

Toward a Definition of Modeling Power for Stochastic Petri Net Models

Gianfranco Ciardo
Duke University

Proc. of the Int. Workshop on Petri Nets and Performance Models, Madison, Wisconsin, August 1987

Abstract

Some insight on the meaning of “modeling power” for Stochastic Petri Net models is given. Extensions characterizing a Stochastic Petri Net are categorized as logical or stochastic. Three logical constructs are shown to be equivalent: inhibitor arcs, transition priorities, and enabling functions associated with the transitions. A direct transformation of Petri Nets with inhibitor arcs into Petri Nets with transition priorities and vice versa is given, determining a bound on the size of equivalent nets in the two models. As a consequence, the higher priority of immediate transitions over timed transitions in the GSPN model is shown to provide the full power of Turing machines.

1 Introduction

The Petri net (PN) model has been extended in several ways by researchers in the performance or reliability field. These extensions embed the model into a timed environment, usually by associating a time to the transitions (sometimes to the places) of the PN. The most general extensions, allowing the usage of stochastic times and probabilities [1, 2, 3, 4, 5, 6, 7, 8], can generally be classified as Stochastic PN models (SPN).

It is certainly interesting to compare different extensions, either to evaluate them individually, or to decide the best suited to model a specific system. The measuring stick for this purpose is the *modeling power* of the model, a concept only partially understood, and never formally defined. We will not try to fill this gap, but we will at least try to clarify some concepts, as a starting point toward a formal definition.

Section 2 introduces the notation used. Section 3 illustrates the separation between logical and stochastic constructs, and, correspondingly, between logic and stochastic modeling power. Section 4 discusses the logical modeling power, and presents a definition useful to compare it for different SPN models. Section 5 contains the proof of equivalence of three different logical constructs, the main result of this paper.

⁰This research was supported in part by an IBM Graduate Fellowship and in part by the Air Force Office of Scientific Research under grant AFOSR-84-0132.

2 Notation

In the following, we assume the reader is familiar with the basic PN model. The main terms are recalled in the appendix. The notions of multiple arc, enabling function, inhibitor arc, and transition priority are reviewed in the next section.

Definition 1 Let's define PN_e as the set of standard PN with the addition of enabling functions on the transitions, PN_i as the set of standard PN with the addition of inhibitor arcs, and PN_p as the set of standard PN with the addition of transition priorities.

Definition 2 Given a (marked) PN A , we define the following symbols:

P_A : the set of places in A .

$|p|$: the number of tokens in place p ($p \in P_A$).

T_A : the set of transitions in A .

$I_A \subseteq P_A \times T_A$: the set of input arcs in A .

$O_A \subseteq T_A \times P_A$: the set of output arcs in A .

$H_A \subseteq P_A \times T_A$: the set of inhibitor arcs in A (if $A \in PN_i$).

$R_A = (V_A, E_A)$: the reachability graph of A . The node set V_A consists of the reachable markings, and the arc set E_A consists of the transition firings from marking to marking. Furthermore the arcs, or RG-transitions, are labeled with the corresponding (PN-)transition: if the initial marking reaches m_1 , t is enabled in m_1 , and m_2 is obtained firing t in m_1 (we write $m_1 \xrightarrow{t} m_2$), then $m_1, m_2 \in V_A$, and $(m_1, m_2) \in E_A$.

If arc multiplicity is allowed, we indicate $(x, y)[k]$ an arc from x to y with multiplicity k . The multiplicity is indicated only when allowed and needed.

3 Logical and stochastic constructs

In performance evaluation and reliability analysis, the standard PN model has been embedded into a timed environment. A constant or random time is associated to either the places or the transitions (not both) of the PN. In the following we assume that time is associated to the transitions.

Most SPN models include additional extensions with the (usually undeclared) goal of increasing the flexibility, but often the improvements are not discussed. We try to examine these extensions, starting with a fundamental distinction among them: *logical constructs* change the firing rule of the transitions; *stochastic constructs* associate a stochastic interpretation to the evolution of the PN.

Examples of logical constructs found in SPN models are (see figure 1):

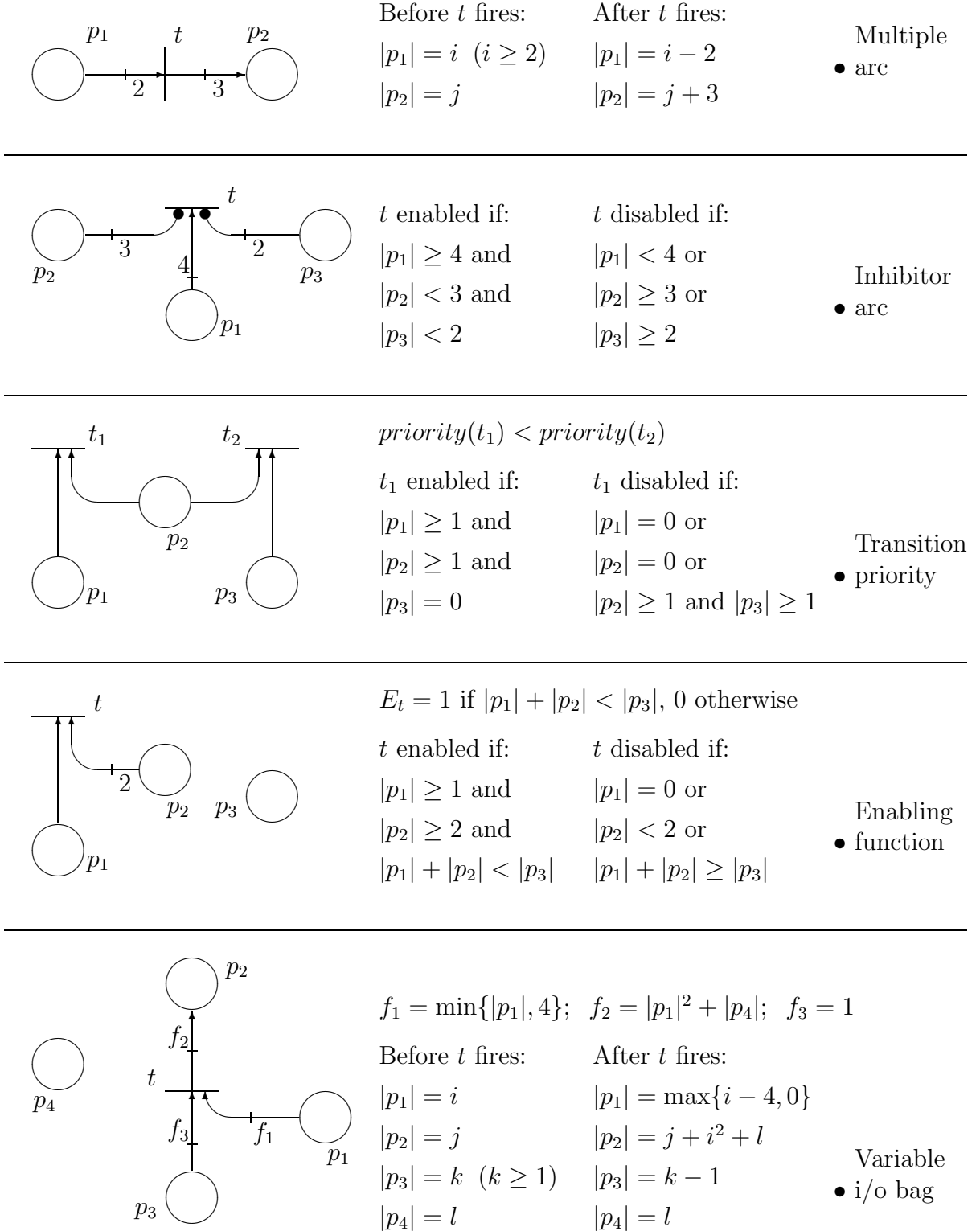


Figure 1: *Some Petri net constructs*

- *multiple arc* [3, 4, 5, 6, 7, 8]
A multiplicity can be associated to the arcs input or output arcs. This extension is almost unanimously used, so it is often included into the standard definition of PN. An input arc from p to t with multiplicity k requires k tokens in p to enable t , and it causes their removal upon firing. An output arc with multiplicity k from t to p causes the addition of k tokens in p when t fires.
- *inhibitor arc* [4, 5, 6]
An inhibitor arc with multiplicity k from p to t disables t if k or more tokens are in p . In particular, if the multiplicity is 1, t is disabled unless p is empty.
- *transition priorities* [5, 6]
If each transition is assigned a fixed priority (non negative integer), the enabling rule is modified so that, in each marking, only the enabled transitions with the highest priority are really enabled, while the remaining ones are disabled.
- *enabling functions* [5, 8]
An enabling function $E_t : N^P \rightarrow \{0, 1\}$ (where P is the number of places) can be defined on each transition t ; if $E_t(m) = 1$, t is enabled in marking $m = (|p_1|, |p_2|, \dots, |p_P|)$, otherwise t is disabled.
- *variable input/output bag* [8]
The input or output bags can be defined as a function of the marking, similarly to the enabling function, instead of being fixed (determined by the arcs of the PN). This extension, especially if used together with the previous one, deeply changes the nature of the PN. The model is described by a set of $3T$ functions (input function, output function, and enabling function), where T is the number of transitions, plus the initial marking, a vector of P integers, but this information is difficult, if not impossible, to represent graphically. Furthermore, no study on the analysis of net invariants or other logical properties of PN with variable input/output bags has been done, nor does it seem likely that it can be done.

The category of stochastic constructs can be subdivided into two groups:

- *probability distribution for the firing times of the transitions*
This broad category encompasses many different choices, according to the allowed distribution types and other features:
 - constant times [9] *vs.* random times [1, 3, 4, 5, 6, 7, 8]
 - discrete times [9, 1, geometric] *vs.* continuous times [3, 4, 6, 8, 1, exponential]
 - memoryless distributions (geometric for discrete time, exponential for continuous time) [1, 3, 7, 6, exponential-only] *vs.* general distributions [4, 6, phase-type]
 - state change policies [5, 8], describing how to consider the remaining firing time for each enabled transition when the SPN changes the marking because of a transition firing.

- interruption policies [5, 8], describing how to consider the remaining firing time for transitions once enabled, then disabled, then subsequently enabled again.
- *relative probability of firing for conflicting transitions*
Two different interpretations exist, when several transitions are in conflict. The transition(s) to fire can be *preselected* independently of their firing times using relative probabilities supplied by the user or automatically determined with some rule [7], or the *minimum firing time rule* can be used, so that only the transition(s) scheduled to fire at the closest future time among the enabled ones may fire [1, 3, 4, 5, 6, 8]. This second policy may still need the specification of probabilities, since contemporary events may have a non-null probability. Constructs used to specify the probabilities needed for conflicting transitions are:
 - random switch [3], defined as a set of immediate transitions (transitions having firing distribution equal a null constant, so that they always fire as soon as they are enabled), and their relative probability of firing.
 - probabilistic arc [4, 5], defined as an arc of an hypergraph, connecting a transition to a set of places, so that its firing causes the placement of a token in exactly one of the places in the set, according to a certain probability distribution.
 - input or output case [8], respectively (roughly) equivalent to the random switch and the probabilistic arc.

The interested reader should consult the references for more information on stochastic extensions. In the following we focus on the logical extensions.

We believe that the proliferation of SPN “models” should be better considered as the proliferation of SPN “packages”: a question like “*Does this model allow inhibitor arcs?*” should be often expressed as “*Were the inhibitor arcs implemented in the package for this model?*”, because this decision is often left to the implementor. The real question is “*How different this model really is from other analogous models?*” The answer must be based on the true modeling power of the model, a concept different from the flexibility (constructs) of the related package.

It has been shown [10] that PN with inhibitor arcs have the same representational power as Turing machines (TM). Since TM are the most general model of computation known, it may be concluded that PN with inhibitor arcs can represent any situation, any system, so that other constructs, and in particular the stochastic constructs, are redundant.

In practice we do not use the SPN as a TM. We want instead to use the SPN to model the system, where the markings reachable in the underlying PN roughly correspond to the possible states of the system when time is not considered, while the times between change of marking in the SPN correspond to activities in the system (this distinction is sometimes blurred, for example the reader may think of the presence of a subnet with exponentially distributed transitions to simulate an Erlang distribution).

If the distinction between logical and stochastic constructs is kept, the modeling power of a SPN model can be divided into two aspects: logical modeling power, and stochastic modeling power. The first aspect describes what (discrete) state spaces can be expressed by the underlying PN, and the difficulty to express them. The second aspect describes what stochastic processes can be mapped to these state spaces.

A third consideration could be done, about the difficulty of studying a SPN to obtain certain measures from it: certain packages can easily compute a wide range of measures defined on the net. For example, the presence of general distributions usually allows a simulative study only; also, if all transitions have exponential distributions, a transient analysis is usually more difficult than a steady analysis. On the other side, the difficulty of the solution is determined by the type of stochastic process and of required measures, so it should be the same for the same study on the same system, independently of the particular SPN model chosen.

In the following section we consider the logical modeling power of SPN models, while we leave the much harder problem of categorizing the stochastic power of SPN models to future work.

4 Logical level of modeling

We consider the logical level of modeling as dealing with the problem of generating the correct state space (reachability graph) for a certain system. Note that it may be impossible to completely free this problem from timing considerations. For example, the reachability graph of a GSPN with immediate transitions can be smaller if the transition type (timed or immediate) is considered. This difference in the reachability graphs is eliminated by using additional logical constructs in the underlying PN, namely by explicitly assigning a higher priority to the immediate transitions.

More difficult is the presence of non-null constant firing times. The reachability graph of the underlying PN may properly contain the one obtained when timing is considered, but the state of the system is not completely described by the marking of the SPN, since the remaining firing times for the (non memoryless) transitions is part of the state [7].

We ignore these difficulties since the easiest scenario (all exponentially distributed firing times) is interesting enough for our purposes.

We talk about equivalence of two PN models by finding a relationship between their reachability graphs. If a PN A has a finite reachability graph, a trivially equivalent PN can be defined, having one place for each marking in V_A , one input arc, one transition, and one output arc for each RG-transition in E_A , and one token in the place corresponding to the initial marking of A . This construction shows that any PN with finite reachability set can be represented (in any reasonable sense) by a large but simple PN, being conservative and safe (exactly one token in the PN in any configuration). This consideration shows how the study of the equivalence of PN models should be concerned with the *sizes* of equivalent PN.

For PN with infinite reachability graphs, the same consideration to the size should be given, but, in addition, a theoretical difference may exist, according to the extensions allowed in the PN. For example consider the PN $\in PN_i$ presented in figure 2, together with its reachability graph. The reader familiar with Peterson's book [11, page 175] will recognize that we borrowed his example, where standard PN are said to be unable to generate the language $L = \{w : \exists n w \in (a^n b^n)^*\}$. We simply translated the example from the realm of firing sequences and language theory to the one of reachability graphs. It is clear how no standard (finite) PN can generate the reachability graph of figure 2, otherwise it would generate the language L , if t_2 is labeled with the character a and t_3 is labeled with the character b . So,

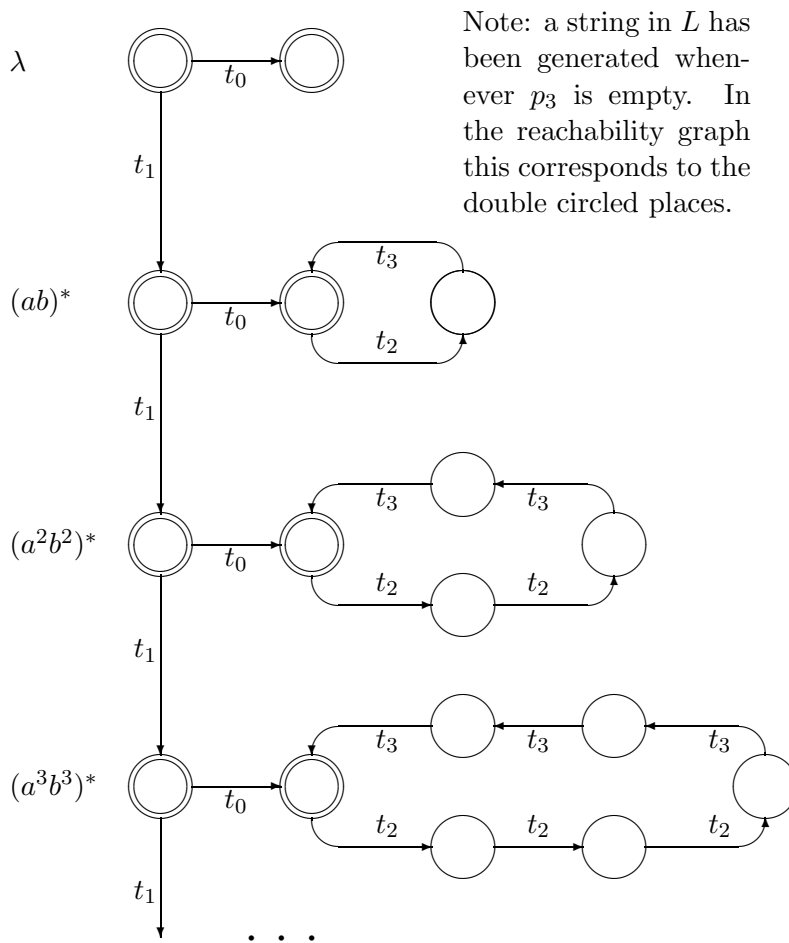
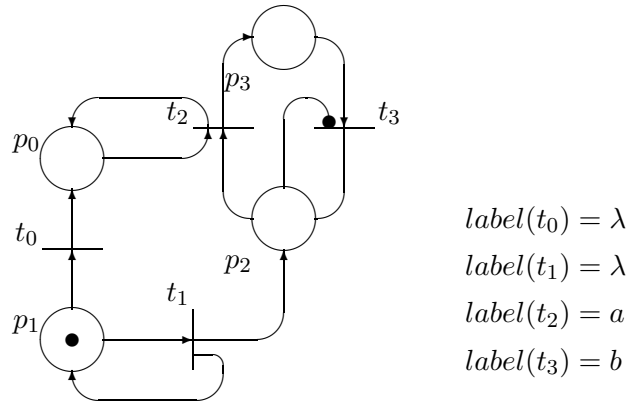


Figure 2: A Petri net generating $L = \{w : \exists n w = (a^n b^n)^*\}$ and its reachability graph

there exist infinite reachability graphs that cannot be generated by a standard PN of any size, while they can be generated by a $PN \in PN_i$ (note that there exists infinite graphs that cannot be generated by PN_i , since the set of infinite graphs is not countable, while PN_i is countable, even considering marked PN).

We give now a definition that allows us to compare two PN. Note that it is formulated as a containment relationship, not as an equivalence one.

Definition 3 A PN A with place set P_A , transition set T_A , initial marking m_A , and reachability graph R_A is said to RG-simulate a marked PN B with place set P_B , transition set T_B , initial marking m_B , and reachability graph R_B if:

1. (Transition mapping) $\exists \tau : T_A \rightarrow T_B \cup \{\epsilon\}$, $\epsilon \notin T_B$
2. (Marking mapping) $\exists \mu : V_A \rightarrow V_B$
3. (Consistency) τ and μ satisfy:
 - $\mu(m_A) = m_B$
 - $\forall k > 0 \forall m_{i_0}, m_{i_1}, \dots, m_{i_k} \in V_A \forall t_{i_1}, t_{i_2}, \dots, t_{i_k} \in T_A$
 $m_{i_0} \xrightarrow{t_{i_1}} m_{i_1} \xrightarrow{t_{i_2}} m_{i_2} \dots \xrightarrow{t_{i_k}} m_{i_k} \implies \forall j \ 1 \leq j \leq k :$
 $(\mu(m_{i_{j-1}}) = \mu(m_{i_j}) \wedge \tau(t_{i_j}) = \epsilon) \vee \mu(m_{i_{j-1}}) \xrightarrow{\tau(t_{i_j})} \mu(m_{i_j})$
 - $\forall k > 0 \forall \mu_{i_0}, \mu_{i_1}, \dots, \mu_{i_k} \in V_B \forall \tau_{i_1}, \tau_{i_2}, \dots, \tau_{i_k} \in T_B$
 $\mu_{i_0} \xrightarrow{\tau_{i_1}} \mu_{i_1} \xrightarrow{\tau_{i_2}} \mu_{i_2} \dots \xrightarrow{\tau_{i_k}} \mu_{i_k} \implies \exists c \geq k$
 $\exists m_{i_0}, m_{i_1}, \dots, m_{i_c} \in V_A \exists t_{i_1}, t_{i_2}, \dots, t_{i_c} \in T_A :$
 $m_{i_0} \xrightarrow{t_{i_1}} m_{i_1} \xrightarrow{t_{i_2}} m_{i_2} \dots \xrightarrow{t_{i_c}} m_{i_c} \wedge \forall j \ 1 \leq j \leq c :$
 $(\tau(t_{i_j}) = \epsilon \implies \mu(m_{i_{j-1}}) = \mu(m_{i_j})) \wedge$
 $(\tau(t_{i_j}) \neq \epsilon \implies \exists l \ 1 \leq l \leq k : \mu(m_{i_{j-1}}) = \mu_{i_{l-1}} \wedge \mu(m_{i_j}) = \mu_{i_l} \wedge \tau(t_{i_j}) = \tau_{i_l})$

The above definition is not intended to be applied when T_B contains transitions whose input and output bag are identical (whose firing does not change the marking). We do not to give a definition including these “self-transitions” for simplicity only.

The transition mapping condition is needed to interpret the firing of a transition in A in terms of a transition in B : a transition in A can either correspond to exactly one transition in B , or to no transition at all, if it is mapped to ϵ . Transitions in A mapped to ϵ by τ can be considered as performing bookkeeping in A .

The marking mapping condition states that any marking of A must correspond to a marking in B , so that the state of B can always be obtained by considering the state of A .

The consistency condition is the most important, since τ and μ can always be defined. The reachability graphs R_A and R_B must satisfy a requirement: there must exist a surjective function from the paths in R_A to the paths in R_B , in a way consistent with the τ and μ mappings. In particular, the firing of a bookkeeping transition changes the marking of A , but it does not change the image of the marking of A in B through μ .

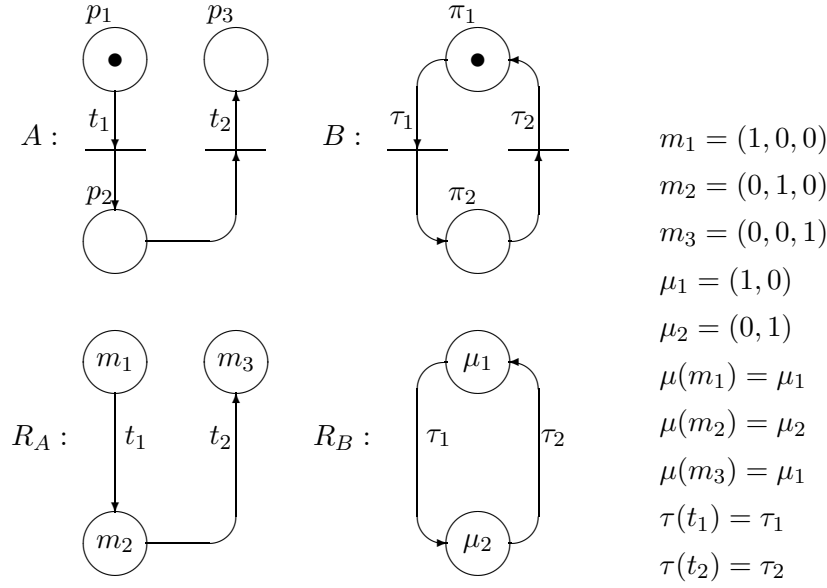


Figure 3: A does not RG-simulate B

If the reader is wondering why transition firing sequences of arbitrary length must be considered, and not single transition firings, figure 3 may clarify: A would RG-simulate B if the definition were restricted to single transition firings, while A does not simulate B in most reasonable interpretations, surely not in the one we are interested in, since A fires exactly twice, then stops, while B is alive (a transition is alive if from any reachable marking there exist a firing sequence leading to a marking enabling the transition; a PN is alive if all its transitions are alive).

The marking mapping μ implies a place mapping π , but in a broad sense. It may not be true that $\exists \pi : P_A \rightarrow P_B \cup \{\epsilon\}$ such that, for each place $p \in P_B$, $|p|$ can be obtained as a linear combination of $|q_i|$, for $q_i \in \{q : \pi(q) = p\}$, for each marking in V_A . If π exists, it is possible to define μ starting from it, instead of the other way around. In the constructions of the theorems we present, π exists.

This definition of simulation was chosen because it is useful for our purposes. The reader can foresee that the transitions of A mapped to ϵ can be defined as immediate, so that the bookkeeping activity is instantaneous, and the stochastic processes represented by A and B may coincide.

5 Some equivalent PN extensions

Theorem 1 $PN_i \equiv PN_p \equiv PN_e$, in the sense that:

$\forall A \in PN_e, \exists B \in PN_i : B$ RG-simulates A ,

$\forall A \in PN_i, \exists B \in PN_p : B$ RG-simulates A ,

$\forall A \in PN_p, \exists B \in PN_e : B$ RG-simulates A .

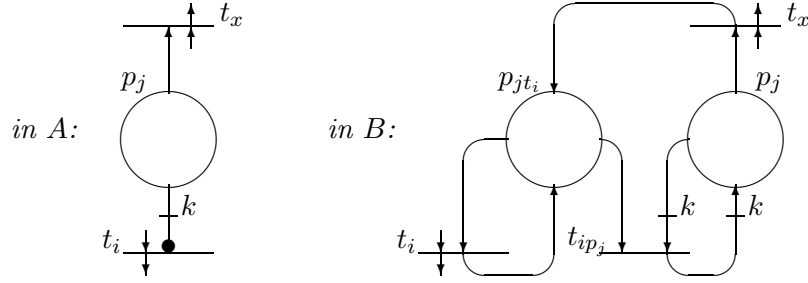


Figure 4: *Representation of an inhibitor arc*

We prove this theorem only informally. In [10] it is shown that PN_i can simulate any TM, and a PN in PN_e can be simulated by a TM, since the enabling functions are computable. The proof of the second statement is given in the next theorem, together with a bound on the size of $B \in PN_p$ simulating $A \in PN_i$. Finally, PN_e can be considered a generalization of PN_p , since an enabling function E_t can be defined to evaluate to 1 only when a certain set S of transitions is not enabled, a condition describable using only the number of tokens in certain places. If S corresponds to the set of transitions having higher priority than t , then t is enabled only if the transitions in S are disabled, and this is exactly what can be achieved using priorities. Incidentally, a similar construction shows that PN_e are a generalization of PN_i as well. \square

In the two following theorems we deal with the *size* of PN in PN_i RG-simulating PN in PN_p , and vice versa. No interesting statement can be made on the size of PN in PN_i or PN_p RG-simulating PN in PN_e , since the enabling functions can be arbitrarily complex, requiring arbitrarily large PN in PN_i or PN_p to simulate them.

Theorem 2 Given $A \in PN_i$, with P places, T transitions, and H inhibitor arcs, $\exists B \in PN_p$, with at most $P + H$ places, $T + H$ transitions, and two levels of priority that RG-simulates it.

Proof. We give a constructive proof where each inhibitor arc is simulated by adding a place and a transition with high priority. Consider a PN $A \in PN_i$ with $(p_j, t_i)[k] \in H_A$, and p_j input to other transitions, for example t_x . In figure 4, the relevant portion of A appears, together with its “translation” in B .

Let’s see on the example of figure 4 how the simulation works. The goal is to have a token in p_{jt_i} if and only if $|p_j| < k$, so that t_i is disabled if $|p_j| \geq k$. The initial marking of B has $|p_{jt_i}| = 1$ if $|p_j| < k$, 0 otherwise. Transition t_{ip_j} eliminates the token from p_{jt_i} as soon as $|p_j| \geq k$, for example after a transition adds tokens in p_j , because it has higher priority. On the other side, every time a transition t_x removes tokens from p_j , $|p_j|$ may have been decreased below k , and a token should be put in p_{jt_i} . To achieve this behavior, an output arc from transition t_x to p_{jt_i} is added, the token is removed by t_{ip_j} if p_j still contains at least k tokens.

To show that B RG-simulates A , we give the τ and μ functions, while the proof that the consistency condition is met is not given for brevity (the reader can easily check it).

If $T_A = \{t_1, t_2, \dots, t_n\}$, then $T_B = T_A \cup \{t_{jp_i} : (p_i, t_j) \in H_A\}$, and τ can be defined as:

$$\forall t \in T_B : \tau(t) = \begin{cases} t & \text{if } t \in T_A \quad (t \text{ has low priority}) \\ \epsilon & \text{otherwise} \quad (t \text{ has high priority}) \end{cases}$$

that is, the additional transitions are for bookkeeping only, while the remaining transitions in T_B have an exact match in T_A .

The marking mapping is defined starting from a place mapping. If $P_A = \{p_1, p_2, \dots, p_l\}$, then $P_B = P_A \cup \{p_{it_j} : (p_i, t_j) \in H_A\}$, and π can be defined as:

$$\forall p \in P_B : \pi(p) = \begin{cases} p & \text{if } p \in P_A \\ \epsilon & \text{otherwise} \end{cases}$$

while μ is defined as (P_B contains l' places):

$$\mu(|p_1|, |p_2|, \dots, |p_l|, |p_{l+1}|, \dots, |p_{l'}|) = (|p_1|, |p_2|, \dots, |p_l|)$$

that is, the additional places do not influence the μ mapping, or, in other words, markings in V_B differing only in the additional places are mapped by μ to the same marking of V_A . \square

Note In the definition of GSPN, immediate transitions have priority over timed transitions, so two levels of priority are implicitly defined. Then GSPN have an underlying TM model at the logical level, even if inhibitor arcs and enabling functions are not allowed, and if the marking dependent functions expressing rates and probabilities cannot evaluate to zero (otherwise they would *de facto* constitute an enabling function).

Theorem 3 Given $A \in PN_p$ with P places, T transitions, and I input arcs, $\exists B \in PN_i$ with at most $P + T + 3$ places, $2T + I + 2$ transitions, and $T + 2I + 1$ inhibitor arcs that RG-simulates it.

Note that the bound is the relevant portion of the theorem, since it is easy to see that a TM can simulate any $A \in PN_p$, and we know that TM can be simulated by PN_i .

Proof. Also for this theorem we give a constructive proof; unfortunately the transition priority mechanism is not localized, so the equivalent net is complex. We follow figure 5. If $P_A = \{p_1, p_2, \dots, p_l\}$, then $P_B = P_A \cup \{p_{t_i} : t_i \in T_A\} \cup \{p_{FILL}, p_{EMPTY}, p_{COUNT}\}$. If $T_A = \{t_1, t_2, \dots, t_n\}$, then $T_B = T_A \cup \{t_{ip_j} : (p_j, t_i) \in I_A\} \cup \{t_{t_i} : t_i \in T_A\} \cup \{t_{FILL}, t_{EMPTY}\}$. The description of the input, output, and inhibitor arcs in B is cumbersome, so we illustrate it only for the transitions in figure 5. Let's define $N(t)$ as the number of transitions in T_A having higher priority than $t \in T_A$. In particular, $N(t) = 0$ if t is of highest priority.

The evolution of B iterates between two phases. In phase 1 (e.g., in the initial marking), a token is in p_{FILL} , and the transitions declare whether they are disabled by adding a token to p_{COUNT} (initially empty). Initially only the transitions with highest priority can declare; if each of them declares to be disabled, then the transitions with the next highest priority can declare, and so on, until a transition does not declare to be disabled, and fires (all the transitions may declare to be disabled, if A can reach a marking where no transition is enabled). When that transition fires, B enters phase 2: the token has been moved from p_{FILL} to p_{EMPTY} , and all the places of the form p_{t_i} plus p_{COUNT} need to be emptied. This

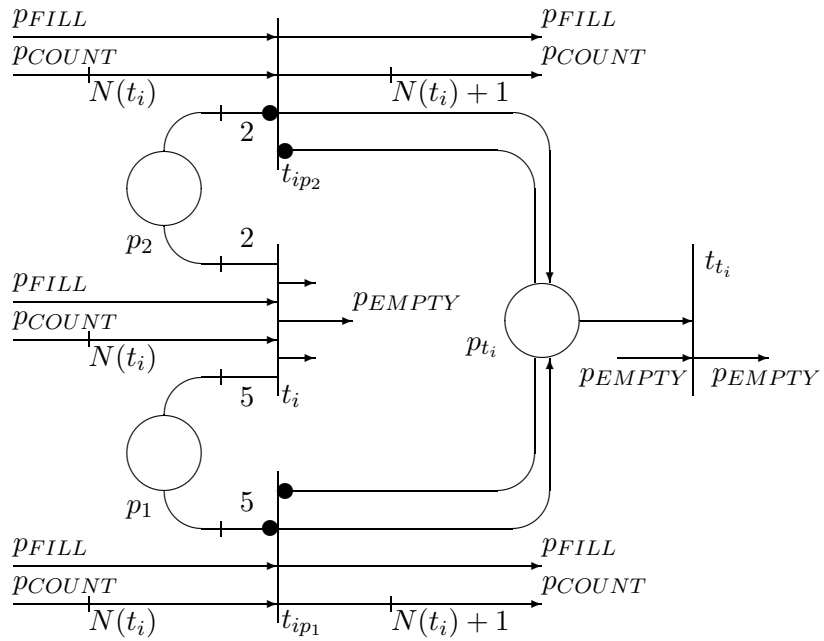
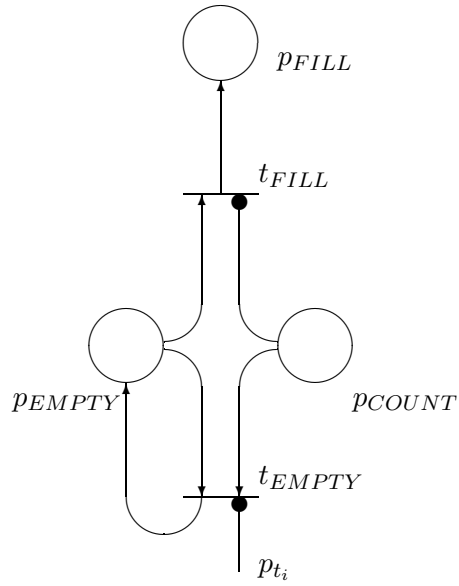


Figure 5: Representation of priorities

is achieved first by firing the enabled transitions of the form t_{t_i} , then by repeated firings of t_{EMPTY} . When p_{COUNT} is finally empty, t_{FILL} can fire, the token is moved from p_{EMPTY} back to p_{FILL} , and a new phase 1 can start.

The main difficulties are performing the declarations and sequentializing them in the correct order. The declaration is achieved by adding a “declaring transition” t_{ip_j} for each input place p_j to transition t_i , firing only if the input condition expressed by the arc is not satisfied, and if no other input condition has already been declared unsatisfied (otherwise t_i would multiply declare, and the number of tokens in p_{COUNT} would not reflect the number of *different* transitions that have already declared: place p_{t_j} is needed for this test). The sequentialization is achieved by having an input arc from p_{COUNT} to t_{ip_j} and to t_i , with cardinality $N(t_i)$, so that t_i can either fire or declare to be disabled only if each transition with priority higher than t_i has already declared to be disabled.

The definition of the τ and μ (π) mappings, is exactly the same as the one for the previous theorem, that is, the additional places and transitions are needed for bookkeeping only. Again we leave to the reader the proof that the consistency condition holds. \square

Since there is at least one transition of lowest priority, let’s call it t_k , the bounds of the theorem could be decreased by one, corresponding to the elimination of p_{t_k} from P_B , t_k from T_B , and (p_{t_k}, t_{EMPTY}) from H_B (lowest priority transitions do not need to declare whether they are disabled).

Comment In [12], M. Hack proved a similar equivalence between PN_i and PN_p , we outline here the main differences. There the interest is in the languages generated by the PN. Avoiding the introduction of λ -transitions, transitions not labeled with any character of the alphabet, is the goal. The corresponding construct in the SPN world is the immediate transition, widely accepted and freely used. Similarly, the size of equivalent nets is not given relevance in [12], while it is fundamental in practice, when SPN are used to model a system. Given these basic differences in the approach, it is not surprising to observe differences in the results.

Hack proves that if a language L is generated by $A \in PN_i$, then L can be generated by $B \in PN_p$, where B does not contain λ -transitions, but no bound in the size of B is given (it appears to be exponential in the size of A); furthermore, many priority levels may be needed. We have proved that the same simulation can be achieved, if the restriction on the λ -transition is eliminated, with a bound on the size of B . The bound is given by the number of inhibitor arcs, of the order of the product $P \times T$, or, in other words quadratic in the number of nodes of A . If no multiplicity is allowed on the inhibitor arcs (as in [12]), it is easy to see that only one extra place and transition are needed, for each place inhibiting *any number of transitions*, so the bound is reduced to be linear in the number of nodes of A . Another characteristic of our proof is that only two priority levels are needed.

The proof in the other direction reflects similar considerations: Hack is able to simulate $A \in PN_p$ with $B \in PN_i$ not containing λ -transitions, but his proof is less direct than ours (A is first transformed into a Restricted PN, then into a PN with inhibitor arcs), and no bound on the size of B is given. We require λ -transitions, but the proof is more direct, and a bound is given (the bound is quadratic in the number of nodes of A , being related to the number of input arcs in A).

Defining the size of A as $|P_A| + |T_A| + |I_A| + |O_A| + |H_A|$, our bounds are linear in the

size of the modeled PN, even if multiple arcs are allowed.

6 Conclusion

In this paper we have discussed the logical “equivalence” of SPN models, when they allow inhibitor arcs, transition priorities, or enabling functions. These three extensions are the most general, and they may well coexist, even if equivalent, to increase design flexibility. The model becomes in any case as powerful as Turing Machines, theoretically difficult to analyze, since most questions about them are undecidable. In particular, the priority of immediate transitions over timed transitions in the GSPN model is enough to achieve this power, even without the addition of inhibitor arcs or other constructs.

We tried to make clear how any SPN model should be categorized as having a TM or a “standard” PN power at the logical level of modeling. A further categorization would be possible by restricting the modeling power (e.g. safe PN, conservative PN, etc), but apparently this is contrary to the trend shown by recent modeling packages.

On a practical basis, the distinction between bounded and unbounded PN is fundamental. SPN packages usually deal only with bounded PN, where all the problems of undecidability disappear, given that we can assert that the PN is bounded.

Further work is needed in the area of stochastic modeling power, possibly in combination with the logical modeling power.

APPENDIX: The basic PN model

In this appendix we recall the main terms of the PN model, see [11] for a complete definition. We do define a multiplicity on the arcs, but this feature can be considered an extension, even if the most widely accepted, and, as such, it is presented together with the other extensions.

A PN is a directed graph whose nodes are partitioned into two sets, *places* and *transitions*. Arcs can only connect a place to a transition (*input arcs*), or a transition to a place (*output arcs*). A multiplicity (positive integer) may be attached on each arc. The following definitions apply to the PN without multiple arcs, if the multiplicity is restricted to have value 1.

The *input (output) bag* for a transition is the bag [11, Appendix A] constituted by the input (output) arcs, considered with their multiplicity. Each place may contain any number of *tokens*. All the tokens are indistinguishable. A *marking* is a bag representing the configuration of tokens in the places of the PN. It can be thought as the state of the PN.

Regarding the evolution of the PN, the following terms are fundamental. A transition is *enabled* if its input bag is a subbag of the (current) marking. When a transition is enabled, it can *fire*, leading the PN into a different marking, obtained by subtracting its input bag from and adding its output bag to the current marking. A *firing sequence* is a sequence of transition firings. A marking is *reachable* if it is obtained by a firing sequence starting in the *initial marking*. The *reachability set (graph)* is the set (graph) of all the reachable markings (connected by arcs representing the transition firings). A set \mathbf{S} of transitions is a *conflicting transition set* in a marking \mathbf{m} if the contemporary firing of all the transitions of \mathbf{S} is impossible in \mathbf{m} , or, in other words, if the sum of the input bag of the transitions in \mathbf{S} is not a subbag of \mathbf{m} .

References

- [1] M. K. Molloy, *On the integration of delay and throughput measures in distributed processing models*. PhD thesis, UCLA, Los Angeles, California, 1981.
- [2] M. Ajmone Marsan, G. Balbo, and G. Conte, "A class of Generalized Stochastic Petri Nets for the performance evaluation of multiprocessor systems," *ACM Trans. Comp. Syst.*, vol. 2, pp. 93–122, May 1984.
- [3] M. Ajmone Marsan, G. Balbo, G. Ciardo, and G. Conte, "A software tool for the automatic analysis of Generalized Stochastic Petri Net models," in *Modelling Techniques and Tools for Performance Analysis* (D. Potier, ed.), pp. 155–170, INRIA, Elsevier Science Publishers B.V. (North Holland), 1985.
- [4] J. Bechta Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola, "Extended Stochastic Petri Nets: applications and analysis," in *Performance '84* (E. Gelenbe, ed.), (Paris, France), North-Holland, Dec. 1984.
- [5] J. Bechta Dugan, A. Bobbio, G. Ciardo, and K. Trivedi, "The design of a unified package for the solution of stochastic Petri net models," in *Proc. of the Int. Workshop on Timed Petri Nets*, (Torino, Italy), July 1985.
- [6] A. Cumani, "ESP - A package for the evaluation of stochastic Petri nets with phase-type distributed transitions times," in *Proc. of the Int. Workshop on Timed Petri Nets*, (Torino, Italy), July 1985.
- [7] M. Holliday and M. Vernon, "A Generalized Timed Petri Net model for performance analysis," in *Proc. of the Int. Workshop on Timed Petri Nets*, (Torino, Italy), July 1985.
- [8] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: structure, behavior, and application," in *Proc. of the Int. Workshop on Timed Petri Nets*, (Torino, Italy), July 1985.
- [9] C. V. Ramamoorthy and G. S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Softw. Eng.*, vol. 6, Sept. 1980.
- [10] T. Agerwala, "A complete model for representing the coordination of asynchronous processes," Hopkins Computer Research Report 32, Johns Hopkins University, Baltimore, Maryland, July 1974.
- [11] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.
- [12] M. Hack, "Petri net languages," Technical Report 159, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, Mar. 1976.