Proceeding of the 2006 IEEE
Conference on Automation Science and Engineering
Shanghai, China, October 7-10, 2006

Improving analysis and simulation of continuous Petri Nets

L. Recalde, C. Mahulea, M. Silva
Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain
Email: {lrecalde, cmahulea, silva}@unizar.es

***Abstract*— State explosion problems in the analysis of discrete event dynamic systems have lead to relaxed views, fluidification in particular. Proceeding in that way continuous Petri nets have been defined. For infinite servers semantics, switching linear systems appear through the presence of minimum operators due to synchronizations. The analysis of the ODEs defining the semantics of those hybrid models suffers from two problems: the existence of the *minimum* operators and *stiffness*. In discrete models, an approximated way to deal with stiffness was to classify the transitions into *immediate* and *timed* (thus markings are divided into *vanishing* and *tangible*). Avoiding synchronizations (hence minimum operators) and removing immediate transitions at net level becomes a crucial issue for continuous Petri nets because timed continuous net models and ODEs are isomorphous. The application to an example taken from the literature illustrates our purpose and the computational advantages obtained.**

***Index Terms*— Continuous Petri nets, immediate transitions, improving simulation**

I. Introduction

State explosion is a major problem for the analysis of discrete event systems, for which closed formulas describing the possible behaviors are extremely difficult to obtain. Fluidification of discrete event models described by Petri nets (PNs) is a promising way of dealing with that kind of systems, if population loads are relatively significant [1], [2], even if some paradoxical behaviors may appear.

Proceeding in that way, continuous Petri nets (contPN) have been defined. Among the more interesting timed semantics is infinite servers semantics [3] which, through the presence of minimum operators due to synchronizations, leads to switching linear systems. The analysis of the ODEs defining the semantics of those hybrid models suffers from two problems: (1) the existence of the already mentioned *minimum* operators, that introduces non linearities in the model; and (2) *stiffness*, which is problematic with simulation or with numerical solution methods. Minimum operators can be avoided if synchronizations are removed. In discrete models, dummy synchronizations that can be eliminated appear when implicit places [4] are considered. Here a more subtle concept is introduced: *implicit arcs*. In this case the place (variable) cannot be removed, but it is not used in the minimum operator. If all the output arcs of a place are implicit, the place is implicit, and it can be removed. If the corresponding minimum operators are removed, a linear system with linear inequality constraints, associated to the nonnegativity of the markings, is obtained. However, it can be seen that in the autonomous (non forced) model the inequalities are redundant.

In discrete models, an approximate way to deal with stiffness was to drastically classify the transitions into *immediate* and *timed* (thus markings into *vanishing* and *tangible*) [5]. The same kind of classification can be done for contPN. However, removing immediate transitions at net level becomes a crucial issue in contPNs, because the semantics of timed continuous net models is directly expressible as ODEs with minimum operators, for which reasonable solutions are frequently obtained by means of numerical integration. Observe at this point that, even Join-Free nets, that is, nets without synchronizations, (whose dynamics can be modelled with linear ODEs) may suffer from stiffness.

Hence, the basic goals of this paper are: (1) reduce the number of minimum operators both for any timing (implicitness in the autonomous model) and for a particular timing (implicitness in the timed model); and (2) provide the semantics for immediate transitions and some rules to reduce their number, since they are not easy to deal with in simulation or numerical computations.

The paper is organized as follows: Section II presents the notation and basic concepts of contPNs. Immediate transitions are introduced in the model in Section III and an algorithm is provided for their simulation. Section IV is devoted to trying to remove synchronizations, both in autonomous and in timed models. Sections V and VI provide some rules to reduce the number of immediate transitions. Section VII shows the application to an example taken from the literature.

II. Notation and general concepts for autonomous and timed net systems

***Definition** 2.1:* A continuous Petri net (contPN) system is a pair $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, where $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ is a Petri net structure (a set of places, P, a set of transitions, T, and the pre and post incidence matrices, $\boldsymbol{Pre}$ and $\boldsymbol{Post}$), and $\boldsymbol{m}_0$ is the initial marking.

A transition $t \in T$ is enabled at $\boldsymbol{m}$ iff $\forall p \in {}^\bullet t$, $\boldsymbol{m}(p) > 0$ and its enabling degree is:

$$enab(t, \boldsymbol{m}) = \min_{p \in {}^\bullet t} \left\{ \frac{\boldsymbol{m}(p)}{\boldsymbol{Pre}(p,t)} \right\}$$

This work was partially supported by the project CICYT and FEDER DPI2003-06376.

An enabled transition t can fire in any amount $0 \leq \alpha \leq enab(t, \boldsymbol{m})$ leading to a new marking $\boldsymbol{m}' = \boldsymbol{m} + \alpha \cdot \boldsymbol{C}(P, t)$, where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the token-flow matrix.

In a timed model, the state equation has an explicit dependence on time $\boldsymbol{m}(\tau) = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}(\tau)$. To simplify, assume first that all the transitions are timed. Deriving with respect to time, $\dot{\boldsymbol{m}}(\tau) = \boldsymbol{C} \cdot \dot{\boldsymbol{\sigma}}(\tau)$ is obtained. Let us denote $\boldsymbol{f} = \dot{\boldsymbol{\sigma}}$, since it represents the *flow* through the transitions. In this paper it will be assumed that every transition has at least one input place. *Infinite servers semantics* or (variable speed [2]) will be considered here. Under this semantics, the flow of a transition t is given by the product of $\boldsymbol{\lambda}(t)$ and its enabling degree, i.e., $\boldsymbol{f}(t) = \boldsymbol{\lambda}(t) \cdot \text{enab}(t, \boldsymbol{m}) = \boldsymbol{\lambda}(t) \cdot \min_{p \in {}^\bullet t}\{\boldsymbol{m}(p)/\boldsymbol{Pre}(p, t)\}$, leading to nonlinear systems. In discrete nets, if the servers are made explicit infinite and finite servers semantics are equivalent. This is not true in the continuous case. For a broad class of nets it is proved in [3] that infinite servers semantics always provides a better approximation.

III. Immediate transitions

Numerical analysis of models with immediate transitions may suffer from *stiffness*. This is always problematic both for simulation and for numerical solution techniques. The last case is clear at the level of the Markov chain generation for discrete markovian models [5]. The problem is also important in the context of fluidified models that have to be solved by means of numerical techniques. Immediate transitions appear as a simplification of a timed model when some transitions are "much quicker" than others, case in which they are reduced to "fireable in zero time". In a certain sense, transitions could be logically classified, according to their speeds, into timed (producing a job in a sensible time) and immediate (leading to some vanishing markings). Of course, immediate transitions may be defined at several levels of immediateness, but here we will simplify the presentation assuming that transitions are either timed or immediate, i.e., $T = T_T \cup T_I$, $T_T \cap T_I = \emptyset$.

In discrete markovian models immediate transitions can be removed when the Markov Chain is generated. It is a kind of compilation technique because the tangible Markov chain (in which no vanishing state remains) will not suffer from stiffness anymore. Reduction of immediate transitions at net level for discrete Petri nets deserved also some attention in the literature [5]. The advantage obtained in this case derives from the isomorphism between the reachability graph of the PN model without immediate transitions, and the state transition rate diagram of the underlying markovian model. In the context of timed continuous net systems, obtaining stiffness-free models is particularly interesting if numerical integration has to be done, because the net model and the derived ODE system are isomorphic. Even if the introduction of immediate transitions was historically done in order to cope with stiffness, since the *routing* of clients in service networks is much quicker than the treatments to be performed, they can be seen at a logical modelling level as a way to decouple routing from services. But immediate transitions are also helpful, for example, if *setup times* are much shorter that *processing times*, or if simple synchronizations (*rendez-vous*) among classes of clients are present. A real positive number is associated to each transition $t \in T_I$. This number is represented as $\boldsymbol{r}(t)$ and defines the firing rates of the transitions in a conflict. That is, if t_1 and t_2 are both enabled and in conflict relation, the firing of t_1 divided by the firing of t_2 will be equal to $\boldsymbol{r}(t_1)/\boldsymbol{r}(t_2)$.

The numerical integration of timed contPN with immediate transitions can be seen as done at two levels: (1) if at least one immediate transition is enabled, stop the flows through all timed transitions and fire immediate transitions until none is enabled; (2) after that, continue computing flows through timed transitions. Numerical techniques developed for ODEs can be applied to the above computational schema if flows through immediate transitions are solved.

For step 1, since all timed transitions are stopped, assume they are deleted from the net. In the remaining model some transitions may be in *conflict*. Two transitions $t, t' \in T$ are in *conflict relation* at marking $\boldsymbol{m}$ iff there exist $k, k' \in \mathbb{R}_{>0}$ such that $\boldsymbol{m} \geq k \cdot \boldsymbol{Pre}(P, t)$ and $\boldsymbol{m} \geq k' \cdot \boldsymbol{Pre}(P, t')$, but $\boldsymbol{m} \ngeq k \cdot \boldsymbol{Pre}(P, t) + k' \cdot \boldsymbol{Pre}(P, t')$. For this, it is necessary that ${}^\bullet t \bigcap {}^\bullet t' \neq \emptyset$ and in that case it is said that t and t' are in structural conflict relation. The structural conflict relation is not transitive, and the *c*oupled conflict relation is defined as its transitive closure. Each equivalence class is called a coupled conflict set, that will be denoted as CCS_i, and SCCS will represent the set of all the coupled conflict sets.

Compute the coupled conflict sets of the immediate transitions and then, for each coupled conflict set, fire the enabled transitions according to their rates. That is, assume t_1 and t_2 in Fig. 1 are immediate transitions with rates $\boldsymbol{r}(t_1) = \alpha$ and $\boldsymbol{r}(t_2) = \beta$. To be consistent with the way immediate transitions are used in discrete nets, t_1 and t_2 should be fired according to their relative rates, until one of the input places goes empty. If it is p_2 that empties first ($\boldsymbol{m}(p_1) \geq \frac{\alpha}{\alpha+\beta} \cdot \boldsymbol{m}(p_2), \boldsymbol{m}(p_3) \geq \frac{\beta}{\alpha+\beta} \cdot \boldsymbol{m}(p_2)$), we are done since t_1 and t_2 are not further enabled. Otherwise, fire the transition that remains enabled until one of its input places is empty.

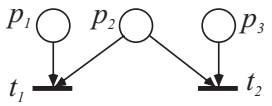

Fig. 1. Conflict between immediate transitions.

Hence, at a certain marking $\boldsymbol{m}$, it is possible that not all the transitions in a coupled conflict set are enabled. This introduces a new concept, *effective structural conflict relation* at marking $\boldsymbol{m}$, which is the same as structural conflict relation, but applied to the net without the disabled transitions.

Let $\boldsymbol{q}(t_i)$ be the amount in which t_i is fired. At a marking $\boldsymbol{m}$, the algorithm to be applied to each $\text{CCS}_i \in \text{SCCS}$ is:

INPUT: $\boldsymbol{m}$, $\text{CCS}_i = \{t_1, \cdots, t_g\}$

Compute the subset of transitions in CCS_i that are enabled at the current marking $\boldsymbol{m}$

Let CI= the effective coupled conflict sets of CCS_i at $\boldsymbol{m}$

while $CI \neq \emptyset$ **do**

Let $c = \{t_1, t_2, \ldots, t_k\} \in CI$

Solve the following linear programming problem (LPP):

$$\max \boldsymbol{q}(t_1) + \boldsymbol{q}(t_2) + \cdots + \boldsymbol{q}(t_k)$$

$$s.t.$$

$$\begin{cases} \frac{1}{\boldsymbol{r}(t_1)} \cdot \boldsymbol{q}(t_1) = \frac{1}{\boldsymbol{r}(t_2)} \cdot \boldsymbol{q}(t_2) = \cdots = \frac{1}{\boldsymbol{r}(t_k)} \cdot \boldsymbol{q}(t_k) \\ \boldsymbol{m}(p_j) - \sum\limits_{t_h \in p_j^\bullet} \boldsymbol{Pre}(p_j, t_h) \cdot \boldsymbol{s}(t_h) \geq 0, \forall p_j \in {}^\bullet t_i \\ \boldsymbol{q}(t_1), \boldsymbol{q}(t_2), \cdots, \boldsymbol{q}(t_k) \geq 0 \end{cases}$$

Fire $\{t_1, t_2, \cdots, t_k\}$ with $\boldsymbol{q}(t_1), \boldsymbol{q}(t_2), \cdots, \boldsymbol{q}(t_k)$ and let $\boldsymbol{m}'$ be the obtained marking, i.e. $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C} \cdot \boldsymbol{q}$.

Let $CI = (CI \setminus c) \cup \{$ the effective coupled conflict sets of c at $\boldsymbol{m}'\}$, and $\boldsymbol{m} = \boldsymbol{m}'$

end while

If the net does not have self-loop arcs associated to immediate transitions, at each execution of the loop at least one of the transitions will not be enabled anymore. Otherwise it may happen that the procedure has to be repeated several times until one place gets empty.

If sequences of several immediate transitions exist, the order in which the elements of the SCCS are visited is important and should be obtained first. It has to ensure that when applying the algorithm to one coupled conflict set, transitions belonging to a previously fired set do not become enabled. This order can be solved as long as we are not dealing with a circuit of immediate transitions, which clearly is a modelling error.

The net in Fig. 2, taken from [6], models a simple manufacturing cell. The size of its reachability set is not very big and so it could have been analyzed as discrete. However, we have rather kept its token load small so as to observe the quality of the continuous approximation. Notice that the computational effort for the discrete analysis will increase in a more loaded system, while it will not significatively change for the continuous model, and the quality of the approximation usually improves with more loaded systems. To simplify the presentation, only the steady state results will be compared, although the simulation will also give the transient behavior. The throughput of the system as discrete is 0.0412, while it is 0.0415 as continuous. However, the simulation time for the continuous model until the steady state is obtained is 186 seconds using Matlab in a Pentium IV 3.2 MHz. Quite large for such a simple net. Efficient algorithms for the solution of linear ODEs exist, however the relative abundance of immediate transitions and synchronizations slows down the simulation. Hence, our goal is to avoid both synchronizations and immediate transitions when possible.

IV. Removing minimum operators

A. Autonomous models: implicit arcs and places

In discrete models the existence of minimum operators is not so important. However, in continuous models their existence is much more troublesome: they produce switches in the set of equations that define the evolution of the model and prevent the removal of immediate transitions. Hence it is important to avoid them when possible.

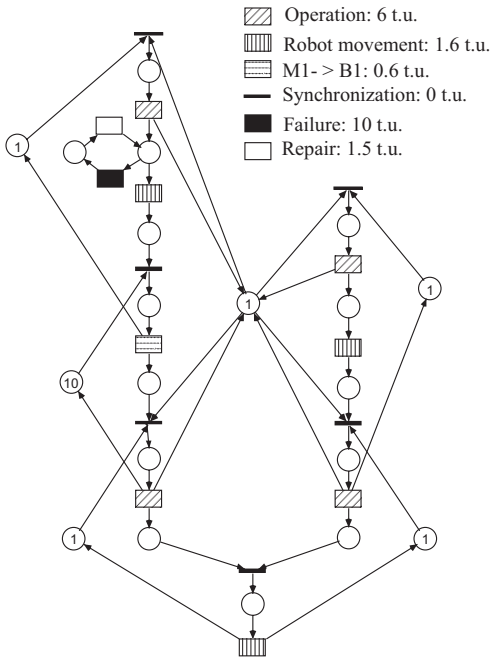

Fig. 2. Example of a simple manufacturing cell taken from [6].

In discrete models, a place is said to be implicit if it can be removed without changing the behavior of the system [4]. For continuous models it is important even to know if a place is never the one that stops a synchronization. This is true not only for simulation, but also for other analysis techniques. For example, in [7] an upper bound of the throughput of the system in steady state was computed using a branch and bound algorithm. The branching was related to synchronizations, hence removing synchronizations would simplify that problem also. Of course, if all the output arcs of a place are implicit, this place will never stop anything, and can be completely removed. In that case we will say that the place is implicit.

***Definition** 4.1:* An input arc (p, t) is implicit in $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ if p is never the place that defines the enabling of t. That is, for every reachable marking $\boldsymbol{m}$, $\frac{\boldsymbol{m}(p)}{\boldsymbol{Pre}(p,t)} \geq \frac{\boldsymbol{m}(p')}{\boldsymbol{Pre}(p',t)}$ $\forall p' \in {}^\bullet t$.

Place p is implicit if all its output arcs are implicit.

This is equivalent to saying that for any reachable marking $\boldsymbol{m}$, the following system has no solution, where s represents the amount in which t is fired.

$$\begin{cases} \boldsymbol{m}(P') - \boldsymbol{Pre}(P', t) \cdot s \geq \boldsymbol{0}, \ P' = P \setminus \{p\} \\ \boldsymbol{m}(p) - \boldsymbol{Pre}(p, t) \cdot s < 0 \\ s \geq 0 \end{cases} \quad (1)$$

Using that any reachable marking has to verify the state equation, a sufficient condition for (p, t) being implicit is that the solution to the following linear programming problem (LPP), z, verifies $\boldsymbol{m}_0(p) \geq z$.

$$\begin{array}{rll} z = \max & \boldsymbol{Pre}(p, t) \cdot s - \boldsymbol{C}(p, T) \cdot \boldsymbol{\sigma} & \\ s.t. & \boldsymbol{m} - \boldsymbol{C} \cdot \boldsymbol{\sigma} = \boldsymbol{m}_0 & \\ & \boldsymbol{m}(P') - \boldsymbol{Pre}(P', t) \cdot s \geq \boldsymbol{0}, \ P' = P \setminus \{p\} & \\ & \boldsymbol{m}, \boldsymbol{\sigma} \ z \geq 0 & \end{array} \quad (2)$$

Notice that $\boldsymbol{m} = \boldsymbol{m}_0$, $\boldsymbol{\sigma} = \mathbf{0}$ and $s = 0$ is always solution of the equations in (2). Hence, applying duality, the solution to this LPP is the same as the solution of this other LPP:

$$\begin{array}{rl} z = \min & \boldsymbol{y}^T \cdot \boldsymbol{m}_0 \\ s.t. & \boldsymbol{y}^T \cdot \boldsymbol{C}(P', T) \leq \boldsymbol{C}(p, T) \\ & \boldsymbol{y}^T \cdot \boldsymbol{Pre}(P', t) \geq \boldsymbol{Pre}(p, t) \\ & \boldsymbol{y} \geq \mathbf{0} \end{array} \tag{3}$$

For place p to be implicit, putting together all the equations related to its output arcs, a sufficient condition is that $\boldsymbol{m}_0(p) \geq z$, with z defined as follows:

$$\begin{array}{rl} z = \min & \boldsymbol{y}^T \cdot \boldsymbol{m}_0 \\ s.t. & \boldsymbol{y}^T \cdot \boldsymbol{C}(P', T) \leq \boldsymbol{C}(p, T)) \\ & \boldsymbol{y}^T \cdot \boldsymbol{Pre}(P', p^\bullet) \geq \boldsymbol{Pre}(p, p^\bullet) \\ & \boldsymbol{y} \geq \mathbf{0} \end{array} \tag{4}$$

This condition simplifies one deduced for discrete PN [4].

B. On implicitness for timed contPNs

In the previous subsection only the autonomous model has been considered, not taking time into account. Hence the results will be valid for any timing of the model. However, timing will introduce additional restrictions in the dynamics of the system, and so arcs that were not implicit in general might be so for a particular timing.

It is not easy to give general rules for this kind of implicit arcs. However, some situations can be dealt with. Assume for example a net with a par-begin par-end in which the synchronization is an immediate transition (see Fig. 3, taken from [5]). It is clear that for this net always one of the arcs in the synchronization will be implicit. In this case, since $\boldsymbol{\lambda}(T_{par1}) = 2$ and $\boldsymbol{\lambda}(T_{par2}) = 1$, the arc (p_5, t_{syn}) is implicit. Even more, in this case place p_5 is implicit, and can be removed. Its marking can always be deduced using that $\boldsymbol{m}(p_5) = \boldsymbol{m}(p_4) + \boldsymbol{m}(p_6) - \boldsymbol{m}(p_3)$.

This can be generalized to the case in which some tokens appear in one place of the par-begin par-end subnet. To simplify, let us consider first a net with only a par-begin par-end connected by a place (see Fig. 4). If $\boldsymbol{\lambda}(t_2) \leq \boldsymbol{\lambda}(t_3)$, then for any $q \geq 0$, the arc (p_5, t_4) will be implicit. If $\boldsymbol{\lambda}(t_2) > \boldsymbol{\lambda}(t_3)$, it may also happen that (p_5, t_4) is implicit if q is large enough. To get the lower q for this, we need to ensure that always $\boldsymbol{m}(p_4) = 0$ and $\boldsymbol{m}(p_5) \geq 0$. At the beginning, the evolution of the system can be described by the following equations:

$$\begin{array}{l} \dot{\boldsymbol{m}}(p_1) = -\boldsymbol{\lambda}(t_1) \cdot \boldsymbol{m}(p_1) + \boldsymbol{\lambda}(t_2) \cdot \boldsymbol{m}(p_2) \\ \dot{\boldsymbol{m}}(p_3) = \boldsymbol{\lambda}(t_1) \cdot \boldsymbol{m}(p_1) - \boldsymbol{\lambda}(t_3) \cdot \boldsymbol{m}(p_3) \\ \boldsymbol{m}(p_1) + \boldsymbol{m}(p_2) = k \\ \boldsymbol{m}(p_1) + \boldsymbol{m}(p_3) + \boldsymbol{m}(p_5) = k + q \end{array}$$

Integrating the system, it can be deduced that if $q \geq k \cdot \frac{1/\boldsymbol{\lambda}(t_2) - 1/\boldsymbol{\lambda}(t_3)}{1/\boldsymbol{\lambda}(t_2) + 1/\boldsymbol{\lambda}(t_1)}$, then $\boldsymbol{m}(p_5) \geq 0$.

Of course, in general the net will not be so simple. However, it can be proved that, as long as the flow of transition t_1 in the original net is at most that of t_1 in the basic par-begin par-end net, if (p_5, t_4) is implicit in the basic net it will also be implicit in the complete net. Hence, the idea is to find $\boldsymbol{\lambda}(t_1)$ and k such that the relationship between the flows of t_1 in both nets is guaranteed. For example, a simple way is to define k as the maximum number of tokens in an input place of t_1 (of course, if there are more than one place, take the minimum among them) and $\boldsymbol{\lambda}(t_1)$ the same as it is in the original net or, if this transition were immediate, a previous timed one.

In this schema it has been assumed that the synchronization was immediate. In fact, this is no restriction since the case with a timed synchronization can be reduced to this one by unfolding the timed transition into an immediate synchronization followed by the timed transition.

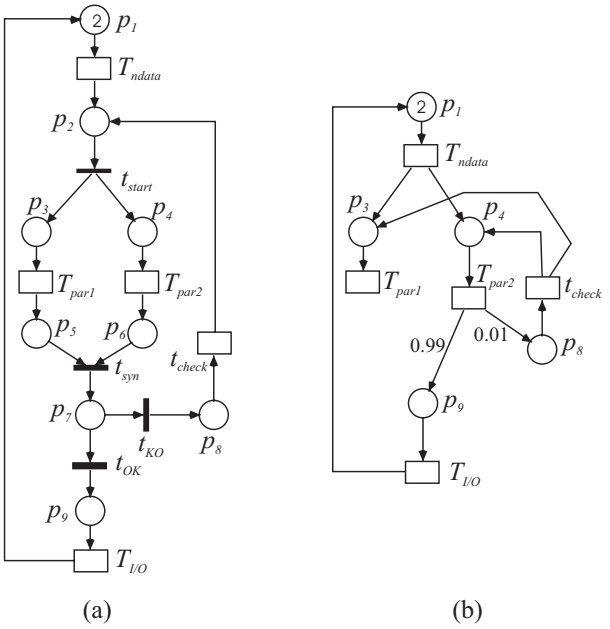

Fig. 3. (a) Net taken from [5] and (b) its reduction as a contPN

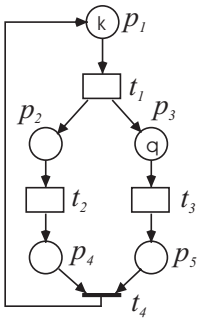

Fig. 4. Par-begin par-end net

V. Merging of Equal Conflicts: timed and immediate transitions

A particular kind of conflicts are those in which the preconditions of the transitions are all the same. In discrete PNs, they are called *equal conflicts*. For contPN the same idea can be slightly generalized, since the weights of the arcs are not

really as important as in the discrete case. We will say that t_i and t_j are in *topologically equal conflict relation*, if a constant k exists such that $\forall p \in P$, $\boldsymbol{Pre}(p,t_i) = k \cdot \boldsymbol{Pre}(p,t_j)$. This is an equivalence relation.

Any set of transitions in topologically equal conflict relation can be reduced to a single transition, t. Although the idea of the transformation is the same both for timed and immediate transitions, the new rates and arc weights are not computed the same way. The reason is that in a conflict among immediate transitions, their firing rate does not depend on the enabling, while it does in the timed case. To simplify the presentation here we will consider that the set has only two elements: t_1 and t_2. Then $\forall p \in P$:

- Timed transitions
  - $\boldsymbol{Pre}(p,t) = \boldsymbol{Pre}(p,t_1)$
  - $\boldsymbol{Post}(p,t) = \frac{1}{\boldsymbol{\lambda}(t_1)+\boldsymbol{\lambda}(t_2)} \cdot (\boldsymbol{Post}(p,t_1) \cdot \boldsymbol{\lambda}(t_1) + \boldsymbol{Post}(p,t_2) \cdot \boldsymbol{\lambda}(t_2) \cdot \frac{\boldsymbol{Pre}(p,t_1)}{\boldsymbol{Pre}(p,t_2)})$
  - $\boldsymbol{\lambda}(t) = \boldsymbol{\lambda}(t_1) + \boldsymbol{\lambda}(t_2)$
- Immediate transitions,
  - $\boldsymbol{Pre}(p,t) = \boldsymbol{Pre}(p,t_1)$,
  - $\boldsymbol{Post}(p,t) = \frac{\boldsymbol{Pre}(p_1,t_1)}{\boldsymbol{r}(t_1)\cdot\boldsymbol{Pre}(p_1,t_1)+\boldsymbol{r}(t_2)\cdot\boldsymbol{Pre}(p_1,t_2)} \cdot (\boldsymbol{r}(t_1) \cdot \boldsymbol{Post}(p,t_1) + \boldsymbol{Post}(p,t_2) \cdot \boldsymbol{r}(t_2))$

This rule can be applied for example to merge transitions t_{OK} and t_{KO} in Fig. 3.a. Let us call this transition t_{OK-KO}.

VI. Agglomeration of immediate transitions in timed contPNs

Pre- and post-fusion rules for discrete PNs were first introduced by Berthelot in [8]. In their original form, they were designed for autonomous discrete nets, but they have been extended/refined afterwards for different kinds of timing [9], [10]. Similar rules can also be applied for continuous nets. Fluidification simplifies the problem of weights in the input arcs of a transition, which were difficult to deal with in the discrete case, and also the problem of transitions in a topologically equal conflict, since they can be merged using the result in the previous section.

Let p be a place with only one output transition t which does not have other input place. That is ${}^\bullet t = p$ and $p^\bullet = t$, see Fig. 5.a. Both p and t can be removed, and the input transitions of p will now split the flow among the output places of t. More formally, let ${}^\bullet p = \{t_1, \cdots t_k\}$, $t^\bullet = \{p_1, \cdots p_r\}$ and $\boldsymbol{m}_0(p) = 0$. In the reduced net, $\boldsymbol{Post}(p_i,t_j) = \boldsymbol{Post}(p,t_j) \cdot \boldsymbol{Post}(p_i,t)/\boldsymbol{Pre}(p,t)$. In the figure the input transitions are represented as timed, but the transformation can be used also with immediate transitions.

This rule allows to remove transitions t_{start}, t_{sync} and t_{OK-KO} in the net in Fig. 3.a (recall that place p_5 had been removed before). Hence, in this case all the immediate transitions of the net have been removed.

A similar rule can be obtained changing the role of immediate and timed transitions. On one side this rule is slightly more restrictive because it requires that none of the immediate transitions is in structural conflict relation with any other transition (that is, all the immediate transitions are persistent) and their only output is p. On the other side, it is more general since it allows synchronizations in t (see Fig. 6.a). The arc weights are computed as in the previous rule.

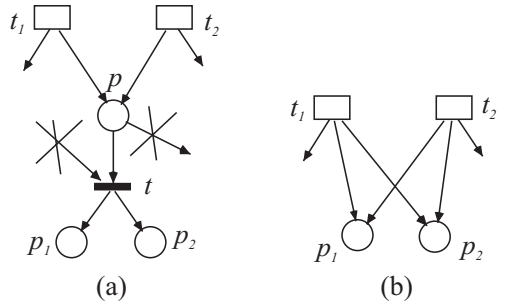

Fig. 5. Removing a non-synchronizing immediate transition that is not in conflict relation with any other transition

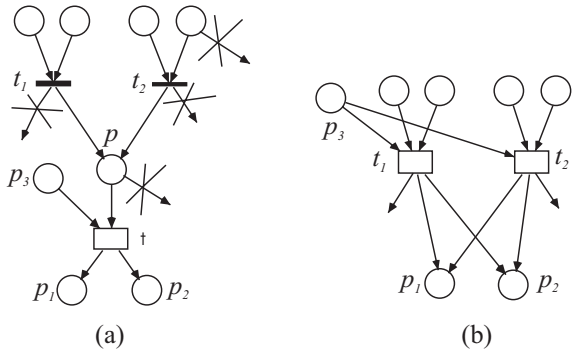

Fig. 6. A kind of symmetric rule to the one presented in Fig. 5

Other situation that can be simplified can be seen in Fig. 7.a. This kind of situation appears in the example we will study in the following section, taken from [5]. In this case, a timed transition is followed by an immediate one. It is not difficult to see that although a synchronization appears at the immediate transition, it will never stop the flow that comes from the input transition as long as no other transition consumes tokens from the place represented as p_2. That is, p_2 could have other output arcs if they are equal-weighted self-loops. Then, (p_2, t_2) is implicit and so, transition t_2 will fire as soon as tokens arrive to p_1, and its firing takes no time. Hence, both transitions can be fused. Observe that this could not have been done if both were timed, or if t_1 were immediate and t_2 were timed.

VII. A second example

In this section, we will apply some of the previous rules to a manufacturing example from the literature. This net can be seen in Fig. 8 and has been taken from [5] (Fig. 97), adding the arcs indicated in the text to avoid deadlock situations. Transitions $part_a$ and $part_b$ are in EQ relation and can be grouped, and the transition that appears can be fused with the previous one. Transitions $outM_2^a$ and $outM_2^b$ can be fused with their previous transitions (rule in Fig. 5). Transitions inM_1^a and inM_3^b can be reduced applying the rule shown in

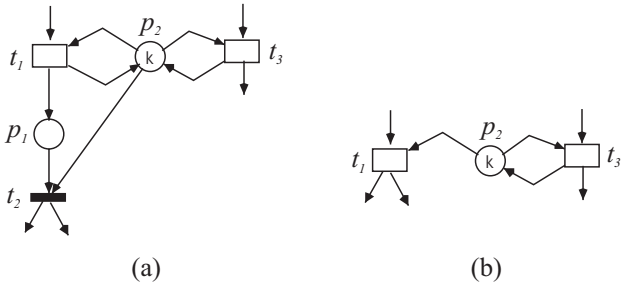

Fig. 7. All the tokens that arrive to place p_1 will immediately go through

TABLE I
COMPARISONS FOR THE NET IN FIG. 8

	Discrete	Continuous	
		Original	Reduced
Throughput(LU)	0.0446	0.051	0.051
Throughput(mv^a_{5-LU})	0.0357	0.040	0.040
Throughput(mv^b_{5-LU})	0.0089	0.010	0.010
Effort (in seconds)		197	146

Fig. 7. Hence, the only immediate transitions that remain are $outM^a_1$, inM^a_2, inM^b_2 and $outM^b_3$.

The results can be seen in Table I. First, observe that throughput at steady state is quite similar to the discrete case. Since all the reduction techniques we have applied are exact, the performance of the original continuous net and of the reduced one are the same. The simulation time has been reduced approximately in 25%.

VIII. CONCLUDING REMARKS

In this paper, a semantics for immediate transitions in contPN has been presented. The introduction of this kind of transitions means that the simulation of a model has to be done in two steps: after each time step, any enabled immediate transition has to be fired before going on with the timed ones.

Synchronizations are related to minimum operators, which introduce nonlinearity in the model. Some sufficient conditions have been presented that allow to detect "false synchronizations" both in the autonomous model (hence for any timing) and in the model with a particular timing. Of course, not all the synchronizations can be removed in general, this would mean that the model is in fact a linear ODEs system.

Immediate (and timed) transitions can also be removed in other cases. Some simple rules have been presented here, mainly when conflicts and synchronizations are not mixed. We have applied all the previous ideas to an example, to show that this may significatively reduce the simulation effort. However, there are still situations in which none of the rules can be applied. Further work is required in order to improve and generalize these practical transformation rules.

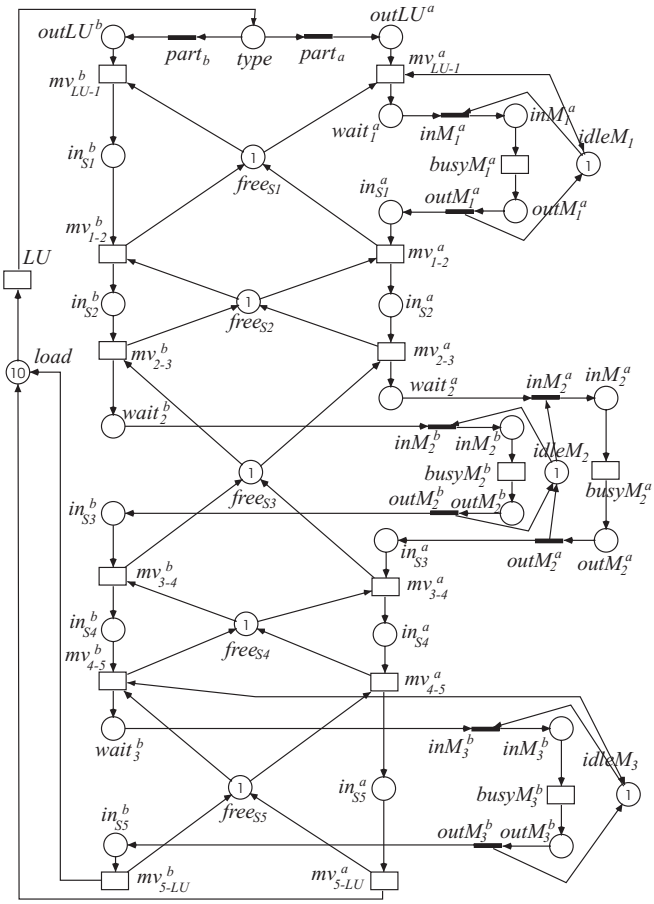

Fig. 8. A manufacturing example taken from [5]

REFERENCES

[1] M. Silva and L. Recalde, "On fluidification of Petri net models: from discrete to hybrid and continuous models," *Annual Reviews in Control*, vol. 28, no. 2, pp. 253–266, 2004.

[2] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*. Springer-Verlag, 2004.

[3] C. Mahulea, L. Recalde, and M. Silva, "On performance monotonicity and basic servers semantics of continuous Petri nets," in *Accepted at WODES06*, 2006.

[4] M. Silva, E. Teruel, and J. M. Colom, "Linear algebraic and linear programming techniques for the analysis of net systems," in *Lectures in Petri Nets. I: Basic Models*, ser. Lecture Notes in Computer Science, G. Rozenberg and W. Reisig, Eds. Springer, 1998, vol. 1491, pp. 309–373.

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

[6] M. Silva and E. Teruel, "A systems theory perspective of discrete event dynamic systems: The Petri net paradigm," in *Symposium on Discrete Events and Manufacturing Systems. CESA '96 IMACS Multiconference*, P. Borne, J. C. Gentina, E. Craye, and S. El Khattabi, Eds., Lille, France, July 1996, pp. 1–12.

[7] J. Júlvez, L. Recalde, and M. Silva, "Steady-state performance evaluation of continuous mono-t-semiflow Petri nets," *Automatica*, vol. 41, no. 4, pp. 605–616, 2005.

[8] G. Berthelot, "Checking properties of nets using transformations," in *Advances in Petri Nets 1985*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer, 1986, vol. 222, pp. 19–40.

[9] J. C. Mugarza, H. Camus, J.-C. Gentina, E. Teruel, and M. Silva, "Reducing the computational complexity of scheduling problems in petri nets by means of transformation rules." in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'98)*, Oct. 1998, pp. 19–25.

[10] R. H. Sloan and U. Buy, "Reduction rules for time Petri nets," *Acta Informatica*, vol. 33, no. 7, pp. 687–706, 1996.