Clarifying the Priority Specification of GSPN: Detached Priorities *

Enrique Teruel(1), Giuliana Franceschinis(2), and Massimiliano De Pierro(3)

(1) Dpto. Informática e Ingeniería de Sistemas, CPS, Universidad de Zaragoza, Spain (eteruel@posta.unizar.es)

(2) Dip. Scienze e Tecnologie Avanzate, Univ. Piemonte Orientale, Alessandria, Italy (giuliana@di.unito.it)

(3) Dipartimento di Informatica, Università di Torino, Italy

Abstract

The global multilevel priority definition in GSPN, while being most convenient with many respects, poses some modelling problems related with confusion and indirect conflicts, which may lead to undesired effects in the definition of the underlying stochastic process. While some of these problems have been recognized and dealt with in previous works on the definition of GSPN, through the introduction of extended conflict sets, some others, that will be illustrated in this paper, were not covered. To overcome all these problems, in this paper a syntactical subclass of GSPN, called detached priorities GSPN (dpGSPN), is proposed. In dpGSPN no confusion or indirect conflicts are possible, and according to our experience, modelling power is not substantially sacrificed. To facilitate the modelling, a possible method to derive a suitable global priority definition from local information is outlined.

Keywords — *Stochastic Petri nets, priorities, conflict, confusion, modelling methodology.*

1. Introduction

Priorities were early introduced as an extension to PN [11, 12]: transitions can be classified into *priority classes*, or a *relation between transitions* can be specified [5, 3]. In any case, priorities modify the enabling rule by preventing a transition firing when another transition having higher priority is also enabled. Priorities were originally introduced with the aim of enlarging the expressive power of the formalism. In fact, the same as with other extensions such as inhibitor arcs, priorities lead to the modelling power of Turing machines in the unbounded case. Even in the bounded case these extensions enlarge the modelling power if some kind of concurrent semantics is of interest [7, 5].

Priorities have found another important motivation in the case of *generalized stochastic PN* [1]: time delays of very different magnitude caused computational and numerical problems for the solution that could be avoided by assuming that "fast" transitions occurred immediately; this implies that *immediate transitions* always occur before timed transitions, that is, they have priority. But the introduction of immediate transitions was not so immediate as it seemed to be at first sight. Due to their zero delay, conflicts between immediate transitions could not be resolved by a race policy (as conflicts between timed transitions were). Random switches were specified by assigning weights to immediate transitions, but the appropriate weight assignment became a major modelling issue, since it required the knowledge of the reachable markings in order to know the conflicts between immediate transitions that could arise. This led to a revised definition of GSPN, allowing the specification of random switches at the net level whenever the net satisfies certain properties [6, 2]. One such property is absence of confusion in immediate transitions firing sequences, called *stochastic confusion*. Confusion is the situation where different sequentializations of the firing of concurrent transitions require different conflict resolutions. Therefore, although being concurrent seems to indicate that the order of firings does not matter, the stochastic behaviour depends on which sequentialization is taken. If (the possibility of) stochastic confusion is detected, a given sequentialization can be forced by specifying suitable priorities between immediate transitions, that is, by using multilevel priorities. As a side effect, multilevel priorities make it possible that (unexpected) *indirect conflicts* appear between transitions in the same priority level not sharing any input place. The confusion and indirect conflict problems become specially unconvenient when the model is intended to be obtained from models of parts of the system: even if these submodels are

*This work was supported by the HCM project MATCH (CHRX-CT94-0452), and by the integrated action HI1998-0196 within a bilateral Italy-Spain exchange programme. The work of E. Teruel was also supported by Project CICYT TAP98-0679.

checked, the complete model must be checked again, and often it requires careful modeller intervention [10].

The problem of a complete and correct specification of the stochastic process underlying a model described using different flavours of stochastic PN formalism with immediate transitions, has been tackled also in [8] and [13], where the complete and correct specification of the model is checked *during the construction of the underlying stochastic model.* Instead, the approach in the revised definition of GSPN [6], and the one proposed in this paper, proceed reasoning at the *net level.* In this paper we introduce a syntactical subclass of GSPN where neither stochastic confusion nor indirect conflicts appear. Working with net subclasses to prevent confusion has not been generally adviced due to the fact that topological subclasses being confusion-free (e.g., free choice nets) are very limited from a modelling point of view. Here, instead of restricting the net structure, a *restriction to the priority specification* is proposed.

Some preliminary notions, particularly concerning the conflict structure of a net, are introduced in Section 2. The problems originated by the unrestricted use of multilevel priorities are illustrated in Section 3. A restriction to the priority specification, leading to the subclass of *detached priorities* GSPN, is introduced in Section 4. A possible method to obtain such a global priority specification from local modelling considerations is outlined and illustrated in Section 5. The case of models obtained by superposition over immediate transitions is briefly considered in Section 6.

2. Preliminaries

The GSPN formalism

The fundamentals of PN and GSPN are well-know [12, 2]. GSPN have been successfully used for the performance modelling and analysis of systems from diverse application fields. A GSPN system is a 7-tuple $\langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{w}, \mathbf{pri}, \mathbf{m_0} \rangle$ where P and T are the sets of places and transitions, $\mathbf{Pre}$ and $\mathbf{Post}$ are $|P| \times |T|$-dimensional matrices of natural numbers representing the pre-(post-)incidence functions, $\mathbf{w}$ is a $|T|$-dimensional array of weights, that are to be interpreted as rates for timed transitions and as weights for immediate transitions to be used in probabilistic conflict resolution, $\mathbf{pri}$ is a $|T|$-dimensional vector representing global transition priorities, and $\mathbf{m_0}$ is a $|P|$-dimensional vector representing the initial marking. For simplicity in this work we have omitted inhibitor arcs, that are instead included in the complete definition of GSPN.

We introduce a special denotation, used in Sections 5 and 6, to indicate that a transition t_i is immediate, although its priority is not (yet) specified: $pri_i = \phi$.

The conflict and causality structure of a net

Transition t_i is said to be in *effective conflict relation* with t_j at marking $\mathbf{m}$ when $\mathbf{m} \xrightarrow{t_i} \mathbf{m}'$ and the enabling degree of t_j decreases from $\mathbf{m}$ to $\mathbf{m}'$. (Notations: disregarding priorities, a transition t is enabled at marking $\mathbf{m}$ when $\mathbf{m} \geq \mathbf{Pre}[P,t]$ componentwise. Its occurrence, denoted by $\mathbf{m} \xrightarrow{t} \mathbf{m}'$, yields marking $\mathbf{m}' = \mathbf{m} + \mathbf{Post}[P,t] - \mathbf{Pre}[P,t]$. The enabling degree of a transition t at marking $\mathbf{m}$ captures the number of simultaneously enabled instances of t: $\max\{k \in \mathbb{N}_+ \mid \mathbf{m} \geq k \cdot \mathbf{Pre}[P,t]\}$. Priorities modify the enabling rule by preventing a transition firing when another transition having higher priority is also enabled.) If "conflicting" transitions have different priority, then the conflict is deterministically solved using priority. We call this kind of conflict *prioritized conflict*, although it is not an actual conflict, since the low priority transition is not actually enabled. Hereafter when we talk of effective conflict we mean effective conflict between transitions with the same priority[1]. In GSPN, effective conflict between timed transitions is solved using a race policy. When the conflicting transitions are immediate, the probability associated with each immediate transition firing is obtained as a function of the weights of the set of enabled transitions in the current marking.

The notion of (effective) conflict is dynamical, it depends on the marking. The static structure of the net contains information on the *potential* dynamic conflicts. The very basic net construct used to model conflicts is a place with more than one output transition, i.e., a *choice place.* Transition t_i is said to be in *structural conflict relation* with t_j when ${}^\bullet t_i \cap {}^\bullet t_j \neq \emptyset$ (and $pri_i = pri_j$). This relation is reflexive and symmetric, but not transitive. Its transitive closure is named *coupled conflict relation*, and it partitions the transitions of a net into *coupled conflict sets.* The coupled conflict set containing t is denoted by $\mathrm{CCS}(t)$. A particular kind of conflicts are those where all the outcomes have the same precondition: transitions t_i and t_j (with the same priority) are said to be in *equal conflict relation*, when $t_i = t_j$ or $\mathbf{Pre}[P,t_i] = \mathbf{Pre}[P,t_j] \neq \mathbf{0}$. This equivalence relation further partitions the transitions into *equal conflict sets.* The equal conflict set containing t is denoted by $\mathrm{EQS}(t) \subseteq \mathrm{CCS}(t)$. A CCS which is also an EQS is called a *trivial* CCS hereafter.

Transition t_i is said to be *causally connected* to t_j

[1] This definition of effective conflict slightly differs from that in [2], which is intended also to include (one kind of) indirect conflict. Here we deal with indirect conflicts separately instead.

at marking $\mathbf{m}$ when $\mathbf{m}\xrightarrow{t_i}\mathbf{m}'$ and the enabling degree of t_j increases from $\mathbf{m}$ to $\mathbf{m}'$. The very basic net construct used to model causality is a place connecting two transitions. Transition t_i is said to be *structurally causally connected* to t_j when $t_i{}^\bullet \cap {}^\bullet t_j \neq \emptyset$. For short, we say t_i is *upstream* t_j.

3. The problems

Let us illustrate the confusion and the indirect conflict problems by simple examples.

Example of confusion

The net in Figure 1 models a production system consisting of two machines that are loaded by a shared robot. Both machines have an input buffer for one part. When the corresponding machine is idle, it immediately starts working on the new part; otherwise, it must finish the current job, and in the meanwhile it cannot be loaded again.

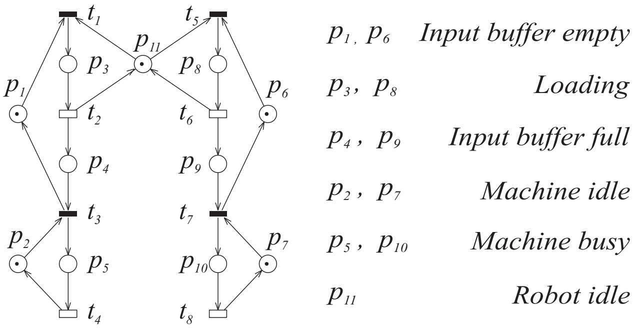

Figure 1. A GSPN model that can show stochastic confusion and indirect conflict.

Assume first that no priorities are specified (other than that of immediate transitions over timed ones), and that the weights of immediate transitions are all one. Now the model shows stochastic confusion: consider, for instance, the (tangible) marking $p_2 + p_3 + p_6 + p_7$. When t_2 fires, the (vanishing) marking $p_2 + p_4 + p_6 + p_7 + p_{11}$ is reached, from which two tangible markings can be reached after some immediate firings (see Figure 2). In particular, the marking $p_1 + p_5 + p_7 + p_8$ can be reached by the firing of either t_5 followed by t_3, without effective conflicts, or t_3 followed by t_5 winning its conflict with t_1. Although t_3 and t_5 are concurrently enabled at $p_2 + p_4 + p_6 + p_7 + p_{11}$, depending on which sequentialization is taken the probability to reach $p_1 + p_5 + p_7 + p_8$ varies!

Faced to this problem, the modeller can decide that t_3 and t_7 should have priority over t_1 and t_5, so they

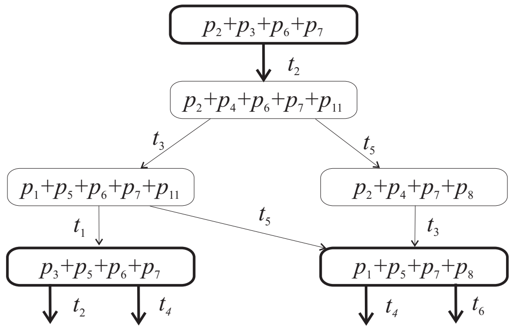

Figure 2. Part of the reachability graph of the GSPN in Figure 1. Tangible markings have a thick border.

can be given priority two to remove confusion: the path t_5 followed by t_3 is removed from the reachability graph in Figure 2. The modeller may well have decided that t_1 and t_5 had priority, it depends on the actual behaviour of the system. In such case, only the path t_5 followed by t_3 would remain in Figure 2, and of course the performance figures would be different.

Example of indirect conflict

Consider again the same example net, but assume now that conflicts to allocate the robot are resolved deterministically in favour of the machine on the left, so t_1 has priority over t_5, e.g., priority two is assigned to t_1. In such case, t_3 can become in effective (indirect) conflict with t_5, for instance at the marking $p_2 + p_4 + p_6 + p_7 + p_{11}$: if t_3 is fired, then t_1 becomes enabled, and, having priority over t_5, it fires disabling t_5. (In Figure 2, the arc from $p_1 + p_5 + p_6 + p_7 + p_{11}$ to $p_1 + p_5 + p_7 + p_8$ disappears.) From the point of view of t_5, it is like t_3 were in conflict with it, because after the choice of firing t_3, the token in p_{11} is removed without a chance for t_5 to grab it. Therefore, the modeller should have specified a conflict resolution policy between t_3 and t_5, although in principle they do not appear to be conflicting transitions because they do not share any input place, and most likely they are not meant to be conflicting transitions. Notice that, prior to the decision of solving local conflicts between t_1 and t_5 in favour to the former, there was no conflict between t_3 and t_5!

It would be advisable that, the same as before, the modeller clarifies the behaviour of the system. For instance, if t_3 and t_7 should have priority, then they can be assigned priority three, and the need to specify probabilities to solve the indirect conflict between t_3 and t_5 disappears.

Pseudo-conflict: a weird sort of indirect conflict

Indirect conflicts such as the above one are taken into account in the definition of *extended conflict sets (ECS)* in GSPN (see [6, 2]). But there is another (weird) sort of indirect conflict that was not taken into account. Let us illustrate it on the GSPN model in Figure 3. We assume that immediate transitions t_4 and t_5 (sharing input place p_9) have priority two, while transitions t_2 and t_3 have priority one. Although t_2 and t_3 appear to be independent (in GSPN terminology, they are in different ECS), it can be easily checked that their firings are not at all independent. After firing t_1, they become simultaneously enabled, and the relative probability of firing one or the other influences the steady state probability of markings. Moreover, if an arbitrary firing order between them was defined, relying in their presumed independence, a different state space would be obtained, containing three tangible states rather than five! Notice that the same problem would appear if t_4 and t_5 had different priorities while the priority of t_2 and t_3 was lower than the priorities of both t_4 and t_5.

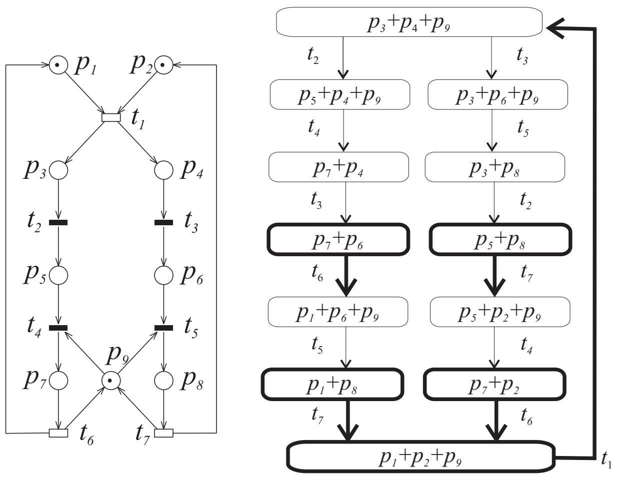

Figure 3. A net showing a pseudo-conflict, and its reachability graph.

Due to the higher priority of t_4 and t_5, transitions t_2 and t_3 happen to be in a sort of indirect conflict relation: depending on which one fires first, different sequences of higher priority transitions can be fired, leading to different states. Since the firing of one transition does not reduce the enabling degree of the other, but the final effect is very much like they were in conflict, as the reachability graph clearly shows, this situation will be called *pseudo-conflict* hereafter. It can appear whenever some immediate transitions upstream a conflict (possibly through a "chain" of more immediate transitions, see the example of Figure 5 later) have the same priority, lower than that of the transitions in the conflict (and those in the "chain"). Observe that no pseudo-conflict would appear if the priorities of t_2 and t_3 were different, because their relative ordering would have been fixed by the modeller. No problem would appear either if the priorities of t_2 and t_3 were higher than the priorities of both t_4 and t_5, because they both would fire before the conflict downstream was solved, or said in other words, they would "push" tokens towards the conflict.

4. Detached priorities GSPN

All situations above share a common pattern: two transitions which are not intended to interfere, meaning that they are not in the same conflict set (in the same CCS), have the same priority and happen to become enabled at the same time, but their firing order is relevant: firing one or the other first produces different effects. More precisely, in all cases we find some structurally conflicting immediate transitions, let's call them t_i and t_j, and either t_i or t_j or both are preceded by different ("chains" of) immediate transitions upstream. Depending on the priorities, there are several cases: when the conflicting transitions and a transition upstream one of them have the same priority, stochastic confusion can arise; when a conflicting transition, say t_i, has lower priority than the other one, and a transition upstream t_j has the same priority as t_i, indirect conflict can arise; when both conflicting transitions are preceded by lower priority upstream transitions, and there is a transition t'_i upstream t_i and a transition t'_j upstream t_j which have the same priority, pseudo-conflict can arise.

Basically, the solution to these problems can be approached in two ways:

- *Detect* the problems, and inform the modeller, so the specification can be completed accordingly. This can be done while generating the underlying stochastic model [8, 13], or at the net level, as required in the net-level GSPN definition [6].
- *Prevent* the problems, by using a subclass of nets were they cannot appear. This approach has not been generally adviced due to the fact that topological subclasses being confusion-free (e.g., free choice nets) are very limited from a modelling point of view.

Here, instead of restricting the net structure, a *restriction to the priority specification* is proposed: immediate transitions not intended to be in effective conflict have *different* priorities, so only the firing of transitions which are intended to be in conflict can interfere.

Definition 1 *A* dpGSPN *(for* detached priorities *GSPN) is a GSPN such that $pri_i = pri_j$ implies that t_i is in coupled conflict relation with t_j.*

Since detached priorities do not allow that two immediate transitions in different CCS are simultaneously enabled, trivially only transitions in the same CCS can interfere, and the firing policy for these is specified by the weights. Therefore, compared to general GSPN, *dpGSPN cannot exhibit stochastic confusion*, what is a desired feature, and *cannot exhibit indirect conflicts* either, because in dpGSPN possibly conflicting transitions always belong to the same CCS (in GSPN terminology, the ECS of dpGSPN are simply the CCS).

The absence of indirect conflicts might have been a modelling limitation, but we could not find a convincing example showing that the indirect conflict is a useful modelling feature. Actually, whenever we found a model with indirect conflicts, we could easily build an alternative model with normal conflicts that appeared to be clearer than the former. Moreover, from the examples above, it appears that when indirect conflict situations are possible in a model, the modeller may be required to define the relative weights of transitions that are structurally far away from each other, and which are not perceived as (or intended to be) conflicting actions. For these reasons, we believe that *indirect conflicts should be avoided.*

Now the question is whether dpGSPN are expressive enough to conveniently model practical situations, but this is a difficult question to answer, because "modelling convenience" or "practical situations" are not formal concepts that could be proved. By reviewing several examples from diverse application domains, no limitations were observed. In these examples, we first gave a local priority specification, and then built the global dpGSPN priority specification from it. The complete formalization of this method is not included in this paper. We instead devote the rest of the paper to illustrate it, to show how practical situations can be conveniently modelled using dpGSPN. In a first step of the method, the modeller imposes deterministic or probabilistic conflict resolutions for the intended conflicts in the model. In a second step, the modeller determines the desired priority ordering between CCS that could lead to confusion or indirect conflict problems. Finally, the global dpGSPN priorities can be computed transparently to the modeller. The second step is the one that would require further formalization: in principle, the modeller could be asked to order *all* pairs of CCS, but this is clearly not necessary. For instance, obviously, immediate transitions belonging to different "islands" of immediate transitions (i.e., not connected by paths of other immediate transitions) can be ordered in one way or the other without affecting the results. Better structural conditions to determine which orderings are actually required would relieve the modeller from the need to specify them.

5. A method to assign priorities

Starting from a GSPN where **pri** simply classifies transitions into immediate and timed, i.e., $pri_i \in \{0, \phi\}$, in order to define a suitable dpGSPN priority specification, we propose to: first, define *partial* priorities (between pairs of transitions, see below) to reflect modelling needs; and second, compute *global* priorities, that is, a suitable **pri** without ϕ's, for stochastic model generation and solution purposes only.

Definition of partial priorities

Partial priorities represent the modeller specification of the behaviour of the system. The specification should be able to state that some structurally conflicting transitions have the same priority, so their conflicts are solved probabilistically, or one has priority over another one. Partial priorities are defined by an irreflexive, asymmetric, and transitive partial relation between groups of transitions, which can be represented by an acyclic digraph, that we shall call the *partial priority (PP) graph* of the net: nodes are transitions, or groups of transitions with the same priority, and there is a directed arc from t to t' when t has priority over t'. (It is not needed to draw the "redundant" arc from t to t'' in case there are arcs from t to t', and from t' to t''.)

We distinguish two kinds of partial priorities: inner and outer. *Inner* priorities are specified between transitions in structural conflict relation, and they are meant to specify conflict resolution policies. *Outer* priorities can be (in some cases they *must* be) specified between transitions not in structural conflict relation, and they are meant to clarify the model behaviour, particularly in order to break confusion and indirect conflicts.

For convenience, the modeller may start with a graph where (immediate) transitions are nodes and there is an undirected arc joining two nodes when the corresponding transitions are in structural conflict relation. We shall call this graph the *structural conflict (SC) graph* of the net. Obviously, the connected components of the SC graph correspond to the CCS of the net. (It must be observed that these CCS shall later be partitioned into smaller CCS by the definition of priorities; at this initial stage, though, the CCS are determined by the structural conflict relation alone, because all immediate transition priorities are the same, namely ϕ.) In order to specify inner priorities, for each arc in the SC graph the modeller must either define an orientation, to state a priority ordering, or group the adjacent nodes, to state that both transitions have the same priority. Whenever a cycle appears in the graph, a contradiction is signaled, and it must be removed. Let us call the graph obtained after this step the *inner priorities (IP) graph*. The nodes of this graph define a partition of the "initial" CCS. Since they correspond to the CCS of the final net, we call them CCS hereafter.

To obtain the PP graph, some additional arcs are required between these CCS, namely those specifying outer priorities needed to break potential confusions or indirect conflicts. The question now is: which outer priorities are *actually required*? As commented above, we give here just some preliminary ideas, while a detailed formal algorithm, taking also into account inhibitor arcs, and possibly mutual exclusion relations, together with its implementation in a tool to support the modelling process, is left for future work. To break a confusion, an outer arc is required connecting a non-trivial CCS and transitions (or more precisely, the CCS containing these transitions) upstream a subset of transitions in the CCS. If these arcs, when there are several, outgo the non trivial CCS, that is, the tokens are not "pushed" towards the conflict, a pseudo-conflict can appear. To break an indirect conflict, an outer arc is required connecting the low priority part of a prioritized conflict (direct or pseudo) and transitions upstream the high priority part. Again, if the orientation is outgoing the low priority part of the prioritized conflict, and there are more transitions upstream it, a pseudo-conflict could appear. To break a pseudo-conflict, an outer arc is required connecting the involved transitions. Notice that the request for outer arcs may require iterative application, as shown by the example in Figure 5 below.

Computation of global priorities

The definition of partial priorities establishes a partial ordering of groups of transitions, according to modelling needs. But, typically, transitions are not totally ordered, even the resulting PP graph may well not be connected, showing some "islands" of immediate transitions. If the specification is complete, meaning that all the required inner and outer priorities have been specified by the modeller, then fixing an arbitrary total ordering, of course consistent with the partial one, does not affect the solution, but it can avoid the generation of many irrelevant vanishing markings [6]. The same idea of introducing artificial priorities has been exploited for the generation of reduced reachability graphs in untimed PN [4]. A simple way to compute global priorities for immediate transitions respecting the partial priority specification is fixing a total ordering, by performing topological sorting [9] in each connected component of the partial priority graph, and arbitrary sorting of the different connected components.

Examples

Coming back to the example in Figure 1, the modeller starts with the net shown in the figure, where some transitions are timed (priority zero) and others are immediate (priority ϕ). First, the modeller defines inner and outer partial priorities. For inner priorities, in this case it reduces to the solution of conflicts between t_1 and t_5. Two cases will be considered:

Case a. Conflicts between t_1 and t_5 are solved probabilistically: they form a group, and the IP graph has no arcs. (In this case suitable weights for the random switch must be defined.)

Case b. Transition t_1 has priority over t_5: the IP graph contains only one arc from t_1 to t_5.

For outer priorities (between CCS), for Case a, to break potential confusions, it is required to specify the relative priority of t_3 and t_7 with respect to $\{t_1, t_5\}$, while for Case b, to break potential indirect conflicts, it is required to specify the relative priority of t_3 and t_5. Two possible PP graphs, one for each case, are represented in the acyclic digraphs of Figure 4. None of them requires further outer priorities so this completes the specification model. Together with the timed transition rates, and the weights in Case a, all the information needed for performance analysis is given. For Case a, topological sorting of the only component of the PP

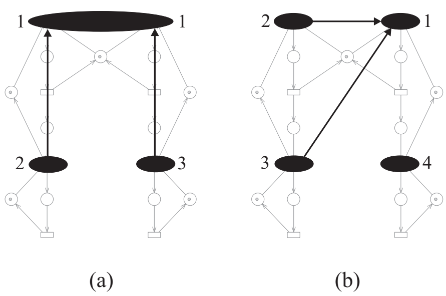

Figure 4. Illustration of the definition of partial priorities and the computation of global priorities for the net in Figure 1.

graph can lead to two equivalent solutions, with different relative priorities of t_3 and t_7. For Case b, different options in the topological sorting of the connected component, and the arbitrary sorting of the isolated transition, can lead to four equivalent solutions. One possible solution for each case is displayed on Figure 4.

The net in the following example, a minor variation of the one in Figure 3, shows that the requirement of outer arcs might be incremental. Let us consider only a few of the many possible cases, which are displayed in Figure 5, where also possible assignments of global priorities are shown.

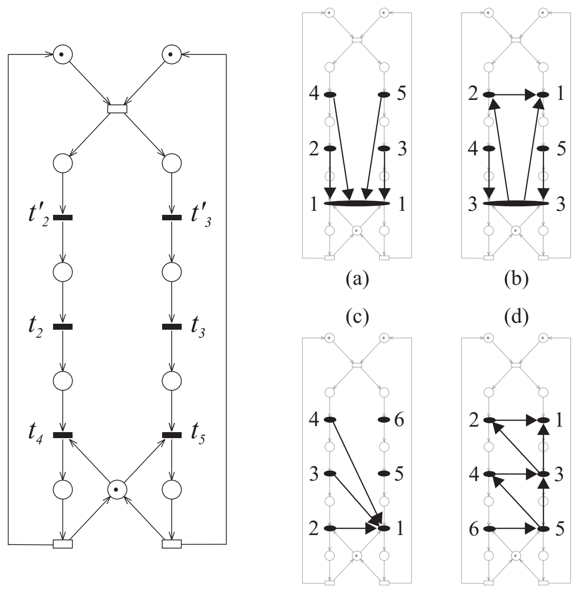

Figure 5. Illustration of the stepwise requirement of outer arcs.

Assume first that the priority of t_4 and t_5 is the same. To break potential confusions, it is required to specify the relative priority of t_2 and t_3 with respect to $\{t_4, t_5\}$. Let both be "pushing". We still have to decide the relative priority of t'_2 and t'_3 with respect to $\{t_4, t_5\}$. If it is also "pushing" we are done (Case a). If it is not, then a pseudo-conflict between t'_2 and t'_3 appears, requiring to order these transitions (Case b).

Assume instead that t_4 and t_5 have different priorities, say t_4 has priority over t_5. To break potential indirect conflicts, it is required to specify the relative priority of t_2 and t_5: If it is "pushing" then we still have to break the potential indirect conflict between t'_2 and t_5. If it is also "pushing" we need not bother about t_3 and t'_3, so we are done (Case c). If t_5 has priority over t_2 instead, then we have to specify the relative priority of t_3 and t_5, because depending on it a pseudo-conflict can appear. Assume it is not "pushing" either (i.e., t_5 has also priority over t_3), so the pseudo-conflict between t_2 and t_3 appears, requiring to order these transitions. Say t_2 has priority over t_3. Then we have to break an indirect conflict between t'_2 and t_3. If again we choose the not "pushing" orientation, the relative priority of t'_3 is needed, and if it is again lower we still have to order t'_2 and t'_3 (Case d).

6. Application to layered models

Among the many ways of building models by composition, we concentrate here in the case of *superposition over common immediate transitions*. This is a fairly general composition principle, that is used for instance in the PSR methodology [10], originally intended for integrating hardware and software models. Similar methodologies have been proposed for manufacturing systems, such as GRAMAN [14], although the latter is not specific for performance models.

The following example is adapted from [2, Fig. 98]. Consider a manufacturing system with one load-unload station (LU), three machining stations (M1, M2, M3), and one automated guided vehicle (AGV) for transportation between stations. There are n pallets circulating in the system. An incoming raw part, loaded in LU, can be used to produce a part of type "A" or a part of type "B". "A" parts are machined by M2 or M3, and "B" parts are machined first by M1, and then by M2 or M3. When finished, they are unloaded in LU. Intermediate buffers of unlimited capacity are available at the input and output of the stations. Since M2 and M3 form a cluster (they can perform the same operations) they share their intermediate buffers: two at the input, one for each part type, and one at the output. This system can be modelled by composing the model of the workplan (top-left in Figure 6) with the models of the different stations or resources (bottom-left in Figure 6). Timed transitions appear only in resource models, because operations are performed by resources, while the workplan model describes the possible operation sequences. It is assumed in the AGV model that travel times are identically distributed irrespective of the actual travel, so it contains a single operation path which shall be replicated as needed for the superposition with different transitions in the workplan. On the other hand, M2 and M3 take a different amount of time to machine a part of type "A" or "B", so their models show different operation paths. The composite model, after removing some obvious redundant places, is shown in Figure 6, right.

Assume that M2 is faster with A's while M3 is faster with B's. It is natural then to solve eventual conflicts

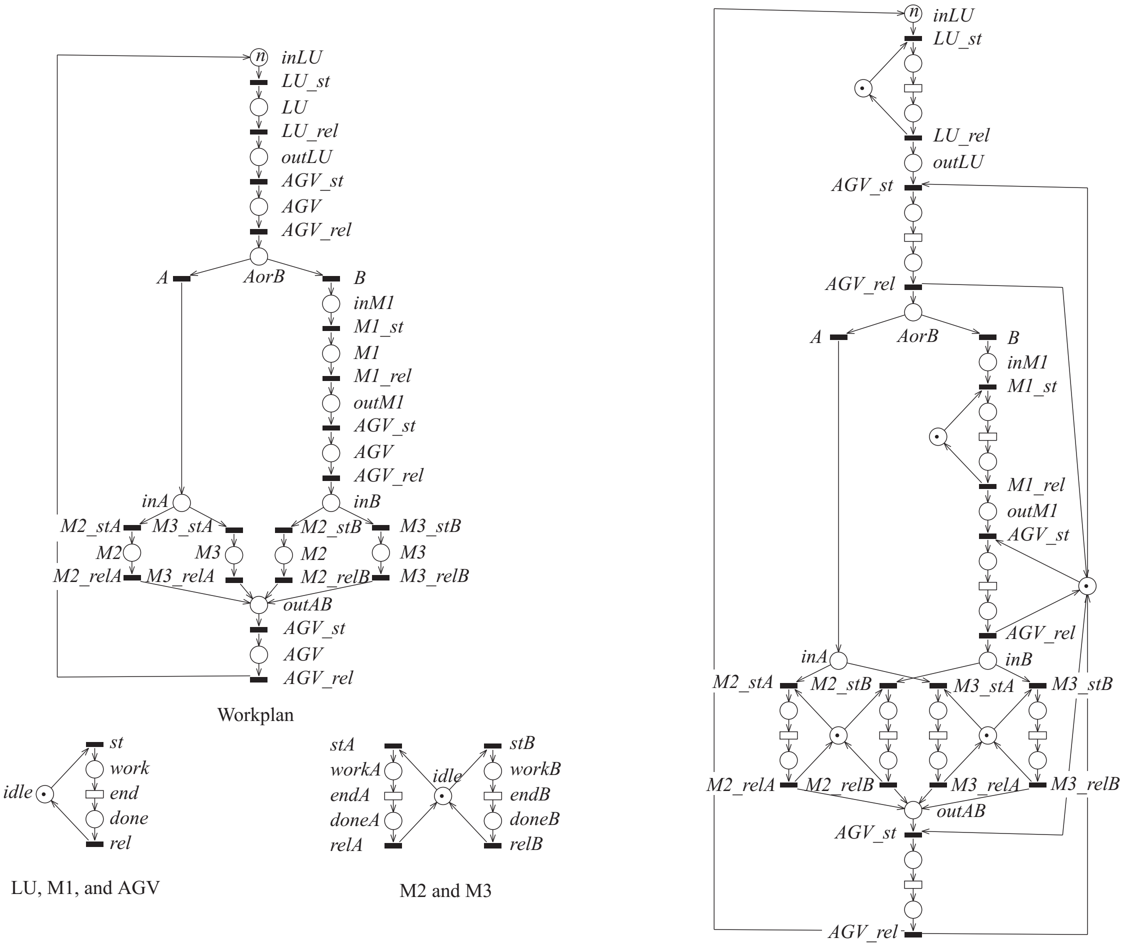

Figure 6. Model of the manufacturing system. Submodels (workplan, and resources), and composite model. Unlabelled places and transitions in the composite model "inherit" submodel's labels prefixed by the name of the submodel, but these are not shown for graphic convenience.

in favour of the faster machine, by specifying suitable priorities at the level of resource models: stA has priority over stB in M2, while stB has priority over stA in M3, that is, stA in M2 and stB in M3 have priority two, while all the other immediate transitions, in this and the other submodels, have priority one. For the composite model, it "appears natural" in this case to assign priority two to $M2\_stA$ and $M3\_stB$ and one to all the others, even though this is somehow contradictory to the fact that these transitions have the same priority in the workplan submodel! Moreover, new potential indirect conflicts or stochastic confusions may appear in the composite model.

In order to derive the priority specification compositionally, instead, the submodels would have some partial priority definition (both inner and outer), that could be incomplete, reflecting that a given part of the model leaves some undeterminism to be resolved by the composition with another one. After performing the union of the PP graphs of the submodels, i.e., some transitions are grouped whenever they are grouped in one submodel, and an arc is drawn from t to t' whenever one of the submodels specifies that t has priority over t', it is possible that some cycles appear. For instance, a self-loop appears when one submodel specifies a priority ordering between two transitions which are in the same node in another submodel. In such case, the priority definitions of submodels are found to be contradictory, and the modeller should analyze the problem and then change the submodels. If there are no cycles, then it is possible to obtain a coherent priority definition for the composite model. This definition may well not be complete, even when it was complete for each submodel, because the new connections

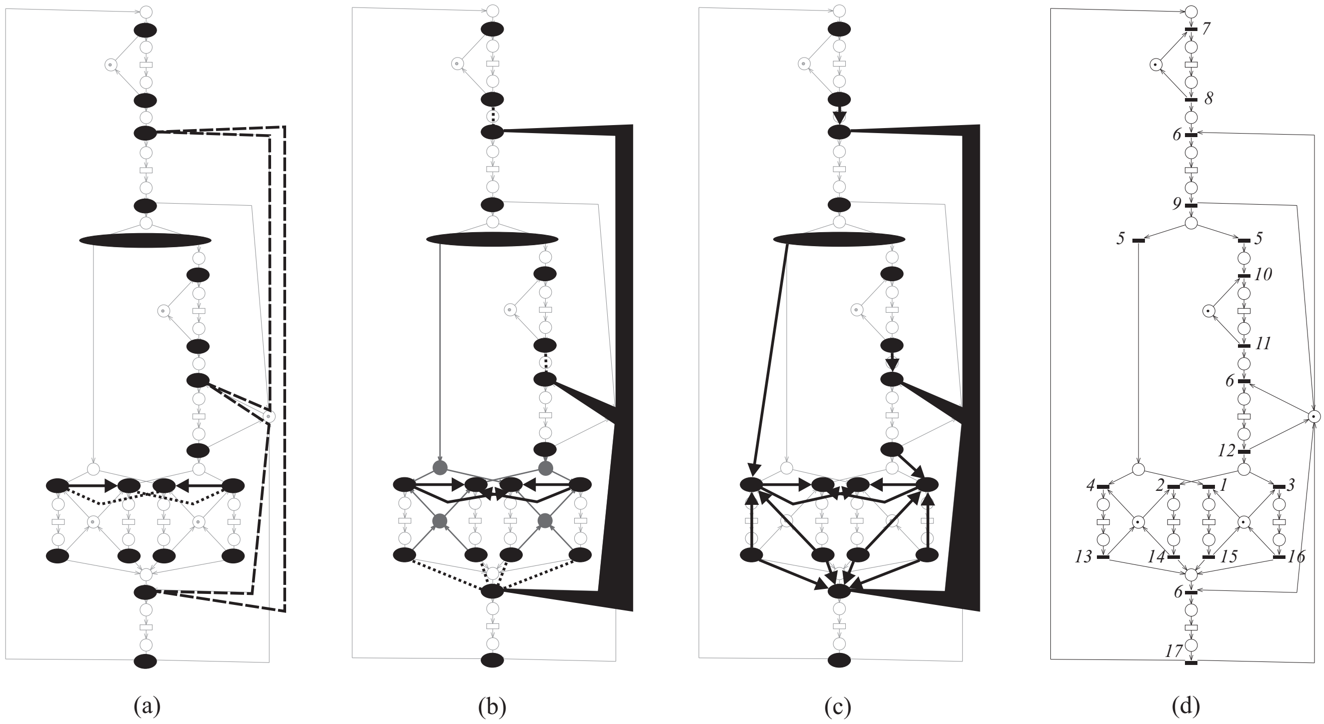

Figure 7. Specifying the priorities for the net in Figure 6: (a) Union of PP graphs (dashed and dotted arcs are illustrative); (b) Inner priorities completed (dotted arcs are illustrative); (c) Final PP graph; (d) GSPN model (transition labels indicate priorities).

make necessary to add some outer priorities: Instead of producing stochastic confusion or indirect conflicts unadvertedly, the necessity of further specification of the behaviour of the model is precisely signaled at the instant of the composition. This further specification is introduced locally, and the subsequent computation of global priorities will account for it producing a model that respects the modelling needs.

Coming back to the example in Figure 6, the modeller starts defining submodel priorities in the submodels nets. At this point the modeller most likely defines only some inner priorities, such as *stA* has priority over *stB* in M2 and *stB* has priority over *stA* in M3. The modeller can specify also that some choices are to be solved probabilistically. For instance, the choice between A and B in workplan, with appropriate weights according to the production-mix.

The union of the PP graphs of the composite model is shown in Figure 7.(a). In some cases, the submodel specification is sufficient. For instance, the choice between A and B, private from workplan, keeps the identical priorities specification (and the weights). But this is not always the case, the specification is not yet complete. For instance, the choice between the three AGV_st is not yet specified by any local submodel, it appeared by the composition; in Figure 7.(a) the three corresponding arcs of the SC graph between these transitions for which no orientation has been decided yet are shown dashed. From M2 and M3, there is an arc from *M2_stA* to *M2_stB*, and another one from *M3_stB* to *M3_stA*, as shown in Figure 7.(a). In case that workplan does not specify priorities between these transitions, there are still some arcs of the SC graph without orientation, shown dotted in Figure 7.(a).

In Figure 7.(b) a given assignment of the missing inner priorities is shown: we assume a probabilistic resolution of the AGV conflict, so the three transitions are grouped in a single node, the ∃-shaped node, and priorities are imposed in favour of the faster machine in each case: *M2_stA* has priority over *M3_stA*, and *M3_stB* has priority over *M2_stB*. Finally, once inner priorities are completed, some outer priorities are required to break potential confusions involving the AGV_st transitions, shown dotted, and some potential indirect conflicts involving the machine choices, where the relevant causal connections are shaded. A set of outer arcs are added leading to the PP graph in Figure 7.(c). (Notice that, except for one case involving the choice between

A and B and the machine choices, all outer arcs involved "*rel*" transitions, i.e., "release resource", which are rather artificial, they are due to the composition methodology: they are all immediate persistent transitions that fire as soon as the precedent timed transition fires, and both firings are representing the same "physical" event, the end of the corresponding operation modelled by the timed transition, so both firings should not be separated, in fact, one could have removed these transitions easily performing post-fusion. So it could have been "pre-defined" that the priority of these transitions is always higher than that of any choice transition, and the modeller would have been relieved from the necessity to specify them.)

Once partial priorities are completely specified, global priorities are computed easily, the same as in Section 5, leading for instance to the GSPN model in Figure 7.(d), where the only probabilistic choices are the one between A and B, and the one involving the AGV, and these are the only immediate transitions which have the same priority.

7. Discussion and Concluding Remarks

The unrestricted use of global priorities makes modelling cumbersome and error-prone. In this paper a new kind of indirect conflict originated by multilevel priorities, which is not covered by the current definition of extended conflict sets in GSPN, has been signaled. To avoid these problems, a restriction has been proposed, leading to the subclass of detached priorities GSPN (dpGSPN), which are stochastic confusion free, and where all possibly conflicting transitions belong to the same coupled conflict set.

Possibly, the question whether too much "practical modelling power" is sacrificed can only be answered by practice. Anyhow, in order to give some insight on the possibility to satisfy habitual modelling needs by using dpGSPN, a net-level method to derive suitable (global) priorities, fulfilling the dpGSPN restriction, from (local) modelling considerations has been illustrated. In this method, the modeller defines partial priorities of immediate transitions, as a relation between them, and global priorities are transparently computed for solution purposes only. The local nature of the priority specification facilitates the composition of models, which had been signaled as a major modelling difficulty with standard GSPN.

Currently we are working on a further formalization of this method, and its implementation in a tool, trying to reduce to a minimum the request for information from the modeller.

Acknowledgement

We are grateful to the reviewers, whose comments helped us to better organize and improve the paper.

References

[1] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems. *ACM Trans. on Computer Systems*, 2(2):93–122, 1984.

[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

[3] F. Bause. On the analysis of Petri nets with static priorities. *Acta Informatica*, 33(7):669–686, 1996.

[4] F. Bause. Analysis of Petri nets with a dynamic priority method. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 215–234. Springer, 1997.

[5] E. Best and M. Koutny. Petri net semantics of priority systems. *Theor. Computer Science*, 96:175–215, 1992.

[6] G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte. Generalized stochastic Petri nets: A definition at the net level and its implications. *IEEE Trans. on Software Engineering*, 19(2):89–107, 1993.

[7] G. Chiola, S. Donatelli, and G. Franceschinis. Priorities, inhibitor arcs, and concurrency in P/T nets. In *Proc. 12^{th} Intern. Conference on Application and Theory of Petri Nets*, pages 182–205, Aarhus, Denmark, June 1991.

[8] G. Ciardo and R. Zijal. Well-defined stochastic Petri nets. In *Procs. Fourth International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '96)*, pages 278–284. IEEE Computer Society, 1996.

[9] N. Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, 1974.

[10] S. Donatelli and G. Franceschinis. The PSR methodology: Integrating hardware and software. In J. Billington and W. Reisig, editors, *Application and Theory of Petri Nets 1996*, volume 1091 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 1996.

[11] M. H. T. Hack. *Decidability Questions for Petri Nets*. PhD thesis, M.I.T., Cambridge, MA, USA, Dec. 1975.

[12] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[13] M. A. Qureshi, W. Sanders, A. P. van Moorsel, and R. German. Algorithms for the generation of state-level representations of stochastic activity networks with general reward structures. *IEEE Trans. on Software Engineering*, 22(9):603–614, 1996.

[14] J. L. Villarroel, J. Martínez, and M. Silva. GRAMAN: A graphic system for manufacturing system design. In S. Tzafestas, A. Eisinberg, and L. Carotenuto, editors, *IMACS Symp. on System Modelling and Simulation*, pages 311–316. Elsevier, 1988.