From: *Proceedings of PNPM '99: the 8th International Workshop on Petri Nets and Performance Models*, Zaragoza, Spain, September 8–10, 1999

An Efficient Well-Specified Check

Daniel D. Deavours and William H. Sanders *

Department of Electrical and Computer Engineering
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL, U.S.A.
E-mail: {deavours,whs}@crhc.uiuc.edu

Abstract

A longstanding problem with generalized stochastic Petri nets and extensions is that of what to do when more than one zero-timed event is scheduled to occur at the same time. If the order is left unspecified, it could lead to ambiguity that affects reward variables. Stochastic activity nets (SANs) [6, 7] have used the well-specified condition to avoid this problem. However, the existing algorithm to perform the well-specified check is computationally complex, proportional to the number of paths through unstable markings. We provide some theoretical results that allow us to make use of a much more efficient algorithm, with complexity proportional to the number of arcs between unstable markings.

Keywords: Stochastic Petri nets, stochastic activity networks, Markov process, well-specified, well-defined.

1. Introduction

Stochastic Petri nets have emerged as a popular way of expressing performance and dependability models. They have been successfully used in building performance and reliability models with complex structural interactions. The ability to generate a Markov chain in a fairly straightforward way makes analytical/numerical solution possible, and the relatively simple structure yields fast simulators when numerical solution is impractical.

Extensions to SPNs, including generalized SPNs (GSPNs) [5], allow for both timed and immediate transitions. A fundamental problem that arises from immediate transitions is that of what to do when multiple immediate transitions are enabled at the same time. The network may behave differently depending on which transition is chosen to fire first. A similar problem arises if several timed transitions have the same firing time, which can happen if the transition delay is deterministic or has a discrete or mixed distribution.

Historically, there have been several approaches to solving this problem. An early solution [5] was to assign probabilities to immediate transitions. Since it is not always easy or practical to foresee all the possible combinations of enabled immediate transitions, the state-space generator prompts the user to assign probabilities to the set of enabled immediate transitions. This solution could be tedious to the user for larger, complex models.

Later refinements to GSPNs assigned priorities and weights to immediate transitions [4]. The probability of choosing an immediate transition is then the ratio of the weight of the immediate transition to the sum of the weights of enabled immediate transitions. While this is a reasonable solution, it still requires that the user have some implicit knowledge of what sets of immediate transitions may be enabled at one time. This can be difficult for larger models. Also, it sometimes leads to over-specification, when the user is required to specify weights that are irrelevant. Furthermore, the modeler may make an error in building the model, in which case the result of over-specification is legal but unanticipated and incorrect behavior. Thus, over-specification can be undesirable.

A different approach is to use an *extended conflict set* (ECS) [1]. Weights in transitions are used only to choose among transitions within the ECS. This is an attempt to reduce the complexity of assigning weights to immediate transitions. The order in which immediate transitions of different ECSs fire may not matter; this means that the firing of a transition in one ECS may not affect the enabling or firing rule of a transition in a different ECS. This rule is called *defined at the net level*. There are structural tests [1] to determine whether a conflict between ECSs exists, but these methods may declare that a conflict may exist where one

*This work was supported, in part, by DARPA/ITO under Contract No. DABT63-96-C-0069. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA/ITO.

does not exist. The only known accurate test is a comprehensive test at state-generation time.

A similar problem exists when more than one timed transition is scheduled to fire at the same time. Priorities, weights, and ECSs may be applied, but unless there is a type of global weight-assignment system, or a test at state-generation time, there is a possibility of ambiguity.

There are several other tests that may be performed at state-generation time that are aimed at reducing the burden of specification and eliminating over-specification, while still ensuring that the behavior of a GSPN is completely specified. One of these tests is based on the notion of a well-defined GSPN [2]. A GSPN is well-defined if the underlying stochastic process is completely described at every step, including all events that occur in zero time. A check that a model is well-defined can be done when exploring the state space by an efficient algorithm that does not add to the asymptotic running time of the state-generation algorithm. However, the well-defined definition may be too restrictive, since some ambiguities do not change the behavior of the stochastic process in a measurable sense (for example, with respect to a reward structure). Therefore, a more relaxed definition called *well-defined with respect to a reward structure* was defined and applied [2]. We discuss some of the details of this definition later. An efficient algorithm to detect whether a GSPN is well-defined with respect to a reward structure has been given in [2].

In parallel with the evolution of GSPNs, stochastic activity networks (SANs) [6, 7] have developed a technique to address a closely related problem. An activity (similar to a transition) has *cases*, which have probabilities assigned to them. When an activity completes, a case is chosen. Since SANs have cases, there is no need to use weights on instantaneous activities to specify choices between different behaviors in the SAN. However, if multiple instantaneous activities are enabled at the same time, there may be some ambiguity as to which activity completes first. As long as the probability of going to a next stable (similar to tangible) marking and obtaining an impulse reward is independent of the order in which instantaneous activities complete, we say that the SAN is well-specified. A SAN must be well-specified if its behavior is to be analyzed probabilistically. There are some subtle distinctions, which we discuss formally in Section 3, between "well-defined with respect to a reward structure" and "well-specified."

Algorithms have been developed to determine whether a SAN is well-specified [3, 8, 9]. However, these algorithms require the searching of all paths from a stable state to the set of next stable states. This makes the algorithm very computationally complex, which substantially slows down the state-space generation if there is a large number of interacting instantaneous activities.

We note that while the well-defined definition has historically been applied to GSPNs with ECSs and the well-specified definition has historically been applied to SANs, there is nothing unique about the definitions that ties them to these specific formalisms. GSPN ECSs are similar to SAN instantaneous activities, and GSPN immediate transitions are similar to SAN cases. (We discuss SANs in more detail in Section 2.1.) Since we focus on the well-specified definition in this paper, we discuss both definitions as applied to SANs.

In this paper, we show that the two definitions are in fact equivalent. The implication of this is that the well-specified check can now be performed much more efficiently than was previously known [3].

We begin, in Section 2, by reviewing SANs and introducing the configuration graph. In Section 3, we formally present the definitions of well-specified and well-defined. We prove that the two definitions are equivalent in Section 4, and show the algorithm as applied to the configuration graph in Section 5. We conclude in Section 6.

2. SANs

2.1. Review of SANs

Stochastic activity networks (SANs) have similarities to stochastic Petri nets. A SAN is composed of places, activities (similar to transitions), input gates, and output gates. Places hold tokens, as they do in Petri nets. The number of tokens in places of the SAN is called the *marking* of the SAN. Formally, let P be the (finite) set of places in the SAN. A marking is then a mapping $\mu : P \rightarrow I\!N$.

Activities in a SAN may be either *timed* or *instantaneous*. Each timed activity has a (possibly general) probability distribution function assigned to it, which describes the time an activity takes to complete once it becomes enabled. Timed activities are drawn as hollow vertical bars. Instantaneous activities have no time distribution associated with them; the delay between enabling and completion is "instantaneous" with respect to time in the model. Instantaneous activities are drawn as solid vertical lines. Each activity has a (positive) number of cases associated with it, which probabilistically describe a choice of next markings a SAN may enter after the completion of the activity. A case is drawn as a circle on an activitiy, usually on the right.

Each input gate has several input arcs and a single output arc. Input arcs are connected to places in the SAN, and the output arc connects to an activity. An input gate is made up of two functions: an input gate predicate, which is a general marking-dependent expression used to determine whether an activity is enabled, and an input gate function, which is a general marking-change function used to change the SAN marking when an activity completes. The input gate

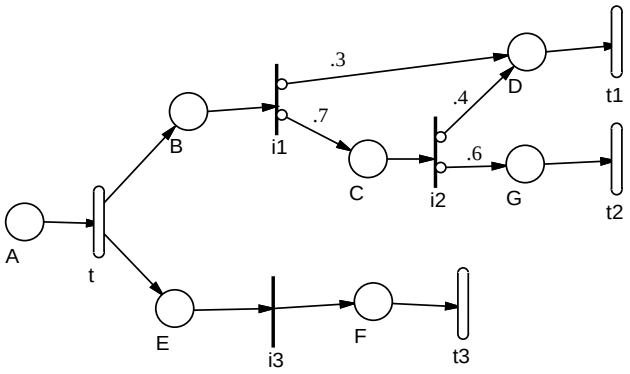

Figure 1. SAN example.

function may change the marking of any of the connected places. Input gates are drawn as right-pointing triangles.

An output gate has input arcs that connect activity cases to the output gate, and output arcs that connect the gate to places. An output gate also has a general function that describes how the marking changes when the connected activity completes, and it may change the marking of the connected places in a general way. Output gates are drawn as left-pointing triangles.

If a place is directly connected to an activity by an arc, the behavior of the SAN is the same as that of an SPN. When an activity completes, a case is chosen, the input gate functions are executed, and then the output gate functions of the output gates that are connected to the chosen case are executed.

An example SAN is given in Figure 1. This example shows a SAN with places, timed activities, and instantaneous activities with and without cases. Gates are a powerful extension, and allow for general, marking-dependent behavior. For the sake of illustration, we omit gates from our example.

2.2. SAN execution

The execution of a SAN specifies how the marking of the SAN changes over time, and which activity completions and case selections cause these changes. An activity is *enabled* if all the places connected to the activity by input arcs have more than one token, as in Petri nets, and if all the input gate predicates are true.

Instantaneous activities always have priority over timed activities; that is, if there are timed and instantaneous activities enabled in the same marking, an instantaneous activity will complete. If several timed activities are enabled and no instantaneous activities are enabled, the activity with the minimum delay is chosen to complete first.

The "yields" function specifies how the marking changes when an activity completes. We say that the completion of activity a and the choice of case c in marking μ *yields* marking μ', and we write $\mu \stackrel{a,c}{\rightarrow} \mu'$.

We make use of the reflexive, transitive closure of the yields relation, $\stackrel{*}{\rightarrow}$, to construct the set of reachable markings of a SAN. From the initial marking of a SAN, μ_{init}, we can define the set of all reachable markings in the SAN as $R(SAN, \mu_{\text{init}}) = \{\mu | \mu_{\text{init}} \stackrel{*}{\rightarrow} \mu\}$. A reachable marking can be stable or unstable. A *stable marking* is one in which no instantaneous activities are enabled; an *unstable marking* is a marking in which one or more instantaneous activities are enabled. We denote the set of stable reachable markings as $SR(SAN, \mu_{\text{init}}) \subseteq R(SAN, \mu_{\text{init}})$, and the set of unstable reachable markings as $UR(SAN, \mu_{\text{init}}) \subseteq R(SAN, \mu_{\text{init}})$.

The execution of a SAN is described, formally, using configurations. A *configuration* is a triple $<\mu, a, c>$, and represents the completion of activity a and the choice of case c in marking μ. We say that a configuration is *stable* if the marking of the configuration is a stable marking; a configuration is *unstable* if the marking of the configuration is unstable. We use the symbol "$-$" for "don't care" elements in configurations. The execution of a SAN is described in terms of a sequence of configurations called a *step*. There is a certain type of step called a *stable step*, which is a sequence of configurations in which the first configuration is a stable configuration, and subsequent configurations are unstable configurations. The last configuration must yield a stable marking. (Note that the last configuration is not a stable configuration, but it must yield a stable marking.)

2.3. Configuration graph

We use a configuration graph to aid us in describing the zero-timed behavior of a SAN. A configuration graph has similarities to the state-space graph of the unstable markings between stable markings. The primary difference is that we distinguish between markings in which there are non-quantified choices (multiple instantaneous activities enabled) and those in which there are quantified choices. This helps us considerably in our analysis.

A configuration graph is a directed acyclic graph. Each node is identified with a marking of the SAN, although the marking is not necessarily unique. It has a root node, which is the initial stable marking, and leaves, which are stable markings. The intermediate nodes correspond to unstable markings. We label the nodes with markings, e.g., μ_i. If a marking is a stable marking, then we use a hat, e.g., $\hat{\mu}_i$.

Configuration graphs have two types of nodes: case nodes and decision nodes. A *decision node* is a node in which more than one instantaneous activity is enabled. Let μ_i be the marking associated with a decision node. We call the node μ_i. Let a be an instantaneous activity enabled in marking μ_i. Then μ_i has a child node called μ_i^a, which represents marking μ_i in which immediate activity a is chosen

to complete. Node μ_i has a child node for each instantaneous activity that is enabled in μ_i.

The other type of node is a case node. A *case node* is a node in which there is only one instantaneous activity enabled, or, if multiple instantaneous activities are enabled, only one is chosen to complete. Let μ_i^a be a case node associated with marking μ_i, where instantaneous activity a is chosen to complete first. Let μ_j be a marking that is reached from marking μ_i when activity a completes and a case is chosen. Using SAN terminology, we can say that there exists the sequence of configurations $<\mu_i, a, c><\mu_j, -, ->$. The arcs from a case node to its children are labeled with the probability of reaching the child node, which is computable from case probabilities. Note that for case node μ_i^a, if multiple instantaneous activities are enabled in marking μ_i, the probabilities are computed using cases on activity a. Case node μ_i^a has a child for each marking reachable by completing a in marking μ_i.

The root node is a special node. Let $EN(\hat{\mu}_0)$ be the set of timed activities enabled in marking $\hat{\mu}_0$. We identify the root node with the stable marking $\hat{\mu}_0$ and a particular timed activity $t \in EN(\hat{\mu}_0)$, which is chosen to complete first. We label the root node $\hat{\mu}_0^t$ throughout this paper.

The leaf nodes of the tree are identified with the next stable marking, and the intermediate nodes are identified with unstable, intermediate markings. The children of the root node $\hat{\mu}_0^t$ are identified with the markings that are reachable from marking $\hat{\mu}_0$ by the firing of activity t. The arcs from the root node to the children are labeled with the probability of reaching the child node, and can be easily computed from the case probabilities on the timed activity. A path through the configuration graph corresponds directly to a stable step.

To facilitate the following discussion, we introduce the notion of a child set. The child set of a node μ_i is simply the set of child nodes in the configuration graph, and we write this as $C(\mu_i)$. Let $IEN(\mu_i)$ be the set of immediate activities enabled in the unstable marking μ_i. If μ_i is a decision node, then $C(\mu_i)$ contains case nodes of the form μ_i^a, $\forall a \in IEN(\mu_i)$. If μ_i is a decision node (which occurs if $|IEN(\mu_i)| = 1$), then $C(\mu_i)$ contains the nodes reachable from μ_i by firing the enabled instantaneous activity.

Note that SANs are by definition stabilizing. A SAN is *stabilizing* if the number of stable steps from any stable marking is finite. This means that the configuration graph is acyclic and has a finite number of nodes.

Recall the example SAN in Figure 1. We show the configuration graph of this SAN in Figure 2, where place A initially has one token, and all other places have none. The enumeration of the markings of this configuration graph is given in Table 1.

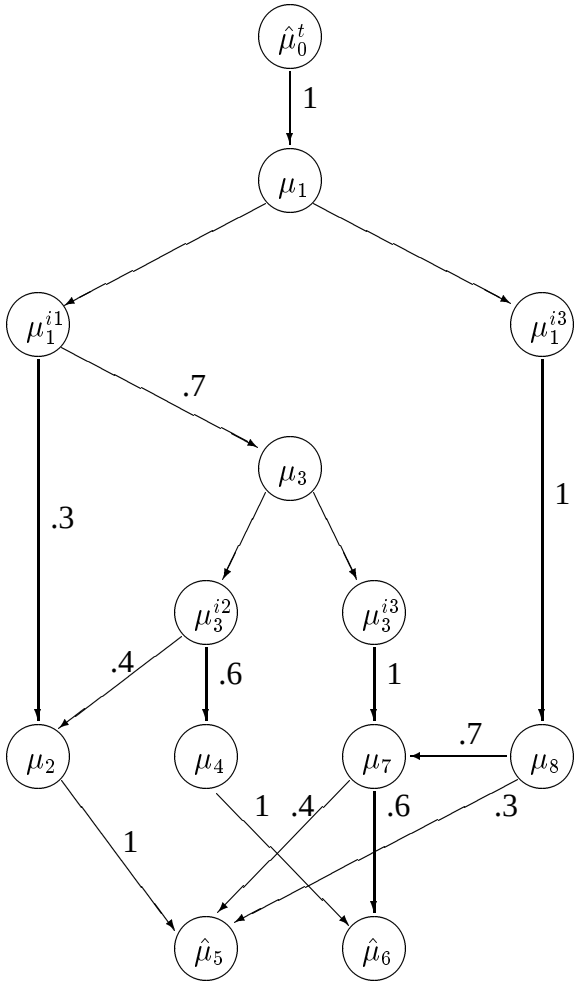

Figure 2. Configuration graph of SAN.

Table 1. Enumeration of markings.

Marking	A	B	C	D	E	F	G
μ_1	0	1	0	0	1	0	0
μ_2	0	0	0	1	1	0	0
μ_3	0	0	1	0	1	0	0
μ_4	0	0	0	0	1	0	1
$\hat{\mu}_5$	0	0	0	1	0	1	0
$\hat{\mu}_6$	0	0	0	0	0	1	1
μ_7	0	0	1	0	0	1	0
μ_8	0	1	0	0	0	1	0

3. Definitions

Before we discuss the well-defined and well-specified definitions, we present a list of symbols and definitions for the reader's convenience.

μ A marking of a SAN.

$\hat{\mu}$ A stable marking of a SAN.

$R(SAN, \mu_{\text{init}})$ Set of reachable markings in a SAN.

$SR(SAN, \mu_{\text{init}})$ Set of reachable stable markings in a SAN.

$UR(SAN, \mu_{\text{init}})$ Set of reachable unstable markings in a SAN.

$EN(\mu)$ The set of enabled activities in marking μ.

$IEN(\mu)$ The set of enabled instantaneous activities in marking μ.

$C(\mu)$ The set of children of μ in the configuration graph. (The configuration graph is described later.)

μ^a Marking μ in which a is chosen to complete. Note that $a \in IEN(\mu)$.

μ_i Marking μ_i. Subscripts on μ are used to distinguish markings. Marking μ_i is not the same as marking μ_j. However, marking μ_i is the same as marking μ_i^a.

$\hat{\mu}_0^t$ Stable marking μ_0 in which timed activity t is chosen to complete.

$P\{\hat{\mu}|\hat{\mu}_0^t\}$ The probability of reaching stable marking $\hat{\mu}$ from $\hat{\mu}_0$ by a stable step after t completes.

$P\{\mu_i|\hat{\mu}_0^t\}$ The probability of reaching unstable marking μ_i by a sequence of immediate activity completions after timed activity t completes in marking $\hat{\mu}_0$.

$P\{\hat{\mu}|\mu_i, \hat{\mu}_0^t\}$ The probability of reaching stable marking $\hat{\mu}$ from $\hat{\mu}_0$ by a stable step that includes $<\mu_i, -, ->$ after t completes in marking $\hat{\mu}_0$.

$P\{\hat{\mu}|\bar{\mu}_i, \hat{\mu}_0^t\}$ The probability of reaching stable marking $\hat{\mu}$ from $\hat{\mu}_0$ by a stable step that does not include $<\mu_i, -, ->$ after t completes in marking $\hat{\mu}_0$.

$P\{\hat{\mu}, im|\hat{\mu}_0^t\}$ The probability of reaching stable marking $\hat{\mu}$ and obtaining impulse reward im from $\hat{\mu}_0$ by a stable step that includes $<\mu_i, -, ->$ after t completes in marking $\hat{\mu}_0$.

p_{ij} $P\{\mu_j|\mu_i\}$ if $\mu_i \rightarrow \mu_j$, where $\rightarrow$ is the "yields" function. Note that $p_{ij} > 0$ or it is undefined.

$NS(\hat{\mu}_0^t)$ The set of stable markings reachable from marking $\hat{\mu}_0$ after t completes in a stable step.

$NU(\hat{\mu}_0^t)$ The set of unstable markings reachable from marking $\hat{\mu}_0$ after t completes before reaching a stable marking.

IM The set of allowable impulse rewards.

Quantified A probability is *quantified* if it can be computed without any additional information. A probability is said to be *non-quantified* if it is not quantified.

Configuration A triple $<\mu, a, c>$ representing the completion of activity a and choice of case c in marking μ.

Stable configuration The configuration $<\mu, a, c>$ is *stable* if μ is a stable marking. The configuration is *unstable* if μ is an unstable marking.

Stable step A sequence of configurations that consist of the following: first, a stable configuration; then, a number of unstable configurations; and finally, a terminating configuration that yields a stable configuration. The sequence must follow a legal execution of the SAN.

Stabilizing SAN A SAN is *stabilizing* if the number of stable steps from any stable state is finite.

3.1. Well-defined

We begin by describing the approach taken in [2], where the term well-defined is introduced. There are actually three different definitions (we believe) of well-defined in this paper. In [2], the underlying stochastic process of an SPN model is defined to include the state of the SPN, the time in which state changes occur, and the sequence of transition firings. SANs use the sequence of configurations to describe the underlying stochastic process. Although the description of the underlying stochastic process differ in SANs and GSPNs, both afford enough information to apply the well-specified or well-defined definition.

A *well-defined* GSPN is informally defined as an GSPN in which the underlying stochastic process is fully determined. Non-determinism may occur when several enabled immediate transitions belong to different extended conflict sets (ECSs). The order in which transitions fire may affect the probability of reaching some tangible state, and the order is not specified. According to the well-defined definition, sequences of transition firings, even if they happen in zero time, must always yield a completely determined underlying stochastic process. This allows for no ambiguity, which means that if more than one immediate transition is

enabled, all of them must belong to the same ECS. If applied to SANs, that would require that only one immediate activity be enabled at any one time. This has the potential to be overly restrictive.

A more relaxed definition that Ciardo defines is *well-defined with respect to a reward structure* [2]. Simply stated, this definition requires that the reward structure be fully determined, or quantified. We say that a probability is *quantified* if it can be computed without any additional information. This has some interesting side effects. For example, any SPN with no (or trivial) reward structure is well-defined with respect to that reward structure. A more important side effect of this definition is that the probability of reaching the next tangible state may not be quantified, due to ambiguity. If no impulse rewards are affected, some global knowledge about the evolution of the stochastic process and reward structure is needed in order to determine whether the ambiguity will eventually affect the reward structure. While this definition is an interesting idea, it is very difficult to check in practice.

A more practical definition of well-specified is also offered in [2], in algorithm form. This is a more restrictive form of the definition of well-defined with respect to a reward structure. It adds two additional restrictions. First, the probability of reaching a stable marking from any unstable marking must be quantified (no probability confusion). Second, the probability of obtaining an impulse reward by reaching a stable marking from any unstable marking must be quantified (no reward confusion).

To state this precisely, we introduce some notation. Let $NU(\mu_i)$ be the set of unstable markings reachable from μ_i in zero time, let $NS(\hat{\mu}_0^t)$ be the set of stable markings reachable by a stable step from marking $\hat{\mu}_0$ after t completes, and let IM be the set of all impulse rewards. A SAN meets the more restrictive definition of well-defined with respect to a reward condition if $\forall \hat{\mu}_0 \in SR(SAN, \mu_{\text{init}})$, $\forall t \in EN(\hat{\mu}_0)$, $\forall \mu_i \in NU(\hat{\mu}_0^t)$, and $\forall \hat{\mu} \in NS(\hat{\mu}_0^t)$, the following are true:

1. $P\{\hat{\mu}, |\mu_i, \hat{\mu}_0^t\}$ is quantified (no probability confusion).
2. $P\{\hat{\mu}, im|\mu_i, \hat{\mu}_0^t\}$ is quantified $\forall im \in IM$ (no reward confusion).

By $P\{\hat{\mu}, |\mu_i, \hat{\mu}_0^t\}$, we mean the probability of reaching the next stable marking $\hat{\mu}$ in the configuration graph rooted by $\hat{\mu}_0^t$ given that μ_i is visited. This is the definition of *well-defined* that we are adopting, and that we hereafter use in this paper.

The algorithm for checking if a model is well-defined involves determining the probability distribution of next stable markings at every unstable marking. If computing the distribution at each node in the configuration graph can be done in constant time, and $G = (E, V)$ is the configuration graph, then the algorithm runs in $O(E + V)$ time.

3.2. Well-specified

The definition of well-specified essentially says that the marking of the SAN and the reward variables must be completely described at all points in time. Non-quantified choices (i.e., multiple enabled instantaneous activities) that introduce some ambiguity are allowable so long as the distribution of next stable markings and accumulated reward is independent of these non-quantified choices. The definition of well-specified requires that from a stable marking, the distribution of next stable markings must be quantified. However, it does not appear to require that the distribution of next stable markings be quantified for all unstable markings, as the well-defined definition does. This has led previous attempts at designing an algorithm for the well-specified check to enumerate all possible stable steps. We show in the next section that this apparent ambiguity can not exist.

A SAN is well-specified if for any stable marking, the probability of reaching a next stable marking and obtaining an impulse reward is quantified. We say a SAN is *well-specified* if $\forall \hat{\mu}_0 \in SR(SAN, \mu_{\text{init}})$, $\forall t \in EN(\hat{\mu}_0)$, and $\forall \hat{\mu} \in NS(\hat{\mu}_0^t)$, the following are true:

1. $P\{\hat{\mu}|\hat{\mu}_0^t\}$ is quantified, and
2. $P\{\hat{\mu}, im|\hat{\mu}_0^t\}$ is quantified $\forall im \in IM$.

There is a subtle difference between the definitions of well-specified and well-defined. Let $\hat{\mu}_0^t$ be the root of a configuration graph. The well-specified definition requires that the probability of reaching the next stable marking and obtaining an impulse reward be quantified from $\hat{\mu}_0^t$, the root of the configuration graph. The well-defined definition requires that the probability of reaching the next stable marking and obtaining an impulse reward be quantified for all nodes in the configuration graph.

This distinction is important, because the algorithms derived from these two definitions are substantially different in their computational complexity. A well-specified SAN, by definition, is assumed to be capable of having some unstable marking for which the probability of reaching the next stable marking and obtaining an impulse reward is not determined. This causes the well-specified algorithm to be computationally complex, because it must enumerate all possible paths.

The algorithm to check whether a SAN is well-specified has been developed in [3], and involves enumerating and storing all possible paths (stable steps), which can be prohibitive for models with larger networks of instantaneous activities, or even for models with smaller networks that are frequently used. If $G = (E, V)$ is the configuration graph, then the algorithm can run in $O(2^E)$ time.

In the next section, we show that these two definitions are actually equivalent. Hence, the algorithm to test whether

a net is well-defined may be used to determine whether a SAN is well-specified.

4. Equivalence of definitions

4.1. Condition 1

Recall that a SAN is well-specified if $\forall \hat{\mu}_0 \in SR(SAN, \mu_{\text{init}})$, $\forall t \in EN(\hat{\mu}_0)$, and $\forall \hat{\mu} \in NS(\hat{\mu}_0^t)$, the following are true:

1. $P\{\hat{\mu}|\hat{\mu}_0^t\}$ is quantified, and
2. $P\{\hat{\mu}, im|\hat{\mu}_0^t\}$ is quantified $\forall im \in IM$.

We begin by considering the first part of the definition, and in Section 4.2 we consider the second part.

We begin by stating three rules that are evident and easily derived from the definition of SANs. Let $\hat{\mu}_0 \in SR(SAN, \mu_{\text{init}})$, $t \in EN(\hat{\mu}_0)$, and $\hat{\mu}, \hat{\mu}' \in NS(\hat{\mu}_0^t)$, $\hat{\mu} \neq \hat{\mu}'$.

Rule 1. If node μ_i is a case node, then $P\{\hat{\mu}|\mu_i, \hat{\mu}_0^t\} = \sum_{\mu_j \in C(\mu_i)} P\{\hat{\mu}|\mu_j, \hat{\mu}_0^t\} p_{ij}$.

Rule 2. $P\{\hat{\mu}|\hat{\mu}, \hat{\mu}_0^t\} = 1$.

Rule 3. $P\{\hat{\mu}|\hat{\mu}', \hat{\mu}_0^t\} = 0$.

We use the shortened notation p_{ij} to mean $P\{\mu_j|\mu_i\}$, where $\mu_i \rightarrow \mu_j$. Note that p_{ij} is always greater than zero.

Now we state our first condition.

Condition 1 *If node μ_i is a decision node, then $P\{\hat{\mu}|\mu_i^a, \hat{\mu}_0^t\} = P\{\hat{\mu}|\mu_i^b, \hat{\mu}_0^t\}$ $\forall a, b \in IEN(\mu_i)$, and $\forall \hat{\mu} \in NS(\hat{\mu}_0^t)$.*

This leads us to our first theorem.

Theorem 1 *Condition 1 is a necessary and sufficient condition for a SAN to be well-specified with respect to the first part of the well-specified definition.*

Proof: If Condition 1 holds, then $P\{\hat{\mu}|\mu_i, \hat{\mu}_0^t\}$ is quantified for all nodes μ_i in the configuration graph, so it is certainly quantified for the root node. This shows that Condition 1 is sufficient.

To show that Condition 1 is a necessary condition, we assume, for contradiction, that a well-specified SAN exists that does not meet Condition 1.

Note that all probabilities discussed in the remainder of this proof are assumed to be conditioned on $\hat{\mu}_0^t$, i.e., we write $P\{\hat{\mu}|\mu_i\}$ when we mean $P\{\hat{\mu}|\mu_i, \hat{\mu}_0^t\}$.

First, we note that the well-specified definition implies that $P\{\hat{\mu}\}$ is quantified for all $\hat{\mu} \in NS(\hat{\mu}_0^t)$. Condition 1 implies that $P\{\hat{\mu}|\mu_i\}$ is quantified for all μ_i in the configuration graph. For a contradiction, we can assume that in some configuration graph, Condition 1 does not hold for some nodes.

Based on the contradictory assumption, we can infer that there exists some node, call it μ_i, where the $P\{\hat{\mu}|\mu_i\}$ is quantified for some arbitrary $\hat{\mu} \in NS(\hat{\mu}_0^t)$, but is not quantified for some children of μ_i. (If no node μ_i exists, then our contradictory assumption does not hold.)

There are two possibilities: μ_i is either a decision node or a case node. If μ_i is a decision node, then $P\{\hat{\mu}|\mu_i^a\}$ is non-quantified for some $a \in IEN(\mu_i)$, and for some $\hat{\mu} \in NS(\hat{\mu}_0^t)$, and therefore $P\{\hat{\mu}|\mu_i\}$ is non-quantified. This violates our assumption, so μ_i is not a decision node.

Thus, μ_i must be a case node. Therefore, we can write

$$P\{\hat{\mu}|\mu_i\} = \sum_{\mu_j \in C(\mu_i)} P\{\hat{\mu}|\mu_j\} p_{ij} \, .$$

Our assumption says that for some μ_j, $P\{\hat{\mu}|\mu_j\}$ is not quantified; that is, it is dependent on choices made at (successor) decision nodes.

Before we proceed, we introduce a new term. A *decision* specifies which instantaneous activity is chosen to complete in some decision node. We write a decision for activity a at node μ_i as d_i^a. The decision d_i^a thus quantifies the choice of which activity completes in marking μ_i.

A *decision vector* D is a vector of decisions in which each decision node in the configuration graph has a decision associated with that node in D. A decision vector prescribes all non-probabilistic choices in the configuration graph, and hence quantifies $P\{\hat{\mu}|D\}$ in general, and $P\{\hat{\mu}|\mu_j, D\}$, $\forall \mu_j \in NU(\hat{\mu}_0^t)$, in particular.

We can then define $\mathcal{D}$ to be a subspace defined by a set of decision vectors such that $P\{\hat{\mu}|\mu_j, D\}$ is the same for all $D \in \mathcal{D}$. We claim that there must exist a decision vector $D' \notin \mathcal{D}$ such that $D \in \mathcal{D}$ and D' differ by only one decision. To see this, imagine the decision space as an n-dimensional, finite, discrete space, where n is the number of decision nodes. It is easy to see that there exists a sequence of vectors, starting in $\mathcal{D}$, that trace a path through the whole decision space. Each vector along the path differs from the previous vector by only one decision. Either the path traces the whole space without leaving $\mathcal{D}$, which is a contradiction, or it arrives at the sequence of vectors $D \in \mathcal{D}$ and $D' \notin \mathcal{D}$ that differ in only one decision. Let the decision in which D and D' differ be called d_d^a and d_d^b, that is, D is identical to D' except that d_d^a is an element in D and d_d^b is an element in D'. This decision occurs at node μ_d.

Now we have two decision vectors that differ by only one decision and yield different quantified probabilities. Recall that we assume that $P\{\hat{\mu}|\mu_i\}$ is quantified, but for some child nodes $\mu_j \in C(\mu_i)$, $P\{\hat{\mu}|\mu_j\}$ is not quantified. We can then write

$$P\{\hat{\mu}|\mu_i\} = \sum_{\mu_j \in C(\mu_i)} P\{\hat{\mu}|\mu_j\} p_{ij}$$

as two separate equations,

$$P\{\hat{\mu}|\mu_i\} = \sum_{\mu_j \in C(\mu_i)} P\{\hat{\mu}|\mu_j, D\} p_{ij}\,, \quad (1)$$

$$P\{\hat{\mu}|\mu_i\} = \sum_{\mu_j \in C(\mu_i)} P\{\hat{\mu}|\mu_j, D'\} p_{ij}\,. \quad (2)$$

Taking the difference between (1) and (2), we get

$$0 = \sum_{\mu_j \in C(\mu_i)} \left(P\{\hat{\mu}|\mu_j, D\} - P\{\hat{\mu}|\mu_j, D'\}\right) p_{ij}\,. \quad (3)$$

Now for a chosen $\mu_j \in C(\mu_i)$, we can condition on the event that μ_d is an intermediate node on the path from μ_j to $\hat{\mu}$. Recall that μ_d is the node at which the two decision sets differ in their decision.

$$\begin{aligned} P\{\hat{\mu}|\mu_j, D\} &= P\{\hat{\mu}|\mu_j, \mu_d, D\} P\{\mu_d|\mu_j, D\} + \\ &\quad P\{\hat{\mu}|\mu_j, \bar{\mu}_d, D\} P\{\bar{\mu}_d|\mu_j, D\} \\ &= P\{\hat{\mu}|\mu_d^a, D\} P\{\mu_d|\mu_j, D\} + \\ &\quad P\{\hat{\mu}|\mu_j, \bar{\mu}_d, D\} P\{\bar{\mu}_d|\mu_j, D\} \end{aligned} \quad (4)$$

Similarly,

$$\begin{aligned} P\{\hat{\mu}|\mu_j, D'\} &= P\{\hat{\mu}|\mu_d^b, D'\} P\{\mu_d|\mu_j, D'\} + \\ &\quad P\{\hat{\mu}|\mu_j, \bar{\mu}_d, D'\} P\{\bar{\mu}_d|\mu_j, D'\}\,. \end{aligned} \quad (5)$$

We make use of the following equalities.

$$\begin{aligned} P\{\mu_d|\mu_j, D\} &= P\{\mu_d|\mu_j, D'\} \\ P\{\hat{\mu}|\mu_j, \bar{\mu}_d, D\} &= P\{\hat{\mu}|\mu_j, \bar{\mu}_d, D'\} \\ P\{\bar{\mu}_d|\mu_j, D\} &= P\{\bar{\mu}_d|\mu_j, D'\} \\ P\{\hat{\mu}|\mu_j, \mu_d^a, D\} &= P\{\hat{\mu}|\mu_d^a, D\} \\ P\{\hat{\mu}|\mu_j, \mu_d^b, D'\} &= P\{\hat{\mu}|\mu_d^b, D'\} \end{aligned}$$

Finally, the difference between (4) and (5) becomes

$$\begin{aligned} &P\{\hat{\mu}|\mu_j, D\} - P\{\hat{\mu}|\mu_j, D'\} = \\ &\quad \left(P\{\hat{\mu}|\mu_d^a, D\} - P\{\hat{\mu}|\mu_d^b, D'\}\right) P\{\mu_d|\mu_j, D\}\,. \end{aligned}$$

Now we restate (3) and substitute.

$$\begin{aligned} 0 &= \sum_{\mu_j \in C(\mu_i)} \left(P\{\hat{\mu}|\mu_j, D\} - P\{\hat{\mu}|\mu_j, D'\}\right) p_{ij} \\ &= \sum_{\mu_j \in C(\mu_i)} \left(P\{\hat{\mu}|\mu_d^a, D\} - P\{\hat{\mu}|\mu_d^b, D'\}\right) \times \\ &\qquad P\{\mu_d|\mu_j, D\} p_{ij} \\ &= \left(P\{\hat{\mu}|\mu_d^a, D\} - P\{\hat{\mu}|\mu_d^b, D'\}\right) \times \\ &\qquad \sum_{\mu_j \in C(\mu_i)} P\{\mu_d|\mu_j, D\} p_{ij} \end{aligned}$$

By definition, $p_{ij} > 0$. For some $\mu_j \in C(\mu_i)$, $P\{\mu_d|\mu_j, D\} > 0$ because of the way we chose μ_d. Therefore, the summation sums to a value greater than zero, and the above equation can hold only if $P\{\hat{\mu}|\mu_d^a, D\} = P\{\hat{\mu}|\mu_d^b, D'\}$, which implies that $P\{\hat{\mu}|\mu_j, D\} = P\{\hat{\mu}|\mu_j, D'\}$, which is itself a contradiction.

This contradiction implies that μ_i may not be a decision node. Since μ_i is neither a case node nor a decision node, it can not exist. Thus, Condition 1 is both necessary and sufficient for a SAN to be well-specified with respect to the first part of the definition. ■

Restated, the first part of the well-specified definition says that $P\{\hat{\mu}|\hat{\mu}_0^t\}$ must be quantified for all $\hat{\mu}_0 \in SR(SAN, \mu_0)$. Condition 1 implies that a SAN is well-specified if and only if $P\{\hat{\mu}|\mu_i, \hat{\mu}_0^t\}$, $\forall \mu_i \in NU(\hat{\mu}_0^t)$ is quantified. Thus, there can be no unstable marking in which the probability of reaching some next stable marking is not quantified. Notice that this is also the first part of the well-defined definition.

Specifically, this result provides us with a condition that can be checked at each node in the configuration graph. Condition 1 holds for all nodes in the configuration graph, for all configuration graphs generated by a SAN, if and only if the SAN is well-specified with respect to the first definition of well-specified. Next, we address the second part of the definition of well-specified.

4.2. Condition 2

The second part of the definition for a SAN to be well-specified is that the probability of obtaining a certain impulse reward when entering a particular next stable marking, given that t completes in $\hat{\mu}$, is independent of non-quantified activity choices.

We define $P\{\hat{\mu}, im|\hat{\mu}_0^t\}$ to be the probability that the next stable marking from $\hat{\mu}_0^t$ is $\hat{\mu}$ and the impulse reward is im. Let IM be the set of all possible impulse reward values. Note that then $P\{\hat{\mu}|\hat{\mu}_0^t\} = \sum_{im \in IM} P\{\hat{\mu}, im|\hat{\mu}_0^t\}$.

From the definition of SANs, we can derive the following rules. Let $\hat{\mu}, \hat{\mu}' \in NS(\hat{\mu}_0^t)$, $\hat{\mu} \neq \hat{\mu}'$.

Rule 1. If μ_i^a is a case node and $im(\mu_i, a)$ is the impulse reward accumulated when activity a completes in marking μ_i, then $P\{\hat{\mu}, im|\mu_i^a, \hat{\mu}_0^t\} = \sum_{\mu_j \in C(\mu_i^a)} P\{\hat{\mu}, im - im(\mu_i, a)|\mu_j, \hat{\mu}_0^t\}$.

Rule 2.

$$P\{\hat{\mu}, im|\hat{\mu}, \hat{\mu}_0^t\} = \begin{cases} 1 & im = 0\,, \\ 0 & im \neq 0\,. \end{cases}$$

Rule 3. $P\{\hat{\mu}, im|\hat{\mu}', \hat{\mu}_0^t\} = 0$.

Note that we allow for marking-dependent impulse rewards.

To address the second part of the well-specified definition, we introduce a second condition.

Condition 2 *If node μ_i is a decision node, then $P\{\hat{\mu}, im|\mu_i^a, \hat{\mu}_0^t\} = P\{\hat{\mu}, im|\hat{\mu}_i^b, \hat{\mu}_0^t\}$ $\forall a, b \in IEN(\mu_i)$, and $\forall \hat{\mu} \in NS(\hat{\mu}_0^t)$.*

This leads us to the second theorem.

Theorem 2 *Condition 2 is a necessary and sufficient condition for a SAN to be well-specified with respect to the second part of the well-specified definition.*

Proof: The proof follows the same form as the proof for Theorem 1. ■

As we observed earlier, $P\{\hat{\mu}|\hat{\mu}_0^t\} = \sum_{im \in IM} P\{\hat{\mu}, im|\hat{\mu}_0^t\}$. Notice that the impulse rewards strictly partition the event $\{\hat{\mu}|\hat{\mu}_0^t\}$. From this, we can deduce the following corollary.

Corollary 1 *Condition 2 implies Condition 1.*

The implication of this is, naturally, that only Condition 2 needs to be checked.

Notice that the second condition implies that $P\{\hat{\mu}, im|\mu_i, \hat{\mu}_0^t\}$ is quantified (no reward confusion). This is precisely the second part of the definition of well-defined. Thus, we can state another corollary.

Corollary 2 *A SAN is well-specified if and only if it is well-defined.*

The well-specified definition seems to be a broader definition, and one might presume, as some have ([2, 3]), that some SANs may exist that would meet the well-specified condition but not meet the well-defined condition. We have shown that this is *not* the case, and that the two definitions are in fact equivalent definitions.

The direct consequence of this discovery is that the algorithm to perform the well-specified check can safely be replaced with the well-defined check. The well-defined algorithm also unwittingly performs the well-specified check.

5. Well-specified checker

5.1. Algorithm

Here we present a new algorithm for doing the well-specified check. Naturally, it is similar to the algorithm for the well-defined check [2]. The algorithm performs a depth-first search, applying the rules and the condition at each node in the configuration graph.

To describe the distribution of $NS(\hat{\mu}_0^t)$, we use a vector P that is uniquely indexed by two elements: a marking and an impulse reward. Thus, the vector entry $P(\hat{\mu}, im)$ at a node μ_i is $P\{\hat{\mu}, im|\mu_i, \hat{\mu}_0^t\}$.

In one instance, we use a vector shorthand notation $P(\mu_i, \circ) = P(\mu_i, \circ) + P'(\mu_j, \circ - im)p_{ij}$. This involves two constructs. First, there is the vector-scalar multiplication $P(\mu, \circ) = P'(\mu_i, \circ)p_{ij}$, which means simply $\forall m \in IM$, $P(\mu, m) = P(\mu_j, m)p_{ij}$. The second construct is $P(\mu, \circ) = P'(\mu_i, \circ) + P''(\mu_j, \circ - im)$, which is the vector addition. Again, it is to be interpreted as $\forall m \in IM$, $P(\mu, m) = P'(\mu_i, m) + P''(\mu_j, m - im)$. The algorithm to perform the well-specified check is presented in Figure 3.

```
Generate configuration graph.
P = well-spec-check(root)

algorithm well-spec-check(μ_i) : P
P = 0
if IEN(μ_i) = ∅                    // stable marking
    P(μ_i, 0) = 1
    return P
else if |IEN(μ_i)| = 1             // case node
    {a} = IEN(μ_i)
    im = im(μ_i, a)
    ∀μ_j ∈ C(μ_i)
        P' =well-spec-check(μ_j)
        P(μ_i, ∘) = P(μ_i, ∘) + P'(μ_j, ∘ − im)p_ij
    return P
else if |IEN(μ_i| > 1              // decision node
    μ_i^b ∈ C(μ_i)
    P =well-spec-check(μ_i^b)
    ∀μ_i^a ∈ C(μ_i)
        if P ≠well-spec-check(μ_i^a)
            Not Well Specified
    return P
end algorithm
```

Figure 3. Well-specified algorithm.

5.2. Analysis

If $G = (V, E)$ is the configuration graph, then the algorithm performs the well-specified check in $O(V + E)$ time. This is the same time it takes to do a state-space generation. Therefore, in the asymptotic sense, this algorithm for the well-specified check does not add to the state-space generation time, and therefore it is asymptotically optimal.

The previous well-specified algorithm performs the check in time proportional to the number of paths, which can be exponential in E. Thus, the two conditions provide critical insight, allowing us to build a much more efficient algorithm.

However, if one is simulating the SAN, the configuration graph need not be explicitly constructed to generate a sample path. In that sense, the algorithm is not necessarily optimal. Since a simulator does not need to compute the complete distribution of next stable markings, one might be able to use the structure of the SAN to reduce the amount of work necessary to check whether the SAN is well-specified. We leave this to future work.

6. Conclusion

Well-specified and well-defined definitions are two attempts at managing zero-timed events in stochastic Petri nets and extensions. The well-specified check, adopted in the context of SANs, previously yielded algorithms that have a time complexity that may be exponential in the number of arcs in the configuration graph.

In another context, the well-defined definition was developed. The well-defined check yielded a much more efficient algorithm, having time complexity linear in the number of arcs in the configuration graph. The apparently stricter definition seemed to be the reason a more efficient algorithm could be used.

In this paper, we introduced two conditions that must be true if a SAN is well-specified. These two conditions yield the insight necessary to deduce that the well-specified and well-defined definitions are actually equivalent definitions. Consequently, we can use the more efficient algorithm to perform the well-specified check. This algorithm is essentially the well-defined check, and since the asymptotic running time is no more than that of the state-space generator, it is asymptotically optimal for that purpose.

References

[1] G. Chiola, M. A. Marsan, G. Balbo, and G. Conte. Generalized stochastic Petri nets: A definition at the net level and its implications. *IEEE Trans. Softw. Eng.*, 19:89–107, Feb. 1993.

[2] G. Ciardo and R. Zijal. Well-defined stochastic Petri nets. In *Proc. 4th International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'96)*, pages 278–284, San Jose, California, Feb. 1996.

[3] R. German, A. van Moorsel, M. A. Qureshi, and W. H. Sanders. Algorithms for the generation of state-level representations of stochastic activity networks with general reward structures. *IEEE Trans. Softw. Eng.*, 22:603–614, Sept. 1996.

[4] M. A. Marsan, G. Balbo, G. Chiola, and G. Conte. Generalized stochastic Petri nets revisited: Random switches and priorities. In *Proc. 2nd International Workshop on Petri Nets and Performance Models (PNPM'87)*, pages 44–53, Madison, Wisconsin, Aug. 1987.

[5] M. A. Marsan, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2:93–122, 1984.

[6] J. F. Meyer, A. Movaghar, and W. H. Sanders. Stochastic activity networks: Structure, behavior, and application. In *Proc. International Workshop on Timed Petri Nets*, pages 106–115, Torino, Italy, July 1985.

[7] A. Movaghar and J. F. Meyer. Performability modeling with stochastic activity networks. In *Proc. 1984 Real-Time Systems Symposium*, pages 215–224, Austin, TX, Dec. 1984.

[8] M. A. Qureshi, W. H. Sanders, A. P. A. van Moorsel, and R. German. Algorithms for the generation of state-level representations of stochastic activity networks with general reward structures. In *Sixth International Workshop on Petri Nets and Performance Models*, pages 180–190, Durham, NC, Oct. 1995.

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