this is often happens, especially for large values of θ .

The detection of steady state for the underlying DTMC requires extreme care. We have implemented the steady state detection based on the suggestions given in [64]. The usual test for convergence compares successive iterates, but if the method is converging slowly, the change between successive iterates might be smaller than the error tolerance specified. We might then incorrectly assume that the system has reached steady state when, instead, it is only experiencing slow convergence. To avoid this problem we compare iterates that are spaced m iterations apart, checking the difference between $\hat{\gamma}(i)$ and $\hat{\gamma}(i - m)$. Ideally, m should be varied according to the convergence rate, but this is difficult to implement in practice. For simplicity, we choose m based on the iteration number: $m = 5$ when number of iterations is less than 100, $m = 10$ when it is between 100 and 1000 and $m = 20$ when it is greater than 1000. We also test for steady state only every m iterations, saving computation.

5.3. Cumulative transient analysis. The computation of $\sigma(\theta)$ is similar to that of $\pi(\theta)$. A method similar to Jensen's method for solving Equation 4.8 is given in [55].

Integrating Equation 5.2 with respect to θ yields,

$$\begin{aligned}
 \sigma(\theta) &= \frac{1}{q} \sum_{i=0}^{\infty} \hat{\gamma}(i) \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} \\
 (5.5) \quad &= \frac{1}{q} \sum_{i=0}^{\infty} \hat{\gamma}(i) \left(1 - \sum_{j=0}^i e^{-q\theta} \frac{(q\theta)^j}{j!} \right)
 \end{aligned}$$

This is again a summation of an infinite series which can be evaluated up to the first k significant terms [55] resulting in,

$$(5.6) \quad \sigma(\theta) \approx \frac{1}{q} \sum_{i=0}^k \hat{\gamma}(i) \left(1 - \sum_{j=0}^i e^{-q\theta} \frac{(q\theta)^j}{j!} \right)$$

The error due to truncation can be bounded from above:

$$\begin{aligned} \frac{1}{q} \sum_{i=k+1}^{\infty} \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} &\leq \frac{1}{q} \sum_{i=k+1}^{\infty} (i - (k+1)) e^{-q\theta} \frac{(q\theta)^i}{i!} \\ &\leq \frac{1}{q} \sum_{i=k+1}^{\infty} i e^{-q\theta} \frac{(q\theta)^i}{i!} - \frac{1}{q} \sum_{i=k+1}^{\infty} (k+1) e^{-q\theta} \frac{(q\theta)^i}{i!} \\ &\leq t \sum_{i=k}^{\infty} e^{-q\theta} \frac{(q\theta)^i}{i!} - \left(\frac{k+1}{q} \right) \sum_{i=k+1}^{\infty} e^{-q\theta} \frac{(q\theta)^i}{i!} \end{aligned}$$

Given an error tolerance requirement ϵ , we can compute the number of terms k needed as:

$$k = \min \left\{ j \in \mathbb{N} : \theta \sum_{i=k}^{\infty} e^{-q\theta} \frac{(q\theta)^i}{i!} - \left(\frac{k+1}{q} \right) \sum_{i=k+1}^{\infty} e^{-q\theta} \frac{(q\theta)^i}{i!} \right\}$$

The detection of steady state for the underlying DTMC applies also to Equation 5.6. Two cases arise: $S > k$ and $S \leq k$.

- $S > k$: steady state detection does not take place and $\sigma(\theta)$ is computed using Equation 5.6.
- $S \leq k$: Equation 5.6 is modified as follows:

$$\begin{aligned} \hat{\sigma}(\theta) &\approx \frac{1}{q} \sum_{i=0}^{\infty} \hat{\gamma}(i) \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} \\ &= \frac{1}{q} \sum_{i=0}^S \hat{\gamma}(i) \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} + \frac{1}{q} \hat{\gamma}(S) \sum_{i=S+1}^{\infty} \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} \\ &= \frac{1}{q} \sum_{i=0}^S \hat{\gamma}(i) \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} + \frac{1}{q} \hat{\gamma}(S) \left(\sum_{i=0}^{\infty} \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} - \sum_{i=0}^S \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} \right) \\ &= \frac{1}{q} \sum_{i=0}^S \hat{\gamma}(i) \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} + \frac{1}{q} \hat{\gamma}(S) \left(qt - \sum_{i=0}^S \sum_{j=i+1}^{\infty} e^{-q\theta} \frac{(q\theta)^j}{j!} \right) \\ &= \frac{1}{q} \sum_{i=0}^S \hat{\gamma}(i) \left(1 - \sum_{j=0}^i e^{-q\theta} \frac{(q\theta)^j}{j!} \right) + \frac{1}{q} \hat{\gamma}(S) \left(qt - \sum_{i=0}^S \left(1 - \sum_{j=0}^i e^{-q\theta} \frac{(q\theta)^j}{j!} \right) \right) \end{aligned}$$

6. Conclusion. Markov and Markov reward models are frequently used in performance and dependability analysis of discrete event systems. Models of real systems, however, tend to become extremely large. We have proposed the use of stochastic reward nets for the concise specification and automated generation of Markov reward models. We have presented a formal definition of stochastic reward nets and methods for their structural, temporal, and sensitivity analysis.

A number of papers have dealt with the application of the methods described in this paper to practical modeling problems [33,31,14,48,49]. Nevertheless, practical problems stretch the capabilities of current methods and tools. State truncation [36,48] and decomposition [16,68,11] methods are being investigated to solve the ubiquitous largeness problem.

Stochastic reward nets are extensions of stochastic Petri nets. The reader may consult [50,51,56] and the Springer-Verlag series, *Advances in Petri Nets* for further information on Petri nets. Stochastic extensions of PNs have been covered extensively in the proceedings of a series of IEEE workshops, *Petri Nets and Performance Models*.

REFERENCES

- [1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Menlo Park, CA, USA, 1974.
- [2] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of Stochastic Petri Nets. *IEEE Transactions on Software Engineering*, 15(7):832–846, July 1989.
- [3] M. Ajmone Marsan, G. Balbo, and G. Conte. 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] G. Balbo, G. Chiola, G. Franceschinis, and G. Molinari Roet. On the efficient construction of the tangible reachability graph of generalized stochastic Petri nets. In *Proceedings of the International Workshop on Petri Nets and Performance Models*, Madison, WI, USA, Aug. 1987.
- [5] F. Baskett, K. M. Chandy, R. R. Muntz, and F. Palacios-Gomez. Open, Closed, and Mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):335–381, Apr. 1975.
- [6] M. D. Beaudry. Performance-related reliability measures for computing systems. *IEEE Transactions on Computers*, C-27(6):540–547, June 1978.
- [7] J. Bechta Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola. Extended Stochastic Petri Nets: applications and analysis. In E. Gelenbe, editor, *Performance '84*, Paris, France, Dec. 1984. North-Holland.
- [8] J. Bechta Dugan, K. S. Trivedi, M. K. Smotherman, and R. M. Geist. The Hybrid Automated Reliability Predictor. *AIAA Journal of Guidance, Control and Dynamics*, 9(3):319–331, May 1986.
- [9] J. T. Blake, A. L. Reibman, and K. S. Trivedi. Sensitivity analysis of reliability and performance measures for multiprocessor systems. In *Proceedings of the 1988 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Santa Fe, NM, USA, May 1988.
- [10] G. Chiola. A software package for the analysis of Generalized Stochastic Petri Net models. In *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

- [11] H. Choi and K. S. Trivedi. Approximate Performance Models of Polling Systems using Stochastic Petri Nets. In *Proceedings of the IEEE INFOCOM 92*, Florence, Italy, May 1992.
- [12] G. Ciardo. *Analysis of large stochastic Petri net models*. PhD thesis, Duke University, Durham, NC, USA, 1989.
- [13] G. Ciardo, R. A. Marie, B. Sericola, and K. S. Trivedi. Performability analysis using semi-Markov reward processes. *IEEE Transactions on Computers*, 39(10):1251–1264, Oct. 1990.
- [14] G. Ciardo, J. Muppala, and K. S. Trivedi. Analyzing concurrent and fault-tolerant software using stochastic Petri nets. *Journal of Parallel and Distributed Computing*. To appear.
- [15] G. Ciardo, J. Muppala, and K. S. Trivedi. On the solution of GSPN reward models. *Performance Evaluation*, 12(4):237–253, 1991.
- [16] G. Ciardo and K. S. Trivedi. A decomposition approach for stochastic Petri net models. In *Proceedings of the Fourth International Workshop on Petri Nets and Performance Models (PNPM91)*, Melbourne, Australia, Dec. 1991.
- [17] G. Ciardo, K. S. Trivedi, and J. Muppala. SPNP: stochastic Petri net package. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models (PNPM89)*, Kyoto, Japan, Dec. 1989.
- [18] A. Cumani. ESP - A package for the evaluation of stochastic Petri nets with phase-type distributed transitions times. In *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.
- [19] G. Dahlquist and A. Björck. *Numerical Methods*. Prentice-Hall, Englewood Cliffs, N.J., 1974.
- [20] E. de Souza e Silva and H. R. Gail. Calculating availability and performability measures of repairable computer systems using randomization. *J. ACM*, 36(1):171–193, Jan. 1989.
- [21] E. de Souza e Silva and R. R. Muntz. Queueing networks: solutions and applications. In H. Takagi, editor, *Stochastic Analysis of Computer and Communication Systems*. Elsevier Science Publishers B.V. (North-Holland), 1990.
- [22] B. L. Fox and P. W. Glynn. Computing poisson probabilities. *Commun. ACM*, 31(4):440–445, Apr. 1988.
- [23] P. M. Frank. *Introduction to System Sensitivity*. Academic Press, New York, NY, 1978.
- [24] A. Goyal, W. C. Carter, E. de Souza e Silva, S. S. Lavenberg, and K. S. Trivedi. The System Availability Estimator. In *Proceedings of the Sixteenth International Symposium on Fault-Tolerant Computing*, pages 84–89, Vienna, Austria, July 1986.
- [25] A. Goyal, S. Lavenberg, and K. S. Trivedi. Probabilistic modeling of computer system availability. *Annals of Operations Research*, 8:285–306, Mar. 1987.
- [26] W. K. Grassmann. Means and variances of time averages in Markovian environments. *Eur. J. Oper. Res.*, 31(1):132–139, 1987.
- [27] D. Gross and D. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Oper. Res.*, 32(2):926–944, Mar.-Apr. 1984.
- [28] P. J. Haas and G. S. Shedler. Stochastic Petri net representation of discrete event simulations. *IEEE Transactions on Software Engineering*, 15(4):381–393, Apr. 1989.
- [29] P. Heidelberger and A. Goyal. Sensitivity analysis of continuous time Markov chains using uniformization. In P. J. Courtois, G. Iazeolla, and O. J. Boxma, editors, *Proceedings of the 2nd International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communications Systems*, Rome, Italy, May 1987.
- [30] R. A. Howard. *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*. John Wiley and Sons, New York, NY, 1971.
- [31] O. C. Ibe, H. Choi, and K. S. Trivedi. Stochastic Petri net models of client-server

- systems. *IEEE Transactions on Parallel and Distributed Systems*. to appear.
- [32] O. C. Ibe, R. C. Howe, and K. S. Trivedi. Approximate availability analysis of vaxcluster systems. *IEEE Trans. Reliability*, R-38(1):146–152, Apr. 1989.
- [33] O. C. Ibe and K. S. Trivedi. Stochastic Petri net models of polling systems. *IEEE Journal on Selected Areas in Communications*, 8(10), Dec. 1990.
- [34] A. Jensen. Markoff chains as an aid in the study of Markoff processes. *Skand. Aktuarietidskr.*, 36:87–91, 1953.
- [35] K. Jensen. Coloured Petri nets and the invariant method. *Theoretical Computer Science*, 14:317–336, 1981.
- [36] H. Kantz and K. S. Trivedi. Reliability Modeling of the MARS System: A Case Study in the Use of Different Tools and Techniques. In *Proceedings of the Fourth International Workshop on Petri Nets and Performance Models (PNPM91)*, Melbourne, Australia, Dec. 1991.
- [37] J. Keilson. *Markov Chain Models — Rarity and Exponentiality*. Applied Mathematical Sciences Ser. Vol. 28. Springer-Verlag, 1979.
- [38] F. P. Kelly. *Reversibility and Stochastic Networks*. Wiley, 1979.
- [39] W. Kleinoder. Evaluation of task structures for hierarchical multiprocessor systems. In D. Potier, editor, *Modelling Techniques and Tools for Performance Analysis*. Elsevier Science Publishers B.V. (North Holland), 1985.
- [40] S. S. Lavenberg, editor. *Computer Performance Modeling Handbook*. Academic Press, 1983.
- [41] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcick. *Quantitative System Performance*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1984.
- [42] V. W. Mak and S. F. Lundstrom. Predicting performance of parallel computations. *IEEE Transactions on Parallel and Distributed Systems*, 1(3):257–270, July 1990.
- [43] M. Malhotra and K. S. Trivedi. Higher order methods for the transient analysis of Markov chains. In *Proc. Int. Conf. on the Performance of Distributed Systems and Integrated Communication Networks*, Kyoto, Japan, Sept. 1991.
- [44] J. F. Meyer. Performability: a retrospective and some pointers to the future. *Performance Evaluation*, 14(3-4):139–156, 1992.
- [45] C. Moler and C. F. V. Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20(4):801–835, Oct. 1978.
- [46] M. K. Molloy. *On the integration of delay and throughput measures in distributed processing models*. PhD thesis, UCLA, Los Angeles, CA, USA, 1981.
- [47] J. K. Muppala. *Performance and Dependability Modeling Using Stochastic Reward Nets*. PhD thesis, Department of Electrical Engineering, Duke University, Durham, NC, Apr. 1991.
- [48] J. K. Muppala, A. S. Sathaye, R. C. Howe, and K. S. Trivedi. *Dependability modeling of a heterogenous VAXcluster system using stochastic reward nets*. Ellis Horwood Ltd., 1992. to appear.
- [49] J. K. Muppala, S. P. Woolet, and K. S. Trivedi. Real-time systems performance in the presence of failures. *IEEE Computer*, 24(5):37–47, May 1991.
- [50] T. Murata. Petri Nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–579, Apr. 1989.
- [51] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.
- [52] C. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, Bonn, West Germany, 1962.
- [53] S. Pissanetzky. *Sparse Matrix Technology*. Academic Press, Orlando, FL, USA, 1984.
- [54] A. L. Reibman and K. S. Trivedi. Numerical transient analysis of Markov models. *Computers and Operations Research*, 15(1):19–36, 1988.
- [55] A. L. Reibman and K. S. Trivedi. Transient analysis of cumulative measures of Markov model behavior. *Stochastic Models*, 5(4):683–710, 1989.
- [56] W. Reisig. *Petri Nets*, volume 4 of *EATC Monographs on Theoretical Computer*

- Science*. Springer-Verlag, New York, 1985.
- [57] G. Rozenberg. Introduction to Petri nets, Dec. 1991. Tutorial Notes for the Fourth International Workshop on Petri Nets and Performance Models (PNPM91).
 - [58] R. A. Sahner and K. S. Trivedi. Performance and reliability analysis using directed acyclic graphs. *IEEE Transactions on Software Engineering*, Oct. 1987.
 - [59] R. A. Sahner and K. S. Trivedi. Reliability modeling using SHARPE. *IEEE Transactions on Reliability*, R-36(2):186–193, June 1987.
 - [60] W. H. Sanders and J. F. Meyer. METASAN: a performability evaluation tool based on Stochastic Activity Networks. In *Proceedings of the ACM-IEEE Comp. Soc. Fall Joint Comp. Conf.*, Nov. 1986.
 - [61] M. L. Shooman. *Probabilistic Reliability: An Engineering Approach*. McGraw-Hill, New York, 1968.
 - [62] J. Sifakis. Use of Petri nets for performance evaluation. In H. Beilner and E. Gelenbe, editors, *Measuring, MODelling, and Evaluating Computer Systems*, pages 75–93. North Holland, 1977.
 - [63] R. M. Smith and K. S. Trivedi. The analysis of computer systems using Markov reward processes. In H. Takagi, editor, *Stochastic Analysis of Computer and Communication Systems*. Elsevier Science Publishers B.V. (North-Holland), 1990.
 - [64] W. Stewart and A. Goyal. Matrix methods in large dependability models. Technical Report RC-11485, IBM T.J. Watson Res. Center, Yorktown Heights, NY, 10598, Nov. 1985.
 - [65] D. P. Stotts and P. Godfrey. Place/transition nets with debit arcs. *Information Processing Letters*, 41:25–33, Jan. 1992.
 - [66] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley Massachusetts 02182, 1986.
 - [67] H. Tardif, K. S. Trivedi, and A. V. Ramesh. Closed-form transient analysis of Markov chains. Technical Report CS-1988, Dept. of Computer Science, Duke University, Durham, NC, 27706, June 1988.
 - [68] L. Tomek and K. S. Trivedi. Fixed-Point Iteration in Availability Modeling. In M. Dal Cin, editor, *Informatik-Fachberichte, Vol. 91: Fehlertolerierende Rechensysteme*. Springer-Verlag, Berlin, 1991.
 - [69] K. S. Trivedi. *Probability & Statistics with Reliability, Queueing, and Computer Science Applications*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1982.
 - [70] K. S. Trivedi, J. K. Muppala, S. P. Woollet, and B. R. Haverkort. Composite performance and dependability analysis. *Perf. Eval.*, 14(3-4):197–215, 1992.