Quantitative
Modeling
In
Parallel
Systems

Part 1
Immediate Transitions in Parametric Stochastic Petri Nets

Giovanni Chiola, Susanna Donatelli, and Giuliana Franceschinis
Università di Torino

Approved by: Gianfranco Balbo

Abstract Stochastic Petri nets (SPNs) are a modelling formalism in which state changes are represented by a set of nodes called transitions. State changes usually take places in finite time intervals and in SPNs transitions are associated with exponentially distributed stochastic delays. Generalized SPNs (GSPNs) allow the use of immediate transition as well to model state changes that take negligible amounts of time to occur. SPNs and GSPNs are compared in order to investigate the actual role of immediate transitions. The comparison is based on the ability of the two formalisms to define models that can be "scalable" with respect to the initial marking. An intermediate model between (G)SPN structures and (G)SPN systems called "family" is used as a reference for the comparison. It is shown that GSPN families exist that do not have SPN counter-parts, so that the presence of immediate transitions makes GSPN models more scalable. An algorithm is also given to reduce GSPN to SPN families in some restricted cases. The application of this structural reduction algorithm (when possible) allows a more efficient generation of the state space of GSPN systems.

1.1 Introduction

Scientific results may be developed using different approaches: a mathematical approach is usually driven by sequences of theoretical results, while an engineering approach can be driven by specific application demands. Generalized Stochastic Petri Nets (GSPNs) were introduced following an engineering approach driven by the necessity of developing modelling and performance evaluation techniques and tools adequate for the increasing complexity of (multiprocessor) computer systems design [1, 2]. As a result of their origin GSPNs were initially used in a very pragmatic way as a kind of graphic, compact, implicit representation for continuous-time Markov chain models. Little attention was payed to the "mathematical beauty" of the underlying qualitative modelling tool offered by the Petri net formalism [3]. Not only classical validation techniques based on structural analysis but also some subtle problems in the stochastic model definitions received initially little consideration, such as for example the possibility of producing confused model descriptions [4, 5]. After recognizing all these problems a question rises spontaneous: is the introduction and use of immediate transitions worth the complication of SPN analysis and validation? This paper is an attempt to provide some a-posteriori mathematical consideration that justify the definition of GSPNs from a theoretical as well as from a practical point of view.

1.1.1 The development of GSPNs

Stochastic Petri Nets (SPNs) were independently proposed by Molloy [6, 7] and by Florin and Natkin [8, 9] for the modelling and performance evaluation of discrete, distributed systems that exhibit features not representable by product-form queueing networks. SPNs are P/T nets [10, 3] extended by (exponentially distributed) stochastic timing specification associated with transitions.

The original idea of all authors was to combine qualitative validation and performance evaluation in a single modelling framework. However, all authors concentrated their efforts in the stochastic definition of the model. The main result was the construction of Markov chains isomorphic to the reachability graph in case of exponentially distributed transition firing delays. The stochastic definition was obtained by a labelling of the reachability graph with the average transition speeds from state to state corresponding to net transition firings. The semantics of transition was hence chosen to be similar to "single server queues" with possibility of arbitrary dependency of the firing rate from the departure marking. A complete exploitation of this theoretically powerful feature required of course a good knowledge of the reachability set, thus contradicting the original goal of defining models at a higher net level. Special forms of marking dependency similar to "multiple server" queues where used in several cases to avoid a definition of the stochastic part of the model at the state space level.

Usually SPN models were intended for steady-state performance evaluation, so that ergodicity of the underlying stochastic model is implicitly required most of the times. Ergodicity implies qualitative properties of the Petri net model such as boundedness, liveness, and existence of a home space. Necessary and/or sufficient conditions may be checked on the net structure and initial marking to ensure these behavioural properties.

GSPNs added immediate transitions (transitions that fire in zero time and with priority over timed ones) to the basic SPN model [1]. The extension of several priority levels and of inhibitor arcs was also part of the basic definition of the model [4]. The idea was to increase the modelling convenience of the formalism by allowing some "dynamic state adjustment" to occur in zero time. The introduction of immediate transitions usually leads to GSPN models with simpler graphic representations than the "equivalent" SPN models and that was the practical justification advocated for the addition of immediate transitions.

The concept of equivalence between different SPN or GSPN models itself remained at an intuitive, informal level. Empirically two (G)SPN model representations were considered equivalent if they generated isomorphic Markov chain representations. By thinking at the state space generated by models the feeling was that it should have always been possible to construct an SPN model generating a Markov chain isomorphic to the one generated by reduction to the tangible states of an arbitrary

GSPN model. Hence the conjecture was made that, altough in practice more flexible, the GSPN formalism had the same modelling power of the original SPN formalism without immediate transitions.

1.1.2 Scalable systems and parametric modelling

In the Petri net literature one can find two different notions of models: *net structures* and *net systems* [10, 11, 3]. A net system is what is usually called "Petri net" (PN), i.e. a weighted, bipartite graph together with an initial marking and a firing rule. Given a net system it is possible to construct the set of possible states called reachability set (RS) of the system in order to derive information on "behavioural properties." A net structure is the weighted, bipartite graph that is derived from a PN system ignoring the initial marking. The identification of the PN structure alone is justified by the availability of powerful "structural analysis" techniques such as, for instance, the computation of P- and T-invariants [3]. This kind of analysis is based only on the graph structure; its results are therefore common to all systems differing only for the initial marking. We call the PN structure simply "net" and the PN system simply "system."

A net can be considered as the generator of a class of several possible systems obtained by associating different initial markings with the net. However, systems derived from the same net can have substantially different behaviours. It can even be the case that a system is live while another is deadlocked (and notice that a system subject to deadlock may be viewed as a degenerate case with null steady-state performance). In other words the PN structure in general is not sufficient to define a family of behaviourally similar systems. It is only in the constrained case of "free-choice" nets that some regularity can be found in terms of changing behaviour for changing initial marking: free choice nets are monotonic with respect to liveness for increasing initial markings, i.e. a live free-choice system is guaranteed to remain live if the initial marking is increased. In all cases with structure less constrained than free-choice, any change in the initial marking (increase or decrease) can potentially "kill" the system.

Usually a person who designs a PN model has in mind something which is neither a net nor a system, but some kind of intermediate model where part of the initial marking is fixed and part may vary. In [12] such an intermediate model specification called "family" has been defined. A family is a PN where the initial marking of some places is constant (as in PN systems) and that of other places is expressed as a parameter or as a function of parameters. Systems can be obtained from a family by "instantiating" the parameters to actual values. Borrowing terminology from object oriented programming we can say that a system is an object and a family is a class.

A typical example of system that is naturally modelled in terms of families is that of a computer system in which at most one customer at a time can access shared memory. The number of customers is usually a parameter, while the place that expresses the exclusive access to the memory must have a constant initial marking equal to one in order for the model to be correct (i.e. to enforce mutual exclusion).

Notice that, while SPNs and GSPNs are usually defined in terms of net systems with a given initial marking, most if not all applications developed using these formalism are scalable in nature. Moreover the results presented in literature usually involve plotting performance results versus varying system parameters. There is therefore a mismatch between the formal definitions of (G)SPNs (non parametric) and their actual use (parametric modelling and sensitivity studies). In this paper we argue that, by explicitly taking the parametric nature of (G)SPN models into account, some more insights can be gained on the actual value of the availability of higher priority immediate transitions in the model formalism.

Parametric analysis of P/T nets [10] has been studied in [12], where families are used as a reference to compare P/T nets with and without priorities and inhibitor arcs. In this paper instead we compare generalized stochastic Petri nets (GSPN) [1, 4] and stochastic Petri nets (SPN) [7, 9], in the particular case of bounded (and usually also live) families (the study of liveness and boundedness of PN families has been presented in [12]).

1.1.3 Purposes of the work

The basic question we want to address is: "*Are immediate transitions in GSPNs a real extension with respect to SPNs as far as scalability power of the two formalisms is concerned?*". By "scalability power" we mean the ability of a model of representing scalable modular systems in a parametric way. A parametric model can represent different real systems by simply scaling up or down some of the parameters involved in the model definition, without affecting the structure of the net. For example, well-formed coloured nets in the definition given in [13, 14] can represent a whole set of systems by simply changing the size of the basic color domains. P/T nets can only be parametric with respect to the initial marking, while Condition/Event nets [10] or Elementary Net Systems [15] have no possibility of parametrization that leaves the structure unchanged.

The aim of the paper is twofold. A first goal is understanding the relationships between GSPNs and SPNs and the role that immediate transitions can play in providing a greater scalability power to GSPNs with respect to SPNs. A second goal is showing how in some cases GSPN families can be reduced into equivalent SPN families, and how the use of such reductions can increase the size of GSPN systems for which we can numerically compute a performance solution.

The balance of the paper is as follows. Section 1.2 introduces the basic concepts of GSPN and SPN families and systems. The stochastic semantics of confusion free GSPN systems is also discussed as it is of relevance for the reduction of GSPNs to SPNs. Section 1.3 introduces the basic framework for the comparison of GSPNs and SPNs. Given a generic (bounded) GSPN system an equivalent SPN system is constructed. Subsequently the comparison is stated in terms of families instead of systems and it is then shown that there are GSPN families that do not have SPN counterparts. In Section 1.4 a subclass of GSPNs is defined for which we give an algorithm to transform a GSPN family into an SPN family by removing immediate transitions in an appropriate way. Finally we present an application of the algorithm to an FMS application example and recall a few works [16, 17] where the removal of immediate transitions has been necessary to be able to solve problems of the required size.

1.2 GSPN families and systems

A GSPN family is basically a P/T net with priorities and inhibitor arcs with an associated set of rate and marking parameters. Each transition has an associated function of rate parameters. Places are divided in two subsets: places that have a fixed associated initial marking, and places that have a function of the marking parameters as initial marking. GSPN systems are obtained from a family by instantiating the marking parameters and the rate parameters to actual values.

1.2.1 Definitions

Definition 1 *We call* GSPN-family *the 11-th tuple*

$$\mathcal{N} = (P, T, \Pi(.), W^-(.), W^+(.), W^H(.), M_{par}, MD(.), R_{par}, MP(.), RP(.))$$

where

- $P = P_C \cup P_V$ *is the set of places;* $P \neq \emptyset$ *and* $P_C \cap P_V = \emptyset$*;* P_C *is the set of places to which a constant initial marking is assigned, while* P_V *is the set of places whose initial marking depends on the value of some parameters.*

- T *is the (non empty) set of transitions, such that* $T \cap P = \emptyset$*.*

- $\Pi : T \to I\!N$ *is the priority function.* $T_E = \{\ t \in T\ :\ \Pi(t) = 0\ \}$ *is the set of "timed" transitions* [1] *and* $T_I = \{\ t \in T\ :\ \Pi(t) > 0\ \}$ *is the set of immediate transitions. We require* Π *to be such that* $|T_E| > 0$*. As far as this paper is concerned we assume only two priority values: zero (for timed transitions) and one (for immediate transitions).*

[1] the index E stands for exponential, as the delay associated with timed transitions is exponentially distributed

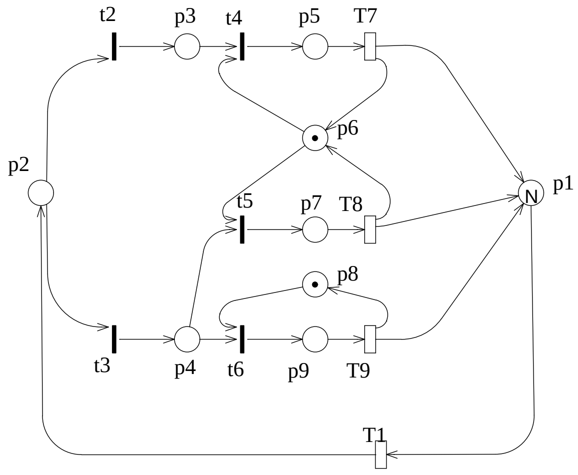


Figure 1.1: A GSPN family model of an FMS system

- $W^-(.), W^+(.), W^H(.) : T \to Bag(P)$ *(that we can also write as* $T \to P \to I\!N$ *are, respectively, the input, output, and inhibitor functions. We also define the input, output, and inhibition sets of a transition* t *respectively as* ${}^\bullet t = \{p \in P : W^-(t)(p) > 0\}$, $t^\bullet = \{p \in P : W^+(t)(p) > 0\}$, *and* ${}^\circ t = \{p \in P : W^H(t)(p) > 0\}$.
- $R_{par} = \{\rho_1, \ldots, \rho_k\}$ *is a set, called the "rate parameters set."*
- $RP : T \to R_{par} \cup I\!R^+$ *associates a rate parameter or a rate value with each transition.*
- $M_{par} = \{\mu_1, \ldots, \mu_m\}$ *is a set, called the "marking parameters set."*
- $MD : M_{par} \to 2^{I\!N}$ *is the set of legal values of each marking parameter.*
- MP *is the function that associates with each place in* P_C *a natural number (constant parameter) and to each place in* P_V *a positive function of the marking parameters. For the sake of simplicity in this paper we limit this function to associate one marking parameter with each place in* P_V *so that we have:*
$$\forall p \in P_C \quad MP(p) \in I\!N$$
$$\forall p \in P_V \quad MP(p) \in M_{par}$$

A GSPN family without any immediate transition ($T_I = \emptyset$) is called an SPN family.

A GSPN family can be seen either as a "partially specified" GSPN system (i.e. a GSPN system in which some of the constants involved are instead parametric) or as a "partially instantiated" net (a net in which there are constraints on the possible initial markings and firing rates). Figure 1.1 shows an example of a GSPN family. This example is used also in Section 1.4 to show an application of the algorithm that eliminates immediate transitions, so that we describe it here in some details.

The GSPN family of Figure 1.1 represents a flexible manifacturing system in which there are a parametric number of users (indicated by the parameter N) and exactly two machines $R1$ and $R2$. Users can require two different kinds of works: the first one (class 1) can be performed only on machine $R1$, while the second one (class 2) can either be done on $R1$ or $R2$. A user presents itself to the machines after a "thinking time" modelled by the timed transitions $T1$. The choice of which type of work the user requires is probabilistically modelled by transitions t_2 and t_3 with the associated parameters α and $1-\alpha$. Immediate transition t_4 (t_5) models the beginning of an access to the machine $R1$ by users of class one (two), while t_6 models the beginning of an access to $R2$ by users of class two. Users need to have exclusive access to machines, and this is modelled by place P_6 for what concern machine $R1$ and by place P_8 for $R2$. These places are assigned a fixed initial marking equal to one. Transition T_7 (T_8) represents the end of work by users of class one (two) on machine $R1$, while T_9 represents the end of work on machine $R2$.

The GSPN model is independent of the number of users, while the existence of a single machine per machine type is fixed. This allows us to define a GSPN family where $P_V = P \setminus \{p_1\}$, $P_C = \{p_1\}$, $M_{par} = \{N\}$, $R_{par} = \{\alpha, \lambda, \rho_{11}, \rho_{12}, \rho_2\}$, $MD(N) = I\!N$; moreover $MP(p_1) = N$, $MP(p_6) = MP(p_8) =$

1, all remaining places are assigned a constant initial marking equal to zero, $RP(T_1) = \lambda$, $RP(t_2) = \alpha$, $RP(t_3) = 1 - \alpha$, $RP(t_4) = RP(t_5) = RP(t_6) = 1$, $RP(T_7) = \rho_{11}$, $RP(T_8) = \rho_{12}$, and $RP(T_9) = \rho_2$. Different GSPN systems can be obtained by taking different values for N, or different values for the rate parameters. It is interesting to note that GSPN systems that are obtained by instantiation of the same family for different values of the marking parameters in general have a different state space. GSPN systems that differ only in the transition rate values have instead the same state space, although they may differ in the rates at which the system changes state.

On a GSPN family we can identify subnets containing independent, higher priority (i.e., immediate) transitions. These subnets, that will be called HIISs in the sequence, are used in Section 1.4 to perform structural reductions of immediate transitions.

Definition 2 *A HΠS is a connected subnet* [2] *containing only immediate transitions together with their input places. A HΠS is maximal if it is not a subnet of another HΠS. From here on, when we use the term HΠS we implicitly assume that it is maximal, unless otherwise specified.*

GSPN families can be instantiated in order to define system models with particular values of rate and marking parameters. The instantiation consists in a textual substitution of actual values for parameter names. The model obtained is called GSPN system and it is defined as follows.

Definition 3 *A GSPN system [4] is a tuple*

$$\Sigma = (P, T, \Pi(.), W^-(.), W^+(.), W^H(.), M_0, \Lambda)$$

where

- *P is the (non empty) set of places;*
- *T is the (non empty) set of transitions, such that $T \cap P = \emptyset$;*
- *$\Pi : T \to I\!N$ is the priority function. As in GSPN families we denote $T_E = \{\ t \in T \ :\ \Pi(t) = 0\ \}$ and $T_I = \{\ t \in T \ :\ \Pi(t) > 0\ \}$ the set of immediate transitions. It is required that Π be such that $|T_E| > 0$.*
- *$W^-(.), W^+(.), W^H(.) \ :\ T \to Bag(P)$ are, respectively, the input, output, and inhibitor functions.*
- *$M_0 : P \to I\!N$ is the initial marking.*
- *$\Lambda \ :\ T \to I\!R^+$. $\Lambda(t)$ is called the rate value of t if $t \in T_E$, and the weight value of t if $t \in T_I$.*

We call SPN system a GSPN system where $T = T_E$.

1.2.2 Dynamic behaviour

The dynamic behaviour of a GSPN system Σ is defined in terms of the so called "token game." A transition t is said to *have concession* in marking M iff $M \geq W^-(t) \quad \wedge \quad \forall p \in^\circ t,\ M(p) < W^H(t)(p)$. A transition t is defined to be *enabled* in marking M iff it has concession in M and $\forall t' \in T$ that have concession in M, $\Pi(t) \geq \Pi(t')$. As a consequence of this definition, only transitions of the same priority level can be enabled in a marking. This means that if an immediate transition is enabled no timed transition is enabled and vice-versa. We use the notation $\mathcal{E}(M)$ to indicate the set of all transitions enabled in marking M.

A marking M is called *vanishing* iff there exists at least one immediate transition enabled in M, and it is called *tangible* otherwise. From now on we assume that the initial marking M_0 is tangible. Any transition t that is enabled in a marking M can fire, producing a new marking:

$$M' \quad = \quad M \quad + \quad W^+(t) \quad - \quad W^-(t) \tag{1.1}$$

[2] Two nodes are defined to be connected if there exists a set of arcs (considered as non directed) that links them.

The firing operation is denoted $M[t\rangle M'$, meaning that $t \in \mathcal{E}(M)$ and that M' satisfies Equation 1.1. We call $W^+(t) - W^-(t)$ the *incidence function* of t; it represents the change of marking due to the firing of t.

We define a transition sequence starting from a marking M as $\sigma_M = (t^1, \ldots, t^k)$ to be any sequence of transitions for which $\exists M_{(1)}, \ldots, M_{(k)}$ such that $M[t^1\rangle M_{(1)}$, and $\forall j : 1 < j \leq k$, $M_{(j-1)}[t^j\rangle M_{(j)}$. Given any transition sequence σ_M we can define its firing count as the bag of transitions in σ_M. A marking M' is said to be reachable from a marking M (denoted by $M[\sigma_M\rangle M'$) iff there exists a sequence σ_M such that $M_{(k)} = M'$.

Given a GSPN system we can compute the set of possible states of the system, that is the set of all possible markings that can be reached from the initial marking by firing any possible transition sequences. We therefore give the following definition.

Definition 4 *The Reachability Set of a GSPN system Σ is denoted $RS(\Sigma)$, and defined as the smallest set of markings such that*

- $M_0 \in RS(\Sigma)$
- $M_1 \in RS(\Sigma) \wedge \exists t \in T : M_1[t\rangle M_2 \Rightarrow M_2 \in RS(\Sigma)$

The reachability set includes both vanishing and tangible markings. From the point of view of the associated stochastic process we are interested only in tangible markings.

Definition 5 *The Tangible Reachability Set $TRS(\Sigma)$ of a GSPN system Σ is the set of all tangible markings of $RS(\Sigma)$*

We can also define to be *tangible* all places that are marked in at least one tangible marking. A place which is not tangible is said to be *vanishing*.

The reachability set of a GSPN system does not include any information on how the system evolves from state to state, information which is instead captured by the reachability graph:

Definition 6 *Given a GSPN system Σ and its reachability set $RS(\Sigma)$ we call Reachability Graph $RG(\Sigma)$ the labelled directed multigraph whose set of nodes is $RS(\Sigma)$ and whose set of arcs A is defined as follows:*

$$\bullet A \subseteq RS(\Sigma) \times RS(\Sigma) \times T$$
$$\bullet \langle M_i, M_j, t\rangle \in A \Leftrightarrow M_i[t\rangle M_j$$

M_0 is taken as the initial node of the graph.

Again, the reachability graph of Σ includes both tangible and vanishing markings. If we concentrate on tangible states, and no loop of vanishing markings exists in $RG(\Sigma)$, we can build what is known as the Tangible Reachability Graph (TRG). Typically the TRG is a graph where the set of nodes is the set of tangible states, and there is one arc between two nodes *for each possible path in the corresponding RG between the same two nodes.* Each arc is labelled with the tags of the transitions that fire in the path. Each path begins with a timed transition possibly followed by one or more immediate transitions (immediate path). We do not provide a formal definition of TRG immediately: we postpone it after the discussion of the stochastic semantics of GSPNs.

1.2.3 Stochastic Semantics of GSPNs

In [4, 18, 19] the dynamics and semantics of GSPN systems are clearly explained in terms of the associated Markovian model (CTMC). We summarize here the main points, mainly for what concerns immediate transitions.

Timed transitions have an associated delay which has a negative exponential probability distribution. The function Λ defines the exponential distribution parameter of each timed transition. As far as this paper is concerned, transitions are assumed to be of the single server type. Other types of servers, like the multiple server or the fully general marking dependent case, would require some modifications in the algorithms proposed. When more than one timed transitions are enabled in the same marking M, the probability that t is the one that actually fires, is given by the expression:

$$\frac{\Lambda(t)}{\sum_{t_k \in \mathcal{E}(M)} \Lambda(t_k)}$$

Please note that, due to the continuous nature of the exponential distribution, the probability that two timed transitions fire at the same time is zero. The rate at which the CTMC leaves a given state is computed as the sum of the firing rates of all transitions enabled in that state.

The situation is different for immediate transitions. Given a state where $\{t_1, t_2, \ldots, t_n\}$ immediate transitions are enabled, what is the probability that t_i, $i = 1, \ldots n$ is the transition that actually fires? Observe that this question is relevant when some transitions in $\{t_1, t_2, \ldots, t_n\}$ are *in conflict*. This computation should require the definition of a distribution mass function for the probability of t_i to fire when $\{t_1, t_2, \ldots, t_n\}$ is the set of the enabled transitions. This approach has since long been recognized to be too cumbersome to follow, as it requires the specification of probabilities also for non conflicting transitions. A first, oversimplified way of addressing this problem could be that of specifying a weight for each immediate transition and to compute the probability of t_i when $\{t_1, t_2, \ldots, t_n\}$ are enabled as the weight $\Lambda(t_i)$ normalized with respect to the sum of the weight of all the transitions which are enabled using the same equation as in the case of timed transitions. This approach suffers from the problem of defining the probability of the transitions in T_I with respect to transitions that are not necessarily in conflict.

All problems of definition of choice probabilities disappear in the case of (extended) *free-choice conflicts*. In this case subsets of immediate transitions can be immediately identified by inspection of the connection of the shared input and inhibition places such that all transitions in a subset are either enabled or disabled simultaneously in every possible marking. Hence the specification of the probabilistic choice is actually feasable at the net level in the sense that it does not require any knowledge on the set of reachable markings. Unfortunately, the restriction of free-choice conflicts only between immediate transitions is too strong and does not allow the representation of contention for shared resources among different customers. The possibility of non free-choice conflicts may yield to situations in which some transitions are in conflict in certain markings while the same transitions are not in conflict in other markings. Hence a probabilistic conflict resolution must vary from one marking to another. Moreover, *stochastic confusion* [4] may arise due to the combination of different conflicts in different portions of the net, so that in general the assignment of probabilities requires a knowledge of the marking of the whole net and of the sequences that have led to that marking.

In [4] and [19] the choice was made to consider only GSPNs which are *confusion-free*, and to define the weight of immediate transitions locally to each set of possibly conflicting transitions (ECS), so that the probability of firing t_i when $\{t_1, t_2, \ldots, t_n\}$ are enabled in a marking M can be expressed as:

$$\frac{\Lambda(t_i)}{\sum_{t \in ECS(t_i) \cap \mathcal{E}(M)} \Lambda(t)} \cdot prob(ECS(t_i) \text{ is the one that fires})$$

Since the net is confusion-free the probability of firing a given ECS in marking M is irrelevant [19]. Hence immediate transitions weights can be defined inside a single ECS without having to worry about the relative probability of different ECSs so that the probability of firing t_i in marking M can be computed indifferently as

$$\frac{\Lambda(t_i)}{\sum_{t \in \mathcal{E}(M)} \Lambda(t)}$$

or as

$$\frac{\Lambda(t_i)}{\sum_{t \in \mathcal{E}(M) \cap ECS(t_i)} \Lambda(t)} \cdot \text{relative weight of } ECS(t_i) =$$
$$= \frac{\Lambda(t_i)}{\sum_{t \in \mathcal{E}(M) \cap ECS(t_i)} \Lambda(t)} \cdot \frac{\sum_{t \in \mathcal{E}(M) \cap ECS(t_i)} \Lambda(t)}{\sum_{\forall ECS enabled} \Lambda(ECS)}$$

or as

$$\frac{\Lambda(t_i)}{\sum_{t \in \mathcal{E}(M) \cap ECS(t_i)} \Lambda(t)} \cdot \frac{1}{\#\text{ of enabled ECSs}}$$

The reason why this is irrelevent is that transitions belonging to different ECSs can never disable one each other and that the firing of a transition inside a given ECS cannot alter the set of enabled transitions in another ECS which was already enabled (this is where the requirement of the system to be confusion-free plays an important role).

As an extreme consequence we can define a total order among ECSs and we can say that the relative weight of $ECS(t)$ is 1 if $ECS(t)$ is the greatest ECS (with respect to the total order) among those enabled, and is zero in all other cases. Observe that, by setting to zero some of the relative weights, we are specifying a null firing probability in a given marking for the transitions belonging to all but one ECS. This is equivalent to defining an appropriate priority structure on immediate transitions. Therefore we are actually eliminating some of the immediate paths from the reachability graph., i.e. we are considering only one out of many possible interleavings of transition firings in different ECSs.

The above discussion about the independence of transitions belonging to different $ECSs$ leads to the following observation: if the TRG keeps track of all possible paths among two tangible states, then it contains redundant information in the immediate transition firing interleavings. Indeed this corresponds to the consideration of all possible interleavings of immediate transitions even when their firing order is irrelevant.

We define here a TRG in which the firing order of the $ECSs$ is not taken into account. Each arc in the TRG is labeled with the (multi)set of transitions that fire in a path between two tangible markings, neglecting the order; for each immediate transition t that fires in the path the label also records the set of transitions of the $ECS(t)$ that were enabled at the time of the firing of t (in order to be able to compute the probability of the path).

Definition 7 *Given a GSPN system Σ and its reachability graph $RG(\Sigma)$ we can define the Tangible Reachability Graph $TRG(\Sigma)$ as a labelled directed multigraph whose set of nodes is $TRS(\Sigma)$ and whose set of arcs A is defined as follows:*

$$A \subseteq TRS(\Sigma) \times TRS(\Sigma) \times T_E \times Bag(TFD)$$

where TFD is the set of "firing descriptors" and $Bag(TFD)$ is the multiset of transition firing descriptors. The set of firing descriptors [3] is defined by

$$TFD = T_I \times 2^{T_I}$$

with the additional constraint that $\forall (t,s) \in TFD, s \subseteq ECS(t)$. We define A as

$$\langle M, M', t_0, x\rangle \in A$$

iff $M[t_o\rangle M_1$, $x = [(t_1,s_1),\ldots,(t_k,s_k)]$ and $\exists$ a sequence of the elements of the multiset x that we indicate with $\{(t'_1,s'_1),\ldots,(t'_k,s'_k)\}$ such that

$$\begin{array}{ll} \exists\ M_2,\ldots,M_k: & M_1[t'_1\rangle M_2[t'_2\rangle \ldots M_k[t'_k\rangle M' \quad \text{and} \\ \forall i, 1 \le i \le k, & s'_i = \{t \in \mathcal{E}(M_i) \cap ECS(t'_i)\} \end{array}$$

Each labelled arc of the TRG contains the information required to compute the total weight of the arc in the corresponding CTMC using the formula: $\prod_{i\in[1\ldots k]} \frac{\Lambda(t_i)}{\sum_{t\in s_i}\Lambda(t)}\Lambda(t_0)$

It is interesting to note that if we only needed to determine the reachable states we could simplify the labels on the arcs. In this case it would be sufficient to include the bag of transitions that fire, without having to keep track of which conflicting transitions are enabled when a transition fires.

1.3 Comparing GSPNs and SPNs

The first problem that we address is the definition of a reasonable and practically meaningful concept of equivalence for GSPN as well as SPN systems. Intuitively, one would like to consider two performance

[3] We denote an element in TFD by $(t_j\{t_{j_1},\ldots,t_{j_{m_j}}\})$, with the associated meaning of "t_j fires when $s = \{t_{j_1},\ldots,t_{j_{m_j}}\}$ are enabled"; we also write (t,s) for short, where $t \in T_I$ and $s \subseteq 2^{T_I}$

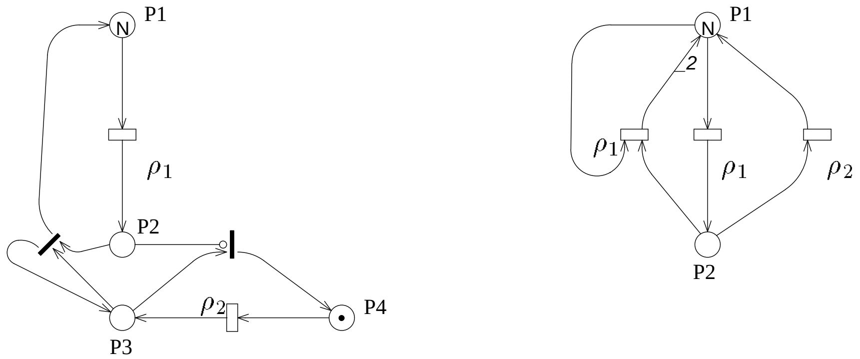


Figure 1.2: A GSPN and an SPN families yielding equivalent TRS, same Markov chain solution, but different TRG.

models to be equivalent if they are comparable from the point of view of the performance indexes that can be computed on them, and if they yield the same results. Since both GSPN and SPN systems are stochastically defined in terms of Markov processes, this requirement implies that two systems should be considered to be equivalent if they produce two equivalent Markov chains. We start by presenting the equivalence notion that we have defined, and by justifying it in terms of the above intuitive criterion. Then in the following subsections we show that, according to our equivalence notion, GSPN systems have always an SPN system counterpart while GSPN families do not always have an SPN family counterpart.

1.3.1 Equivalence notion for GSPN and SPN systems

The first obvious requirement that one can pose to consider two systems to be equivalent is that they yield two Markov chains with the same number of states and with the same (transient and/or steady-state) probability distribution. This would be sufficient to guarantee the possibility of computing the same performance indexes in the two models if the performance indexes themselves were defined directly on the states of the Markov chain. However, since performance indexes in SPN as well as GSPN systems are usually defined at the net level (i.e. in terms of the marking and/or the transition firing but without a previous knowledge of the set of reachable markings) this requirement is not sufficient. A precise correspondence should be established between the marking and/or the firing processes of two models in order to be able to compute the same performance estimates on both.

A first improvement could be obtained by demanding to have a one-to-one mapping between the places of SPN (or the tangible places of GSPN) systems. Hence also the tangible reachability sets of two systems should be equal. In this way all performance indexes defined on the net marking (such as e.g. the distribution of tokens in places, the average numbers of tokens, or the probability of given marking conditions) could be computed in the same way using the two models.

The requirement that the two systems have the same number of tangible places can be too restrictive, as it forces one to consider not to be equivalent also systems that differ only on things that have no influence on the behaviour of the system. Examples of this type are systems that differ only for the presence of implicit places, or places with a constant number of tokens, or vanishing places that do not affect the set of tangible places. We can relax this condition by simply saying that the two systems have the same number of states, and that there are two transformations from the set of places of the first system into that of the second one, and vice-versa, in such a way that any function (including the performance indexes) of the reachable markings of the first system can be translated into a function of the reachable markings of the second system and vice-versa.

In Figure 1.2 a GSPN family and an SPN family are shown that would be equivalent according to all above criteria. These systems have the remarkable characteristics of generating the same TRS and two Markov chains different in structure but with identical solution for any common instantiation of the parameters N, ρ_1, and ρ_2. An example of instantiation in the case $N = 3$ of the two Markov chains is depicted in Figure 1.3. The following three linearly independent state-probability balance

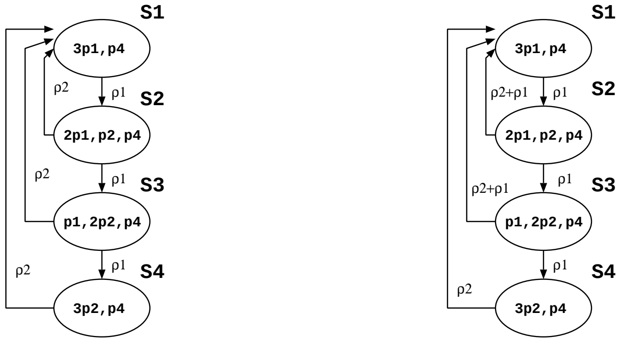


Figure 1.3: The two Markov chains and their unique balance equations for $N = 3$.

equations hold true in both Markov chains, so that the two steady-state probability distributions must be equal.

$$\begin{aligned} \rho_1 \cdot S_3 &= \rho_2 \cdot S_4 \\ \rho_1 \cdot S_2 &= (\rho_1 + \rho_2) \cdot S_3 \\ \rho_1 \cdot S_1 &= (\rho_1 + \rho_2) \cdot S_2 \end{aligned}$$

The drawback of the above type of equivalence is that no relation exists between the transitions of the two models so that, for example, the comparison of transition throughputs does not make sense. Moreover, being the two TRGs different, the equivalence can be shown only by analysis of the Markov chains: nothing can be said by inspection of the two net structures. The proof that the two families are equivalent for any parameter values is obtained by induction on the structure of the state balance equations. In order to achieve a complete comparability of results we introduce an additional requirement on the mapping between the reachability graphs of the (G)SPN systems.

Definition 8 *Two (G)SPN systems Σ_a, Σ_b are equivalent if and only if*

i) *there exist a bijection β among $TRS(\Sigma_a)$ and $TRS(\Sigma_b)$, two transformations described by the matrices $\Delta_{ab} : (P_a \times P_b) \rightarrow Q$ and $\Delta_{ba} : (P_b \times P_a) \rightarrow Q$ and two constant row vectors $\delta_b : P_b \rightarrow I\!N$ and $\delta_a : P_a \rightarrow I\!N$ such that $\forall M_a \in TRS(\Sigma_a)$ and $\forall M_b \in TRS(\Sigma_b) : M_b = \beta(M_a)$ we have*

$$\begin{aligned} M_a \cdot \Delta_{ab} + \delta_b &= M_b \\ M_b \cdot \Delta_{ba} + \delta_a &= M_a \end{aligned}$$

where the marking is considered as a row vector of natural numbers, Q is the set of rational numbers and $\cdot$ is the classical "row by columns" matrix multiplication.

ii) *there is an arc in the $TRG(\Sigma_a)$ between M'_a and M''_a if there is at least one arc between $\beta(M'_a)$ and $\beta(M''_a)$ in the $TRG(\Sigma_b)$ and vice-versa: $\forall M'_a, M''_a \in TRS(\Sigma_a) : \; M'_a \neq M''_a$,*

$$\exists (M'_a, M''_a, t_a, x_a) \in A_{TRG(\Sigma_a)} \iff \exists (\beta(M'_a), \beta(M''_a), t_b, x_b) \in A_{TRG(\Sigma_b)}$$

where A_{TRG} is the set of arcs of the TRG

iii) *the probability distribution of the Markov chain defined by Σ_a is equal to the probability distribution of the one defined by Σ_b.*

Definition 8.i) implies that two systems are equivalent up to implicit and/or constant marking places. In the GSPN and SPN systems that we compare in this paper we often have a one to one relation between tangible places, and therefore the two transformations Δ_{ab} and Δ_{ba} are simply identity matrices and the two constant vectors are identically equal to zero. The constant vector is required for taking into account systems that differ only for one or more places with a constant marking. More complex cases, as the presence of implicit places, require the use of arbitrary matrices. In Definition 8.ii) we do not require a bijection among arcs because we want to consider equivalent (G)SPN systems that contain redundancies in the transitions (two or more transitions whose firing has the same effect on the change of marking).

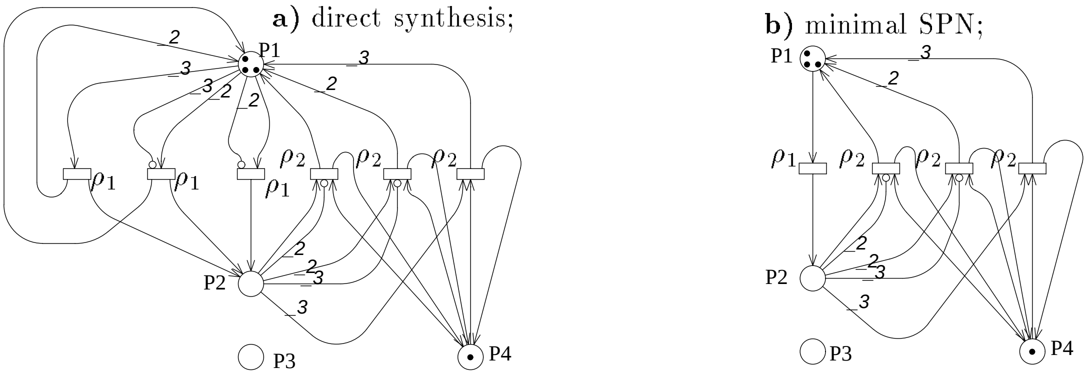


Figure 1.4: An example of equivalent SPN system synthesis.

1.3.2 Synthesis of SPN systems from GSPN systems

The existence of an SPN system Σ_S equivalent to any arbitrarily given GSPN system Σ_G is proven in a constructive way by giving an algorithm that defines such an equivalent SPN system starting from an arbitrarily chosen GSPN system.

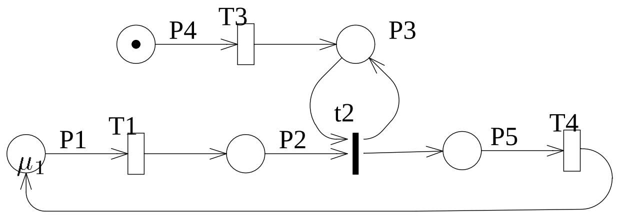
Without any further constraint this construction would be trivial: one could take the structure of the TRG of the GSPN, transform states into places, arcs into timed transitions connected with an input and an output arc, and add exactly one token to the place that represents the initial marking. This construction would of course guarantee the equivalence among the two reachability graphs, but in general it would not be possible to find the two matrices Δ_{ab} and Δ_{ba} that define the transformations from one state space to the other.

A correct construction requires that the same set of tangible places and initial marking of the GSPN system is used for the synthesis of Σ_S. A direct solution would be to introduce in the SPN system exactly as many different transitions as the total number of arcs in $TRG(\Sigma_G)$. Each transition must have the marking of departure of the arc as its enabling condition, and this can be obtained by adding one input and one inhibitor arcs connected to each place p, respectively with weights $M(p)$ and $M(p)+1$. Additionally, each transition should have as many output arcs as the total number of places that contain at least one token in the marking M' reached by the corresponding arc of $TRG(\Sigma_G)$, with multiplicity $M'(p)$. Figure 1.4.a depicts an SPN system obtained by synthesis of the GSPN system in Figure 1.2 for the case of $N = 3$.

The above SPN system satisfies our equivalence definition, but it is not a good representative of the class of SPN systems equivalent to a given GSPN system if a fair comparison is to be made of the modelling power of the two formalisms. This is because the number of transitions is much larger then the number of transitions that a human modeller would use to represent the system with an SPN formalism. Indeed one of the main advantages of using a net based formalism is to obtain a compound representation of many state transitions by means of a substantially fewer number of net transitions (factorization of state transitions in PN transitions). We introduce then the concept of *minimality* of an SPN representation based on the maximum grouping of state transitions into net transitions.

Proposition 1 *Given an arbitrary GSPN system Σ_G it is always possible to construct an SPN system Σ_S that is equivalent to Σ_G according to Definition 8 and that is minimal in terms of number of transitions.*

The proof of the Proposition is given by construction exploiting the synthesis algorithm reported in Appendix 1.6. First the tangible reachability graph $TRG(\Sigma_G)$ is computed. Then the synthesis algorithm is applied to contruct a minimal SPN representation Σ_S such that $TRG(\Sigma_S) = TRG(\Sigma_G)$. The synthesis algorithm is made of two phases: an initialization in which a direct (non minimal) SPN system (without immediate transitions) is defined, followed by a merge phase in which the number of PN transitions is reduced while preserving the original state transitions.



$MP(P1) = \mu_1, \; MP(P4) = 1, \; MP(P3) = MP(P2) = MP(P5) = 0$

Figure 1.5: An example of dependence of the set $\{(W^+(t) - W^-(t))\}$ from the value of parameter μ_1 due to a loop of immediate transitions.

Figure 1.4.b depicts the minimal SPN system obtained by application of the merge phase to the synthesized net in Figure 1.4.a.

From the availability of the synthesis algorithm one could draw the conclusion that immediate transitions are not strictly necessary to represent the resulting Markov chain of a bounded GSPN system. Hence, in this framework of non parametric modelling the extensions provided by GSPNs may be justified only by modelling convenience, and not by actual theoretical needs.

1.3.3 Comparing GSPN and SPN families

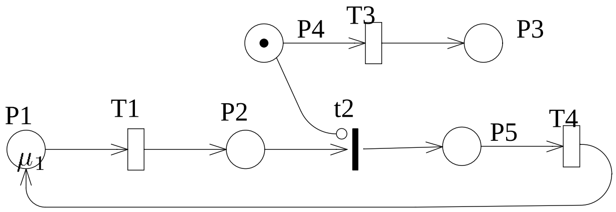
When we consider GSPN families instead of individual systems the synthesis algorithm given in the appendix does not apply any more. Indeed, the structure of the TRG that is the starting point of the synthesis algorithm changes when changing the initial marking. Hence different members of the family could be represented by different minimal SPN system structures according to the synthesis algorithm. In general, given a GSPN family $\mathcal{N}_G$ it is not possible to find a corresponding SPN family $\mathcal{N}_S$ such that pairs of GSPN and SPN systems obtained by the same instantiation from the two families are equivalent in the sense of Definition 8. It is clear that in the general case the two families can be rather different so that it can be impossible to define a common instantiation. However, our goal is comparing GSPN families to SPN families that are obtained by elimination of immediate transitions, so that they do have the same parameter definitions.

The comparison method used to show that priorities (and therefore immediate transitions) in GSPN systems are not strictly essential cannot be used directly for the case of GSPN families (as it is impossible to build a TRG for a family). However, its underlying idea of transition factorization can indirectly help our reasoning by:

- identifying the (structural) characteristics of GSPN families whose instantiation reduces to SPN systems with different structures;
- guiding the sketch of a structural reduction algorithm that applies to a class of GSPN families whose systems all reduce to SPN systems with a common structure for varying instantiations.

The synthesis algorithm factorizes all transitions that have the same incidence function $c(t) = W^+(t) - W^-(t)$ as much as possible. Two transitions having different incidence functions could never be substituted by a single transition without changing the TRG. Hence there would be a possibility of representing a parametric GSPN family without using the priority concept only if a variation of the value of the marking parameters does not change the set of all possible incidence functions as would be generated by the application of the initialization step of the synthesis algorithm.

The GSPN family depicted in Figure 1.5 provides an example in which the set of the incidence functions found in the analysis of the TRG changes for varying values of μ_1. For example there exists a set of incidence functions $\forall 0 < i \le \mu_1, \; c_i \; = \; -i \; P_2 \; - P_4 \; + P_3 \; + i \; P_5$ whose cardinality varies with the value of the parameter μ_1. Notice that the same behaviour can be observed also in the net of Figure 1.6 where the inhibitor arc plays a similar role as the loop on transition t_2 in the previous example.



$MP(P1) = \mu_1,\ MP(P4) = 1,\ MP(P3) = MP(P2) = MP(P5) = 0$

Figure 1.6: An example of dependence of the set $\{(W^+(t) - W^-(t))\}$ from the value of parameter μ_1 due to an inhibition on an immediate transition.

The problem with systems containing substructures of the kind shown in Figures 1.5 and 1.6 is due to the presence of a "cycle" of immediate (high priority) transitions (in the example the cycle contains only one transition, namely t_2) or an inhibitor arc with an analogous behaviour.

The type of operation that can be represented by a similar structure is the 0-time transfer of a variable (marking-dependent) number of tokens from one place into another one. Another possible cause of reduction to different structures is the elimination of inhibitor arcs in bounded nets by means of complementary places. In fact the weight of the test arcs connecting each complementary place to the "inhibited" transitions depends on the bound of the corresponding "inhibitor" place and thus on the initial marking. Different systems in the family with different bounds on the inhibitor places, in general reduce to systems whose structure differs at least for the weight on the test arcs connected to complementary places; anyway the problem could be solved if the maximum bound of each inhibitor place was known.

The considerations just presented justify the statement that GSPNs, thanks to immediate transitions (priorities) which are primitive in the formalism, allow one to parametrize more on the initial marking than SPNs, and thus have a higher modelling power from this point of view. Of course if this potentiality of the extended formalism is not actually exploited and the priorities are merely used for representation convenience, then a structural reduction algorithm can be used that reduces a family with priorities and inhibitor arcs into a corresponding family without priorities (provided that all the systems in the former family are bounded).

1.4 Reducing GSPN families to SPN families

The initial motivation for attempting to reduce GSPN families to SPN families was related to the problem of explosion of the number of vanishing markings in GSPN systems. Often GSPN systems with reasonably small TRGs have a large number of vanishing markings, so that the generation of the TRGs is prevented due to the necessity of storing the whole RGs. Some algorithms have been developed in the past that allow one to reduce the amount of information about vanishing markings stored during the TRG construction [20]. In this work we adopt a more radical approach and attempt the elimination of vanishing markings by eliminating their source, i.e., the immediate transitions. This approach may require a large amount of computation. Nevertheless the effort is worthwhile if it is performed only once for the family, so that the TRG can be computed for every possible instantiation of the GSPN family by properly instantiating the SPN family obtained from the reduction algorithm.

In Section 1.3.3 we have discussed the problem of finding an SPN family equivalent to a given GSPN family, and the conclusion was that this is not always possible.

In Section 1.4.2 we define a subclass of GSPN families (not the largest one) for which the above problem has a solution. Then we propose an algorithm that performs a structural reduction of GSPN families within that subclass into equivalent SPN families. A similar algorithm has been presented in [21] for untimed nets with priority and inhibitor arcs; the main difference between the two algorithms is due to the need of computing transition rates when working with GSPNs. For this reason, in the timed case the net obtained after the reduction can have more transitions, some of which performing the same state transformations but having different enabling conditions and/or different firing rates.

Let us first introduce the ideas underlying the reduction algorithm on an example, and then define the subclass of reducible GSPN family and outline the algorithm itself.

1.4.1 An introduction to structural reduction of GSPNs

Informally, a structural reduction algorithm should find for each timed transition T, the set S_i of all the complete immediate transition sequences [4] that can be made fireable by T. Each sequence is characterized by the multiset of immediate transitions that fire together with their firing probability. Sequences with the same transition multiset but different probabilities are considered separately. The net is then transformed by substituting each T with as many timed transitions as the number of sequences in S_i. Each new transition performs the same state transformation as the corresponding sequence in S_i. After this transformation all immediate transitions become redundant and can thus be removed from the net. Hereafter we explain these concepts on an example.

Figure 1.7 shows an example of reduction of a GSPN family (a) to an equivalent SPN family (b). The basic idea is the following. T has lower priority than t_5, so that T can only fire when t_5 has no concession. Hence, after the firing of T only t_1 and t_2 may be enabled in the ECS comprising t_5, t_1, and t_2. The higher priority of immediate over timed transitions ensures that if the firing of T enables any immediate transition t then the firing sequence "T followed by t" is perceived as an atomic action. It is therefore possible to substitute the firing of T followed by a sequence of immediate transitions by the firing of a new timed transition that replaces T together with all immediate transitions of the sequence.

The GSPN in Figure 1.7 allows the following three possible behaviours: T fires followed by t_1; T fires followed by t_2; T fires, but its firing does not enable any immediate transition. This is all we need to know in order to reduce higher priority transitions into lower priority ones (and it is in fact the approach that has been followed in [21] for untimed nets). Instead, if we want to reproduce also the correct stochastic behaviour in case of conflicts, we have to consider which are the immediate transitions that are enabled when a given immediate transition fires in order to be able to normalize the firing weights and obtain the correct firing probabilities. In our example we have to consider five different cases: T fires followed by t_1 when only t_1 is enabled; T fires followed by t_1 when also t_2 is enabled; T fires followed by t_2 when only t_2 is enabled; T fires followed by t_2 when also t_1 is enabled; T fires and is not followed by any immediate transition because neither t_1 nor t_2 is enabled. We can therefore substitute T, t_1, and t_2 by the five timed transitions "$T{+}t_1a$," "$T{+}t_1b$," "$T{+}t_2a$," "$T{+}t_2b$," and "Ta." We assume that the firing rate of transition T is ω, and the probability weights associated with t_1 and t_2 are respectively α and $1-\alpha$. The firing rates of the replicas of T followed by t_i are computed as the firing rate of T multiplied by the probability that t_i is the immediate transition that fires after T; they are respectively equal to: ω, $\omega\alpha$, ω, $\omega(1-\alpha)$, and ω.

It is interesting to note how enabling/disabling conditions have been translated into input, output, and inhibition function of the input places of the new timed transitions. For example, since the replica "$T{+}t_1a$" represents the "firing of T followed by t_1 when also t_2 is enabled," the W^-, W^+, and W^H functions of the replica are those of T plus a test arc on p_1 (that ensures that t_1 is enabled after T), a test arc of weight 2 on p_2 (that ensures that t_2 is enabled after T), and the arcs that implement the state transformation due to the firing of T followed by t_1 (this implies that the test arc on p_1 is actually an input arc, and that there is an output arc connecting the replica to p_3).

Although the probability that a given immediate transition fires is related to extended conflict sets (ECSs) note that in this case the firing of T can only enable a subset of the ECS. In particular it can only enable transitions in $p^\bullet$. We shall use the notation $CS(T)$ to indicate the subset of transitions of a given ECS that may be enabled by the firing of T (that is to say the subset of transitions of a given ECS that have a place of $T^\bullet$ as input place).

[4]In this context an immediate transition sequence is *complete* when no transitions belonging to the same HIIS are enabled after its firing.

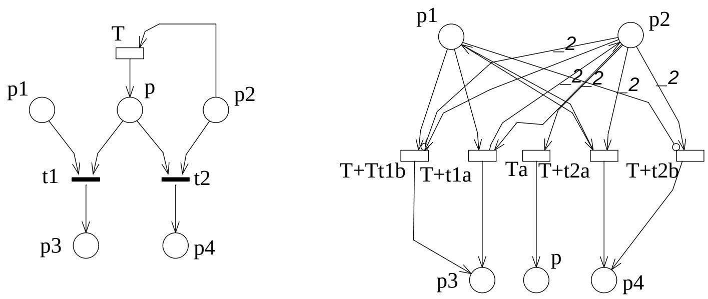


Figure 1.7: Reduction of a non free choice conflict

The number of replicas required to reduce immediate transitions on T grows exponentially with the number of immediate transitions which can be enabled by T (and hence with the cardinality of $p^\bullet$) as we have to consider all possible combinations of enabled/disabled transitions. Moreover, it is not always possible to reproduce a situation in which a subset of transitions is enabled and another is disabled with a single replica.

Consider for example the GSPN obtained by that in Figure 1.7 by adding an input place p_5 to transition t_2. We now have the following enabling condition for "T followed by t_1 when t_2 is not enabled": $p_1 \geq 1 \wedge (p_2 = 0 \vee p_5 = 0)$. The semantics of Petri nets does not allow to translate this boolean condition into the enabling of a single transition: we have therefore to provide two replicas in this case. In the general case we have to introduce as many replicas as the number of "and" clauses in the enabling condition in disjunctive normal form.

Another situation where the number of replicas can rapidly increase is when the timed transition is allowed to put more than one token into the subnet of immediate transitions. In this case in fact it is possible that more than one immediate transition is enabled, hence there are more combinations that have to be taken into account.

The example of Figure 1.7 has only one timed input (i.e., only one timed transition that can add tokens in p), but this is not always the case. When there are more than one timed inputs it is necessary to repeat the reduction on each timed input. For example if there is a timed input T' to place p_2 then t_2 has to be reduced on both T and T'. Only when both reductions have been performed the immediate transition can be deleted from the net.

In this example the immediate transition sequences that can follow the firing of T have length one. If some other immediate transitions could have been enabled by the firing of t_1 or t_2 (i.e., if some other immediate transition had p_3 or p_4 as input places), following the intuitive algorithm sketch outlined at the beginning of this section we should have found all the possible combination of complete immediate transition sequences (in this case of length two) and then create one replica of T for each of them. This method is correct but rather cumbersome to use in practice. Indeed it is much easier to perform the reduction of each sequence step by step, merging the first immediate transition in the sequence into the timed transition directly enabling it and then repeating the same algorithm on the newly created timed transition and the remaining part of the sequence. The algorithm presented in the following section is based on this idea.

1.4.2 A structural reduction algorithm

Let us first define the subclass of GSPN families for which the algorithm produces correct reductions. The imposed constraints can be partitioned into mandatory and convenience constraints. The convenience constraints are imposed only to simplify the statement of the algorithm: they can be relaxed without raising any conceptual problem. The mandatory constraints are due to the fact that some reductions cannot be performed at a purely structural level, but require some knowledge of the initial marking and the boundedness property.

The first convenience constraint for the reduction of a HIIS is that any transition putting tokens into a given HIIS should have a single output arc of weight one connected to that HIIS. This limits the possible concurrency within a HIIS and thus limits the number of possible firing sequences in it. The second convenience constraint requires that the input arcs of immediate transitions are of weight one and the number of input places of immediate transitions is less than or equal to two. This constraint reduces the possible disabling causes for immediate transitions. The removal of these constraints complicates the reduction algorithm: here we present only this simplified version for the sake of readability.

A mandatory constraint is that HIISs do not contain cycles because in all but few special cases the possible evolutions of an HIIS with cycles depends on the maximum number of tokens that can be in the places connected to the cycle itself, hence on the initial marking of the net. The consequence is that a completely structural reduction is in general not possible if cycles are present and the net is unbounded. For the same reason we do not consider the general case of HIISs containing inhibitor arcs, since in most cases they are amenable to cycles.

In the Appendix 1.7 the pseudo-code of the reduction algorithm is given. The algorithm reduces one HIIS subnet at a time on each timed transition that put tokens in it. In particular if there are still immediate transitions that have not been reduced to all their timed inputs, then the algorithm performs a reduction of the type shown in Figure 1.7.

The reduction algorithm modifies only the net structure of the GSPN family and removes only immediate transitions and vanishing places. Hence a bijection exists between the tangible places of the GSPN family and the places of the resulting SPN family. The MP function for the places in the SPN family is thus equal to the MP function for the corresponding places in the original GSPN family. As a consequence also M_{par} and MD are identical.

The rates of the SPN timed transitions are functions of the GSPN transitions rate and weight parametric definitions. Thus the rate parameters needed for the SPN family definition are still R_{par}; function $RP()$ is unchanged for the timed transitions that are not modified by the reduction while is defined by the algorithm itself for the newly generated transitions. This means that given an instantiation of the GSPN family, the same instantiation applies without modifications to the SPN family.

1.4.3 Equivalence between the GSPN family and the reduced SPN family

The structural reduction algorithm presented in Section 1.4.2 transforms a GSPN family $\mathcal{N}_G$ into a SPN family $\mathcal{N}_S$ that has the following properties:

1. Every valid instantiation of the marking and rate parameters of $\mathcal{N}_G$ is also a valid instantiation for the marking and rate parameters of $\mathcal{N}_S$.

2. For any instantiation of the marking and rate parameters of $\mathcal{N}_G$ producing system Σ_G, the system Σ_S obtained by using the same instantiation of the marking and rate parameters on $\mathcal{N}_S$ is equivalent to Σ_G according to Definition 8.

The equivalence can be easily shown since the function from markings of one system to markings of the other system is trivially derived. Moreover each transition sequence between tangible markings in the GSPN system has a corresponding timed transition with the same rate in the SPN system. Hence the CTMC generated by the two systems surely have the same solution since they are identical.

1.4.4 Application of the reduction algorithm

Let us illustrate the application of the algorithm to the GSPN model of an FMS depicted in Figure 1.1. The GSPN family has a single $HIIS$ that includes $\{t_2, t_3, t_4, t_5, t_6\}$ and has four timed transitions as input to the $HIIS$. We can observe that this model satisfies all restrictions (both convenience constrains

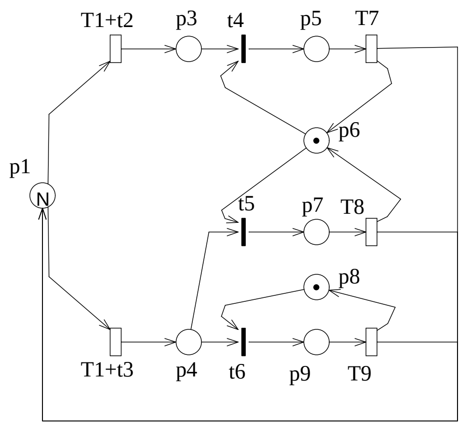


Figure 1.8: Reducing the free choice conflict

and mandatory constraints) posed by the reduction algorithm, and can therefore be completely reduced to an equivalent SPN family.

On this GSPN family we can show the effect of the reduction of immediate transition on the timed transition that enable them in the following three situations:

- the set CS of immediate transitions to be merged into timed transition T form a free choice conflict set;
- the set CS contains only one transition;
- the set CS is a non-free choice conflict set.

We can start reducing the free choice conflict between t_2 and t_3 on its timed input T_1. The resulting family is shown in Figure 1.8. Place P_2 has been deleted since it is vanishing. The two new timed transitions are named "T_1+t_2" and "T_1+t_3." Since this is a free choice conflict t_2 and t_3 are always enabled together, thus their firing probabilities are always α and $1-\alpha$ respectively. Hence the parametric rates of the newly created timed transitions are:

$$RP(T_1 + t_2) = RP(T_1)RP(t_2)/(RP(t_2) + RP(t_3)) = \lambda\alpha$$
$$RP(T_1 + t_3) = RP(T_1)RP(t_3)/(RP(t_2) + RP(t_3)) = \lambda(1-\alpha)$$

In the next step we reduce the conflict set composed of the single immediate transition t_4 on T_1+t_2. We obtain two new timed transitions T_1+t_2a and $T_1+t_2+t_4$ that represent the firing of T_1+t_2 that cannot be followed by t_4 (because place P_6 is not marked) and the firing of T_1+t_2 followed by t_4 (when P_6 is marked). The rate of $T_1 + t_2a$ is of course

$$RP(T_1 + t_2a) = RP(T_1 + t_2) = \lambda\alpha$$

Concerning the rate of $T_1+t_2+t_4$, since the CS contains only one transition, the probability of firing t_4 after $T_1 + t_2$ is $RP(t_4)/RP(t_4) = 1$, so that

$$RP(T_1 + t_2 + t_4) = RP(T_1 + t_2)RP(t_4)/RP(t_4) = \lambda\alpha$$

Notice that at this point we cannot yet delete transition t_4 because we have still to reduce it together with t_5 on transitions T_7 and T_8.
The reduction of t_6 on T_9 is analogous to the one just presented.

We can now consider the more complex case of reduction of a non-free choice conflict set. This is the case of the reduction of t_5 and t_6 on T_1+t_3. The reduction of t_4 and t_5 on both T_7 and T_8 is similar. The procedure produces five replicas. The meaning of the replicas is: $T_1+t_3+t_a$ is the firing of T_1+t_3 that cannot be followed by any immediate transitions because both t_5 and t_6 are not enabled (places P_6 and P_8 are not marked). The rate of this new transition is obviously

$$RP(T_1+t_3a) = RP(T_1+t_3) = \lambda(1-\alpha)$$

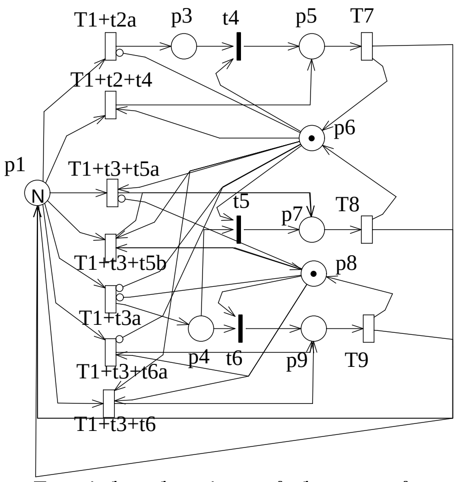


Figure 1.9: Partial reduction of the non free choice conflicts

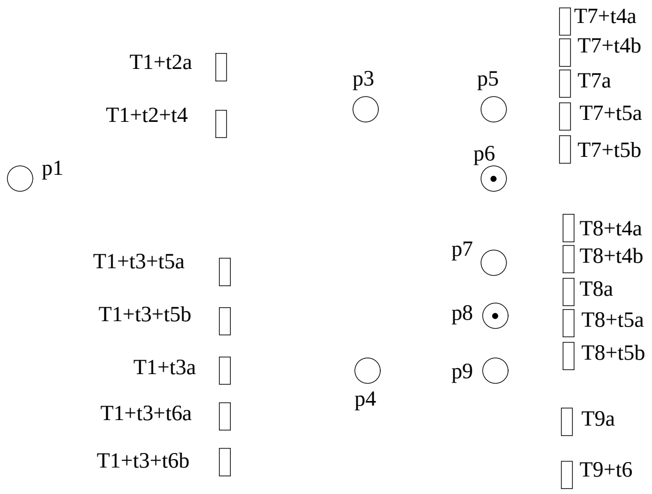


Figure 1.10: Schemata of the SPN equivalent family

$T_1{+}t_3{+}t5a$ represents the firing of $T_1{+}t_3$ followed by t_5 when t_6 is not enabled; its rate is then defined as

$$RP(T_1{+}t_3{+}t5a) = RP(T_1{+}t_3)RP(t_5)/RP(t_5) = \lambda(1-\alpha)$$

$T_1{+}t_3{+}t5b$ represents the firing of $T_1{+}t_3$ followed by t_5 when t_6 is enabled; its rate is then defined as

$$RP(T_1{+}t_3{+}t5b) = RP(T_1{+}t_3)RP(t_5)/(RP(t_5)+RP(t_6)) = \lambda(1-\alpha)\frac{1}{1+1} = \lambda(1-\alpha)\frac{1}{2}$$

$T_1{+}t_3{+}t_6a(b)$ is the firing of $T_1{+}t_3$ followed by t_6 when t_5 is not (is) enabled. The computation of their rates is similar to that shown for the two previous transitions.

Figure 1.9 shows the two partial reductions described above. Observe that in the algorithm the further reduction of t_4 on $T_1{+}\,t_2a$ and of t_5 and t_6 on $T_1{+}\,t_3a$ is prevented by the precondition *T was not generated by a previous reduction of the same CS* on the input parameters of the procedure that performs the reduction (RED_CS).

Figure 1.10 is a schemata of all replicas introduced by the reduction algorithm. The arcs are not shown for the sake of readability. Transitions labelled $T7xx$ ($T8xx$) are the replicas for the reduction of t_4 and t_5 on T_7 (T_8). The two transitions T_9a and $T_9{+}t_6$ represent the reduction of the elementary conflict set t_6 on its input T_9.

It is rather clear from this example that the number of replicas that can be introduced by the algorithm can be very high, so high as to make the net unreadable. But the goal of the reduction algorithm is to produce a net that can generate in a more efficient way the same state space as the original one: the basic idea is in fact that the reduction should be transparent to the user who works at the GSPN level. Another point that deserves attention is the amount of work required to reduce the net: again the reduction can be rather time consuming, but the key observation here is that the reduction has to be computed only once for a GSPN family and it is valid for all GSPN systems of the same family. Therefore it can be very convenient when performing sets of experiments for varying values of the initial marking or of the transitions rate. Moreover, sometimes the reduction of a GSPN to an equivalent SPN can be the only feasible way of solving a GSPN system. This is particularly the case when the number of vanishing states is very high compared to tangible ones.

The reduction algorithm has already been used in some applications. In [16] Ajmone Marsan et al. have used the reduced SPN equivalent to solve GSPN models of multiserver multiqueue systems whose complete state space (tangible + vanishing) could not be generated on the available machines. The most interesting application can be found in [17], where reduction is applied to models of a particular class of polling systems. Again the reduced SPN suffers from the excessive number of replicas, but in that particular case many of the replicas could be merged by taking advantage of the knowledge of the P-invariants of the net. The resulting simplified SPN system is even simpler than the original GSPN (in terms of number of places, transitions and arcs). In particular the stochastic behaviour of the system is more evident from the simplified SPN model.

1.5 Conclusions

The contribution of this work is twofold. From a theoretical point of view we studied the modelling power that immediate transitions bring to GSPN models: the conclusion is that a real advantage can be obtained when using GSPN families to model *classes* of actual systems rather than individual (G)SPN systems. In this case the presence of immediate transitions allows a higher parametrization degree with respect to SPN models. At the beginning of this work the major difficulty we found was the choice of a reasonable concept of equivalence between (G)SPN systems. In particular, the existence of cases of "equivalence" such as the one illustrated in Figure 1.2 between a GSPN and an SPN family that always generate systems with different TRGs was surprising. We chose to forbid this kind of equivalence in order to be able to study the models at the Petri net level rather than at the level of their Markovian processes. Should we have chosen an equivalence notion at the level of Markovian processes we would have probably reached different conclusions concerning the modelling power of GSPNs. Indeed the GSPN family in Figure 1.2 is one of those that do not admit an equivalent SPN family according to our equivalence definition. We are confident, however, that our choice of requiring an equivalence on the structure of the TRGs of the two models is not arbitrary and can be justified by modelling considerations.

From a practical point of view we studied the possibility of an algorithmic structural reduction of immediate transitions as a technique capable of optimizing the reachability graph analysis of GSPNs. A simplified algorithm is given in the appendix, and several extensions have already been studied. The implementation of this kind of algorithms is currently under evaluation in the framework of the *GreatSPN* package [22]. The main practical interest of a structural reduction algorithm as opposed to the usual reduction of vanishing markings at the state space construction level is that the structural algorithm applies to GSPN families rather then GSPN systems. The reduction is performed *once* even in the case of repeated solutions with varying parameters. Hence, even if the efficiency of the implementation of the structural algorithm should turn out to be comparable to the one of the state space reduction (and of course we have good reasons to hope that it would instead be substantially more efficient in several cases such as free choice conflicts) there would be an evident advantage in taking the structural approach.

Besides the actual implementation of the structural reduction algorithm, further work is still needed in order to better understand and formalize the cases of simplification used in [17] as well as the cases of presence of marking-dependent firing rates.

1.6 Appendix: SPN synthesis algorithm from TRG

We now present the sketch of an algorithm that, given a GSPN system Σ_G produces an equivalent SPN system Σ_S with the minimum number of different transitions.

The algorithm first produce an SPN system that has a timed transition for each arc (M', M'', t_0, x) of the TRG of Σ_G. The transition has as input all places of the net that are marked in M' and as output all places that are marked in M''; the weight associated with the transition is that computed from (t_0, x)

Due to the usual hyphothesis of ergodicity and of absence of absorbing loops of vanishing markings we can assume without loss of generality that in the initial marking no immediate transition is enabled, i.e., $\forall t \;:\; M_0[t\rangle,\;\; \Pi(t) = 0$.

Initialization: creation of a first approximation of Σ_S that is not minimal in terms of number of transitions. One timed transition is introduced for each state transition in the $TRG(\Sigma_G)$. The variable $Cset$ contains the set of all triplets *(incidence-function,rate,transition)*, i.e., for each added transition $Cset$ records the state transformation produced by the transition and its associated firing rate.

$P_{\Sigma_S} := P_{\Sigma_G}$;
$M_{0\Sigma_S} := M_{0\Sigma_G}$;
$T_{\Sigma_S} := \emptyset$;
$Cset := \emptyset$;
`foreach` $<$ marking $M' \in TRS(Sigma_G) >$ `do`
    `foreach` $<$ arc $a = (M', M'', r)$ of $TRG(\Sigma_G) >$ `do`
        $<$ create transition $tnew$ such that $\Lambda(tnew) = r >$;
        `foreach` $<$ place $p_j \in P >$ `do`
            $W^-(tnew)[p_j] := M'[p_j]$;
            $W^+(tnew)[p_j] := M''[p_j]>$;
            `if` $\exists M''' \in TRS(Sigma_G) \;:\; M''' \geq M' \;\wedge\; M'''[p_j] > M'[p_j]$
               `then` $W^H(tnew)[p_j] := M'[p_j] + 1$
               `else` $W^H(tnew)[p_j] := 0$
            `endif`
        `endfor`;
        $Cset := Cset \;\cup\; \{((W^+(tnew) - W^-(tnew)), r, tnew)\}$;
        $T_{\Sigma_S} := T_{\Sigma_S} \cup \{tnew\}$
    `endfor`
`endfor`.

At the end of the above algorithm, Σ_S contains one transition per arc of $TRG(\Sigma_G)$. Each transition can fire in exactly one marking of the TRG. Hence this preliminary version of Σ_S by construction produces the given tangible reachability graph $TRG(\Sigma_G)$ as its own reachability graph.

It is clear that there is an exceedingly large number of transitions in Σ_S: the goal of the procedure that we present next is merging transitions that perform the same transformation of state and have the same rate. The second constraint could be relaxed if we allowed marking-dependent firing rates. However, not all transitions with the same incidence function and the same transition rate can be merged into a single one: this possibility depends on the form of the enabling conditions of the transitions to be merged. In particular we cannot merge two transitions if the resulting one is enabled in markings where no transition with that same incidence function (and that same rate) is enabled in the original system: in fact this would add some state transitions which are not present in the original system.

Transition merging: factorization of all transitions that have the same firing effect on markings, the same firing rate, and whose enabling conditions do not introduce any new state transition.

```
foreach < incidence function c' and firing rate r' : ∃(c', r', t) ∈ Cset > do
    repeat
        merge := false;
        foreach M' ∈ TRS(Sigma_G) such that ∃t1 ∈ T_ΣS :
                M'[t1⟩ ∧ c' = W+(t1) − W−(t1) ∧ r' = Λ(t1) do
            if ∃M'' ∈ TRS(Sigma_G) such that ∃t2 ≠ t1 : M''[t2⟩ ∧ (c', r', t2) ∈ Cset
                and ∀M''' ∈ TRS(Sigma_G) such that ∀pj ∈ P
                    M'[pj] ≤ M'''[pj] ≤ M''[pj] ∨ M''[pj] ≤ M'''[pj] ≤ M'[pj]
                    there exists t3 : M'''[t3⟩ ∧ (c', r', t3) ∈ Cset
                then
                    merge := true;
                    < create transition tnew such that Λ(tnew) = r' >;
                    foreach pj ∈ P do
                        W−(tnew)[pj] := min(W−(t1)[pj], W−(t2)[pj]);
                        W+(tnew)[pj] := W−(tnew)[pj] + c'[pj];
                        if W^H(t1)[pj] = 0 or W^H(t2)[pj] = 0
                            then W^H(tnew)[pj] := 0
                            else W^H(tnew)[pj] := max(W^H(t1)[pj], W^H(t2)[pj])
                        endif
                    endfor;
                    T_ΣS := T_ΣS ∪ {tnew};
                    Cset := Cset ∪ {(c', r', tnew)};
                    REMOVE_REDUNDANT(T_ΣS, Cset, c', r')
            endif
        endfor
    until not merge
endfor.
```

The procedure REMOVE_REDUNDANT deletes from the sets all transitions of a given incidence function c and firing rate r whose enabling condition is more restrictive than other transitions with the same incidence function c and firing rate r.

When a new transition *tnew* is added, at least the two transitions t_1 and t_2 become redundant and are then eliminated by REMOVE_REDUNDANT, so that the number of transitions strictly decreases at each merge step. At the end, i.e. when the merge step is not applicable any more for any pair (c,r), the SPN system Σ_S contains the minimum number of transition necessary to generate the given $TRG(\Sigma_G)$. Two transitions with the same incidence function are not merged only if the new transition created by the factorization would be enabled in a marking for which no transition with the same firing effect and rate is enabled in $TRG(\Sigma_G)$. Hence the factorization obtained by this merge step is maximal.

The above algorithm shows that it is always possible to create a minimal SPN system with inhibitor arcs that has the same timed behaviour as a given bounded GSPN system. The distinctive characteristics of the kind of SPN systems synthesized by the above algorithm is that of containing the minimal number of transitions needed to reproduce exactly all the state transitions that can be found in the TRG of the equivalent GSPN system. Thus, in this sense, it represents the best SPN system equivalent to a given arbitrary GSPN system from the point of view of the model conciseness.

The factorization level of the minimal SPN model may be increased if *marking-dependent* firing rates are allowed. If unconstrained marking-dependent rates are allowed for SPNs, the rate component in the above reduction algorithm schema might be just deleted from the $Cset$ variable, thus allowing a total merging of transitions with same firing effect but different firing rates. A marking-dependent rate function can be trivially constructed by enumeration of all firing rates in the different markings in which the resulting merged transition is enabled. More cumbersome, yet conceptually trivial algorithm variations are required if *constrained* marking-dependency is allowed such as for instance in case of *multiple-server* firing semantics.

1.7 Appendix: structural reduction algorithm

We now present an algorithm to reduce a GSPN family to an SPN family, that is to say the algorithm eliminates all immediate transitions from a given GSPN family, to obtain an SPN family equivalent to it in the sense of Section 1.4.3.

As it was already discussed in Section 1.4.2, a GSPN family input to the algorithm has to satisfy a certain number of constraints, in particular it is required that: any timed transition can put at most one token in each $H\Pi S$; input arcs to immediate transitions all have weight one; immediate transitions have at most two input places; there are no cycles of immediate transitions; immediate transitions have no inhibitor arcs.

The algorithm reduces each subnet of immediate transition ($H\Pi S$) on every timed transition whose firing put tokens in it. The algorithm terminates when there are no more immediate transitions to be reduced and therefore no more immediate transitions.

Input : A GSPN family N that satisfies all constraints
Output : The equivalent SPN family
`while` $\exists$HΠS in N `do`
  T_I is the set of immediate transitions of the HΠS
  let T be a timed transition s.t. $((T^\bullet)^\bullet) \cap T_I \neq \emptyset$
  let p be a place s.t. $(p \in T^\bullet) \wedge (p^\bullet \cap T_I \neq \emptyset))$
  let $CS = p^\bullet \cap T_I$
  (* reduce the set of immediate transitions CS on timed transition T *)
  RED_CS(T, p, CS);
  `foreach` $t \in CS$ `do`
    (* decide if it is possible to delete t *)
    `if` $\neg\exists T' \in T_E$ s.t. $(T'^\bullet \cap^\bullet t \neq \emptyset) \wedge$ T' is not a transition generated by a previous reduction of a CS including t `then`
      delete_trans(t)
    `endif`
  `endfor`
`endwhile`.

The following procedure performs the reduction of a set CS of immediate transitions sharing (at least) one input place p, "merging it" into a timed transition T such that $p \in T^\bullet$. An application of the procedure has already been shown with reference to Figure 1.7. Notice that the transformation can add inhibitor arcs but this does not prevent the further reduction of the subnet because inhibitor arcs are always introduced on timed transitions.

`procedure` RED_CS(T, p, CS)
Input : T, p, CS
Precondition: $p \in T^\bullet \cap {}^\bullet(CS)$, $CS \subseteq T_I$ and T was not generated by a previous reduction of the same CS
Output : The reduced net
`begin`
  `foreach` $t_i \in CS$ `do`

(* X is the set of transitions of CS surely enabled with t_i *)
$X := \{t : t \in CS \wedge \mathsf{enab}(t_i) \Rightarrow \mathsf{enab}(t)\}$;
foreach $Y \subseteq (CS \setminus X)$ **do**
$Z := Y \cup X \cup \{t_i\}$;
(* Case of Z enabled and t_i fires *)
$newT$:= create_transition();
$\phi := \bigwedge_{t \in Z} \mathsf{enab}(t) \bigwedge_{t \in CS \setminus Z} \neg\, \mathsf{enab}(t)$;
$\langle$Substitute in ϕ each occurrence of $p \geq 1$ with $true$ $\rangle$
(* Generation of the input, output and inhibition arcs for $newT$ *)
$\langle W^-, W^+, W^H \rangle(newT)$:=compose_arcs($W^-(t), W^H(t), W^+(t) - \{p\}$,
gen_input(ϕ) ,gen_inhib(ϕ), $W^+(t_i) - (W^-(t_i) - \{p\})+$ gen_input(ϕ)));
if $newT$ is legal **then**
$RP(newT) \ := \ RP(T)\ RP(t_i) / \sum_{t \in Z} RP(t)$;
add_trans($newT$)
endif
endfor
endfor;
(* Case of no enabled transitions in CS after T firing *)
$\psi := \bigwedge_{t \in CS} \neg$ enab(t);
(* ψ is the conjunction of the disabling conditions of all transitions in CS *)
$\langle$Substitute in ψ each occurrence of $p \geq 1$ with $true$ $\rangle$;
if ψ is not a contradiction **then**
$newT$:= create_transition();
$\langle W^-, W^+, W^H \rangle(newT)$:= compose_arcs($W^-(T), W^H(T), W^+(T)$,
gen_input(ψ),gen_inhib(ψ) ,gen_input(ψ));
if $newT$ is legal **then**
$RP(newT) \ := \ RP(T)$;
add_trans($newT$)
endif
endif;
(* Delete transition T *)
delete_trans(T);
(*Delete vanishing places *)
foreach p in the new net such that $^\bullet p = \emptyset$ **do**
delete_place(p)
endfor
end.

Function enab transforms the description of input and inhibition arcs of a transition into a boolean expression whose operands are inequalities on the place markings. For example, if we take the GSPN in Figure 1.7 with an additional inhibitor arc from p_1 to t_2 then $enab(t_2) = p \geq 1 \ \wedge \ p_2 \geq 1 \ \wedge \ p_1 < 1$

Procedures gen_input(C_i) and gen_inhib(C_i) are complementary to enab since they generate the input and inhibitor multisets of a transition that has C_i as its enabling condition. A precondition for the applicability of these procedures to a given condition C_i is that it is expressed as a conjunction of simple inequalities on the place markings. For example, if $C_i = (p_1 \geq 1) \wedge (p_2 \leq 0)$ then gen_input(C_i) $= \{p_1\}$ gen_inhibit(C_i) $= \{p_2\}$. If we want to express a more complex condition (i.e. a disjunction of conjunctions of simple inequalities on the place markings: $C = \bigvee_j C_j$) then we need one transition per element C_j of the disjunction, each transition performing the same transformation of state.

These procedures can be used to generate the input, inhibition, and output multisets of a transition that performes a given state transformation x when a certain condition C_i is verified. Assuming that condition C_i already includes the proper requirements on the number of tokens present in the places

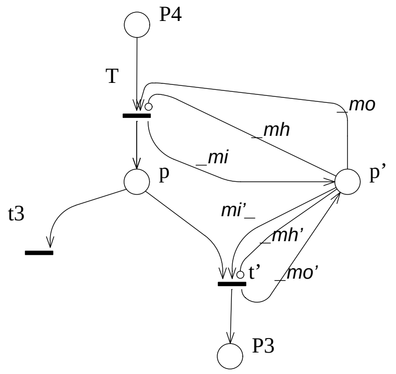


Figure 1.11: Reference figure for the arc composition table

whose marking is decremented by the state transformation, the input and inhibition arcs are simply gen_input(C_i) and gen_inhib(C_i) while the output is computed as $x+$gen_input(C_i).

Procedure delete_trans(t) (add_trans(t)) cancels (adds) transition t from (to) the net. Procedure delete_place(p) cancels place p and all the transitions in $p^\bullet$ from the net. Procedure compose_arcs generates the input, output, and inhibition multisets of a transition that simulates the sequential firing of two transitions whose input, output, and inhibition multisets are passed as arguments to the function itself. Let $\langle W^-(T), W^H(T), W^+(T)\rangle$ and $\langle W^-(t), W^H(t), W^+(t)\rangle$ be the multisets to be combined. If $(W^-(T) \cup W^H(T) \cup W^+(T)) \ \cap \ (W^-(t) \cup W^H(t) \cup W^+(t)) \ = \ \emptyset$ then the resulting multisets are obtained by union of the originating transitions multisets. If this is not the case (i.e. if the situation is similar to that depicted in Figure 1.11) then various possible combinations have to be considered. Observe that in combining the common input, output, and inhibition places we have to keep in mind that the enabling condition of the first transition must be reproduced in conjunction with the enabling condition of the second transition conditioned on the fact that the first transition has just fired.

Procedure compose_arcs generates the input, output, and inhibition multisetsets of a transition that simulates the sequential firing of two transitions whose input, output, and inhibition multisets are passed as arguments to the function itself.

Let $\langle W^-(T), W^H(T), W^+(T)\rangle$ and $\langle W^-(t), W^H(t), W^+(t)\rangle$ be the multisets to be combined. If $(W^-(T) \cup W^H(T) \cup W^+(T)) \ \cap \ (W^-(t) \cup W^H(t) \cup W^+(t)) \ = \ \emptyset$ then the resulting multiset is obtained simply by union of the originating transitions multisets. If this is not the case, i.e. if the situation is similar to that depicted in Figure 1.11, then various possible combinations have to be considered. Observe that in combining the common input, output, and inhibition places we have to keep in mind that we have to reproduce the enabling condition of the first transition in conjunction with the enabling condition of the second transition conditioned by the fact that the the first transition has just fired.

These possibilities are summarized in Table 1.1. The table is organized as follows: six cases are considered that cover all possible parameter combinations. For each case in the first three columns of the first line the arcs connecting transition T to place p' are specified. In the fourth column additional conditions are specified that are necessary for the liveness of the resulting composed transition. In the following lines the input, output, and inhibitor arcs of the composed transition (denoted by $T \bullet t$) are defined. Notice that, due to the convenience constraint that does not allow concurrency within a $H\Pi S$, if T increments the marking of p' then t cannot have p' as an input place (unless it is just a test place). This is expressed more formally at the top of Table 1.1.

Let us explain in some detail the simplest and the most complex cases that are respectively the first and the last one. Notice that to be sure that the resulting transition performs the same state transformation as the sequence of the two original transitions the following statement must hold in any case:

$$W^+(T \bullet t) - W^-(T \bullet t) = m_o + m'_o - m_i - m'_i \tag{1.2}$$

In the first case, the firing of T causes the decrement of the marking of place p'. In order for t to be enabled after the firing of T it is necessary that at least m'_i token are still present in p'. This

See reference Figure 1.11.
N.B. If $m_o > m_i$ then should be $m'_i = 0$

1)	$m_i > 0$	$m_o = 0$	$m_h = 0$	
	$W^-(T \bullet t) = m_i + m'_i$ $W^+(T \bullet t) = m'_o$ if $m'_h = 0$ then $W^H(T \bullet t) = 0$ else $W^H(T \bullet t) = m_i + m'_h$			
2)	$m_i \geq 0$	$m_o > 0$	$m_h = 0$	$(m'_h = 0) \vee (m'_h > (m_o - m_i))$
	if $m_o > m_i$ then $W^-(T \bullet t) = m_i$ else $W^-(T \bullet t) = max(m_i, m'_i + m_i - m_o)$ $W^+(T \bullet t) = m_o + m'_o - m_i - m'_i + W^-(T \bullet t)$ if $m'_h = 0$ then $W^H(T \bullet t) = 0$ else $W^H(T \bullet t) = m'_h - (m_o - m_i)$			
3)	$m_i = 0$	$m_o = 0$	$m_h > 0$	$(m_h > m'_i)$
	$W^-(T \bullet t) = m'_i$ $W^+(T \bullet t) = m'_o$ if $m'_h = 0$ then $W^H(T \bullet t) = m_h$ else $W^H(T \bullet t) = min(m_h, m'_h)$			
4)	$m_i > 0$	$m_o = 0$	$m_h > 0$	
	$W^-(T \bullet t) = m_i + m'_i$ $W^+(T \bullet t) = m'_o$ if $m'_h = 0$ then $W^H(T \bullet t) = m_h$ else $W^H(T \bullet t) = min(m_h, m'_h + m_i)$			
5)	$m_i = 0$	$m_o > 0$	$m_h > 0$	$m'_h > m_o$
	$W^-(T \bullet t) = 0$ $W^+(T \bullet t) = m_o + m'_o$ if $m'_h = 0$ then $W^H(T \bullet t) = m_h$ else $W^H(T \bullet t) = min(m_h, m'_h - m_o)$			
6)	$m_i > 0$	$m_o > 0$	$m_h > 0$	$(m'_h = 0) \vee (m'_h > (m_o - m_i))$
	if $m_o > m_i$ then $W^-(T \bullet t) = m_i$ else $W^-(T \bullet t) = max(m_i, m'_i + m_i - m_o)$ $W^+(T \bullet t) = m_o + m'_o - m_i - m'_i + W^-(T \bullet t)$ if $m'_h = 0$ then $W^H(T \bullet t) = m_h$ else $W^H(T \bullet t) = min(m_h, m'_h - (m_o - m_i))$			

Table 1.1: Definition of function compose_arcs

corresponds to having at least $m_i + m'_i$ tokens before firing the sequence "T,t": hence this is the correct value for $W^-(T \bullet t)$. $W^+(T \bullet t)$ can be computed from Equation (1.2) substituting the expression $m_i + m'_i$ for $W^-(T \bullet t)$ and taking into account the hypothesis that $m_o = 0$. $W^H(T \bullet t)$ is obtained as follows: if $m'_h = 0$ then, being also $m_h = 0$, we need no inhibitor arcs for the composition of the two transitions. If on the contrary $m'_h > 0$ then we require that p' contains less than m'_h tokens after the decrement by m_i tokens due to the firing of T. This means having less than $m'_h + m_i$ tokens in p' before firing T.

In the last case, we have an additional condition on the fourth column; if this condition is not satisfied the sequence $T \bullet t$ can never fire. In fact if t is inhibited by the presence of at least m'_h tokens in p' then we must check that T does not increment p' of such a number of tokens preventing t from firing. Since both m_i and m_o are positive, we have to distinguish between two cases: $m_o > m_i$ (i.e., T increments the marking of p') and $m_o \leq m_i$ (i.e., T does not increment the marking of p'). If $m_o > m_i$ then m'_i is equal to zero by hypothesis and $W^-(T \bullet t)$ is simply m_i. If $m_o \leq m_i$ then m'_i could be positive. In this case we need that at least m'_i tokens are present in p' after the decrement of $(m_i - m_o)$ tokens performed by T. Moreover, in order for T to be able to fire we need at least m_i tokens in p'. Thus we choose the maximum between m_i and $m'_i + m_i - m_o$ as the value for $W^-(T \bullet t)$. The value of W^+ is computed from W^- using Equation (1.2).

Concerning the inhibition of $T \bullet t$, if $m'_h = 0$ then we must only check that the inhibition condition for T is satisfied. Otherwise, we need on one hand that the marking of p' is less than m_h before firing T, and on the other hand that it is less than m'_h after the firing of T. Therefore, if the effect of firing T is to decrement p' marking of $m_i - m_o$ tokens (i.e., if $m_o < m_i$) then it is sufficient to have less than $m'_h + (m_i - m_o)$ tokens in p' before firing T in order to satisfy the inhibition condition of t after its firing. If instead the effect of firing T is to increment p' marking of $m_o - m_i$ tokens (i.e., if $m_i < m_o$) then it is necessary to have less than $m'_h - (m_o - m_i)$ tokens in p' before the firing of T. Notice that the quantity required in the two cases is exactly the same. At this point we can combine the two requirements and obtain the expression in the table: $(p' < m_h) \wedge (p' < m'_h - m_o + m_i)$.

Notice that the prerequisite in the fourth column of case 6, is needed to enforce a positive value of W^H. Anyway, it may still happen that $W^-(T \bullet t) \geq W^H(T \bullet t)$: also in this case the two transitions can never fire one after the other. Notice also that some combinations are not admitted, meaning that the enabling conditions of the resulting transition could never be satisfied, so that the transition should not be generated.

Bibliography

[1] 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), May 1984.

[2] M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems.* MIT Press, Cambridge, USA, 1986.

[3] T. Murata. Petri nets: properties, analysis, and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[4] M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte. Generalized stochastic Petri nets revisited: Random switches and priorities. In *Proc. Int. Workshop on Petri Nets and Performance Models*, pages 44–53, Madison, WI, USA, August 1987. IEEE-CS Press.

[5] M. Ajmone Marsan, G. Balbo, G. Chiola, G. Conte, S. Donatelli, and G. Franceschinis. An introduction to Generalized Stochastic Petri Nets. *Microelectronics and Reliability*, 31(4):699–725, 1991. Special issue on Petri nets and related graph models.

[6] M.K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models.* PhD thesis, UCLA, Los Angeles, CA, 1981. Ph.D. Thesis.

[7] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transaction on Computers*, 31(9):913–917, September 1982.

[8] S. Natkin. *Les Reseaux de Petri Stochastiques et leur Application a l'Evaluation des Systemes Informatiques.* PhD thesis, CNAM, Paris, France, 1980. These de Docteur Ingegneur.

[9] G. Florin and S. Natkin. Les reseaux de Petri stochastiques. *Technique et Science Informatiques*, 4(1), February 1985.

[10] W. Reisig. *Petri Nets: an Introduction.* Springer Verlag, 1985.

[11] E. Best and C. Fernandez. Notations and terminology on Petri net theory. Technical Report 195, GMD, St. Augustin, West Germany, 1986.

[12] G. Chiola, S. Donatelli, and G. Franceschinis. On parametric P/T nets and their modelling power. In *Proc.* 12^{th} *International Conference on Application and Theory of Petri Nets*, Aarhus, Denmark, June 1991.

[13] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. On well-formed coloured nets and their symbolic reachability graph. In *Proc.* 11^{th} *International Conference on Application and Theory of Petri Nets*, Paris, France, June 1990. Reprinted in *High-Level Petri Nets. Theory and Application*, K. Jensen and G. Rozenberg (editors), Springer Verlag, 1991.

[14] G. Chiola and G. Franceschinis. A structural colour simplification in Well-Formed coloured nets. In *Proc. 4th Intern. Workshop on Petri Nets and Performance Models*, pages 144–153, Melbourne, Australia, December 1991. IEEE-CS Press.

[15] G. Rozenberg and P.S. Thiagarajan. Petri nets: basic notions, structure, behaviour. In W. Brawer, W. Reisig, and G. Rozenberg, editors, *Current Trends in Concurrency*, volume 224 of *LNCS*, pages 585–668. Springer Verlag.

[16] M. Ajmone Marsan, S. Donatelli, and F. Neri. GSPN models of Markovian multiserver multiqueue systems. *Performance Evaluation*, 11(4):227–240, 1990.

[17] M. Ajmone Marsan, S. Donatelli, F. Neri, and U. Rubino. On the construction of abstract GSPNS: an exercise in modeling. In *Proc. 4th Intern. Workshop on Petri Nets and Performance Models*, pages 2–17, Melbourne, Australia, December 1991. IEEE-CS Press.

[18] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, 15(7):832–846, July 1989.

[19] G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte. Generalized stochastic Petri nets: Definition at the net level. Submitted for publication, 1989.

[20] G. Balbo, G. Chiola, G. Franceschinis, and G. Molinar Roet. On the efficient construction of the tangible reachability graph of generalized stochastic Petri nets. In *Proc. Int. Workshop on Petri Nets and Performance Models*, Madison, WI, USA, August 1987. IEEE-CS Press.

[21] G. Chiola, S. Donatelli, and G. Franceschinis. Priorities, inhibitor arcs, and concurrency in P/T nets. In *Proc.* 12^{th} *International Conference on Application and Theory of Petri Nets*, Aarhus, Denmark, June 1991.

[22] Giovanni Chiola. *GreatSPN* 1.5 software architecture. In *Proc.* 5^{th} *Int. Conf. Modeling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy, February 1991.