ALGORITHMS FOR THE GENERATION OF STATE-LEVEL REPRESENTATIONS OF STOCHASTIC ACTIVITY NETWORKS WITH GENERAL REWARD STRUCTURES*

Muhammad A. Qureshi[1], William H. Sanders[2],
Aad P. A. van Moorsel[3], and Reinhard German[4]

[1] Bell Laboratories
101 Crawfords Corner
Holmdel, NJ 07733
mqureshi@bell-labs.com

[2] Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801
whs@crhc.uiuc.edu

[3] Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974
aad@bell-labs.com

[4] Prozeßdatenverarbeitung und Robotik
Technische Universität Berlin
Franklinstr. 28/29, 10587 Berlin, Germany
rge@cs.tu-berlin.de

ABSTRACT

Stochastic Petri nets (SPNs) and extensions are a popular method for evaluating a wide variety of systems. In most cases, their numerical solution requires generating a state-level stochastic process, which captures the behavior of the SPN with respect to a set of specified performance measures. These measures are commonly defined at the net level by means of a reward variable. In this paper, we discuss issues regarding the generation of state-level reward models for systems specified as stochastic activity networks (SANs) with "step-based reward structures." Step-based reward structures are a generalization of previously

*The reported work was carried out while M. Qureshi and A. van Moorsel were at The University of Illinois, Urbana-Champaign, and R. German was visiting The University of Illinois, Urbana-Champaign. W. H. Sanders, M. A. Qureshi, and A. P. A. van Moorsel were supported, in part, by NASA Grant NAG 1-1782. R. German was supported by Siemens Corporate, Research and Development.

proposed reward structures for SPNs and can represent all reward variables that can be defined on the marking behavior of a net. While discussing issues related to the generation of the underlying state-level reward model, we provide an algorithm to determine whether a given SAN is "well specified." A SAN is well specified if choices about which instantaneous activity completes among multiple simultaneously enabled instantaneous activities do not matter, with respect to the probability of reaching next possible stable markings and the distribution of reward obtained upon completion of a timed activity. The fact that a SAN is well specified is both a necessary and sufficient condition for its behavior to be completely probabilistically specified and hence is an important property to determine.

I Introduction

Stochastic Petri nets (SPNs) and extensions can compactly represent the behavior of a wide variety of systems. They have the advantage that many attributes of modern computer systems and networks (including contention for multiple resources, fault-tolerance, degradable performance, and timing constraints) can be represented precisely and in a single model. Likewise, through the use of reward structures and variables defined at the network level [1, 2], one can define many useful measures of performance, dependability, and performability on the network representation. Most often, a solution of the specified reward variables is obtained by constructing a finite-state stochastic process representation from the net-level model (e.g., [2, 3, 4]) and then solving the constructed process for the desired variables. When reward structures and variables are used, the resulting state-level representation is a reward model, consisting of a stochastic process and (state-level) reward structure defined on the process.

The algorithms for the generation of the state-level representation depend on the type of SPN representation employed. For example, when standard stochastic Petri nets are used (those without immediate transitions, e.g., [5]), the set of states of the process is taken to be the reachable markings of the net, which can be generated by simple exploration of possible transitions that may fire in markings, starting with the initial marking of the net.

If the net has both timed and immediate transitions (e.g., generalized SPNs (GSPNs) [3], stochastic activity networks (SANs) [4], and stochastic reward nets (SRNs) [2]), the algorithm is not as simple, since the reachable markings are then partitioned into two sets, according to whether they are vanishing or tangible. Vanishing markings (unstable markings in SAN terminology [6]) are those in which there is at least one enabled immediate transition, and hence (since immediate transitions are assumed to have priority over timed transitions) the net spends zero time. Tangible markings (stable markings in SAN terminology [6]) are those in which only timed transitions are enabled, and hence the net spends non-zero time. Since the measures defined on such nets typically depend on the times spent in markings, vanishing markings are not usually taken to be states in the underlying process and are discarded, complicating the generation procedure. Note that there are cases where it is beneficial (from a space perspective) to preserve the vanishing markings in the generated stochastic process [2], but this is not typically done.

In GSPNs and SRNs, "random switches" and priorities are used to specify (probabilis-

tically or deterministically) which transition fires among a set of simultaneously enabled immediate transitions. In the original GSPN definition [3], random switches were specified at the time of state generation, since it was not known *a priori* which sets of immediate transitions would be simultaneously enabled. This was undesirable for two reasons: 1) it required further model specification at generation time, thus not cleanly separating model specification and state generation, and 2) it sometimes required specification of probability distributions (random switches) over large sets of enabled immediate transitions even if the order of their firing did not matter with respect to the probability of reaching possible next tangible markings.

These problems were avoided, to a large extent, in the second GSPN definition [7] and SRN definition [2], where *weights* are assigned to immediate transitions. These weights are assigned at model specification time (thus avoiding the first problem). Probabilities of firing among a set of simultaneously enabled immediate transitions are then determined by summing the weights of all enabled immediate transitions and assigning a probability of firing to the transition equal to its weight divided by the sum of the weights of the enabled immediate transitions. Weights are usually taken to be marking independent, since to do otherwise can, in some cases, lead to "stochastic confusion" [7]. Given a weight specification, the generation of the stochastic process is achieved by an algorithm that first generates all tangible markings and a subnet of the vanishing markings, as determined by an "extended conflict set analysis" [8]. After this is done, the remaining vanishing markings are eliminated by using a reduction algorithm as defined in [9].

SANs differ from GSPNs and SRNs in that they use "cases" to allow the modeler to probabilistically specify uncertainties regarding what happens when an activity (similar to a transition in a Petri net) completes, rather than specifying this choice using a random switch. Each activity has one or more cases and a possibly marking dependent case distribution function, which specifies (probabilistically) which case is chosen when an activity completes. Multiple instantaneous activities (similar to immediate transitions, except that they have cases) can be enabled in unstable markings, but no probability distribution (random switch) is specified to choose among these activities. This cleanly separates probabilistic choices that are present in the system being modeled, which are represented as cases, and choices which are a consequence of the fact that multiple, unrelated events in the system, which are represented by instantaneous activities, may occur at the same time. Furthermore, intentional probabilistic choices can be specified precisely during model specification using

(possibly marking dependent) case probabilities, instead of weights, since case distributions are local to each activity and are not dependent on which activities are enabled in particular markings.

However, allowing simultaneously enabled instantaneous activities without random switches or priorities requires a more sophisticated state generation algorithm in order to insure that the state-level behavior is completely probabilistically specified. In particular, since weights are not used to specify a probabilistic order among simultaneously enabled instantaneous activities, we must insure that these choices do not matter with respect to the probability distribution across next reachable stable markings. Such unspecified choices can give rise to behaviors that are similar to "confusion" exhibited by some Petri nets but are more complicated since case probabilities can be marking dependent. In addition, since the reward structure we consider allows (among other things) impulses to be specified on instantaneous activity completions, we require that the probability distribution on the impulse rewards obtained when transitioning from one stable marking to another does not depend on choices among instantaneous activities. When both of these conditions hold, the SAN is "well specified." Well-specified-ness is a necessary and sufficient condition for the probabilistic behavior of a SAN to be completely specified and for a unique state-level reward model to exist. Previous work along these lines was done by Sanders [10] in 1988, and concurrently by Ciardo and Zijal [11], but both of these works considered a more limited reward structure.

In this paper, we develop an algorithm to generate a state-level representation from a SAN with a very general reward structure, which can capture information regarding sequences of instantaneous activities that can complete, taking the SAN from one marking to another. Although these results are derived in the context of SANs, they are also directly applicable to SPNs with the same reward structure and where weights are not assigned to all transitions at each priority level. Such sequences may represent important system behavior, for example, a sequence of steps in a recovery action or steps leading to a routing decision in a telecommunication switch. In addition, "step-based" reward structures can represent all reward variables that can be defined on the marking behavior of a SAN. The algorithm to generate a state-level representation includes a check to determine if the SAN is well specified and eliminates unstable markings on-the-fly, as each stable state is generated, rather than generating the entire reachability set first and then eliminating the unstable markings, as long as there are no cycles of instantaneous activities without an intervening timed activity. On-the-fly elimination can save a significant amount of memory in the

generation process, avoiding state generation problems that will occur if there are a very large number of unstable markings.

II Review of SAN Terminology

A SAN Primitives

Stochastic activity networks are a stochastic extension to activity networks [6, 4]. Informally, SANs consist of *places*, *activities*, *input gates*, and *output gates*. Places hold tokens, as in Petri nets. Collectively, the distribution of tokens among all the places defines the *marking* of the SAN. Let P be some finite set of places in a SAN. Formally, then, a marking is a mapping $\mu_p : P \rightarrow I\!N$. A set of possible markings of the SAN is the set of functions $M_P = \{\mu_p | \mu_p : P \rightarrow I\!N\}$. If S is a set of places ($S \subset P$, $S \neq \emptyset$), a marking of S is a mapping $\mu_s : S \rightarrow I\!N$. μ_s is called a *partial marking* of the SAN. A set of possible markings of S is the set of functions $M_S = \{\mu_s | \mu_s : S \rightarrow I\!N\}$.

In a SAN model, an event in the system being modeled is represented by either a *timed* or an *instantaneous* activity. Each timed activity is assigned an activity time distribution function which probabilistically specifies its activity time. Activity time distributions can be generally distributed random variables [4]. Instantaneous activities, on the other hand, represent events which are completed in a negligible amount of time. Furthermore, each activity, timed or instantaneous, has a strictly positive number of cases attached to it. Cases associated with an activity provide probabilistic choices about the next marking the SAN will enter after completion of the activity. These probabilistic choices are defined by assigning a discrete probability distribution to the cases of each activity.

Input gates have a finite set of inputs and one output, where the inputs are places and the output is an activity. Each input gate has a computable predicate and function, which are both defined on the associated input places. If the predicate is true and the attached activity completes, the marking of the SAN is changed according to the defined function. In this way, the input gates can put the SAN directly into any desired marking, eliminating the need to build a subnetwork of places and instantaneous activities to implement the function. Output gates are provided for a similar reason and have one input and a finite set of outputs; the input is a case of an activity, while all outputs are places. A computable function is associated with each output gate. If the case attached to the output gate is chosen upon completion of an activity, its function is executed to change the marking of

the SAN.

B SAN Execution

The execution of a SAN is a characterization of the possible completions of activities, selections of cases, and changes in markings. The marking of a SAN can only change by completion of an enabled activity. An activity is *enabled* in a marking if all the directly connected places contain at least one token and, in the case of an input gate connected to the activity, the predicate of the input gate is true. An activity *may complete* in a marking, if it is enabled, and, if it is timed, there are no instantaneous activities enabled. This definition thus gives instantaneous activities priority over timed activities.

The "yields" function describes the marking change that occurs when an activity completes and a case is chosen. Formally, the notation $\mu \stackrel{a,c}{\rightarrow} \mu'$ is used to indicate that the completion of an activity a and choice of the case c yield μ'. The set of all the reachable markings from a given marking is defined in terms of the reflexive, transitive closure of the yield relation, $\stackrel{*}{\rightarrow}$. As a result, the set of all the reachable markings of a SAN $R(SAN, \mu_0)$, where μ_0 is the initial marking, is given by $R(SAN, \mu_0) = \{\mu | \mu_0 \stackrel{*}{\rightarrow} \mu\}$. Reachable markings are classified into *stable* and *unstable* markings. *Stable reachable markings* (denoted by the set $SR(SAN, \mu_0) \subseteq R(SAN, \mu_0)$) are those in which no instantaneous activity is enabled. *Unstable reachable markings* (denoted by $UR(SAN, \mu_0) \subseteq R(SAN, \mu_0)$) are those in which at least one instantaneous activity is enabled.

Furthermore, each event in the evolution of a SAN is defined by a marking-activity-case triple $<\mu, a, c>$, called a *configuration*, where a is an activity with case c, which may complete in μ. A completion of a configuration occurs when the activity a in marking μ completes by choosing case c. We say that a configuration is stable (unstable) if the marking of the configuration is stable (unstable). The execution of a SAN can be described in terms of sequences of configurations, more formally called "stable steps." Each stable step in the execution of a SAN takes it from one stable marking to a next stable marking.

Definition B.1: A *stable step* in the execution of a SAN with initial marking μ_0 is a sequence of configurations,

$$< \mu_1, a_1, c_1 >< \mu_2, a_2, c_2 > \ldots < \mu_n, -, - >,$$

where $n \geq 2$, and

1. $\mu_1, \mu_n \in SR(SAN, \mu_0)$,

2. for $2 \leq i \leq n-1$, $\mu_i \in UR(SAN, \mu_0)$, and

3. for $1 \leq i \leq n-1$, $\mu_i \stackrel{a_i, c_i}{\rightarrow} \mu_{i+1}$.

Performance and dependability measures can be specified in terms of reward variables defined on SANs. In the next section, we will introduce a reward structure that includes and generalizes previously proposed ways of specifying reward variables.

III Step-Based Reward Structure

Reward variables are an established method of specifying performance and dependability measures in SPNs [1, 2]. In this formalism, rate rewards are associated with stable markings, and the reward is accumulated with the specified rate during the time the SAN is in the marking. Likewise, impulse rewards are assigned to events in the SAN execution and are accumulated every time the particular event occurs. In their most general form, these events can be, for example, completions of activities or sequences of activities, choices of cases, and entrances to markings. This gives us great flexibility in defining measures and makes it possible to capture sophisticated measures of worth which depend on times spent on markings, and benefits and costs assigned to events or sequences of events.

As will be seen below, events in a SAN execution can be mapped on stable steps and/or "partial steps" in the SAN execution. A *partial step* is a sequence of configurations in which the number of configurations is less than the number of configurations in a stable step. To support the definition of rewards on stable and partial steps, we therefore introduce in this section a so-called *step-based reward structure*. In this structure, impulse rewards which are assigned to an event on the SAN level may correspond to *sets* of stable or partial steps in the SAN execution. For instance, a reward assigned to an instantaneous activity will be part of any stable step that contains that activity in its sequence of configurations. A formalism to represent such sets based on the notion of stable, unstable, and special extended configurations follows. In particular, these extended configurations can be used to represent a set of partial steps where every element is either a configuration or is of type $< \mu, -, - >$. A "restricted sequence of extended configurations" can be used to represent a set of stable

or partial steps. Formally,

Definition III.1: A *stable extended configuration* is a triple $(\boldsymbol{\mu}, \boldsymbol{a}, \boldsymbol{c})$, where

- $\boldsymbol{\mu}$ is either $\star$, μ_s, or μ_p, where $\star$ is the set of all stable reachable markings, $\mu_s \in M_S$ is a set of stable reachable markings which satisfy the partial marking μ_s, and $\mu_p \in M_P$ is a particular stable reachable marking of the SAN,
- $\boldsymbol{a}$ is either $\star$, s_a, or a, where $\star$ is the set of all timed activities, s_a is a particular set of timed activities, and a is a specific timed activity,
- $\boldsymbol{c}$ is either $\star$, s_c, or c, where $\star$ is the set of all cases, s_c is a particular set of cases, and c is a specific case.

Definition III.2: An *unstable extended configuration* is a triple $(\boldsymbol{\mu}, \boldsymbol{a}, \boldsymbol{c})$, where

- $\boldsymbol{\mu}$ is either $\star$, μ_s, or μ_p, where $\star$ is the set of all unstable reachable markings, $\mu_s \in M_S$ is a set of unstable reachable markings which satisfy the partial marking μ_s, and $\mu_p \in M_P$ is a particular unstable reachable marking of the SAN,
- $\boldsymbol{a}$ is either $\star$, s_a, or a, where $\star$ is the set of all instantaneous activities, s_a is a particular set of instantaneous activities, and a is a specific instantaneous activity,
- $\boldsymbol{c}$ is either $\star$, s_c, or c, where $\star$ is the set of all cases, s_c is a particular set of cases, and c is a specific case.

Definition III.3: A *special extended configuration* $(\mu_p, -, -)$ or $(\mu_s, -, -)$, specifies the event that the SAN enters the reachable marking μ_p or a marking in the set of reachable markings which satisfy the partial marking μ_s, respectively.

Now, sets of stable steps can be defined in terms of restricted sequences of extended configurations. In particular,

Definition III.4: A *restricted sequence of extended configurations* is the sequence

$$(\boldsymbol{\mu_1}, \boldsymbol{a_1}, \boldsymbol{c_1})(\boldsymbol{\mu_2}, \boldsymbol{a_2}, \boldsymbol{c_2}) \ldots (\boldsymbol{\mu_n}, \boldsymbol{a_n}, \boldsymbol{c_n})(\boldsymbol{\mu_{n+1}}, -, -)^{0,1},$$

where $(\boldsymbol{\mu_1}, \boldsymbol{a_1}, \boldsymbol{c_1})$ can be either a stable or an unstable extended configuration and $(\boldsymbol{\mu_i}, \boldsymbol{a_i}, \boldsymbol{c_i})$, $i = 2, 3, \ldots, n$ are unstable extended configurations. The superscript on the special element denotes the number of times it appears in the sequence.

Using the above formalism, a reward structure can be given for rate and impulse rewards, such that rate rewards are associated with stable markings, and impulse rewards are associated with events in the SAN execution. To this end, let the set SEC denote the set of all possible restricted sequences of extended configurations, and let PM be the set of all possible stable markings and partial markings. Given these definitions, the step-based reward structure is defined as follows.

Definition III.5: A *step-based reward structure* is a pair of functions:

- $R : \mathrm{PM} \rightarrow \mathbb{R}$ where for $r \in \mathrm{PM}$, $R(r)$ is the rate of reward accumulated in marking r, if r is a stable reachable marking. If r is a partial marking, then $R(r)$ is the rate of reward accumulated while in stable reachable markings which satisfy the partial marking r.

- $C : \mathrm{SEC} \rightarrow \mathbb{R}$ where for $\tau \in \mathrm{SEC}$, $C(\tau)$ is the impulse reward obtained upon completion of any step (partial or stable) in the restricted sequence of extended configurations τ.

Compared to previous reward structures (e.g., [1, 2]), the step-based reward structure provides a more flexible way of specifying impulse rewards. In particular, previously defined reward structures only allow impulse rewards to be associated with events such as completion of activities [1] or completion of activities in particular markings [2]. In a step-based reward structure, these events correspond to extended configurations $(\star, a, \star)$ and $(\mu, a, \star)$, respectively. Thus, a step-based reward structure covers all previous ways in which impulse rewards are assigned at the stochastic Petri net level. Moreover, the step-based reward structure allows one to specify other types of impulse rewards on the SAN level, since it allows the assignment of impulse rewards to all possible sequences of events that can be mapped to partial or stable steps.

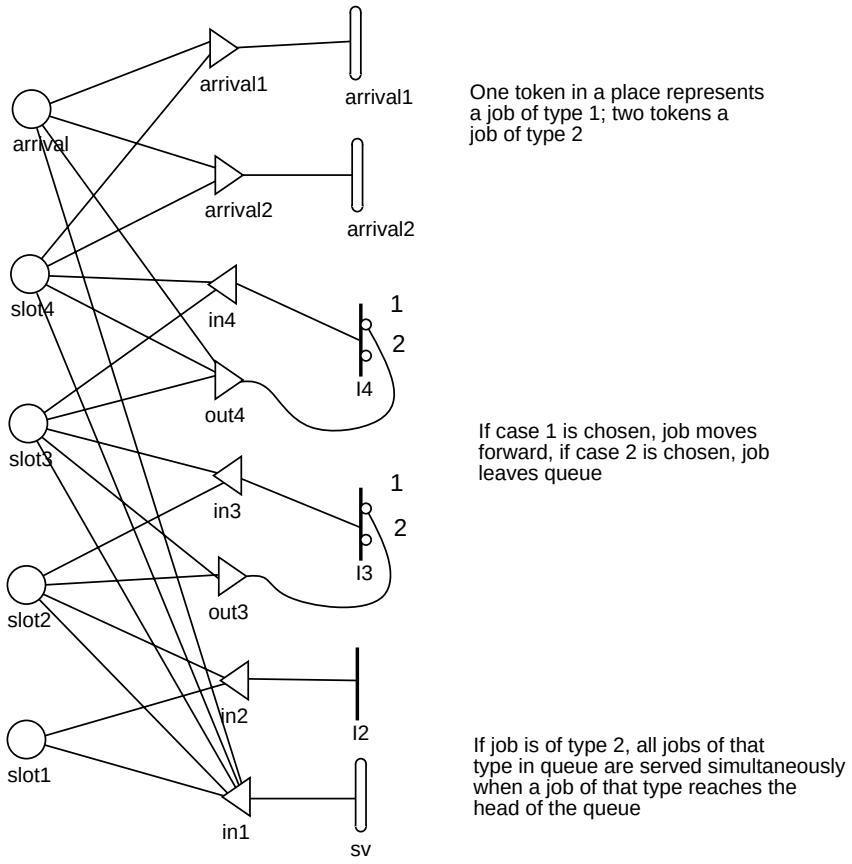

Figure 1: SAN Example

IV Example

To illustrate the use of SANs and step-based reward structures in representing systems and performance measures, consider a service station which consists of a single server and a finite queue. Two types of jobs, job_1 and job_2, arrive at the service station independently of each other. If the queue is full, an arriving job immediately leaves without obtaining any service. If, however, there is space in the queue, the arriving job enters and is placed at the end of the queue.

The server is capable of servicing jobs in two service modes. These service modes depend on the type of the job at the head of the queue. If the first job in the queue is of type job_1, then the server gives this job exclusive access to the server, with some (job-type specific) deterministic service time. If, however, the first job in the queue is of type job_2, then the server simultaneously serves all jobs of type job_2 that are in the queue. After completion of a

service, each remaining job in the queue either moves forward in the queue (with probability p) or becomes impatient and leaves the queue (with probability $(1-p)$). The probability p depends on the number of service periods remaining until the job can acquire the server.

Figure 1 depicts the SAN model of the service station with capacity of four jobs. Places *slot1*, *slot2*, *slot3*, and *slot4* correspond to the first, second, third, and fourth position in the queue, respectively. One token in these places signifies that there is a job of type job_1 in the corresponding position, while two tokens signify a job of type job_2.

The timed activity *sv* and the input gate *in1* model the two service modes of the server. The server takes a different amount of deterministic time to perform each type of service. Hence, the timed activity *sv* has an activity time which depends on the number of tokens in the place *slot1*. After the server completes a service of type job_1, the job from the place *slot1* leaves the system (as specified by gate *in1*). However, when the server completes a service of type job_2, the job from the place *slot1* leaves the system with other jobs of type job_2 (again as specified by gate *in1*). At this point, each remaining job can either move ahead in the queue with probability p or leave the system without getting service with probability $(1-p)$. Input gate *in4*, instantaneous activity *I4*, and output gate *out4* model this event for the job in place *slot*4. In this case, (marking dependent) probability p is associated with *case1* of the activity *I4*, while probability $(1-p)$ is associated with *case2*, which represents leaving of the job from the queue. Similarly, input gate *in3*, instantaneous activity *I3*, and output gate *out3* model the shift or departure of the job in place *slot*3. The job in place *slot*2 always moves ahead in the queue since it is always the next job to be serviced. Note that after the completion of service, shifts and departures are completed in zero time, since *I2*, *I3*, and *I4* are instantaneous activities.

The two timed activities *arrival1* and *arrival2* model the arrival of jobs of type job_1 and job_2, respectively, at the service station. Both of these timed activities have exponentially distributed activity times. After the completion of either of these timed activities, the connected output gate, *arrival1* or *arrival2*, discards the arriving job if the queue is full. Otherwise, the job enters at the fourth position, *slot4*, in the queue. The input gate *in4*, instantaneous activity *I4*, and output gate *out4* shift the job from the fourth position to the third position in zero time if there is no job at the third position. When this happens, *case1* of the activity *I4*, which models this forward movement of a job, is assigned a probability of one, since jobs only become impatient upon completion of a service. Similarly, *in3*, *I3*, and *out3* shift jobs from the third to second position, while *in2* and *I2* shift jobs from the second

to first position. Therefore, if there are zero customers in the queue, an entering customer at the fourth position will move to the first position in zero time. (See the appendix for complete illustration of gate functions and marking dependent case probabilities.)

Given the SAN description of the system, we can define a reward structure corresponding to the performance measure of interest. In particular, let X be the number of jobs that depart from our example queue without obtaining service during some interval of time. Suppose we are interested in finding the mean, variance, and distribution of X. In terms of step-based reward structure, X can be formulated as follows:

$$R(r) = 0, \quad \forall r$$
$$C(\tau) = \begin{cases} 1 & \text{if } \tau \in \{\tau_1 = (\star, I4, c2), \tau_2 = (\star, I3, c2)\} \\ 0 & \text{otherwise.} \end{cases}$$

Since the jobs which become impatient can leave the system by choice of *case2* after completion of instantaneous activity *I3* or *I4*, to compute X, these events must be counted over the observation period. This can be achieved by collecting an impulse reward of one for occurrence of each of these events. Note X cannot be defined in terms of rate rewards.

V Well-Specified SANs

To solve for a desired performance variable, we need to transform a SAN and the associated step-based reward structure into a state-level reward model. The state-level reward model consists of a stochastic process and a reward structure defined on the process. The generation of a state-level reward model from a SAN and associated step-based reward structure requires translation of probabilistic behavior of the SAN, as defined by its execution rules, to the stochastic process level. To a large extent, this translation can be carried out in a straightforward manner by considering the probabilistic behavior of timed activities and case distributions. However, if multiple instantaneous activities are simultaneously enabled, the generation of the process will be complicated by the fact that these activities are not assigned priorities or probabilities. As a consequence, there is a concern whether the SAN specification is completely probabilistically specified, in the sense that it uniquely defines a stochastic process. In this section, we will discuss this issue and give conditions under which a SAN will be "well specified" with respect to a given step-based reward structure. These conditions will be such that they can be checked during the generation of the stochastic process from the SAN and the step-based reward structure.

Recall that the probabilistic behavior in SANs is specified by two types of probability distributions: 1) a case distribution assigned to cases of each timed or instantaneous activity and 2) an activity time distribution assigned to each timed activity. Case distributions quantify the nondeterminism regarding *case choice*, i.e., the choice of one of the cases of the just completed activity. The nondeterminism due to choice of timed activities is quantified by their activity time distributions.[1] However, uncertainty regarding which instantaneous activity, among several, completes is not quantified, since instantaneous activities complete in zero time. For convenience, we will refer to the activity choice among two or more simultaneously enabled instantaneous activities as a *non-quantified activity choice*.

As mentioned in the introduction, the problem of non-quantified activity choices can be circumvented by introducing probabilities or priorities to simultaneously enabled instantaneous activities. From the perspective of the modeler, however, assigning probabilities or priorities is not always natural, since it may require specifying probabilities or priorities on events where the order does not really matter. We therefore take a different approach, where choices that matter to a modeler are specified with cases, and choices among simultaneously enabled instantaneous activities are checked to insure that they do not matter. When this is the case, the SAN with the step-based reward structure is "well specified."

Informally, a SAN with a step-based reward structure is well specified if, for every stable reachable marking $\mu \in SR(SAN, \mu_0)$ and for every timed activity a which may complete in μ:

1. the probability of reaching a particular next stable marking, given that a completes in μ, is independent of non-quantified activity choices, and

2. the probability of obtaining a certain impulse reward when entering a particular next stable marking, given that a completes in μ, is independent of non-quantified activity choices.

If these two conditions are met, the SAN is well specified. To investigate these conditions more formally, we first define two measures on each stable step. The first measure expresses the probability of the occurrence of a particular stable step given that a particular

[1] Note that if multiple activities have discrete or mixed distributions, there is a possibility that they may complete at the same time. This ambiguity can either be resolved in the same manner as done here for instantaneous activities, or by probabilities or priorities, and hence will not be discussed further.

timed activity completes in a particular stable reachable marking. Specifically,

Definition V.1: Let $s = < \mu_1, a_1, c_1 >< \mu_2, a_2, c_2 > \ldots < \mu_n, a_n, c_n >< \mu_{n+1}, -, - >$ be a stable step. Then $P\{s|\mu_1, a_1\}$, *the probability of s given completion of timed activity* a_1 *in* μ_1 is

$$P\{s|\mu_1, a_1\} = p_{a_1}(c_1) \times p_{a_2}(c_2) \times \ldots \times p_{a_n}(c_n),$$

where $p_{a_i}(c_i)$, for $1 \leq i \leq n$, is the probability that case c_i is chosen when activity a_i completes.

The second measure expresses the impulse reward obtained by completion of a particular stable step and depends on the step-based reward structure defined earlier. Specifically,

Definition V.2: $i(s)$, the *impulse reward assigned to a stable step* s, is the sum of the impulse rewards assigned to any partial step that appears in the stable step and the impulse reward assigned to the stable step itself.

Furthermore, we divide stable steps into sets $S(\mu, a)$, where each set contains stable steps which occur upon completion of a particular timed activity a in some particular stable marking μ. Formally,

$$S(\mu, a) = \left\{ s \;\middle|\; \begin{array}{l} s \text{ is a stable step where timed} \\ \text{activity } a \text{ completes in marking } \mu \end{array} \right\}.$$

According to the first condition in our informal definition, we need to check whether all non-quantified activity choices give the same probability of reaching a particular next stable marking for each set $S(\mu, a)$. (We assume that the SAN is stabilizing [4], and hence the set $S(\mu, a)$ is always finite.) To understand the non-quantified activity choices that may occur, note that each stable step in the set $S(\mu, a)$ differs from every other stable step in the set in selecting different activity and/or case choices. If all stable steps in the set $S(\mu, a)$ only differ in their case choice, no non-quantified activity choices need be made when a completes in μ. In this situation, it is obvious that the SAN is well specified with respect to the reward structure. On the other hand, if steps in the set $S(\mu, a)$ differ from each other in that different instantaneous activities complete in the same unstable marking, one or more non-quantified activity choices must be made when a completes in μ. To identify whether

two stable steps s and s' in $S(\mu, a)$ make the same non-quantified activity choices, we define the following relation on $S(\mu, a)$:

$$R = \left\{ (s, s^{'}) \;\middle|\; \begin{array}{l} \text{for every unstable configuration } < \mu, a, c > \\ \text{in } s \text{ and unstable configuration } < \mu^{'}, a^{'}, c^{'} > \\ \text{in } s^{'} \text{ such that } \mu = \mu^{'},\ a = a^{'} \end{array} \right\}.$$

Two stable steps are thus related if, for every unstable marking they share in common, the same activity choice is made. R is a compatibility relation, i.e., it is reflexive and symmetric. Although a compatibility relation does not necessarily define a partition of a set, it does define a covering of a set, by the maximal compatibility classes of the relation. A subset $C(\mu, a) \subseteq S(\mu, a)$ is called a *maximal compatibility class* if every element of $C(\mu, a)$ and no element of $S(\mu, a) - C(\mu, a)$ is related to any element of $C(\mu, a)$.

If a non-quantified activity choice is made, steps that choose different instantaneous activities will belong to different compatibility classes. Therefore, for the first condition in our informal definition to be met, it is necessary that the probability of reaching each possible next stable marking be invariant across maximal compatibility classes. To check this, define $NS(\mu, a)$ as the set of next possible stable markings when a completes in μ.

Definition V.3: Let $\mu \in SR(SAN, \mu_0)$. The set of *next stable markings* for the SAN in μ upon completion of the timed activity a is the set

$$NS(\mu, a) = \left\{ \mu^{'} \;\middle|\; \begin{array}{l} \exists \text{ a stable step } s \text{ from } \mu \text{ to } \mu^{'} \\ \text{such that } a \text{ is the activity of} \\ \text{the first configuration of } s \end{array} \right\}.$$

On the basis of next stable reachable markings, we then partition the stable steps within the compatibility class $C(\mu, a)$ into sets $C(\mu, a, \mu^{'})$ where each set $C(\mu, a, \mu^{'})$ contains the stable steps whose final marking is $\mu^{'} \in NS(\mu, a)$. Formally,

$$C(\mu, a, \mu^{'}) = \left\{ s \;\middle|\; \begin{array}{l} \text{is a stable step from } \mu \text{ to} \\ \mu^{'} \text{ in } C(\mu, a) \end{array} \right\}.$$

To verify that stable steps in different compatibility classes give the same probability of reaching a particular next stable marking, we compute $P_{C(\mu,a)}\{\mu^{'}|\mu, a\}$. $P_{C(\mu,a)}\{\mu^{'}|\mu, a\}$ is the probability of reaching a stable marking $\mu^{'} \in NS(\mu, a)$ when the timed activity a

completes in μ and the set of non-quantified activity choices corresponding to compatibility set $C(\mu, a)$ is chosen. Formally,

$$P_{C(\mu,a)}\{\mu^{'}|\mu, a\} \;=\; \sum_{s \in C(\mu,a,\mu^{'})} P\{s|\mu, a\}. \tag{1}$$

Given this expression, the first condition for a SAN to be well specified holds when, for all $\mu^{'} \in NS(\mu, a)$, $P_{C(\mu,a)}\{\mu^{'}|\mu, a\}$ is the same for all maximal compatibility classes $C(\mu, a)$.

The second condition for a SAN to be well specified can be verified in a similar manner. To do this, we first define the set of impulse rewards $I(\mu, a, \mu^{'})$, where

$$I(\mu, a, \mu^{'}) \;=\; \left\{ i(s) \;\middle|\; \begin{array}{l} s \in S(\mu, a) \text{ and has next} \\ \text{stable marking } \mu^{'} \end{array} \right\}.$$

Furthermore, we partition the set $C(\mu, a, \mu^{'})$ into sets $C(\mu, a, \mu^{'}, i)$, where each set $C(\mu, a, \mu^{'}, i)$ contains the stable steps which have impulse reward equal to $i \in I(\mu, a, \mu^{'})$. Formally,

$$C(\mu, a, \mu^{'}, i) \;=\; \{s \mid s \in C(\mu, a, \mu^{'}) \text{ and } i(s) = i\}.$$

After forming sets $C(\mu, a, \mu^{'}, i)$, we compute $P_{C(\mu,a)}\{\mu^{'}, i \mid \mu, a\}$. For a maximal compatibility set $C(\mu, a)$, $P_{C(\mu,a)}\{\mu^{'}, i \mid \mu, a\}$ is the probability of reaching stable marking $\mu^{'}$ and obtaining impulse reward equal to i when the timed activity a completes in the marking μ. Specifically,

$$P_{C(\mu,a)}\{\mu^{'}, i \mid \mu, a\} \;=\; \sum_{s \in C(\mu,a,\mu^{'},i)} P\{s \mid \mu, a\}. \tag{2}$$

Furthermore, for each compatibility set $C(\mu, a)$, we define $P_{C(\mu,a)}\{i \mid \mu, a, \mu^{'}\}$ as the probability an impulse of i is obtained when stable marking $\mu^{'}$ is reached by completion of timed activity a in μ. $P_{C(\mu,a)}\{i \mid \mu, a, \mu^{'}\}$ can be expressed as:

$$P_{C(\mu,a)}\{i \mid \mu, a, \mu^{'}\} \;=\; \frac{P_{C(\mu,a)}\{\mu^{'}, i \mid \mu, a\}}{P_{C(\mu,a)}\{\mu^{'} \mid \mu, a\}}. \tag{3}$$

Finally, to fulfill the second condition for a SAN to be well specified, $P_{C(\mu,a)}\{i \mid \mu, a, \mu^{'}\}$ must be identical over all maximal compatibility classes $C(\mu, a)$. The second condition can

thus be checked in a manner identical to that for the first condition, resulting in the following precise condition for a SAN to be well specified.

Definition V.4: A SAN with a step-based reward structure is *well specified* if, for every $\mu \in SR(SAN, \mu_0)$, every timed activity a that is enabled in μ, and every $\mu' \in NS(\mu, a)$:

1. $P_{C(\mu,a)}\{\mu' \mid \mu, a\}$ is the same for all maximal compatibility classes $C(\mu, a) \subseteq S(\mu, a)$, and
2. $P_{C(\mu,a)}\{i \mid \mu, a, \mu'\}$ is the same for all maximal compatibility classes $C(\mu, a) \subseteq S(\mu, a)$, for each $i \in I(\mu, a, \mu')$.

When a SAN is well specified, we can thus drop the subscript $C(\mu, a)$ from $P_{C(\mu,a)}\{\mu' \mid \mu, a\}$ and $P_{C(\mu,a)}\{i \mid \mu, a, \mu'\}$, since these probabilities are invariant over different compatibility sets $C(\mu, a) \in S(\mu, a)$. Accordingly, $P\{\mu' \mid \mu, a\}$ gives the probability of entering stable marking μ' when a timed activity a completes in μ. Likewise, $P\{i \mid \mu, a, \mu'\}$ gives the probability an impulse $i \in I(\mu, a, \mu')$ is obtained when the stable marking μ' is reached by completion of timed activity a in μ.

VI Example Revisited

To illustrate the computation of the sets, compatibility classes, and probabilities discussed in this section, we consider again the SAN example in Figure 1. Specifically, we consider the set $S((0, 1, 2, 1, 2), sv)$ which consists of the stable steps that may occur when the timed activity *sv* completes in stable reachable marking (0,1,2,1,2). In this expression, and the following, we write markings as a vector (*arrival*, *slot4*, *slot3*, *slot2*, *slot1*), where the number in a position indicates the number of tokens in the corresponding place. The

set $S((0,1,2,1,2),sv)$ then consists of the following stable steps:

$$
\begin{aligned}
s_1 = \; & <(0,1,2,1,2),sv,c_1><(0,1,0,1,0),I2,c_1> \\
& <(0,1,0,0,1),I4,c_2><(0,0,0,0,1),-,->, \\
s_2 = \; & <(0,1,2,1,2),sv,c_1><(0,1,0,1,0),I2,c_1> \\
& <(0,1,0,0,1),I4,c_1><(0,0,1,0,1),I3,c_2> \\
& <(0,0,0,0,1),-,->, \\
s_3 = \; & <(0,1,2,1,2),sv,c_1><(0,1,0,1,0),I4,c_2> \\
& <(0,0,0,1,0),I2,c_1><(0,0,0,0,1),-,->, \\
s_4 = \; & <(0,1,2,1,2),sv,c_1><(0,1,0,1,0),I4,c_1> \\
& <(0,0,1,1,0),I2,c_1><(0,0,1,0,1),I3,c_2> \\
& <(0,0,0,0,1),-,->, \\
s_5 = \; & <(0,1,2,1,2),sv,c_1><(0,1,0,1,0),I2,c_1> \\
& <(0,1,0,0,1),I4,c_1><(0,0,1,0,1),I3,c_1> \\
& <(0,0,0,1,1),-,->, \\
s_6 = \; & <(0,1,2,1,2),sv,c_1><(0,1,0,1,0),I4,c_1> \\
& <(0,0,1,1,0),I2,c_1><(0,0,1,0,1),I3,c_1> \\
& <(0,0,0,1,1),-,-> .
\end{aligned}
$$

Recall that cases c_1 and c_2 of activity *I4* are assigned probabilities depending on the number of services after which the job in place *slot4* will acquire the server. Similarly, cases c_1 and c_2 of activity *I3* are assigned probabilities depending on the number of services after which the job in place *slot3* will acquire the server. In stable steps s_1, s_2, s_3, and s_4, the probabilities assigned to cases of both activities *I3* and *I4* are identical, since there is always one job ahead of the job in *slot3* and *slot4*. Let p be the probability assigned to case c_1 of the activities *I3* and *I4* in these markings, and $(1-p)$ be the probability assigned to case c_2 of these activities in these markings. Then, in Definition V.1, the summation is carried out over the elements

$$
\begin{aligned}
P\{s_1 \mid (0,1,2,1,2),sv\} &= (1-p) \\
P\{s_2 \mid (0,1,2,1,2),sv\} &= p(1-p), \\
P\{s_3 \mid (0,1,2,1,2),sv\} &= (1-p), \\
P\{s_4 \mid (0,1,2,1,2),sv\} &= p(1-p), \\
P\{s_5 \mid (0,1,2,1,2),sv\} &= p^2, \\
P\{s_6 \mid (0,1,2,1,2),sv\} &= p^2.
\end{aligned}
$$

Furthermore, recall that our performance measure is the number of jobs that depart from the queue without obtaining service. For this measure, according to the Definition V.2, $i(s_1) = i(s_2) = i(s_3) = i(s_4) = 1$ and $i(s_5) = (s_6) = 0$.

Then, according to the definition of the relation R and $S((0,1,2,1,2),sv)$, two sets of

maximal compatibility classes, C_1 and C_2, result, where

$$C_1 \ = \ \{s_1,\ s_2,\ s_5\}, \quad C_2 \ = \ \{s_3,\ s_4,\ s_6\}.$$

Furthermore, according to Definition V.3, the set of next stable markings reachable upon completion of timed activity sv in marking (0,1,2,1,2) can be computed as

$$NS((0,1,2,1,2), sv) \ = \ \{(0,0,0,0,1), (0,0,0,1,1)\}.$$

On the basis of this set of next stable markings, we compute:

$$\begin{aligned}
C_1((0,1,2,1,2), sv, (0,0,0,0,1)) &= \{s_1, s_2\} \\
C_1((0,1,2,1,2), sv, (0,0,0,1,1)) &= \{s_5\} \\
C_2((0,1,2,1,2), sv, (0,0,0,0,1)) &= \{s_3, s_4\} \\
C_2((0,1,2,1,2), sv, (0,0,0,1,1)) &= \{s_6\}.
\end{aligned}$$

Furthermore, using Equation 1, the following probabilities are computed:

$$\begin{aligned}
P_{C_1}\{(0,0,0,0,1) \mid (0,1,2,1,2), sv\} &= (1-p^2), \\
P_{C_1}\{(0,0,0,1,1) \mid (0,1,2,1,2), sv\} &= p^2, \\
P_{C_2}\{(0,0,0,0,1) \mid (0,1,2,1,2), sv\} &= (1-p^2), \\
P_{C_2}\{(0,0,0,1,1) \mid (0,1,2,1,2), sv\} &= p^2.
\end{aligned}$$

These probabilities imply that the first condition required for a SAN to be well specified is true when activity sv completes in marking $(0,1,2,1,2)$.

Similarly, from the knowledge of the impulse reward associated with each stable step, we compute the following sets:

$$\begin{aligned}
I((0,1,2,1,2), sv, (0,0,0,0,1)) &= \{1\}, \\
I((0,1,2,1,2), sv, (0,0,0,1,1)) &= \{0\},
\end{aligned}$$

On the basis of these sets of impulse reward, we then compute:

$$\begin{aligned}
C_1((0,1,2,1,2), sv, (0,0,0,0,1), 1) &= \{s_1, s_2\}, \\
C_1((0,1,2,1,2), sv, (0,0,0,1,1), 0) &= \{s_5\}, \\
C_2((0,1,2,1,2), sv, (0,0,0,0,1), 1) &= \{s_3, s_4\}, \\
C_2((0,1,2,1,2), sv, (0,0,0,1,1), 0) &= \{s_6\}.
\end{aligned}$$

Furthermore, using Equation 2, we can compute the following probabilities:

$$\begin{aligned}
P_{C_1}\{1, (0,0,0,0,1) \mid (0,1,2,1,2), sv\} &= (1-p^2), \\
P_{C_1}\{0, (0,0,0,1,1) \mid (0,1,2,1,2), sv\} &= p^2, \\
P_{C_2}\{1, (0,0,0,0,1) \mid (0,1,2,1,2), sv\} &= (1-p^2), \\
P_{C_2}\{0, (0,0,0,1,1) \mid (0,1,2,1,2), sv\} &= p^2.
\end{aligned}$$

Finally, using Equation 3, we compute:

$$\begin{aligned} P_{C_1}\{1 \mid (0,0,0,0,1) \mid (0,1,2,1,2), sv\} &= 1, \\ P_{C_1}\{0 \mid (0,0,0,1,1) \mid (0,1,2,1,2), sv\} &= 1, \\ P_{C_2}\{1 \mid (0,0,0,0,1) \mid (0,1,2,1,2), sv\} &= 1, \\ P_{C_2}\{0 \mid (0,0,0,1,1) \mid (0,1,2,1,2), sv\} &= 1. \end{aligned}$$

Thus the second required condition in Definition V.4 holds for this SAN and reward structure when activity sv completes in marking $(0,1,2,1,2)$. Identical computations could be made to check the other reachable stable markings, showing that the SAN is indeed well specified with respect to this reward structure.

In the next section, we present an algorithm to generate stable reachable markings of a SAN which embodies the well-specified check.

VII Generation of the State-Level Reward Model

The generation of a stochastic process underlying a SAN and its associated net level reward structure can be done in an algorithmic fashion. We will present in Section A an algorithm which generates the stable reachable markings and determines whether the SAN is well specified, using the approach described in the previous sections. If the SAN is well specified, it generates the sets $NS(\mu, a)$ and $I(\mu, a, \mu')$ and the probabilities $P\{\mu'|\mu, c\}$ and $P\{i|\mu, a, \mu'\}$, where $\mu' \in NS(\mu, a)$ and $i \in I(\mu, a, \mu')$. From this information, the state-level stochastic process and the associated state-level reward structure can be obtained. In Section B, we will discuss in detail the resulting state-level representation if states are chosen to correspond to stable reachable markings.

A Algorithm for Generating Stable Reachable Markings

SANs require that the initial marking of a SAN must be stable. Given a SAN and the initial marking μ_0, for each timed activity enabled in μ_0, the set of stable steps $S(\mu_0, a_i)$ is computed. After computation of a set $S(\mu_0, a_i)$, the conditions for a well-specified SAN are verified. As part of the well-specified check, first sets $NS(\mu_0, a_i)$ and $I(\mu, a, \mu^{'})$ and probabilities $P\{\mu^{'} \mid \mu_0, a_i\}$ and $P\{i \mid \mu_0, a_i, \mu^{'}\}$ are computed, where $\mu^{'} \in NS(\mu_0, a_i)$ and $i \in I(\mu, a, \mu^{'})$. This procedure is then repeated for every stable reachable marking until no more new stable reachable markings are generated. Using this approach, we present an algorithm which, given a SAN and its initial marking, generates all the stable reachable

markings if the SAN is well specified, otherwise it exits with the signal that the SAN is not well specified.

The algorithm is presented in a modular form where the main algorithm, for a stable reachable marking and an activity which may complete in it, invokes a second algorithm which performs the well-specified check.

Algorithm 1: (Generates the set, SR, of stable reachable markings and, for every $\mu \in SR(SAN, \mu_0)$ and every timed activity a which may complete in μ, computes $NS(\mu, a)$, $I(\mu, a, \mu')$, $P\{\mu' \mid \mu, a\}$, and $P\{i \mid \mu, a, \mu'\}$ where $\mu' \in NS(\mu, a)$ and $i \in I(\mu, a, \mu')$.)

$SR = \emptyset$. (set of stable reachable markings)
$U = \{\mu_0\}$. (set of unexplored reachable markings)
While $U \neq \emptyset$.
  For some $\mu \in U$:
    $SR = SR \cup \{\mu\}$.
    Compute the set of enabled timed activities
    E_T in μ.
    For each timed activity $a \in E_T$:
      By invoking Algorithm 2, determine
      if the conditions for the SAN to be well-
      specified hold in a marking μ when a
      completes.

      If the conditions do not hold, signal that
      the SAN is not well specified and exit
      the algorithm.

      Else, return $NS(\mu, a)$, $I(\mu, a, \mu')$,
      $P\{\mu' \mid \mu, a\}$, and $P\{i \mid \mu, a, \mu'\}$.

      $U = (U \cup NS(\mu, a)) - SR$.
While end.

Algorithm 2: (Determines if conditions for a SAN to be well specified hold in a marking μ when activity a completes. If so, return $NS(\mu, a)$, $I(\mu, a, \mu')$, $P\{\mu' \mid \mu, a\}$, and $P\{i \mid \mu, a, \mu'\}$ where $\mu' \in NS(\mu, a)$ and $i \in I(\mu, a, \mu')$.)

Compute $S(\mu, a)$, the set of all stable steps that can
result by completion of timed activity a in stable
reachable marking μ.

Compute $NS(\mu, a)$ using $S(\mu, a)$.

Construct the maximal compatibility sets

$C(\mu, s) \subseteq S(\mu, a)$.

For each maximal compatibility set $C(\mu, a)$:
    Compute $P_{C(\mu,a)}\{\mu^{'} \mid \mu, a\}$ and
    $P_{C(\mu,a)}\{i \mid \mu, a, \mu^{'}\}$.
Next $C(\mu, a)$.

If $P_{C(\mu,a)}\{\mu^{'} \mid \mu, a\}$ and $P_{C(\mu,a)}\{i \mid \mu, a, \mu^{'}\}$ are
identical for all maximal compatibility sets:
    Return $NS(\mu, a)$, $I(\mu, a, \mu^{'})$, $P\{\mu^{'} \mid \mu, a\}$,
    and $P\{i \mid \mu, a, \mu^{'}\}$.
Else:
    Return that the SAN is not well specified.

Note that the algorithm described above requires storage of all stable steps that can occur when transitioning from one stable marking to another. This is necessary since multiple non-quantified activity choices may occur without an intervening stable marking. Simpler algorithms exist when this is not case. See, for example, the algorithm presented by Ciardo and Zijal [11].

B State-Level Representation

The final step in obtaining a state-level representation of a SAN is to construct a state-level reward structure and corresponding stochastic process. Several types of stochastic processes can result from the SAN specification. Ciardo *et al.* [12], for example, distinguish Markov SPNs, semi-Markov SPNs, semi-regenerative SPNs, and generalized semi-Markov SPNs, depending on the type of underlying stochastic process which represents the marking behavior (Markov, semi-Markov, semi-regenerative, and generalized semi-Markov processes, respectively). The above-mentioned stochastic processes are of particular interest since analytic solutions have been derived to solve for the state space occupancy distribution and for reward measures.

In this section, we discuss how a state-level reward structure can be obtained for stochastic processes as mentioned above. In particular, we will assume that the states in the stochastic process correspond to stable reachable markings. Furthermore, we will discuss how the general state-level representation can be simplified if the stochastic process is Markov.

If rate rewards are considered, rates must be associated with certain states (stable

markings) in the generated process. Since more than one rate reward can be assigned to a stable marking (for instance through several partial markings), the total rate reward associated with a state will be the sum of all the rate rewards assigned to the corresponding stable marking. As a result, we then obtain the following rate reward vector $\mathbf{r}$:

$$\mathbf{r} \;=\; [r_1, r_2, \ldots, r_n],$$

where n is the number of states of the stochastic process.

The state-space-level representation for impulse rewards is more complicated because transitions between states must be considered. From the algorithm that generates the stable reachable markings, the following results are available: the set of impulses $I(\mu, a, \mu')$, and the probabilities $P\{\mu'|\mu, a\}$ and $P\{i|\mu, a, \mu'\}$, for all $\mu \in SR$, $\mu' \in NS(\mu, a)$, $i \in I(\mu, a, \mu')$, and all activities a enabled in μ. Since completion of different timed activities (possibly with different associated impulse rewards) can result in a transition between the same states, we distinguish for every transition the activity whose completion causes the transition to occur. Furthermore, since any activity completion can result in various impulse rewards (depending on the instantaneous activities "following" the activity), a probability distribution over all possible impulse rewards is assigned to each activity.

To describe this more formally, let v and w be states in the stochastic process and μ_v and μ_w denote the corresponding stable markings. Then, for all activities a that can result in a transition from state v to state w, $i^a_{v,w}$ is defined as a vector with probabilities

$$P\{\mu_w, i|\mu_v, a\} = P\{i|\mu_v, a, \mu_w\} \cdot P\{\mu_w|\mu_v, a\}.$$

So, $i^a_{v,w}$ has $|I(\mu_v, a, \mu_w)|$ elements. Furthermore, let $i_{v,w}$ be a vector consisting of the vectors $i^a_{v,w}$, i.e.,

$$i_{v,w} = (i^{a_1}_{v,w}, \cdots, i^{a_n}_{v,w}),$$

where $a_1, \cdots, a_n$, are all activities that result in a transition from state v to w (so, $i_{v,w}$ has dimension $\sum_{k=1}^{n} |I(\mu_v, a_k, \mu_w)|$). Then, the state-space-level impulse reward structure is represented by the following matrix:

$$\underline{\mathbf{I}} \;=\; \begin{bmatrix} i_{1,1} & i_{1,2} & \ldots & i_{1,n} \\ i_{2,1} & i_{2,2} & \ldots & i_{2,n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ i_{n,1} & i_{n,2} & \ldots & i_{n,n} \end{bmatrix}.$$

To close this section, we illustrate how the state-space-level definition simplifies if the underlying stochastic process is a continuous-time Markov process. In this case, the stochastic process can be represented by a generator matrix containing the rate of the transitions of the exponentially distributed activities. Let $\lambda_{a_k}(\mu_v), k = 1, \cdots, n$, be the rate of the exponentially distributed activities $a_1, \cdots, a_n$, which result in a transition from state v to w. Then, the (v, w)-th entry in the generator matrix is

$$\lambda_{v,w} = \sum_{k=1}^{n} \lambda_{a_k}(\mu_v) \cdot P\{\mu_w | \mu_v, a_k\}.$$

Furthermore, the distribution over impulse rewards, represented by the vector $i_{v,w}$, can be computed by

$$P\{i|\mu_v, \mu_w\} = \sum_{k=1}^{n} P\{i|\mu_v, a_k, \mu_w\} \cdot P\{a_k|\mu_v, \mu_w\},$$

for all $i \in I(\mu_v, a_k, \mu_w)$, $k = 1, \cdots, n$. In this way, the probability of occurrence of identical impulse rewards is summed for the activities $a_k, k = 1, \cdots, n$. The probability $P\{a_k|\mu_v, \mu_w\}$ is available from the algorithm for generating stable marking, via the relation

$$P\{a_k|\mu_v, \mu_w\} = \frac{\lambda_{a_k}(\mu_v) \cdot P\{\mu_w|\mu_v, a_k\}}{\lambda_{v,w}}.$$

So, we see that if the generator matrix and the probabilities $P\{i|\mu_v, \mu_w\}$ are available, a Markov SAN with step-based reward structure is completely defined on the state level.

VIII Conclusion

Generation of a state-level reward model representation is an integral step in the solution for reward measures defined on stochastic Petri nets. In this paper, we have developed an algorithm to do this for a general (step-based) reward structure defined on a SAN. The reward structure is more general than previously considered reward structures for SPNs and can represent all reward variables that can be defined on the marking behavior of the net. In addition, we have given an algorithm to determine if a SAN is well specified with respect to a step-based reward structure. Being well specified is a necessary and sufficient condition for the SAN to be completely probabilistically specified, and it allows one to construct a state-level reward model from a given SAN and reward structure.

The well-specified check gives one the freedom to probabilistically specify choices that matter, with respect to the system being modeled, using cases, and to *not* specify choices that do not matter but arise because multiple events (whose order does not matter) can happen at the same time. This cleanly separates choices that the modeler cares about, and those he/she does not care about, and allows for full marking dependency on case probabilities. Furthermore, elimination of unstable markings is done on-the-fly, thus reducing the memory requirement to generate a process representation if the SAN has unstable reachable markings, compared to other algorithms. The algorithm has been implemented in *UltraSAN* [13], a package for evaluating systems represented as stochastic activity networks, and which has been successfully applied in a wide variety of application areas.

Acknowledgments The authors would like to thank the anonymous reviewers for their helpful comments. They would also like to thank John Meyer for interesting discussions regarding well-specified and well-defined SANs.

REFERENCES

[1] W. H. Sanders and J. F. Meyer, "A unified approach for specifying measures of performance, dependability, and performability," in *Dependable Computing for Critical Applications* (A. Avizienis and J. C. Laprie, eds.), vol. 4 of *Dependable Computing and Fault-Tolerant Systems*, pp. 215–238, Springer Verlag, 1991.

[2] G. Ciardo, A. Blackmore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi, "Automated generation and analysis of Markov reward models using stochastic reward nets," in *Linear Algebra, Markov Chains, and Queueing Models*, Springer Verlag, 1993.

[3] M. A. Marsan, G. Balbo, and G. Conte, "A class of generalized Petri nets for performance and evaluation of multiprocessor systems," *ACM Transactions on Computer Systems*, vol. 2, pp. 93–122, May 1984.

[4] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: Structure, behavior, and application," in *Proceedings of the International Workshop on Timed Petri Nets*, (Torino, Italy), pp. 106–115, July 1985.

[5] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Transactions on Computers*, vol. C-31, pp. 913–917, September 1982.

[6] A. Movaghar and J. F. Meyer, "Performability modeling with stochastic activity networks," in *Proceedings of the 1984 Real-Time Systems Symposium*, (Austin, TX), December 1984.

[7] G. Chiola, M. A. Marsan, G. Balbo, and G. Conte, "Generalized stochastic Petri nets: A definition at the net level and its implications," *IEEE Transactions on Software Engineering*, vol. 19, pp. 89–107, February 1993.

[8] G. Balbo, G. Chiola, G. Franceschinis, and G. M. Roet, "Construction of the tangible reachability graph of generalized stochastic Petri nets," in *Proceedings of the International Workshop on Timed Petri Nets*, (Madison, Wisconsin), August 1987.

[9] G. Chiola, "A graphical Petri net tool for performance analysis," in *Proceedings of the 3rd International Workshop on Modeling Techniques and Performance Evaluation*, (AFCET, Paris, France), pp. 136–145, March 1987.

[10] W. H. Sanders, *Construction and Solution of Performability Models Based on Stochastic Activity Networks*. PhD thesis, University of Michigan, 1988.

[11] G. Ciardo and R. Zijal, "Well-defined stochastic Petri nets," in *Proceedings 4th International Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '96)*, (San Jose, CA, USA), pp. 278–284, Feb. 1996.

[12] G. Ciardo, R. German, and C. Lindemann, "A characterization of the stochastic process underlying a stochastic Petri net," *IEEE Transactions on Software Engineering*, vol. 20, pp. 506–515, July 1994.

[13] W. H. Sanders, W. D. Obal, M. A. Qureshi, and F. K. Widjanarko, "The *UltraSAN* modeling environment," *Performance Evaluation*, vol. 24, pp. 89–115, 1995.

APPENDIX

Table 1: Input Gate Definitions for SAN Example

Gate	*Definition*
in1	Predicate $MARK(slot1) > 0$
	Function *if* $(MARK(slot1) == 2)$ { *if* $(MARK(slot4) == 2)$ $MARK(slot4) = 0;$ *if* $(MARK(slot3) == 2)$ $MARK(slot3) = 0;$ *if* $(MARK(slot2) == 2)$ $MARK(slot2) = 0;$ } $MARK(slot1) = 0;$ $MARK(arrival) = 0;$
in2	Predicate $MARK(slot2) > 0$ && $MARK(slot1) == 0$
	Function $MARK(slot1) = MARK(slot2);$ $MARK(slot2) = 0;$
in3	Predicate $MARK(slot3) > 0$ && $MARK(slot2) == 0$
	Function *identity*
in4	Predicate $MARK(slot4) > 0$ && $MARK(slot3) == 0$
	Function *identity*

Table 2: Activity Case Probabilities for SAN Example

Activity	*Case*	*Probability*
I3	1	*if* (*MARK*(*arrival*) == *0*) { *if* ((*MARK*(*slot1*) == *1*) || (*MARK*(*slot3*) == *1* && *MARK*(*slot1*) == *2*)) { *return*(*0.75*); } *else if* ((*MARK*(*slot1*) == *0*) || (*MARK*(*slot3*) == *2* && *MARK*(*slot1*) == *2*)) { *return*(*1.0*); } } *else* { *return*(*1.0*); }
	2	*if* (*MARK*(*arrival*) == *0*) { *if* ((*MARK*(*slot1*) == *1*) || (*MARK*(*slot3*) == *1* && *MARK*(*slot1*) == *2*)) { *return*(*0.25*); } *else if* ((*MARK*(*slot1*) == *0*) || (*MARK*(*slot3*) == *2* && *MARK*(*slot1*) == *2*)) { *return*(*ZERO*); } } *else* { *return*(*ZERO*); }
I4	1	*if* (*MARK*(*arrival*) == *0*) { *if* ((*MARK*(*slot1*) == *1* && *MARK*(*slot2*) == *1*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) == *1* && *MARK*(*slot2*) == *2*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *1*)) { *return*(*0.5*); } *else if* ((*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *1* && *MARK*(*slot2*) == *2*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *1*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) > *0* && *MARK*(*slot2*) == *0*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) == *0* && *MARK*(*slot2*) > *0*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *1* && *MARK*(*slot2*) == *0*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *0* && *MARK*(*slot2*) == *1*) ||

Table 3: Activity Case Probabilities for SAN Example

Activity	*Case*	*Probability*
		(*MARK*(*slot4*) == *1* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *2*)) { *return*(*0.75*); } *else if* ((*MARK*(*slot1*) == *0* && *MARK*(*slot2*) == *0*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *0* && *MARK*(*slot2*) == *2*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *0*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *2*)) { *return*(*1.0*); } } *else* { *return*(*1.0*); }
	2	*if* (*MARK*(*arrival*) == *0*) { *if* ((*MARK*(*slot1*) == *1* && *MARK*(*slot2*) == *1*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) == *1* && *MARK*(*slot2*) == *2*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *1*)) { *return*(*0.5*); } *else if* ((*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *1* && *MARK*(*slot2*) == *2*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *1*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) > *0* && *MARK*(*slot2*) == *0*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) == *0* && *MARK*(*slot2*) > *0*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *1* && *MARK*(*slot2*) == *0*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *0* && *MARK*(*slot2*) == *1*) || (*MARK*(*slot4*) == *1* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *2*)) { *return*(*0.25*); } *else if* ((*MARK*(*slot1*) == *0* && *MARK*(*slot2*) == *0*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *0* && *MARK*(*slot2*) == *2*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *0*) || (*MARK*(*slot4*) == *2* && *MARK*(*slot1*) == *2* && *MARK*(*slot2*) == *2*)) {

Table 4: Activity Case Probabilities for SAN Example

Activity	*Case*	*Probability*
		return(ZERO); } } *else* { *return(ZERO);* }

Table 5: Output Gate Definitions for SAN Example

Gate	*Definition*
arrival1	*if (MARK(slot4) == 0) MARK(slot4) = 1;* *MARK(arrival) = 1;*
arrival2	*if (MARK(slot4) == 0) MARK(slot4) = 2;* *MARK(arrival) = 1;*
out3	*MARK(slot2) = MARK(slot3); MARK(slot3) = 0;*
out4	*MARK(slot3) = MARK(slot4); MARK(slot4) = 0;*