

Discrete Deterministic and Stochastic Petri Nets

Dipl.-Inform. Robert Zijal*, TECHNISCHE UNIVERSITÄT BERLIN, Berlin, Germany
Prof. Gianfranco Ciardo**, COLLEGE OF WILLIAM AND MARY, Williamsburg, USA
Prof. Günter Hommel, TECHNISCHE UNIVERSITÄT BERLIN, Berlin, Germany

Abstract

Petri nets augmented with timing specifications gained a wide acceptance in the area of performance and reliability evaluation of complex systems exhibiting concurrency, synchronization, and conflicts. The state space of time-extended Petri nets is mapped onto its basic underlying stochastic process, which can be shown to be Markovian under the assumption of exponentially distributed firing times. The integration of exponentially and non-exponentially distributed timing is still one of the major problems for the analysis and was first attacked for continuous-time Petri nets at the cost of structural or analytical restrictions. We propose a *discrete deterministic and stochastic Petri net* (DDSPN) formalism with no imposed structural or analytical restrictions where transitions can fire either in zero time or according to arbitrary firing times that can be represented as the time to absorption in a finite absorbing *discrete-time Markov chain* (DTMC). Exponentially distributed firing times are then approximated arbitrarily well by geometric distributions. Deterministic firing times are a special case of the geometric distribution. The underlying stochastic process of a DDSPN is then also a DTMC, from which the transient and stationary solution can be obtained by standard techniques. A comprehensive algorithm and some state space reduction techniques for the analysis of DDSPNs are presented, including the automatic detection of conflicts and confusions, which removes a major obstacle for the analysis of discrete-time models.

Keywords: Discrete time extended Petri nets, reward processes, conflicts and confusions.

1 Introduction

Petri nets (PNs) [12] proved to be a powerful graphical and mathematical modeling tool that allows to describe and analyze complex systems exhibiting concurrency, synchronization, and conflicts.

The ability to model timed and probabilistic behavior is essential in the field of performance and reliability evaluation. This need leads to various different extensions of the PN formalism, where the class of *stochastic Petri nets* (SPNs) gained the widest acceptance. In SPNs, firing time delays are specified by probability distributions associated to transitions. SPNs are often classified as continuous or discrete time, depending on the type of firing time distributions and on the underlying stochastic process.

Deterministic and stochastic Petri nets (DSPNs) [3] represent the most important continuous-time approach where transitions can fire either in zero time or after a constant (deterministic) or exponentially distributed time delay. The initial definition of DSPNs imposed the structural restriction that concurrent deterministic activities cannot be present. This problem was theoretically solved in [8]. However, the solution is not feasible in

* Robert Zijal was supported by the German Research Council (DFG) under grant Ho 1257/7-1.

** Gianfranco Ciardo was partially supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480.

practice because it leads to a state space explosion when a larger number of concurrent deterministic activities has to be considered.

Discrete time stochastic Petri nets [11] instead belong to the discrete approach, where transition firing times are specified by geometric distributions which approximate the exponential distribution arbitrarily well in discrete time. Other approaches having an underlying discrete-time stochastic process have been presented in [15] (Timed Petri nets) and in [10] (Generalized Timed Petri nets), but they do not achieve the modeling power of DSPNs.

The mixture of deterministic and stochastic firing times still imposes severe problems on the quantitative analysis of a time-extended PN, since the state space needs to be generated and mapped onto the basic underlying stochastic process. Our work attacks this problem by adopting a pure discrete-time approach. However, conflicts and confusions among transition firings are more likely to occur in discrete than in continuous time, since transitions are allowed to fire only at certain discrete instants of time. Thus, simultaneous firing attempts of multiple transitions, including timed ones, can take place. The detection of the sets of transitions involved in conflicts and confusions is a precondition for the correct specification of probabilistic firing weights resolving these situations. This is an important and often neglected issue especially for discrete-time models.

In [13], *Discrete time Deterministic and Stochastic Petri nets* (dtDSPNs) were introduced where transitions fire either in zero time or after a constant or geometrically distributed time delay without any structural restriction. The deterministic time delay is then modeled as a special case of the geometric distribution. In dtDSPNs, the problem of conflicts and confusions is relaxed to a certain degree by an unconventional approach. The sequentialization of simultaneously fireable timed transitions is not enforced, which leads to the elimination of confusion situations for timed transitions. The drawback of this approach is that a dtDSPN model can generate states which are not covered by the classical Petri net theory.

A more general approach was proposed in [4] with *Discrete Time Markovian SPNs* (DTMSPNs), where firing time distributions are specified by arbitrary finite absorbing DTMCs. It has been proven in [4] that the underlying stochastic process of a DTMSPN is a DTMC, provided that the modeler detects and resolves all conflicts and confusions manually, possibly a very difficult task. This drawback lead in [6] to the development of a new method for the automatic detection of conflicts and confusions applicable to all types of stochastic Petri nets. This approach is independent of structural PN properties and is solely based on the state space generation of a given model, so that only actually occurring conflicts and confusions are detected. This is not the case for the structural tests employed in continuous-time approaches, which are based on necessary, not sufficient, conditions. Thus, structural tests can lead to an overspecification of a given model resulting in a more difficult correct interpretation of obtained results measures.

The work presented in this paper combines the results of [13], [4], and [6], while removing the mentioned drawbacks of [13] and [4]. We define *discrete deterministic and stochastic Petri nets* (DDSPNs). In DDSPNs, transitions can fire either in zero time or after a time delay specified by arbitrary finite absorbing DTMCs without any structural restriction. Firing time distributions of a DDSPN include the geometric and the deterministic distribution as a special case. Any other discrete distribution that can be expressed by a finite absorbing DTMC can be freely defined, such as the discrete uniform distribution. We adapt the general approach for the automatic detection of conflicts and confusions from [6] and integrate it into the solution method for the analysis of DDSPNs. This is a

major improvement of the usability of DDSPNs, since the usual structural tests for the conflict and confusion detection cannot be applied on a discrete-time scale where timed transitions are involved whose firing time distributions are independent of structural Petri net properties. Together with the solution method, a new algorithm for the complex and non-trivial state-space generation is presented, mapping a DDSPN onto a DTMC, from which again the transient and stationary solution can be obtained using standard techniques. In addition, some state space reduction techniques for DDSPNs are proposed to relax the inherent problem of state space explosion. The DDSPN solution technique has been implemented and integrated into the software tool TimeNET [9].

Section 2 defines untimed PNs. Section 3 presents the complete DDSPN formalism. A corresponding state space reduction method is discussed in Section 4. Numerical results for two examples, obtained from the implementation, are shown in Section 5, followed by concluding remarks in Section 6.

2 The PN formalism

We recall the (extended) PN formalism (see [6]). A PN is a tuple

$$(P, T, D^-, D^+, D^\circ, \succ, g, \mu^{[0]}) \quad \text{where:}$$

- P is a finite set of *places*, which can contain tokens. A marking $\mu \in \mathbb{N}^{|P|}$ defines the number of tokens in each place $p \in P$, indicated by μ_p (when relevant, a marking should be considered a column vector). D^-, D^+, D° , and g are “marking-dependent”, that is, they are specified as functions of the marking.
- T is a finite set of *transitions*, so that $P \cap T = \emptyset$.
- $\forall p \in P, \forall t \in T, \forall \mu \in \mathbb{N}^{|P|}, D_{p,t}^-(\mu) \in \mathbb{N}, D_{p,t}^+(\mu) \in \mathbb{N}$, and $D_{p,t}^\circ(\mu) \in \mathbb{N}$ are the multiplicities of the *input arc* from p to t , the *output arc* from t to p , and the *inhibitor arc* from p to t , when the marking is μ , respectively.
- $\succ \subset T \times T$ is an acyclic (*pre-selection*) *priority relation*.
- $\forall t \in T, \forall \mu \in \mathbb{N}^{|P|}, g_t(\mu) \in \{0, 1\}$ is the *guard* for t in marking μ .
- $\mu^{[0]} \in \mathbb{N}^{|P|}$ is the *initial marking*.

Places and transitions are drawn as circles and rectangles, respectively. The number of tokens in a place is written inside the place itself (default is zero). Input and output arcs have an arrowhead on their destination, inhibitor arcs have a small circle. The multiplicity is written on the arc (default is the constant 1); a missing arc indicates that the multiplicity is the constant 0. The default value for guards is the constant 1.

Let $\mathcal{E}(\mu)$ be the set of transitions *enabled* in marking μ . A transition $t \in T$ is enabled in marking μ if, and only if, its guard evaluates to 1, its input and inhibitor arc conditions are satisfied, and no other transition with pre-selection priority over t is enabled (this is well defined because \succ is acyclic):

$$g_t(\mu) = 1 \wedge \forall p \in P, D_{p,t}^-(\mu) \leq \mu_p \wedge (D_{p,t}^\circ(\mu) > \mu_p \vee D_{p,t}^\circ(\mu) = 0) \wedge \forall u \in \mathcal{E}(\mu), u \not\succeq t.$$

If a transition $t \in \mathcal{E}(\mu)$ fires, a new marking $\mathcal{M}(t, \mu)$ is generated from μ by subtracting the *input bag* $D_{\bullet,t}^-(\mu)$ and adding the *output bag* $D_{\bullet,t}^+(\mu)$:

$$\mathcal{M}(t, \mu) = \mu - D_{\bullet,t}^-(\mu) + D_{\bullet,t}^+(\mu) = \mu + D_{\bullet,t}(\mu),$$

where $D = D^+ - D^-$ is the incidence matrix. \mathcal{M} can be extended to its reflexive and transitive closure by considering the marking reached from μ after firing a sequence of transitions. The *reachability set* is then given by $\mathcal{S} = \{\mu : \exists \sigma \in T^* \wedge \mu = \mathcal{M}(\sigma, \mu^{[0]})\}$, where T^* indicates the set of transition sequences.

3 The DDSPN formalism

Basic definitions of the DDSPN formalism are explained in Section 3.1. Sections 3.2 and 3.3 examine the DDSPN state space and introduce the concept of well-defined DDSPNs, which is needed for the correct generation of the underlying stochastic process of a DDSPN model. Finally, Section 3.4 proposes an algorithm for the reduced reachability graph generation of a well-defined DDSPN from which the underlying stochastic process can be derived and numerically analyzed.

3.1 Basic Definitions

Informally, a DDSPN is obtained by associating a *discrete-time phase distribution* (DTP) to each PN transition for the modeling of discrete firing times. A DTP is represented by a finite absorbing *discrete-time Markov chain* (DTMC) $\{X_{i\delta} | i \in \mathbb{N}\}$ with an underlying constant time-step $\delta > 0$; the states of the DTMC are referred to as *phases*, to make a clear distinction between DTP states and the overall state space of a DDSPN.

Each phase of a DTP corresponds to a possible distribution of the *remaining firing time* (RFT) of a transition. Fig. 1 shows, for instance, the DTP representations of a geometric and a discrete uniform distribution, respectively. Phase 0 is absorbing and represents the case that a RFT reached zero and the corresponding transition is allowed to fire.

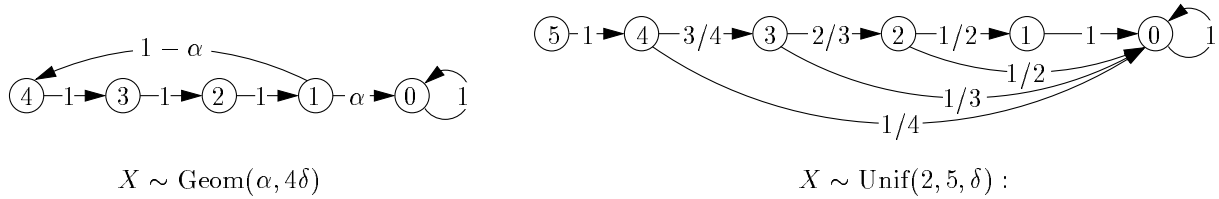


Figure 1 DTP representations of a geometric and a discrete uniform distribution.

The geometric distribution approximates the exponential distribution in discrete time arbitrarily well as δ decreases. The constant firing time can be seen as a special case of the geometric distribution, when the firing probability α is set to 1 (e.g., $\text{Const}(4\delta) = \text{Geom}(1, 4\delta)$ in Fig. 1). An immediate transition (firing in zero time) can be modeled by a DTP containing a single absorbing phase 0. A more detailed presentation of DTPs and of corresponding discrete firing time distributions can be found in [4, 14].

Consequently, a state s of a DDSPN consists of two discrete components, the marking μ and the vector ϕ containing the phase for each transition:

$$s = (\mu, \phi) \in \mathbb{N}^{|P|} \times \mathbb{N}^{|T|}.$$

Each entry ϕ_t of ϕ represents the current phase of the DTP associated to transition t .

Definition 3.1 Formally, a DDSPN is a tuple

$$(P, T, D^-, D^+, D^0, \succ, g, \mu^{[0]}, \Phi, G, F, \phi^{[0]}, \succ, C, w) \quad \text{where:}$$

- $(P, T, D^-, D^+, D^0, \succ, g, \mu^{[0]})$ defines an extended PN as introduced in Section 2.
- $\forall t \in T, \Phi_t \subset \mathbb{N}$ is the finite set of phases of the DTP associated to transition t .
- $\forall \mu \in \mathcal{S}, \forall t \in T, \forall i, j \in \Phi_t, G_t(\mu, i, j)$ is the probability that the phase of transition t changes from i to j in marking μ at the end of one time-step δ . Hence, $\sum_{j \in \Phi_t} G_t(\cdot, i, j) = 1$. G_t specifies the one-step transition probability matrix of the DTP of an enabled transition t in isolation. The phase of a disabled transition does not change in isolation: $G_t(\mu, i, i) = 1$ if $t \notin \mathcal{E}(\mu)$.

All combinations of possible new phases for all enabled transitions must be considered when ϕ is changed at the end of a step of length δ . This leads to the construction of the set $\mathcal{G}(\mu, \phi)$, such that $\forall \phi' \in \mathcal{G}(\mu, \phi)$, ϕ' is a possible combination of phases for all transitions:

$$\mathcal{G}(\mu, \phi) = \times_{t \in T} \Phi_t^G \quad \text{where} \quad \Phi_t^G = \bigcup_{\phi'_t: G_t(\mu, \phi_t, \phi'_t) > 0} \{\phi'_t\}.$$

- $\forall \mu \in \mathcal{S}, \forall t \in \mathcal{E}(\mu), \forall u \in T, \forall i, j \in \Phi_u, F_{t,u}(\mu, i, j)$ is the probability that the phase of transition u changes from i to j when transition t fires in marking μ . F is used for the specification of *race policies* [1] for transitions. See [14] for a more detailed discussion of *race policies* in DDSPNs.

Again, all combinations of possible new phases for all transitions need to be considered when ϕ is changed by the firing of t in μ leading to the construction of the set $\mathcal{F}(t, \mu, \phi)$, such that $\forall \phi' \in \mathcal{F}(t, \mu, \phi)$, ϕ' is a possible combination of phases for all transitions:

$$\mathcal{F}(t, \mu, \phi) = \times_{u \in T} \Phi_u^F \quad \text{where} \quad \Phi_u^F = \bigcup_{\phi'_u: F_{t,u}(\mu, \phi_u, \phi'_u) > 0} \{\phi'_u\}.$$

- $\forall t \in T, \phi_t^{[0]} \in \Phi_t$ is the initial phase of transition t at time 0.
- $\succ \subset T \times T$ is an acyclic post-selection priority relation.
- $C \subset 2^T$ is a partition of T into locally defined weight classes, such that:

$$\forall C_x, C_y \in C, C_x \neq C_y \Rightarrow C_x \cap C_y = \emptyset \quad \text{and} \quad \bigcup_{C_x \in C} C_x = T.$$

Let C_t be the local weight class containing transition $t \in T$. By setting $C_x = T$, we can model a global weight definition as in [4].

- $\forall \mu \in \mathcal{S}, \forall t \in \mathcal{E}(\mu), w_t(\mu) \in \mathbb{R}^+$ is the firing weight for t in marking μ .

□

In a DDSPN, a transition may only fire in a state where it is a *candidate*. For this reason, the enabling rule of Section 2 needs to be extended by the following definition.

Definition 3.2 A transition $t \in T$ is a *candidate* (to fire) in state s iff it is enabled, its phase is zero, and no other candidate has post-selection priority over it (this is well defined because \succ is acyclic). Let $\mathcal{C}(s)$ be the set of *candidates* in state $s = (\mu, \phi)$, such that: $\forall t \in \mathcal{C}(s), t \in \mathcal{E}(\mu) \wedge \phi_t = 0 \wedge (\forall u \in T, u \not\succeq t \vee u \notin \mathcal{C}(s))$ □

Consequently, the firing rule of Section 2 is extended from markings to states for DDSPNs. Transitions involved in conflicts or confusions (see Sec. 3.3) are grouped into the same weight class where the corresponding marking-dependent firing weights are used for the probabilistic resolution of such situations. Namely, the probability that transition $t \in \mathcal{C}(s)$ is chosen to fire in state $s = (\mu, \phi)$, given that one or more transitions of its weight class C_t are also candidates in s , is

$$\hat{w}_{t|\mathcal{C}(s) \cap C_t}(\mu) = \frac{w_t(\mu)}{\sum_{u \in \mathcal{C}(s) \cap C_t} w_u(\mu)}, \quad \text{such that} \quad \sum_{t \in \mathcal{C}(s) \cap C_t} \hat{w}_{t|\mathcal{C}(s) \cap C_t}(\mu) = 1.$$

Candidate transitions belonging to different weight classes have no influence on the calculation of the firing probability, so that, in DDSPNs, firing probabilities are defined only among transitions belonging to the same (locally defined) weight class. If conflicts or confusions do not exist or have not yet been recognized, each transition $t \in T$ of a DDSPN model belongs by default to its own weight class C_t , such that for any state s :

$$\forall t \in T, |C_t| = 1 \quad \text{and} \quad \forall t \in \mathcal{C}(s), \hat{w}_{t|\mathcal{C}(s) \cap C_t}(\mu) = 1.$$

Consider, for example, the DDSPN of Fig. 2(a) which contains only immediate transitions

(firing in zero time) represented by thin bars. Sharing the same input place p_1 , transitions t_1 and t_2 are conflicting in state $(\mu, \phi) = (100, 00)$ and belong therefore to the same weight class $C_{t_1} = C_{t_2} = \{t_1, t_2\}$.

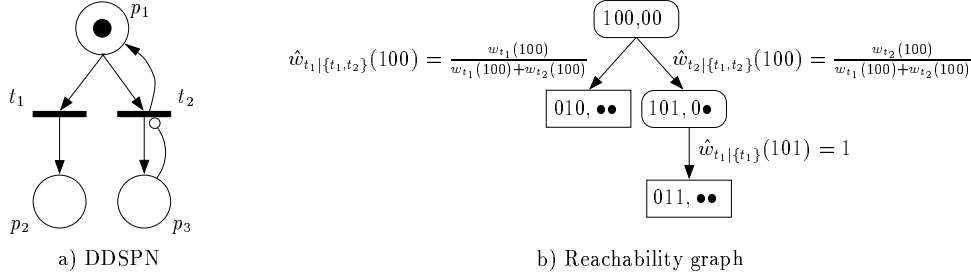


Figure 2 Computation of firing probabilities.

Fig. 2(b) shows the underlying reachability graph and the state transition probabilities computed by the corresponding firing probabilities. If t_2 fires, a token is placed on p_1 , enabling t_1 again in state $s = (101, 0\bullet)$, and a token is placed on p_3 , disabling t_2 because of the corresponding inhibitor arc. Note that the firing probability of t_1 evaluates to 1 in state s (no conflict), since $\mathcal{C}(s) = \{t_1\}$ and subsequently $\mathcal{C}(s) \cap C_{t_1} = \{t_1\}$.

3.2 The DDSPN State Space

The underlying stochastic process of a DDSPN is a DTMC

$$\{(\mu^{[k]}, \phi^{[k]}) | k \in \mathbb{N}\}$$

with state space $\mathcal{S} \subseteq \mathbb{N}^{|P|} \times \mathbb{N}^{|T|}$. The time-step of the DTMC is given by δ , such that $(\mu^{[k]}, \phi^{[k]}) \in \mathcal{S}$ is a DDSPN state at step k at time $k\delta$.

Note that this Section serves also as an introduction to the algorithm of Sec. 3.4.

We adopt the terminology of [2] and call a state s *tangible* if its sojourn time is greater zero (i.e., $\mathcal{C}(s) = \emptyset$), *vanishing* otherwise. Thus, \mathcal{S} consists only of tangible states.

Consider a tangible state $s^{[k]} = (\mu^{[k]}, \phi^{[k]})$ at step k . At the next step $k + 1$, the time must first be advanced by advancing the phases of all enabled transitions in $\phi^{[k]}$. Then, transitions whose phases were advanced to zero and which became candidates must be fired sequentially in any possible order and in zero time. This generates sequences of vanishing states which must be traversed until one or more tangible states are reached, representing the completion of the transitional stage from step k to $k + 1$. A more formal and detailed investigation of the intermediate vanishing states yields the following:

- $s^{[k+1]0} = (\mu^{[k+1]0}, \phi^{[k+1]0})$ is the first state reached from $s^{[k]}$ where
 - only the time is advanced: $\phi^{[k+1]0} \in \mathcal{G}(\mu^{[k]}, \phi^{[k]})$ and consequently
 - no firing and no change of marking occurred: $\mu^{[k+1]0} = \mu^{[k]}$.
- $s^{[k+1]i} = (\mu^{[k+1]i}, \phi^{[k+1]i})$, $i \in \{1, 2, \dots, n\}$ denotes the i -th state entered after the firing of a transition $t^i \in \mathcal{C}(s^{[k+1]i-1})$, such that
 - t^i fires: $\mu^{[k+1]i} = \mathcal{M}(t^i, \mu^{[k+1]i-1})$ and
 - race policies are applied to RFTs of transitions: $\phi^{[k+1]i} \in \mathcal{F}(t^i, \mu^{[k+1]i-1}, \phi^{[k+1]i-1})$.

After n possible firings in n vanishing states $s^{[k+1]i}$, $i = 0, \dots, n - 1$, we denote the first reachable tangible state as

$$s^{[k+1]} \stackrel{\text{def}}{=} s^{[k+1]n}.$$

Note that $s^{[k+1]0} = s^{[k+1]}$ if $\mathcal{C}(s^{[k+1]0}) = \emptyset$, that is, if no firing occurs.

We just described a single state sequence $s^* = (s^{[k+1]i} | i \in \{0, 1, \dots, n\})$ leading from $s^{[k]}$ to $s^{[k+1]}$. For better readability, let $s^{[k]} = s$ and $s^{[k+1]} = \tilde{s}$.

Following from these considerations, the set $S_{s,\tilde{s}}$ of all state sequences from $s = (\mu, \phi)$ to all possible $\tilde{s} = (\tilde{\mu}, \tilde{\phi})$ is given by

$$S_{s,\tilde{s}} = \left\{ s^* = \left(\tilde{s}^0 = (\tilde{\mu}^0, \tilde{\phi}^0), \tilde{s}^1 = (\tilde{\mu}^1, \tilde{\phi}^1), \dots, \tilde{s}^n = (\tilde{\mu}^n, \tilde{\phi}^n) \right) \mid \begin{array}{l} \forall \tilde{\phi}^0 \in \mathcal{G}(\mu, \phi), \quad \tilde{s}^0 = (\mu, \tilde{\phi}^0) \quad \text{and} \\ \forall i \in \{1, 2, \dots, n\}, \quad \forall t^i \in \mathcal{C}(\tilde{s}^{i-1}), \quad \forall \tilde{\phi}^i \in \mathcal{F}(t^i, \tilde{\mu}^{i-1}, \tilde{\phi}^{i-1}), \\ \tilde{s}^i = (\mathcal{M}(t^i, \tilde{\mu}^{i-1}), \tilde{\phi}^i) \end{array} \right\}.$$

The probability of a single state sequence $s^* \in S_{s,\tilde{s}}$ is then computed as

$$\Pr\{s^* | s^* \in S_{s,\tilde{s}} \wedge \tilde{s} \in s^*\} = g \cdot \prod_{i=1}^n (f^i \cdot F^i), \quad \text{where:}$$

- $g = \prod_{t \in T} G_t(\mu, \phi_t, \tilde{\phi}_t^0)$
is the probability for a single combination of phases $\tilde{\phi}^0 \in \mathcal{G}(\mu, \phi)$.
- for a firing transition $t^i \in \mathcal{C}(\tilde{s}^{i-1})$, such that $\tilde{\mu}^i = \mathcal{M}(t^i, \tilde{\mu}^{i-1})$:
 - $f^i = \hat{w}_{t^i | \mathcal{C}(\tilde{s}^{i-1}) \cap C_{t^i}}(\tilde{\mu}^{i-1})$ is the firing probability.
 - $F^i = \prod_{u \in T} F_{t^i, u}(\tilde{\mu}^{i-1}, \tilde{\phi}_u^{i-1}, \tilde{\phi}_u^i)$
is the probability for a single combination of $\tilde{\phi}^i \in \mathcal{F}(t^i, \tilde{\mu}^{i-1}, \tilde{\phi}^{i-1})$.

Fig. 3 shows a possible sequence of states leading from s to \tilde{s} and the involved probabilities. Instead of considering an example, we prefer a symbolical description because, compared to continuous-time approaches, the state space of DDSPNs is in general more complex and, hence, more difficult to exemplify.

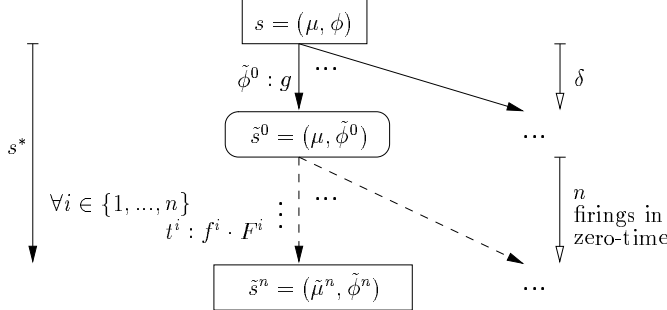


Figure 3 A sequence of states s^* leading from $s = s^{[k]}$ to $\tilde{s} = s^{[k+1]}$.

3.3 Well-defined DDSPNs

The underlying stochastic process of a DDSPN introduced in Section 3.2 takes only the tangible state space and the sojourn time in a particular state into consideration. However, for the analysis of a DDSPN, a more detailed process is needed, extending the definition of Section 3.2, to take into account the firing of transition sequences leading from one tangible state to another.

Definition 3.3 The underlying stochastic process for a DDSPN, or *basic process*, is

$$\{(\sigma^{[k]}, s^{[k]}) | k \in \mathbb{N}\}, \quad \text{where } \sigma^{[k]} = (t^1, \dots, t^n) \in T^*, \quad k > 0$$

is the k -th sequence, of $n \in \mathbb{N}$ transitions, to fire, at time $k\delta$, beginning from state $s^{[k]0}$ and reaching state $s^{[k]n} = s^{[k]}$, such that $s^{[k]i-1} \xrightarrow{t^i} s^{[k]i}$ for $i = 1, \dots, n$. ($\sigma^{[0]} = NULL$ and $s^{[k]0}$ is obtained from $s^{[k-1]}$ by advancing the time from $(k-1)\delta$ to $k\delta$). \square

In practice, we are normally interested in stochastic reward processes derived from the basic process. Without going into too much detail (see [5] for a discussion of the use of reward rates and impulses to define measures of interest), we give the following:

Definition 3.4 A stochastic process $\{Y^{[k]} \in \mathbb{R} \mid k \in \mathbb{N}^+\}$ is a *reward process* derived from the basic process through the reward structure (ρ, r) if it is defined as:

$$Y^{[k]} = \sum_{0 < j \leq k} \left(\rho(\mu^{[j-1]}) \cdot \delta + \sum_{t^i \in \sigma^{[j]}} r_{t^i}(\mu^{[j]^{i-1}}) \right)$$

where the reward rates $\rho : \mathbb{N}^{|\mathcal{P}|} \rightarrow \mathbb{R}$ describe the rate at which reward is accumulated in a particular marking and the reward impulses $r : (T \times \mathbb{N}^{|\mathcal{P}|}) \rightarrow \mathbb{R}$ describe the impulse accumulated when a particular transition is fired in a particular marking. \square

Definition 3.5 A DDSPN is *well-defined with respect to a reward structure* (ρ, r) if

$$\Pr\{Y^{[k]} = y\}$$

is completely determined by the elements of the DDSPN, where $\{Y^{[k]} \in \mathbb{R} \mid k \in \mathbb{N}^+\}$ is the reward process defined by applying the reward structure (ρ, r) to the basic process of the DDSPN. Consequently, a DDSPN is well-defined if it is well-defined with respect to all reward structures. \square

A DDSPN is free of (stochastic) conflicts and confusions if it is *well-defined*, a precondition for the feasibility of its analysis. From a stochastic point of view, there is no difference between conflict and confusion. Both express the fact that the choice of sequentialization for contemporary firing attempts of transitions leads to different reachable states or disables formerly simultaneously enabled transitions. As a consequence, different values for certain reward measures of interest may be computed; if the DDSPN does not provide a way to choose among them, the DDSPN cannot be analyzed, i.e., Def. 3.5 is not fulfilled and the DDSPN is not well-defined.

To demonstrate the need for Def. 3.5 in discrete-time, we will show that a DDSPN might not be well-defined, even if the underlying untimed PN does not contain conflicts or confusions. Fig. 4 shows a DDSPN and its underlying basic process. Transitions t_1 and t_2 , firing after a constant time c , are in different weight classes, so that the firing probability α is not defined initially. According to the classical (untimed) PN theory, both transitions are *concurrent* because the firing of t_1 does not disable t_2 and vice versa (no conflict or confusion).

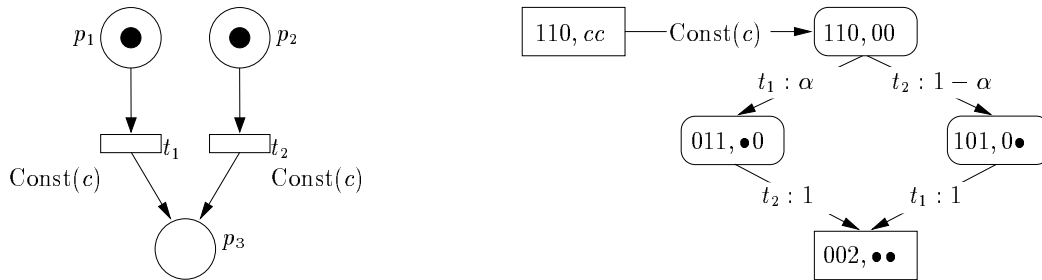


Figure 4 Concurrent but not well-defined.

Nevertheless, the DDSPN is not well-defined with respect to the reward process $Y^{[k]} =$ “the number of firings of t_1 occurring while $\#(p_2) = 1$, during the interval $(0, k\delta]$ ”. This reward process calculates two different results, each with probability one, depending on the firing sequence of t_1 and t_2 in the vanishing state $(110, 00)$ after the advance of time c . A probabilistic resolution of this non-deterministic situation is achieved by grouping both transitions into the same weight class, so that $\alpha = \hat{w}_{t_1|\{t_1, t_2\}}(110)$.

The general concept of well-defined SPNs and its implications have been extensively discussed in [6] where more details and examples can be found.

3.4 Reduced Reachability Graph Generation

In this section, we will propose an algorithm for the construction of the (finite) reduced reachability graph (RRG) and for the calculation of the impulse rewards of a well-defined DDSPN. The algorithm also tests whether the DDSPN is well-defined. The cost for this test is small, because it is based on the state space of the RRG and on the impulse reward measures. Rate rewards are not affected by conflicts and confusions, since they are calculated before any transition firing occurs. Therefore, the calculation of rate rewards is omitted, for the sake of better readability, but it can be easily included into the algorithm. Formally, the algorithm is given:

- a DDSPN $(P, T, D^-, D^+, D^o, \succ, g, \mu^{[0]}, \Phi, G, F, \phi^{[0]}, \succ, C, w)$, and
- a set of impulse reward functions $M = \{r^1, \dots, r^{|M|}\}$, where $r_t^m(\mu) \in \mathbb{R}$ is the impulse reward obtained when firing transition t in marking μ according to the m -th reward structure, $1 \leq m \leq |M|$.

If the DDSPN is well-defined with respect to all reward structures, the algorithm computes the underlying tangible state space \mathcal{S} and all state transitions $\mathcal{P}_{\mathcal{S}} = \bigcup_{s \in \mathcal{S}} \mathcal{P}_s$, such that a single *path set* \mathcal{P}_s contains all state transitions starting from state s . Hence, given that a tangible state \tilde{s} is reachable from a state s , there is a tuple $(\gamma_{\tilde{s}}, \eta_{\tilde{s}}) \in \mathcal{P}_s$ containing

- the corresponding state transition probability $\eta_{\tilde{s}} \in (0, 1]$, and
- a vector $\gamma_{\tilde{s}} = (\gamma_{\tilde{s}}^1, \dots, \gamma_{\tilde{s}}^{|M|}) \in \mathbb{R}^{|M|}$, which stores the accumulated reward value $\gamma_{\tilde{s}}^m$, for every impulse reward function $r^m \in M$.

A single $(\gamma_{\tilde{s}}, \eta_{\tilde{s}}) \in \mathcal{P}_s$ also represents the aggregated individual probabilities and accumulated impulse rewards of possible multiple paths along vanishing states leading from s to \tilde{s} . The nonzero entries of the one-step transition probability matrix \mathbf{P} for the underlying DTMC of a DDSPN are then given by: $\forall s, \tilde{s} \in \mathcal{S}, \forall \mathcal{P}_s \in \mathcal{P}_{\mathcal{S}}, \forall (\gamma_{\tilde{s}}, \eta_{\tilde{s}}) \in \mathcal{P}_s : \mathbf{P}_{s, \tilde{s}} = \eta_{\tilde{s}}$. If the expected accumulated impulse rewards up to time $k\delta$, $E[Y^{[k]} \mid s^{[k]} = s]$ are known, the expected accumulated impulses up to time $(k+1)\delta$ are given by

$\forall k \in \mathbb{N}, \forall s, \tilde{s} \in \mathcal{S}, \forall \mathcal{P}_s \in \mathcal{P}_{\mathcal{S}} :$

$$E[Y^{[k+1]} \mid (s^{[k]} = s \wedge s^{[k+1]} = \tilde{s})] = E[Y^{[k]} \mid s^{[k]} = s] + \begin{cases} \gamma_{\tilde{s}} & \text{if } \exists (\gamma_{\tilde{s}}, \eta_{\tilde{s}}) \in \mathcal{P}_s, \\ 0 & \text{otherwise.} \end{cases}$$

Standard numerical methods (power method, SOR) can be employed for the transient or stationary solution of the processes of interest.

If the DDSPN is not well-defined, the algorithm issues an error message and needs to be restarted after a conflict or confusion situation has been resolved by the means of priority or weight definitions. See [6] for a more detailed discussion of non-well-defined SPNs and their implications. Briefly, the algorithm consists of two procedures:

- In “generateRRG” (see Fig. 5(a)), the time is advanced in a given tangible state $s^{[k]} = s$ leading to $s^{[k+1]0} = \tilde{s}^0$.
- In “traverse” (see Fig. 6(a)), subsequent vanishing states are recursively traversed starting from $s^{[k+1]0} = \tilde{s}^0$ until tangible states $s^{[k+1]} = \tilde{s}$ are reached.

Three types of parameters exist for the procedure headers of Fig. 5(a) and Fig. 6(a): *call by value* (in), *call by reference* (out), and *call by value-reference* (inout).

Time-Advancement

The algorithm is exercised with the call “generateRRG($\mathcal{S}, \mathcal{P}_{\mathcal{S}}$)”. The set \mathcal{S}^{next} contains the tangible states which have not yet been visited. It is assumed that the initial state to be visited $(\mu^{[0]}, \phi^{[0]})$ is tangible (see [14] for a vanishing initial state).

Fig. 5(b) outlines the execution of “generateRRG”. The while-loop of the procedure visits

```

procedure generateRRG( out:  $\mathcal{S}, \mathcal{P}_{\mathcal{S}}$  )
 $\mathcal{S} = \emptyset; \mathcal{P}_{\mathcal{S}} = \emptyset;$ 
 $\mathcal{S}^{next} = (\mu^{[0]}, \phi^{[0]});$ 
while  $\mathcal{S}^{next} \neq \emptyset$  do
  choose a state  $s = (\mu, \phi)$  from  $\mathcal{S}^{next};$ 
   $\mathcal{S}^{next} = \mathcal{S}^{next} \setminus \{s\};$ 
   $\mathcal{P}_s = \emptyset;$ 
  foreach  $\tilde{\phi}^0 \in \mathcal{G}(\mu, \phi)$  do
     $\tilde{s}^0 = (\mu, \tilde{\phi}^0);$ 
     $g = \prod_{t \in T} G_t(\mu, \phi_t, \tilde{\phi}_t^0);$ 
    if  $\mathcal{C}(\tilde{s}^0) = \emptyset$  then    #  $\tilde{s}^0$  IS TANGIBLE
      if  $\tilde{s}^0 \notin \mathcal{S}$  then
         $\mathcal{S} = \mathcal{S} \cup \{\tilde{s}^0\}; \mathcal{S}^{next} = \mathcal{S}^{next} \cup \{\tilde{s}^0\};$ 
         $\mathcal{P}_s^0 = \{(\gamma_{\tilde{s}^0}, g) \mid \gamma_{\tilde{s}^0} = 0\};$ 
      else    #  $\tilde{s}^0$  IS VANISHING
         $\text{traverse}(\tilde{s}^0; \mathcal{S}, \mathcal{S}^{next}; \mathcal{P}_{\tilde{s}^0});$ 
         $\mathcal{P}_s^0 = \bigcup_{(\gamma_{\tilde{s}}, \eta_{\tilde{s}}) \in \mathcal{P}_{\tilde{s}^0}} \{(\gamma_{\tilde{s}} \cdot g, \eta_{\tilde{s}} \cdot g)\};$ 
       $\mathcal{P}_s = \text{unify}^-(\mathcal{P}_s, \mathcal{P}_s^0);$ 
     $\mathcal{P}_{\mathcal{S}} = \mathcal{P}_{\mathcal{S}} \cup \mathcal{P}_s;$ 
end procedure

```

a)

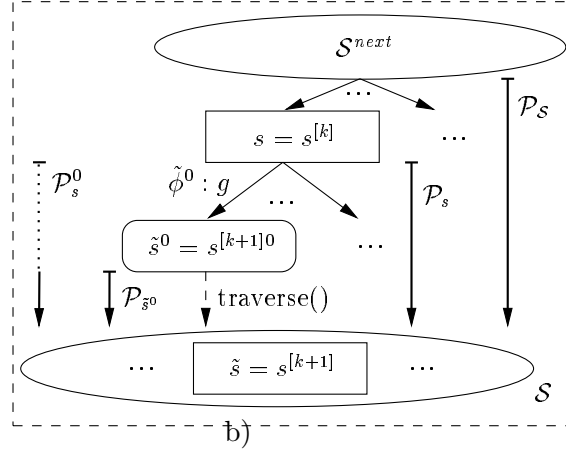


Figure 5 Generation of the reduced reachability graph.

all states of \mathcal{S}^{next} and calculates the set of paths \mathcal{P}_s , accumulated for every iteration in $\mathcal{P}_{\mathcal{S}}$, for each state $s \in \mathcal{S}^{next}$. The for-loop advances the time for one step δ by generating, with every iteration, the new state \tilde{s}^0 with probability g depending on its possible combination of next phases $\tilde{\phi}^0 \in \mathcal{G}(\mu, \phi)$. Moreover, for every \tilde{s}^0 , it generates the set of paths \mathcal{P}_s^0 leading from s to tangible states via \tilde{s}^0 and unifies them afterwards in the path set \mathcal{P}_s covering all existing state transitions originating in s .

Hence, if \tilde{s}^0 is tangible, it is added to the sets of tangible states \mathcal{S} and \mathcal{S}^{next} , if not already there, and a single initial direct path \mathcal{P}_s^0 to \tilde{s}^0 with probability g is created with no impulse rewards, since no transition firing lead to \tilde{s}^0 . If \tilde{s}^0 is vanishing, the call “ $\text{traverse}(\tilde{s}^0; \mathcal{S}, \mathcal{S}^{next}; \mathcal{P}_{\tilde{s}^0})$ ” computes the path set $\mathcal{P}_{\tilde{s}^0}$ (zero-time state transitions starting from \tilde{s}^0) from which \mathcal{P}_s^0 is afterwards obtained by multiplying all impulse rewards and path probabilities of $\mathcal{P}_{\tilde{s}^0}$ with the probability g of reaching \tilde{s}^0 from s .

The function $\text{unify}^-(\mathcal{P}_s, \mathcal{P}'_s)$ unifies two different path sets, whose origin lies in the same state s , in a way that possible multiple paths to equal tangible states are merged.

Recursive Traversal of Vanishing States

Fig. 6(b) outlines the execution of “ traverse ”. The first for-loop of the procedure in Fig. 6(a) partitions all candidate transitions of \tilde{s}^{i-1} into sets of candidate transitions \hat{C}_x belonging to the same weight class C_x . The second for-loop fires all transitions of a particular set \hat{C}_x , so that, with every iteration, a single candidate transition $t^i \in \hat{C}_x$ is fired in marking $\tilde{\mu}^{i-1}$ according to its firing probability f^i leading to $\tilde{\mu}^i$. For each firing of transition t^i , the third for-loop applies the corresponding race policies to all phases of $\tilde{\phi}^{i-1}$ and generates, with every iteration, the new state $\tilde{s}^i = (\tilde{\mu}^i, \tilde{\phi}^i)$ with probability F^i depending on the possible combination of next phases $\tilde{\phi}^i \in \mathcal{F}(t^i, \tilde{\mu}^{i-1}, \tilde{\phi}^{i-1})$. Analogously to the for-loop of “ generateRRG ”, it first generates, for every \tilde{s}^i , the set of paths $\mathcal{P}_{\tilde{s}^{i-1}}^i$ leading from \tilde{s}^{i-1} to tangible states via \tilde{s}^i and unifies them afterwards in the path set $\mathcal{P}_{\tilde{s}^{i-1}}^i$ covering all existing state transitions initiated by firing transitions of \hat{C}_x in \tilde{s}^{i-1} . Again, if \tilde{s}^i is tangible (terminating recursive calls), it is added to the sets of tangible states \mathcal{S} and \mathcal{S}^{next} , if not already there. Moreover, a single initial direct path $\mathcal{P}_{\tilde{s}^{i-1}}^i$ with probability $f^i F^i$ (i.e.,

procedure `traverse`(**in**: $\tilde{s}^{i-1} \equiv (\tilde{\mu}^{i-1}, \tilde{\phi}^{i-1})$; **inout**: $\mathcal{S}, \mathcal{S}^{next}$; **out**: $\mathcal{P}_{\tilde{s}^{i-1}}$)

$\mathcal{P}_{\tilde{s}^{i-1}} = \emptyset$;

foreach $C_x \in C$ **do**

$\hat{C}_x = \mathcal{C}(\tilde{s}^{i-1}) \cap C_x$; $\mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_x} = \emptyset$;

foreach $t^i \in \hat{C}_x$ **do** # FIRE CANDIDATE

$\tilde{\mu}^i = \tilde{\mu}^{i-1} - D_{\bullet, t^i}^-(\tilde{\mu}^{i-1}) + D_{\bullet, t^i}^+(\tilde{\mu}^{i-1})$;

$f^i = \hat{w}_{t^i | \hat{C}_x}(\tilde{\mu}^{i-1})$;

foreach $\tilde{\phi}^i \in \mathcal{F}(t^i, \tilde{\mu}^{i-1}, \tilde{\phi}^{i-1})$ **do**

APPLY RACE POLICIES

$\tilde{s}^i = (\tilde{\mu}^i, \tilde{\phi}^i)$;

$F^i = \prod_{u \in T} F_{t^i, u}(\tilde{\mu}^{i-1}, \tilde{\phi}_u^{i-1}, \tilde{\phi}_u^i)$;

if $\mathcal{C}(\tilde{s}^i) = \emptyset$ **then** # \tilde{s}^i IS TANGIBLE

if $\tilde{s}^i \notin \mathcal{S}$ **then**

$\mathcal{S} = \mathcal{S} \cup \{\tilde{s}^i\}$; $\mathcal{S}^{next} = \mathcal{S}^{next} \cup \{\tilde{s}^i\}$;

$\mathcal{P}_{\tilde{s}^{i-1}}^i = \{(\gamma_{\tilde{s}}, f^i F^i) \mid \forall m \in \{1, \dots, |M|\},$
 $\gamma_{\tilde{s}}^m = r_{t^i}^m(\tilde{\mu}^{i-1}) f^i F^i\}$;

else # \tilde{s}^i IS VANISHING

`traverse`(\tilde{s}^i ; $\mathcal{S}, \mathcal{S}^{next}$; $\mathcal{P}_{\tilde{s}^i}$);

$\mathcal{P}_{\tilde{s}^{i-1}}^i = \bigcup_{(\gamma_{\tilde{s}}, \eta_{\tilde{s}}) \in \mathcal{P}_{\tilde{s}^i}} \{(\gamma_{\tilde{s}}', \eta_{\tilde{s}}' f^i F^i) \mid \forall m \in \{1, \dots, |M|\},$
 $\gamma_{\tilde{s}}'^m = (\gamma_{\tilde{s}}^m + r_{t^i}^m(\tilde{\mu}^{i-1}) \eta_{\tilde{s}}) f^i F^i\}$;

$\mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_x} = \text{unify}^-(\mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_x}, \mathcal{P}_{\tilde{s}^{i-1}}^i)$;

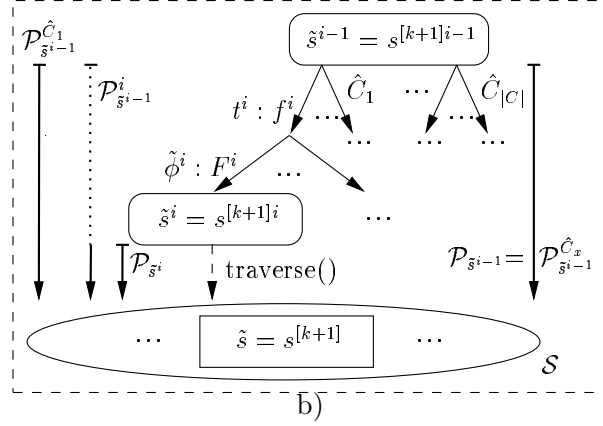
if $\mathcal{P}_{\tilde{s}^{i-1}} = \emptyset$ **then** $\mathcal{P}_{\tilde{s}^{i-1}} = \mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_x}$;

else if $\mathcal{P}_{\tilde{s}^{i-1}} \neq \mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_x}$ **then**

stop; # ERROR, DDSPN NOT WELL-DEFINED

end procedure

a)



b)

Figure 6 Traversing recursively vanishing states.

$\Pr\{\tilde{\mu}^i\}\Pr\{\tilde{\phi}^i\}$) for reaching the tangible \tilde{s}^i is created together with instantaneous impulse rewards gained by the firing of t^i in $\tilde{\mu}^{i-1}$ leading to \tilde{s}^i . If \tilde{s}^i is vanishing, the subsequently reachable states are explored by the recursive call “`traverse`(\tilde{s}^i ; $\mathcal{S}, \mathcal{S}^{next}$; $\mathcal{P}_{\tilde{s}^i}$)” that computes the path set $\mathcal{P}_{\tilde{s}^i}$ (assuming that the vanishing reachability graph created from \tilde{s}^i is acyclic, finite, and that no conflict or confusion occurred). $\mathcal{P}_{\tilde{s}^{i-1}}^i$ is then obtained from $\mathcal{P}_{\tilde{s}^i}$ by adding the instantaneous impulse rewards of t^i to $\mathcal{P}_{\tilde{s}^i}$ and by adjusting all probabilities — for reaching a tangible state \tilde{s} from \tilde{s}^{i-1} via \tilde{s}^i — so that the probability of a particular path and of the accumulated impulse rewards equals correctly to $\eta_{\tilde{s}}' = f^i F^i \eta_{\tilde{s}}$.

Integrated Detection of Conflicts and Confusions

Conflicts and confusions exhibit a non-deterministic behavior that, for DDSPNs, can occur only in a vanishing state \tilde{s}^{i-1} when multiple candidate transitions $t \in \mathcal{C}(\tilde{s}^{i-1})$ attempt to fire in zero time leading to tangible states \tilde{s} with different stochastic results.

Indeed, the DDSPN evolution during instants of time where there is no firing is completely determined by the assumption of a race behavior.

Yet, conflicts and confusions cannot arise in \tilde{s}^{i-1} among candidate transitions of the same weight class \hat{C}_z , because weights are always defined within a given weight class, so that:

- Contemporary firing attempts are resolved probabilistically by the individual firing probability f^i for each candidate $t^i \in \hat{C}_z$, where: $\sum_{t^i \in \hat{C}_z} f^i = 1$.
- All path probabilities are known, resulting in: $\sum_{(\gamma_{\tilde{s}}, \eta_{\tilde{s}}) \in \mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_z}} \eta_{\tilde{s}} = 1$.

Therefore, conflicts and confusions can occur only among candidate transitions belonging to different weight classes. Consequently, a DDSPN is not well-defined if a vanishing state \tilde{s}^{i-1} is encountered — at the end of the first loop of “traverse” — with at least two differing path sets, such that: $\mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_x} \neq \mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_y}$, where $\hat{C}_x \neq \hat{C}_y$.

To remove conflicts or confusions, the modeler must then apply one of the following actions to transitions of \hat{C}_x and \hat{C}_y , before restarting the algorithm:

- Priorities can be specified to prevent conflicting transitions from becoming simultaneous candidates, hence from attempting to fire at the same time:
 - Pre-selection priorities disable a conflicting transition before the advance of time.
 - Post-selection priorities enforce a particular sequence for contemporary firings.
- Merge the corresponding weight classes C_x and C_y and (re)define appropriate weights for them. Hence, candidate transitions, involved in conflicts or confusions, are grouped into the same weight class $C_z = C_x \cup C_y$.

If no further warnings are produced by the algorithm, the DDSPN is well-defined because the following necessary condition for the absence of conflicts and confusions was fulfilled:

$$\forall C_x, C_y \in C : \mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_x} = \mathcal{P}_{\tilde{s}^{i-1}}^{\hat{C}_y} = \mathcal{P}_{\tilde{s}^{i-1}}.$$

That is, candidate transitions, belonging to different weight classes, reach from all vanishing states \tilde{s}^{i-1} the same tangible states with the same probabilities and with the same accumulated rewards, regardless of the order in which they are fired.

4 State Space Reduction

In Sec. 3.4, it has been shown how the one-step state transition probability matrix \mathbf{P} of the underlying (finite) DTMC of a DDSPN is obtained from its RRG where the holding time in each state of the DTMC is equal to the basic underlying time-step δ .

A considerable reduction of state space can be achieved during the generation of the RRG if tangible states are encountered where it is possible to advance the phases of all enabled transitions for a multiple of δ until a probabilistic split or a phase equal to zero (vanishing state) is reached — a condition often met by deterministic transitions.

In other words, a sequence of intermediate tangible states with state transition probabilities equal to one can be omitted if it was generated solely by single phase advancements of one δ with probabilities equal to one, so that no change of marking occurred (see [14] for more details). Informally, only the phase vectors of a sequence of tangible states are discarded, since these states are not needed anymore for the further computation of the underlying stochastic process of a particular DDSPN. The corresponding equal markings are merged into one, while preserving their individual holding times, so that the computation of all measures of interest is still possible¹.

Therefore, fewer tangible states are generated, resulting in a smaller state transition probability matrix $\hat{\mathbf{P}}$ where the holding time in a particular state s is no longer equal to one δ , but to $h_s = x\delta$, $x \in \mathbb{N}^+$. Now, the holding time of each state contains the holding times of omitted tangible states where the markings were the same.

This approach of state space reduction is named *embedding* with reference to the corresponding underlying stochastic process of the DDSPN which is now a discrete-time semi-Markov process where $\hat{\mathbf{P}}$ and $h \in \mathbb{R}^{+|\mathcal{S}|}$ describe an *embedded* DTMC obtained from

¹ Phase vector components of tangible states influence only the temporal evolution of DDSPNs and are not required for the computation of measures of interest.

the *embedded* RRG. The size of the state space depends in general on

- the size and structure of a particular DDSPPN.
- the size of the basic underlying time-step δ .
- the number of phases of firing time distributions (DTPs) specified for the timed transitions of a DDSPPN model.

A small time-step δ and DTPs with a large number of phases lead to a fast growth of the state space. Thus, a careful choice of timing parameters helps reducing the state space, too. If embedding is used, the size of the state space depends, in addition, on the maximum possible phase advancements with probability one of all enabled transitions in tangible states.

Since \mathbf{P} and $\hat{\mathbf{P}}$ are usually sparse matrices, sparse storage schemes should be employed. If \mathbf{P} is an irreducible DTMC, the stationary probability distribution over \mathcal{S} , given by the vector $\pi \in [0, 1]^{|\mathcal{S}|}$, is obtained by solving a system of linear equations with standard techniques (Gauss-Seidel, SOR): $\pi = \pi\mathbf{P}$ and $\sum_i \pi_i = 1$.

In case of embedding, the stationary solution can be computed from $\hat{\mathbf{P}}$ and h employing the following well-known method for semi-Markov processes [7]:

- We first solve the system of linear equations $\varpi = \varpi\hat{\mathbf{P}}$ and $\sum_i \varpi_i = 1$ for the embedded stationary probabilities ϖ .
- Then, we rescale ϖ using the holding times: $\forall s \in \mathcal{S}, \varpi'_s = \varpi_s \cdot h_s$.
- Finally, we normalize the rescaled probabilities ϖ' and obtain the stationary probability distribution: $\pi = \frac{\varpi'}{\sum_i \varpi'_i}$.

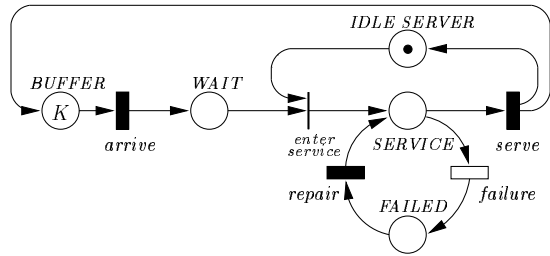
Subsequently, measures of interest can be derived from π .

5 Examples

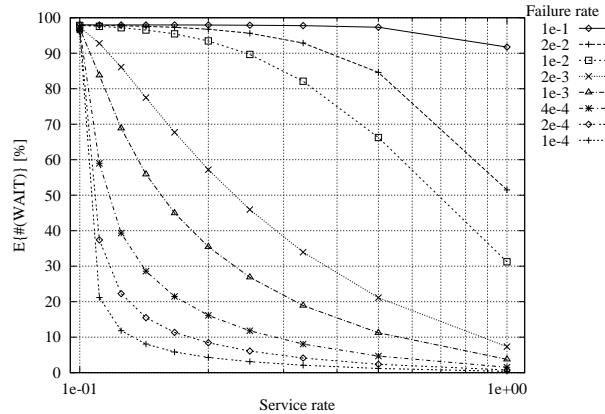
This section illustrates the modeling power of DDSPPNs by presenting an example containing several deterministically timed activities. Consider the processing station of an automated manufacturing system where raw parts arrive at constant time intervals. A machine tool processes each raw part for a constant time period. The tool wears off and needs to be replaced after a stochastically timed delay whose value depends on the tool quality and on the material of the processed parts. The time delay for the replacement is constant. The processing station can then be characterized by a D/D/1/K queuing system where the service station (tool) is subject to stochastic failures (wearout) and deterministically timed repairs (tool replacement).

Fig. 7(a) shows the corresponding DDSPPN model. Raw parts, represented by tokens, arrive with the firing of the deterministic transition *arrive* and wait for service in place *WAIT* until the service station is empty and operable. A single token in place *IDLE SERVER* and the immediate transition *enter service* permit only one part at a time to enter the service station which consists of the place *SERVICE* and of the deterministic transition *serve*. The firing of *serve* stands for the completion of the processing of a single part. The failure and repair of the service station are represented by the geometric and deterministic transitions *failure* and *repair*, respectively.

We consider a system with $K = 50$ parts, a constant deterministic arrival rate of $\lambda = \frac{1}{10sec}$, and a constant deterministic repair rate of $\frac{1}{10^3sec}$. The deterministic service rate ν is varied from $\frac{1}{10sec}$ to $\frac{1}{1sec}$ and the geometric failure rate is varied from $\frac{1}{10sec}$ to $\frac{1}{10^4sec}$. The basic underlying time-step of the model equals to $\delta = 1sec$. The measure of interest of the stationary solution is the average number of waiting raw parts $E\{\#(WAIT)\}$ in place *WAIT*



a) DDSPN model



b) Numerical results

Figure 7 D/D/1/K queuing system with failure and repair.

depending on the varying service and failure rates. The goal of our performance evaluation is to determine which minimum performance of the server, in terms of speed (service rate) and dependability (failure rate), suffices to achieve a desired average percentage of waiting raw parts. Fig. 7(b) shows the corresponding curves, where $E\{\#(WAIT)\}$ (in %) is plotted vs. the firing rates of transition *serve* and transition *failure*.

The state space of the DDSPN consists of 101 tangible markings. Depending on the deterministic service rate, from 5,930 to 125,153 tangible states are generated and the evaluation time varies from 9 to 56 seconds, respectively, on a *Sun Sparc Station-20 (Solaris)*. However, employing the embedding technique for the stationary solution of this particular model leads to a state space reduction of 86.4% to 90.8% and the evaluation time varies from 7 to 17 seconds, again depending on the deterministic service rate.

To some limited degree, DDSPNs can be used also for the approximation of continuous-time models containing exponential and deterministic transitions without structural restrictions; i.e., exponentially distributed times are approximated by geometrically distributed ones. A familiar continuous-time method is to use Erlang- κ distributions for the approximation of constant firing times.

To compare the Erlang- κ and our approach of DDSPNs, we choose a simple M/D/1/K queuing system for which an exact numerical stationary solution can be computed [3].

Δ_{rel} of $E\{\#(WAIT)\}$	DDSPN: δ , memory, CPU time			Erlang: κ phases, memory, CPU time		
10 %	1/2 sec	1.6 MB	1 sec	10	3.0 MB	7 sec
5 %	1/4 sec	1.7 MB	1 sec	20	7.3 MB	34 sec
2 %	1/10 sec	2.0 MB	3 sec	50	36 MB	5 min
1 %	1/20 sec	2.7 MB	4 sec	80	90 MB	16 min
0.1 %	1/200 sec	13 MB	58 sec	not available		

Table 1 Computational effort for the approximation of a M/D/1/K queuing system.

The system can be obtained from Fig. 7(a) by removing the failure/repair subnet and by assuming that transition *arrive* is exponential. Let $K = 50$, $\lambda = \frac{1}{10sec}$, and $\nu = \frac{1}{5sec}$. Table 1 shows the computational efforts to achieve a certain approximation accuracy measured in terms of the relative error Δ_{rel} . It can be observed that our approach achieves better results with less computational effort for the particular model under consideration.

6 Conclusion

The results in [4], [6], and [13] have been combined, introducing the DDSPN formalism where deterministic and stochastic firing times of transitions can be mixed without structural restrictions while providing integrated automatic conflict and confusion detection

on a discrete-time scale. A new solution method combining [4] and [6] and a previously not available algorithm for mapping a DDSPN onto its underlying stochastic process have been presented from which a direct implementation followed. Thus, an additional tool component for TimeNET [9] has been developed with new features based on discrete time as demonstrated for a typical queuing application example. Considerable state space reduction can be achieved for a given DDSPN model by carefully choosing timing parameters and, more importantly, by means of embedding. Even so, the DDSPN formalism still leads to a large state space due to the additional phase components in the state.

References

- [1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of Stochastic Petri Nets. *IEEE Trans. on Softw. Eng.*, 15(7):832–846, July 1989.
- [2] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of Generalized Stochastic Petri Nets for the performance analysis of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(1):93–122, May 1984.
- [3] M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In G. Rozenberg, editor, *Advances in Petri Nets 1987, Lecture Notes in Computer Science*, vol. 266, pp. 132–145. Springer, 1987.
- [4] G. Ciardo. Discrete-time Markovian Stochastic Petri Nets. In W. J. Stewart, editor, *Numerical Solution of Markov Chains '95*, pp. 339–358, Raleigh, NC, USA, Jan. 1995.
- [5] G. Ciardo, A. Blakemore, P. Chimento, J. Muppala, and K. Trivedi. Automated generation and analysis of Markov reward models using Stochastic Reward Nets. In C. Meyer and R. J. Plemmons, editors, *Linear Algebra, Markov Chains, and Queueing Models*, vol. 48 of *IMA Volumes in Mathematics and its Applications*, pp. 145–191. Springer, 1993.
- [6] G. Ciardo and R. Zijal. Well-defined Stochastic Petri Nets. In *Proc. 4th Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'96)*, pp. 278–284, San Jose, CA, USA, Feb. 1996. IEEE Comp. Soc. Press.
- [7] E. Çinlar. *Introduction to Stochastic Processes*. Prentice-Hall, 1975.
- [8] R. German. *Analysis of Stochastic Petri Nets with Non-Exponentially Distributed Firing Times*. PhD thesis, Technical University of Berlin, Berlin, Germany, 1994.
- [9] R. German, C. Kelling, A. Zimmermann, and G. Hommel. TimeNET – a toolkit for evaluating non-Markovian Stochastic Petri Nets. *Perf. Evaluation*, 24:69–87, 1995.
- [10] M. Holliday and M. Vernon. A Generalized Timed Petri Net model for performance analysis. *IEEE Trans. on Softw. Eng.*, 13(12):1297–1310, Dec. 1987.
- [11] M. Molloy. Discrete time Stochastic Petri Nets. *IEEE Trans. on Softw. Eng.*, 11(4):417–423, April 1985.
- [12] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, Bonn, West Germany, 1962.
- [13] R. Zijal. Discrete Time Deterministic and Stochastic Petri Nets. In G. Hommel, editor, *Proc. Int. Workshop - Quality of Communication-Based Systems*, pp. 123–136, TU-Berlin, Germany, Sept. 1994. Kluwer.
- [14] R. Zijal and G. Ciardo. Discrete Deterministic and Stochastic Petri Nets. ICASE Technical Report 96-72, Institute for Computer Applications in Science and Engineering, NASA/Langley Research Center, Hampton, VA, 1996.
- [15] W. Zuberek. Timed Petri Nets, definitions, properties, and applications. *Microelectronics and Reliability*, 31(4):627–644, 1991.