Scenario durations characterization of t-timed Petri nets using linear logic

B. Pradin-Chézalviel*, R. Valette*, L.A. Künzle**

* LAAS-CNRS, 7 av. du Col. Roche, 31077 Toulouse Cedex 4, France
** CEFET-PR, Avenida 7 de setembro, Curitiba PR, Brasil

Abstract

This paper aims to handle scenario durations of t-timed Petri nets without constructing the class graph. We use a linear logic characterization of scenarios based on the equivalence between reachability in Petri nets and provability of a class of linear logic sequents. It has been shown that it was possible to characterize a scenario with concurrency induced both by the Petri net structure and by the marking. This approach is based on the rewriting the linear logic proof of the sequent. But this approach is limited because some structural concurrency cannot be expressed. In this paper we develop a new approach based on a canonical proof of the sequent. It does not explicitly characterize the scenario but it delimits its duration through an algebraic symbolic expression. It allows handling non safe Petri nets and structures which cannot be uniquely characterized by "sequence" and "parallel" operations.

1 Introduction

The characterization of the behavior of concurrent systems represented by Petri nets, and more specifically properties concerning the reachability problem of some states, may be addressed by two different approaches:

- either the proof of the reachability of a set of states, delimited by the fact that some logical proposition is true (but these methods need to construct state graphs),
- or the proof that, from a class of initial markings, a set of transitions verifying some partial order constraints can be fired, producing then a class of final markings.

The first kind of approach is well suited when no explicit quantitative timing consideration is involved. Modal logic and various classes of temporal logics are then particularly efficient. When explicit durations are attached to transition firings, it is then possible to build the reachability class graph [Be 92, Me 83, Me 85] in place of the reachable marking graph. But various difficulties exist:

- the delimitations of the durations of the scenarios are imprecise because the reachability class graph is built in relative time (at each transition firing the time is reset to zero),
- the state space is very large and the concurrency relations are not exploited (if t_1 and t_2 are concurrent, then the sequences $t_1\,t_2$ and $t_2\,t_1$ are both explicitly considered),

- the durations have to be delimited by values, so, no symbolic computation is offered.

The second kind of approach is important for scheduling problems (in manufacturing systems for instance), for diagnosis etc. It allows the computation of scenario durations taking into account the partial order constraints. These durations are necessary to derive the delivery dates of products in a manufacturing system or to define the likelyhood of sequences of fault events by comparing the duration of the corresponding scenarios with the time interval between the current observation and the last well known state.

This approach exhibiting partial order relations among transition firings has typically been based on unfoldings. Transitions are duplicated when they are fired again and places when a token is put again into them. Generally, the algorithms are restricted to safe Petri nets because situations for which a transition is concurrent with itself are not clear. A graphical tool has been developed [DE 97].

Our approach differs from this one because it is neither based on a graphical reasoning nor on a model checking procedure. In place of using the semantical aspect of logic, we use the syntactical one. We do not check if a graphical structure is a model of some formula, we prove a sequent.

The main benefits of this logical framework is that it is possible to formally define the notion of scenario (it is a sequent). The concurrency and partial order relations are taken into account, but are not turned explicit. This framework does not implies any restrictions such as the fact that the net is safe or without any loop (but the combinatorial explosion is not avoided when it is the consequence of conflicts which are not solved). It is possible to introduce quantitative durations by adding annotations to the logical atoms and associating algebraic operations on these annotations to the rules which are used to prove the sequent. The proofs are not altered by the annotations. It is therefore possible to derive algebraic expressions for scenarios durations in function of the durations attached to the transitions.

Linear logic, as defined by J.Y.Girard [Gi 87, Gi 95], has been chosen to formalize this approach because it allows the logical expression of the notion of state change. Representing state changes within a pure logical framework becomes possible because the two rules of sequent calculus expressing the fact that logical propositions are eternal truth (contraction and weakening) have been suppressed. This modification has two consequences:

- as equivalent ways of introducing the connectives *"and"* and *"or"* in classical logic are no longer equivalent in linear logic, these two connectives are split into four ones: *"times"*, *"par"*, *"with"* and *"plus"*,

- the operators *"times"* and *"par"* are no longer idempotent.

In linear logic, propositions are considered as resources which are consumed and produced at each state change. The only connectives used in this paper are the connective *"times"* denoted by $\otimes$ and *"linear implies"* denoted by $-\!\circ$. The first one represents the simultaneous availability of various resources and the second one the availability of a state change. That is, $A \otimes B$ denotes the fact that A and B are simultaneously available and $A -\!\circ B$ that by consuming A a possible state change produces B. The rules of linear intuitionistic logic ILL which are used in this paper can be found in annex at the end of the paper.

This paper exploits the equivalence between the reachability of a marking M' from a marking M in a Petri net and the provability of a family of sequents in the multiplicative fragment of linear logic. As a matter of fact, the proof tree constructed for proving the linear logic sequent (using the

sequent calculus rules) is used to point out the strictly necessary order relations existing among transition firings (those which are a logical consequence of the Petri net structure and the initial marking M). When it is possible to extract these order relations, the duration of the scenario can be exactly deduced with an algebraic formula.

Section 2 presents the relationships between linear logic and Petri nets. Starting from past work, it will be shown how it is possible, in some cases, to derive an algebraic expression, based on the connectives *"sequence"* and *"parallel"* expliciting the partial order among the transition firings in a scenario. This approach takes into consideration the order relations induced by the structure of the Petri net, but also the ones which are a consequence of the markings. From this expression, it is easy to obtain an algebraic expression of the scenario duration. But this approach is limited. An example of Petri net for which the partial order relations and the concurrency cannot be explicited by means of the two connectives is given.

Section 3 introduces then a new approach. The order relations are no longer explicitly represented by an algebraic equation, they are just scanned when building the proof tree and temporal labels are attached to the atoms denoting the tokens in order to compute the scenario duration. We have no longer an algebraic expression of the scenario, but we preserve an algebraic expression of the scenario duration. The discussion is restricted to the case of scenarios which are completely specified, that is scenarios in which no un-resolved conflict occurs.

Section 4 first presents the algorithm for a canonical construction of the proof tree complemented with the duration computation.
In order to see how structural concurrency is handled, the Petri net previously considered in section 2 (for which no algebraic expression of the scenario exists) is considered and the scenario duration is correctly expressed. This example points out the ability to consider some non trivial structural concurrency relations.
This section also presents how it is possible, using the proposed algorithm, to consider examples where markings induce some supplementary concurrency relations. We then deal with non safe Petri nets and with transitions concurrently fired with themselves.
The proposed method can be seen as a symbolic simulation.

Finally, the conclusion compares this approach with other ones and gives some hints for the case of scenarios which are not completely specified.

2 Relations between linear logic and Petri nets

2.1 Past work

Early in 1989, the relationships between linear logic and Petri nets have been established [Br 89]. An equivalence between reachability in Petri net theory and provability for the multiplicative fragment of linear logic has been proved through category theory [Ma 91]. Some papers are also concerned with the fact that Petri nets could be considered as models of linear logic formulas [EW 90].

However, it is the approach of C. Gunter and V. Gehlot [Gu 89, Ge 92] which has really motivated us. They have used this equivalence to try to formalize and characterize transition firing scenarios.

2.1.1 Transition firing scenarios

Informally, a transition firing scenario in a Petri net is a multi-set of transition firings (denoting events) which permit to reach a final marking M' from an initial marking M. Each transition has to be fired the number of times it appears in the multi-set. These transitions cannot be fired in any order. The transition firings have to respect the constraint that only enabled transitions can be fired. This constraint induces a partial order among the transition firings. This partial order depends on the structure of the Petri net, but also on the marking M. It is relatively complex because a transition may be fired concurrently with itself (if the token loads of its input places are sufficient) and the partial order constraints which have to be verified by the transitions may differ according to their firing occurrences. For example, it is possible that the first firing of t_3 is preceded by the first firing of t_1 and that the second firing of t_3 is preceded by the first firing of t_2. It is why, in general, partial order constraints as well as independence (or concurrency) relations cannot be directly defined on the alphabet T of the transition names.

In Petri net theory, there are two ways for defining a scenario. The first one is the notion of a firing sequence s which is an ordered list of transitions. The scenario is then represented by :

$$M \xrightarrow{s} M' \tag{1}$$

When firings are assumed to be instantaneous, this representation is generally sufficient. When explicit quantitative durations are attached to the transitions [Ra 73], it is necessary to express the concurrency (independence) relations and/or the partial order relations existing among the transition firings of the scenario. In addition, when industrial applications are involved, it is not reasonable to restrict to safe Petri nets or to Petri nets for which a transition cannot be fired concurrently with itself. For instance, in a manufacturing system it is possible that two parts of the same type are concurrently machined on two machines of the same pool.

2.1.2 Computation of scenario durations

The fact that the duration of the scenario is not the sum of the durations on the transition firings depends on the partial order and the issue is not fundamentally different if durations are attached to places [Si 77] or if they are enabling durations of transitions [Me 76].

An important point to be underlined is that the partial order among the transition firings is also a consequence of the initial marking M and cannot uniquely be derived from the Petri net structure or from the difference between the final marking M' and the initial marking. This means that the fundamental equation:

$$M' - M = C.\overline{s} \tag{2}$$

where C is the incidence matrix and $\overline{s}$ the characteristic vector of the scenario (describing the corresponding multiset of transition firings) will be of little help.

To illustrate this point, let us consider the Petri net in figure 1. If it is assumed that the firing duration of t_1 is d_1 and if the initial marking is such that there are two tokens in place A and one token in place C and if the final marking is such that there are two tokens in B and one in C, then an intuitive simulation shows that the scenario duration is $d_1 + d_1$.

In contrast, if the initial marking is such that there are two tokens in C as well as in A, then t_1 is fired concurrently with itself and the scenario duration is d_1.

In this paper, we will assume that the considered class of nets is the t-timed Petri net one, for which the durations are firing durations. It does not seem to have a strong impact on our approach and, as a first step, it is less complex.

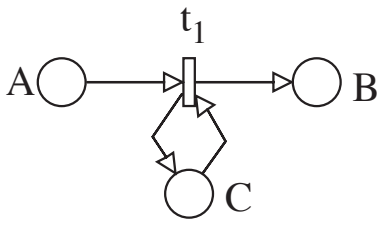

Figure 1: Example of scenarios depending on the initial marking

2.1.3 Gehlot's approach

In his approach, [Ge 92, Gu 89] V. Gehlot proposes to characterize partial order among transition firings by an algebraic expression based on two operators: *"sequence"* denoted by ";" and *"parallel"* denoted by "$\|$".

In the above example (figure 1), the two previously described scenarios would be respectively expressed by $(t_1; t_1)$ and $(t_1 \parallel t_1)$.

The issue is then, how is it possible to derive this algebraic expression from the Petri net and the initial marking of the scenario. V. Gehlot suggested to start from any proof tree proving the scenario in linear logic and to transform it, by means of a set of re-writing rules, into another one corresponding to maximal concurrency (and therefore a minimization of the partial order constraints). As intuitively the introduction of a partial order constraint between two transition firings corresponds to the use of a *"cut"* rule in a proof tree, V. Gehlot has suggested that rules similar to those used to eliminate *"cuts"* would be the solution. The algebraic expression is then directly obtained from the proof tree with maximal concurrency.

2.2 Turning markings explicit

In his PhD, L.A. Künzle [Ku 97, Pr 99] showed that it was possible to derive, from the algebraic characterization of the sequence, an algebraic expression of its duration, even in the case of a fuzzy trapezoidal delimitation of the firing durations. He also pointed out that V. Gehlot was only taking into account the partial order relations which were a consequence of the Petri net structure, and that in consequence, the concurrency which could be introduced by the marking was not taken into account.

Basically, V. Gehlot denotes transitions by proper axioms and does not explicitly represent markings. For instance, the Petri net in figure 1 is represented by:

$$A \otimes C \vdash B \otimes C \tag{3}$$

As a consequence, whatever the initial marking of the scenario, and therefore whatever the initial token load of place C, only formula $(t_1; t_1)$ will be derived.

2.2.1 A new logical representation of Petri nets

In the approach developed at LAAS [Gi 97a], the representation of Petri nets by means of linear logic is different and does not involve the use of added proper axioms. Transitions are denoted by linear logic propositions which are consumed when fired. The number of times a transition is consumed in a sequent corresponds to the number of times it has been fired in the corresponding scenario.

A marking M is a monomial in $\otimes$, that is a marking is represented by $M = A_1 \otimes A_2 \otimes \ldots \otimes A_k$ where A_i are place names. For instance, the two initial markings of the scenarios considered on the Petri net in figure 1 are $A \otimes A \otimes C$ and $A \otimes A \otimes C \otimes C$.

A transition is an expression of the form:

$$M_1 \multimap M_2 \tag{4}$$

where M_1 and M_2 are markings. For example, transition t_1 of the Petri net in figure 1 is noted $A \otimes C \multimap B \otimes C$.

A sequent represents a transition firing (and not a transition as in Gehlot's approach). The sequent corresponding to one firing of transition $M_1 \multimap M_2$ from a minimal initial marking is the following:

$$M_1, (M_1 \multimap M_2) \vdash M_2 \tag{5}$$

Straightforwardly, the proof of such a sequent results from the rule "$\multimap$L" which introduces linear implication in the left part of a sequent in linear logic (a sequent calculus proof is read from bottom to top):

$$\frac{\dfrac{}{M_1 \vdash M_1}\,\text{id} \quad \dfrac{}{M_2 \vdash M_2}\,\text{id}}{M_1, (M_1 \multimap M_2) \vdash M_2}\,\multimap\text{L} \tag{6}$$

In the context of this new representation, the equivalence between Petri net reachability and linear logic provability has been proved in [Gi 97b] without need of category theory. It is more precise than the classical one because it involves the number of times each transition is fired. This result can be expressed by the following theorem.

Theorem 1 *Let σ be a list of transition names separated by commas (where transition t_i appears n_i times), then the two propositions are equivalent:*

- *$M \xrightarrow{s} M'$ where s is a sequence of firings of the transitions of σ (transition t_i is fired n_i times)*
- *Sequent $M, \sigma \vdash M'$ is provable in the strict framework of linear logic (without added proper axioms).*

If the sequent is provable, then there exists at least one sequence. But more than one sequence s can correspond to a list σ. In this framework, L.A. Künzle [Ku 97, Pr 99], developed a mechanism for rewriting proof trees inspired from that of Gehlot but taking into account the initial marking of the scenario when it is larger than the minimal required one. So, in the case of the Petri net in figure 1 and with an initial token load of 2 in place A, two different algebraic expressions are derived according to the fact that the initial token load of C is 1 $(t_1; t_1)$ or 2 $(t_1 \parallel t_1)$.

After rewriting, it is possible to directly derive from the proof tree an algebraic expression of the scenario using the two operators ";" (for sequence) and "$\|$" (for concurrency).

2.2.2 Calculating scenario duration

From any algebraic expression, it is easy to calculate its duration. Duration of sequential firings of 2 transitions is obtained by adding the two durations while concurrent firing duration is obtained by a maximum operation. For the two previous scenarios, if duration of t_1 is d_1 we get respectively $2.d_1$ for $(t_1; t_1)$ and d_1 for $(t_1 \parallel t_1)$.

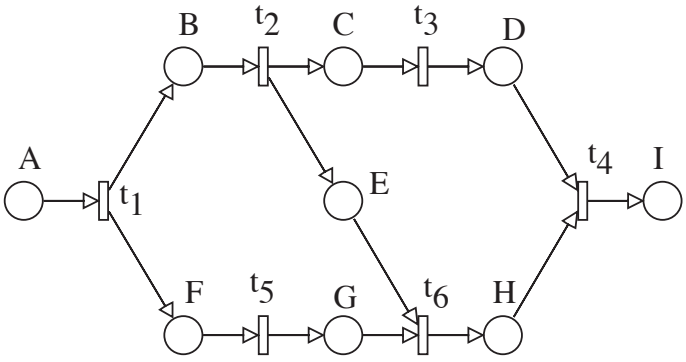

Figure 2: Example of scenario not expressible by ";" and "||"

2.2.3 Limitations of the above approach

Another result of L.A. Künzle's PhD is that some scenarios in Petri nets cannot be accurately characterized by means of the two operators ";" and "||". It is indeed not possible to accurately characterize the scenario starting from one token in A and ending with one token in I with the Petri net in figure 2.

As a matter of fact, three algebraic expressions may be written for this scenario σ:

$$t_1; (t_2 \parallel t_5); (t_3 \parallel t_6); t_4 \tag{7}$$

$$t_1; ((t_2; t_3) \parallel t_5); t_6; t_4 \tag{8}$$

$$t_1; t_2; (t_3 \parallel (t_5; t_6)); t_4 \tag{9}$$

From each expression a different algebraic formula can be derived for the scenario duration. For instance expression 7 gives (let d_i be the duration of t_i):

$$duration(\sigma) = d_1 + max(d_2, d_5) + max(d_3, d_6) + d_4 \tag{10}$$

When variables d_i are replaced by actual values, the obtained duration of the scenario may be larger than the actual one because each of the three scenario expressions adds at least one supplementary partial order constraint which is not a consequence of the Petri net structure and of the initial marking. For example, expression 7 induces a precedence relation between the firings of t_3 and t_5. According to the values of variables d_i this precedence relation may be verified or not by the scenario. If it is the case the scenario duration is correct, if not it is larger than the actual value. Let us, for example consider the following values:

$$d_1 = 0 \quad d_2 = 1 \quad d_3 = 2 \quad d_4 = 0 \quad d_5 = 2 \quad d_6 = 1 \tag{11}$$

The actual scenario duration is 3 whereas the formulas derived from expressions 7, 8 and 9 all give the value 4.

3 A new approach for scenario duration

3.1 Motivation

We have just seen that linear logic was able to handle partial order relations between transition firings. However, it has been illustrated that some scenarios could not be adequately character-

ized. This means that the explicit representation of partial order (*"cut"* rule) and concurrency in a proof tree is not general.

On the other hand, even in the case of the counter example in figure 2, it is possible, by simulation, to compute the duration of the scenario for given values of the transition firing durations. The question is: is it possible to formalize this computation in linear logic and therefore to derive in any case an algebraic expression of the scenario duration? In other words: is it possible to construct proof trees equivalent to a kind of symbolic simulation of a scenario in a Petri net?

The purpose of this section is to answer these questions. It is shown that indeed a symbolic algebraic computation of the scenario duration can be done, but the price to pay is the fact that partial order relations and concurrency among the transition firings are not explicitly represented.

In contrast to the preceding approaches ([Gu 89, Pr 99]), this one is not based on rewriting rules but on a canonical proof tree. This means that in place of deriving a proof without any restriction and then rewriting it in order to increase the concurrency, we restrict to a canonical procedure to derive the proof of the specific sequents representing scenarios. This procedure does not introduce partial orders if they are not a consequence of the Petri net structure or of the initial marking of the scenario. Any provable sequent (denoting a scenario) can be proved by this canonical procedure because each step of this procedure exactly reflects a transition firing in the net and provability and reachability proofs are equivalent.

3.2 Principles

3.2.1 Temporal labels

In linear logic, propositions are produced and consumed, exactly in the same way as tokens during the token game. In order to compute the scenario duration, we associate with each proposition denoting a token in a place its production date. For example, if a token is produced in place B at date d_1, we will write the logical proposition $B(d_1)$.

3.2.2 Rule introducing linear implication on the left

Proof (equation 6) shows that, with our linear logic representation of Petri nets, firing a transition corresponds to the introduction of a linear implication in the left part of the sequent. The corresponding rule in linear logic sequent calculus is noted ("—∘L"). If we want the proof tree to be similar to a symbolic execution of a scenario $M, \sigma \vdash M'$, then we have to define a canonical way of building it, just by applying this rule.

Let us consider a list of transitions $\sigma = t, \sigma'$ such that transition t represented by $M_1 — \circ M_2$ is enabled by the initial marking. This means that this marking can be written $M = M_1 \otimes M_3$. We can therefore derive the following proof tree:

$$\dfrac{\dfrac{\dfrac{}{M_1 \vdash M_1}\,\text{id} \quad M_2, M_3, \sigma' \vdash M'}{M_1, M_3, (M_1 — \circ M_2), \sigma' \vdash M'}\,— \circ \text{L}}{M_1 \otimes M_3, (M_1 — \circ M_2), \sigma' \vdash M'}\,\otimes \text{L} \tag{12}$$

Proof tree 12 points out the fact that it is necessary, before applying rule "—∘L" to break down the monomial in $\otimes$ describing the marking. This is done by the rule "$\otimes$L" which introduces connective $\otimes$ in the left. In proof tree 12 marking $M = M_1 \otimes M_3$ is broken down into two

monomials which become logically independent M_1 and M_3 (they are now only connected by the meta connective ",").

At the top of the proof tree 12 we have a sequent $(M_2, M_3, \sigma' \vdash M')$ which no longer contains transition t. It also contains two monomials connected the meta connective ",": M_2 and M_3. It appears then that in order to base a proof on the repetitive use of the rule "—∘L" and to correctly take into account the presence of tokens which are not required for firing the transition at hand, it is necessary to break down the monomial in $\otimes$ into a list of atoms separated by the meta connective ",". Each atom has an associated temporal label. The update of the temporal labels is detailed in the sequel, but in this example, if the label of M_1 is d and if the duration of t_1 is d_1, then the label of M_2 will be $d + d_1$. The label of M_3 remains unchanged.

3.2.3 Current step

Definition 1 *A* Current step *in a sequent is a list of atoms corresponding to place names, separated by the meta connective "," each one having an attached temporal label.*

Let us consider the Petri net in figure 2, an example of current step is "$B(d_B), F(d_F)$" where d_B is the production date of the token in B. Another one could be (it is not the case for the considered scenario) "$B(d_{B1}), B(d_{B2}), F(d_F)$", which points out the fact that the tokens are turned individuals by the temporal label. Two tokens in the same place but which have not been produced at the same time are differentiated.

A current step does not necessarily corresponds to a reachable marking of the scenario for specific values of the transition durations. For example, the existence of the current step:

$$C(d_C), E(d_E), G(d_G)$$

in a proof tree does not mean that the marking: $C \otimes E \otimes G$ will effectively be reached. This marking will be reached for some transition durations and not for other ones. What is independent of the transition durations is the fact that during one time interval there will be a token in place C, that during another time interval there will be a token in E and that during a third one there will be a token in G. These three time intervals may be disjoint.

Let us now consider the fragment of current step $E(d_E), G(d_G)$. As the places E and G are input places of a unique transition t_6, the tokens present in them have to be consumed by the firing of t_6. They will remain in E and G until this firing. The fragment of marking $E \otimes G$ will thus necessarily be reached. It will be reached at time $max(d_E, d_G)$, and on this date transition t_6 will be immediately fired (we assume here that transitions are fired as soon as possible).

It is always possible to transform a marking (or a marking fragment produced by a transition firing) into a current step by the repetitive application of the rule "⊗L". An example is given in the proof 12. The computation of the temporal label is simple, if the marking of the fragment has been produced on date d, then all the atoms will have the same label d. For example if $C \otimes E$ has been produced on date d_{CE}, the corresponding current step will be $C(d_{CE}), E(d_{CE})$.

3.3 Scenarios and conflicts

Let Ec be a current step, and let t be a transition of σ represented by $M_1 —\circ M_2$. The current step Ec and M_1 can be both considered as multi-sets. Rule "—∘L" can be used if and only if M_1 is included in Ec (all the atoms of M_1 are present in Ec and with a multiplicity which is equal or less).

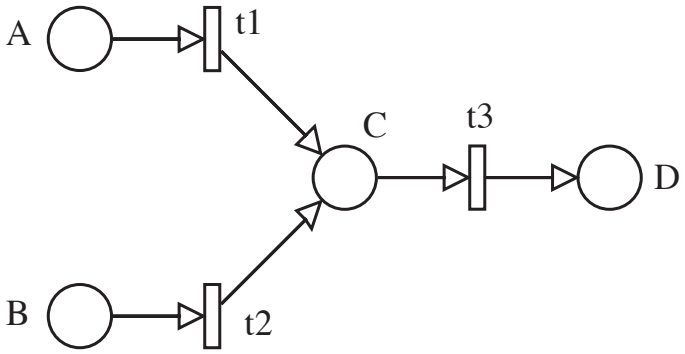

Figure 3: Example of scenario with conflicting tokens

3.3.1 Conflicting tokens

Definition 2 *Let Ec be a current step and t a transition represented by $M_1 — \circ M_2$ of a list σ of a sequent $Ec, \sigma \vdash M$, it is said that the tokens in place A are conflicting for t at current step Ec if and only if the multiplicity of A in Ec is greater than that of A in M_1 and their temporal labels are not completely ordered.*

In order to illustrate this definition, we are looking at three cases where conflicting tokens can be generated:

- Let us assume, for instance, that for some scenario over the net in figure 2 the current step $F(d_{F1}), F(d_{F2})$ is reached. We assume that transition t_5 has to be fired once in the remaining part of the scenario. Transition t_5 may be fired either with the token $F(d_{F1})$ or with the token $F(d_{F2})$. If d_{F1} differs from d_{F2} it is clear that the remaining part of the scenario will be different. If we want to consider all the alternatives, it is clear that two different proof trees (or two fragments of proof trees) have to be built: one corresponding to the firing of t_5 by $F(d_{F1})$, the other of that of t_5 by $F(d_{F2})$.

  If, for any values of the transition durations, d_{F2} (for instance) can be proved to be greater than d_{F1} and if transition t_5 is the only output transition of place F which has to be fired in the scenario, then it is obvious that the duration of the second proof tree will always be larger than the first one. It is well known, for example, that in an event graph the best performance is always obtained by firing transitions as soon as they are enabled [Ra 80]. It is why we restrict the notion of conflicting tokens to the case in which temporal labels are not completely ordered.

- Let us consider now the net in figure 3 and the scenario going from marking $A \otimes B$ to $C \otimes D$ by firing once each transition t_1, t_2 and t_3. Because of the marking t_1 and t_2 are concurrently fireable. If we have no information about t_1 and t_2 durations, two scenarios have to be considered: either duration d_1 is less than d_2 and t_3 is fired after the firing of t_1 or, reverserly d_1 is greater than d_2 and t_3 is fired after the firing of t_2. The durations of these two scenarios are respectively $max(d_2, d_1 + d_3)$ and $max(d_1, d_2 + d_3)$. We again get a case where it is necessary to construct two proof trees. This case illustrates the possibility of having no order relation between the temporal labels of the tokens.

- The same problem appears if a transition is concurrently fired with itself: it is possible to get the scenario duration by a unique formula if tokens are completely ordered. If not, several proof trees have to be generated.

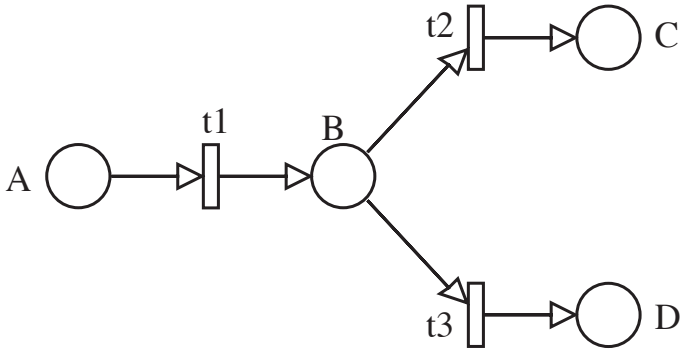

Figure 4: Example of scenario with conflicting transitions

3.3.2 Conflicting transitions

Definition 3 *Let Ec be a current step and t_1 and t_2 be two different transitions represented by $M_{11} \multimap M_{12}$ and by $M_{21} \multimap M_{22}$ which are elements of a sequent $Ec, \sigma \vdash M$ characterizing a scenario. Transitions t_1 and t_2 are conflicting at Ec if and only if there is an atom A belonging to Ec, M_{11} and M_{21}.*

As in the case of conflicting tokens, there is a choice, and the proof tree will be in general different for the various alternatives. Consider for example the Petri net in figure 4 and the scenario going from marking $A \otimes B$ to $C \otimes D$ by firing once each transition t_1, t_2 and t_3. As place B contains only one token, either t_2 or t_3 is fired first. If we have no information about how the conflict is solved, two scenarios have to be considered. If t_2 is fired first, t_3 will only be fired after the firing of t_1 and the duration is $max(d_2, d_1 + d_3)$. Reversely, if t_3 is fired first, the duration is $max(d_3, d_1 + d_2)$: it is necessary to construct two proof trees.

More generally, as a conflict involves two different transitions, this implies that there is at least a place with more than an output arc and it is known that the shortest duration is not necessarily obtained when transitions are fired as soon as they are enabled. In a manufacturing system this means that sometimes, it is better to reserve an available resource for a critical operation which will be possible soon than to immediately start a non critical but long operation. An example of such issues can be found in [Si 90].

It is why in definition 3 it is not stated that transitions t_1 and t_2 have both to be enabled. In fact, if t_1 is not enabled at Ec, this does not guarantee that it could not become enabled after the firing of another transition, say t_3 and that for specific values of the transition durations it is better to wait and prevent the enabled transition t_2 to be fired. Such a scheduling choice is equivalent to adding a partial order constraint between the firing of t_1 and that of t_2 (firing t_2 after t_1) which is neither a logical consequence of the structure of the Petri net nor of the initial marking of the scenario.

3.3.3 Incompletely specified scenarios

Definition 4 *Let us consider a sequent $M, \sigma \vdash M'$. If during the construction of the canonical proof tree conflicting tokens or conflicting transitions appear, then the corresponding scenario is said to be incompletely specified*

In such scenarios, the partial order relations among the transition firings are indeed not completely specified. If conflicting transitions appear, firing transitions as soon as they are enabled is

not necessarily the good choice: external decisions have to be made. They correspond to scheduling the transition firings of the scenario in order to choose the optimal duration one. If conflicting tokens appear, it means we get a current step where at least one place contains several tokens whose labels are not completely ordered. In such a case, it is necessary to construct several proof trees: they represent possible behaviors depending on actual values.

A specific problem arises with conflicting tokens: the algorithm used to construct the logical characterization of the scenario must ensure that all situations with possible conflicting tokens will be detected (all the possible canonical proof trees are not equivalent with respect to this problem). Consider for example the Petri net in figure 3 and the scenario going from $A \otimes B$ to $C \otimes D$ by firing once t_1, t_2 and t_3. As we do not know which of the two durations d_1 or d_2 is the smallest, we previously showed that two proof trees have to be constructed: the conflicting tokens are $C(d_1), C(d_2)$. The algorithm used to construct the proof tree has to explicitly point out this conflict. For example, after applying the logical rule corresponding to the firing of t_1, the current step is $C(d_1), B(0)$ and both rules for t_2 and t_3 can be applied. As t_2 can add a token in place C, the algorithm has to first apply the rule for t_2 in order to point out all possible conflicting tokens.

3.3.4 A class of completely specified scenarios

In this paper, we only focus on completely specified scenarios. In order to be certain that considered scenarios are completely specified ones on one hand and to get a simple algorithm on the other hand, we choose to only consider (as a first stage) scenarios corresponding to event graphs.

This condition can appear restrictive but, doing so we want to show which types of results can be obtained without using a very complex algorithm. In particular, we will consider in the rest of this paper Petri nets which were not correctly characterized with the previous approach in section 2.2 and also scenarios involving non safe markings.

The studied Petri nets are not limited to event graphs but before starting the construction of the proof tree, we will check that the scenario itself is an event graph: two transitions of this scenario have no common input places or output ones. This condition firstly ensures that there are no conflicting transitions and secondly that if a place contains several tokens they have been produced by the same transition and, consequently, it is possible to completely order their temporal labels. As it is an event graph, transitions are fired as soon as they are enabled.

4 Computation of the duration of a completely specified scenario

When the scenario is completely specified, at each current step, an atom (a token) can only be used to fire one transition instance of the list σ of the remaining transitions to be fired in the scenario. As a consequence, when for a current step, more than a transition of the remaining list is enabled it is that they are concurrent and their firing order has no consequence on the duration. Their firing dates are computed with different clocks (the temporal labels of the different tokens). Due to the notion of current step, concurrent clocks are explicitly handled. This would not have been the case with current markings.

4.1 Algorithm

It is then possible to present the following algorithm for the computation of completely specified scenarios.

Step 0 : verify the scenario is an event graph.

Step 1 : initial. Repetitively apply the "$\otimes$L" rule in order to separate all the atoms of the initial marking and assign to each of them the temporal label "0". This first step constructs the first current step. Go to step 2.

Step 2 : find a linear implication which can be eliminated. Search a transition of the transition list such that all the atoms of its left part are present in the current step with at least the sufficient multiplicity. If no such transition is found, the final marking cannot be reached and the sequent is not provable, (there is a deadlock in the Petri net on the date corresponding to the latest temporal label of the current step). Go to step 3.

Step 3 : transition firing. Apply rule "—$\circ$L" for the corresponding transition, let it be t_i for instance. In the upper left part of the rule, either an identity sequent ($A \vdash A$) is obtained, or a sequent of the form ($A_1, A_2 \vdash A_1 \otimes A_2$) which will be proved by applying rule "$\otimes$R". In the upper right part of the rule, the obtained sequent contains the output marking produced by transition t_i and the list of the remaining atoms of the current step after having removed those which were enabling t_i. Go to step 4.

Step 4 : computation of the temporal labels. The remaining atoms (upper right part of the rule) keep their values unchanged. The label of the output marking is derived by applying function "max" to the labels of the atoms which were enabling t_i (date for the earliest firing of t_i) and by summing it to the duration of t_i. By applying "$\otimes$L" repetitively, the output marking of transition t_i is then broken down into atoms. The temporal labels of these new atoms are equal to that of the marking. If any input place of t_i contains more tokens than necessary for enabling t_i, the consumed ones will be those which temporal labels are the smallest ones (transitions are fired as soon as possible). If the sequent at the upper right part of the rule contains at least one transition, go to step 2, otherwise go to step 5.

Step 5 : final. The remaining sequent is either an identity one ($A \vdash A$), or a sequent of the form ($A_1, A_2 \vdash A_1 \otimes A_2$). Otherwise the sequent is not provable. The scenario duration is then the result of the application of function "max" to the temporal labels of the remaining atoms A_i (the earliest date on which the final marking is reached).

By applying this algorithm, we can derive the scenario duration (in a first stage only for completely specified scenarios) without turning explicit the partial order constraints and the concurrency relations. This duration corresponds to a policy for which transitions are fired as soon as they are enabled and as the scenario is completely specified, it is the shortest schedule.

If a partial order among two transition firings has to be established, this can be done by examining the proof tree. In fact, each atom in a current step denotes a precedence relation between the transition firing (an application of rule "—$\circ$L") which has produced it and the transition firing which has consumed it. In contrast, the concurrency relations cannot be derived straightforwardly.

As the initial marking of the scenario is taken into account, partial order constraints are defined among transition firings, not among transitions (a transition may be fired several times in different contexts).

4.2 Structural concurrency

4.2.1 First example

Let us illustrate the main steps of the algorithm on the Petri net in figure 2 by computing the duration of the scenario:

$$A, t_1, t_2, t_3, t_4, t_5, t_6 \vdash I$$

Transitions are represented by the following expressions:

$$\begin{array}{lcl} t_1 \text{ (duration } d_1) & : & A \multimap B \otimes F \\ t_2 \text{ (duration } d_2) & : & B \multimap C \otimes E \\ t_3 \text{ (duration } d_3) & : & C \multimap D \\ t_4 \text{ (duration } d_4) & : & D \otimes H \multimap I \\ t_5 \text{ (duration } d_5) & : & F \multimap G \\ t_6 \text{ (duration } d_6) & : & E \otimes G \multimap H \end{array} \tag{13}$$

Transition names will only be replaced by their corresponding expressions in linear logic in the steps in which they are fired (application of rule "$\multimap$L").

This scenario can be considered because it corresponds to an event graph. The initial step is straightforward. For each transition, we write the rule. The current sequent (remaining part of the scenario) is written under the bar (bottom of the rule). This sequent starts with the current step, and then contains the list of the remaining transitions to be fired. The first one is the transition under examination. The upper right sequent is the one which will be examined in the next step.

Let us consider transition t_1 and detail this step. We have:

$$\dfrac{\dfrac{}{A \vdash A}\,\text{id} \quad \dfrac{B(d_1), F(d_1), t_2, t_3, t_4, t_5, t_6 \vdash I}{(B \otimes F)(d_1), t_2, t_3, t_4, t_5, t_6 \vdash I}\,\otimes\text{L}}{A(0), (A \multimap B \otimes F), t_2, t_3, t_4, t_5, t_6 \vdash I}\,\multimap t_1 \tag{14}$$

In the new current step, transitions t_2 and t_5 are enabled (they are not conflicting and can then be fired in any order). Whatever the choice, the same atoms (C, E, G) with the same temporal labels will be derived. Let us consider t_2 first and skip the exhibition of rule "$\otimes$L":

$$\dfrac{\dfrac{}{B \vdash B}\,\text{id} \quad C(d_1 + d_2), E(d_1 + d_2), F(d_1), t_3, t_4, t_5, t_6 \vdash I}{B(d_1), F(d_1), (B \multimap C \otimes E), t_3, t_4, t_5, t_6 \vdash I}\,t_2 \tag{15}$$

Let us now consider t_5:

$$\dfrac{F \vdash F \quad C(d_1 + d_2), E(d_1 + d_2), G(d_1 + d_5), t_3, t_4, t_6 \vdash I}{C(d_1 + d_2), E(d_1 + d_2), F(d_1), (F \multimap G), t_3, t_4, t_6 \vdash I}\,t_5 \tag{16}$$

Now transitions t_3 and t_6 are enabled. Let us consider t_3 firstly:

$$\dfrac{C \vdash C \quad D(d_1 + d_2 + d_3), E(d_1 + d_2), G(d_1 + d_5), t_4, t_6 \vdash I}{C(d_1 + d_2), E(d_1 + d_2), G(d_1 + d_5), (C \multimap D), t_4, t_6 \vdash I}\,t_3 \tag{17}$$

Then let us fire t_6:

$$\dfrac{E, G \vdash E \otimes G \quad D(d_1 + d_2 + d_3), H(d_1 + max(d_2, d_5) + d_6), t_4 \vdash I}{D(d_1 + d_2 + d_3), E(d_1 + d_2), G(d_1 + d_5), (E \otimes G \multimap H), t_4 \vdash I}\,t_6 \tag{18}$$

where the upper left part is not directly an identity and involves the use of rule "$\otimes$R" as follows:

$$\frac{\dfrac{}{E \vdash E}\,\text{id} \quad \dfrac{}{G \vdash G}\,\text{id}}{E, G \vdash E \otimes G}\,\otimes\text{R} \tag{19}$$

The fact that rule "$\otimes$R" has to be employed denotes the fact that the tokens E and G have to be synchronized and that the earliest time at which transition t_6 can be fired is

$$max((d_1 + d_2), (d_1 + d_5)) = d_1 + max(d_2, d_5)$$

And finally let us fire t_4:

$$\frac{D, H \vdash D \otimes H \qquad SEQ_{fin}}{D(d_1 + d_2 + d_3), H(d_1 + max(d_2, d_5) + d_6), (D \otimes H \multimap I) \vdash I}\,t_4 \tag{20}$$

with

$$SEQ_{fin} = I(d_1 + max((d_2 + d_3),\ (max(d_2, d_5) + d_6)) + d_4), \vdash I$$

As the scenario was completely specified, its duration has been completely characterized by a unique proof tree. The algebraic expression of the duration is:

$$duration(\sigma) = d_1 + max((d_2 + d_3), (max(d_2, d_5) + d_6)) + d_4 \tag{21}$$

which can be written:

$$duration(\sigma) == d_1 + max(d_2 + d_3,\ d_2 + d_6, d_5 + d_6) + d_4 \tag{22}$$

This value is the accurate one whatever the values of the transition durations. We can check that for the values specified in table 11 the scenario duration is 3, the correct value. For a classical simulation, based on a notion of current marking, it would have been necessary to specify values in order to know on each date, the current marking. The approach presented here can be seen as a symbolic simulation.

4.2.2 Second example

Let us point out another interesting example. Using this algorithm it is possible to consider scenarios corresponding to cycles in the Petri net. For example, if there is a transition t_7 in the Petri net in figure 2 going from place I to place A (as represented in figure 5), it is possible to consider the scenario σ':

$$A, t_1, t_2, t_3, t_4, t_5, t_6, t_7 \vdash A$$

Constructing the proof tree we get the duration of this scenario:

$$duration(\sigma') = d_1 + max((d_2 + d_3), (max(d_2, d_5) + d_6)) + d_4 + d_7 \tag{23}$$

These two scenario examples have shown how the method permits to correctly characterize non elementary concurrent transition firings but in these examples concurrency was only generated by the structure of the Petri net. In next section we are going to consider concurrency generated by markings.

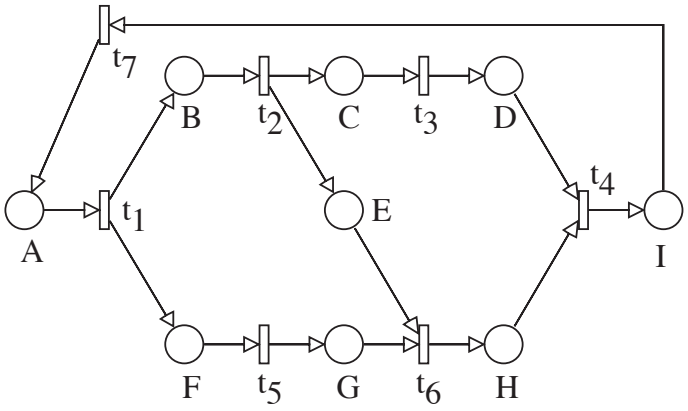

Figure 5: Example of scenario with a loop

4.3 Concurrency induced by markings

4.3.1 Concurrent firings of a transition

Let us first look at a simple net, the one in figure 1 and apply the algorithm to the scenario:

$$A{\otimes}A{\otimes}C, t_1, t_1 \vdash B{\otimes}B{\otimes}C$$

Firing transition t_1, we have:

$$\frac{\dfrac{}{A \vdash A}\,\text{id} \quad A(0), B(d_1), C(d_1), t_1 \vdash B{\otimes}B{\otimes}C}{A(0), A(0), C(0), (A{\otimes}C{\multimap}B{\otimes}C), t_1 \vdash B{\otimes}B{\otimes}C}\,t_1 \tag{24}$$

Then, we can fire t_1 once more:

$$\frac{\dfrac{}{A \vdash A}\,\text{id} \qquad SEQ_1}{A(0), B(d_1), C(d_1), (A{\otimes}C{\multimap}B{\otimes}C) \vdash B{\otimes}B{\otimes}C}\,t_1 \tag{25}$$

with

$$SEQ_1 = B(max(0, d_1) + d_1), B(d_1), C(max(0, d_1) + d_1) \vdash B{\otimes}B{\otimes}C$$

The logical proof of SEQ_1 is:

$$\frac{\dfrac{}{B \vdash B}\,\text{id} \quad \dfrac{\dfrac{}{B \vdash B}\,\text{id} \quad \dfrac{}{C \vdash C}\,\text{id}}{B, C \vdash B{\otimes}C}\,{\otimes}\text{R}}{B, B, C \vdash B{\otimes}B{\otimes}C}\,{\otimes}\text{R} \tag{26}$$

and thus, the duration is obtained by considering the max of the labels of all tokens. In this example, this value is $2.d_1$: it is the correct one because, as there is only one token in place C in the initial marking, t_1 cannot be concurrently fired with itself.

Consider now the scenario:

$$A{\otimes}A{\otimes}C{\otimes}C, t_1, t_1 \vdash B{\otimes}B{\otimes}C{\otimes}C$$

Firing transition t_1, we have (we just give the main points of the proof):

$$\frac{A \vdash A \quad A(0), C(0), B(d_1), C(d_1), t_1 \vdash B \otimes B \otimes C \otimes C}{A(0), A(0), C(0), C(0), (A \otimes C \multimap B \otimes C), t_1 \vdash B \otimes B \otimes C \otimes C} \, t_1 \tag{27}$$

Then, we can fire t_1 once more (using the tokens with the smallest temporal labels):

$$\frac{A \vdash A \quad B(d_1), C(d_1), B(d_1), C(d_1) \vdash B \otimes B \otimes C \otimes C}{A(0), C(0), B(d_1), C(d_1), (A \otimes C \multimap B \otimes C) \vdash B \otimes B \otimes C \otimes C} \, t_1 \tag{28}$$

The duration is now d_1 because of the concurrent firing of t_1 with itself. It must be pointed out that there are no conflicting tokens because at the first step the two tokens in place C have the same temporal label (0) and at the second step their labels are completely ordered (one is (0) and the other one (d_1)).

4.3.2 Concurrent firings between different transitions

In previous examples we have shown that concurrency of a transition with itself was correctly considered. We are now looking at non structural concurrency between different transitions. Let us consider again the net in figure 2 and the scenario σ_1:

$$A \otimes E, t_1, t_2, t_3, t_4, t_5, t_6 \vdash I \otimes E$$

The duration obtained when constructing the proof tree is:

$$duration(\sigma_1) = d_1 + max(d_2 + d_3, d_5 + d_6) + d_4 \tag{29}$$

The scenario presented here exactly involves the same transitions as the one considered in section 4.2.1 (formula 22) but the initial and final markings are different. In this case t_6 can be fired before t_2 and, consequently, the scenario duration is no more dependent of $d_2 + d_6$.

Let us consider a last example on this net, the scenario σ_2 again involves the same set of transitions but differs on initial and final markings:

$$A \otimes F, t_1, t_2, t_3, t_4, t_5, t_6 \vdash I \otimes F$$

The duration is:

$$duration(\sigma_2) = max(d_1 + d_2 + d_3, d_1 + d_2 + d_6, d_5 + d_6) + d_4 \tag{30}$$

This result shows that t_5 can be concurrently fired with t_1, although these two transitions are tied up by a structural precedence relation: the scenario duration is no more dependent of $d_1 + d_5$.

4.4 Discussion

In the context of the above examples, let us compare our approach with Petri net unfordings and with the construction of PERT graphs for the computation of the overall duration of a set of tasks with partial order constraints.

The first point is that in any case we directly operate on the Petri net, no specific graph has to be constructed for each scenario. Scenarios are sequents to be proved and the Petri net structure

is simply used to write down the formulas denoting the transitions. For example, it is possible to consider that all the scenarios are defined on the net represented in figure5.

The first scenario (presented in section 4.2.1):

$$A, t_1, t_2, t_3, t_4, t_5, t_6 \vdash I$$

is such that the three paths of the corresponding PERT graph (from A to I via C, E and G) are all present in the Petri net. The unfolding procedure would have been straightforward.

The second scenario (presented in section 4.2.2):

$$A, t_1, t_2, t_3, t_4, t_5, t_6, t_7 \vdash A$$

just requires the duplication of place A, the paths are those of the net. It remains simple.

The third scenario (see section 4.3.2):

$$A \otimes E, t_1, t_2, t_3, t_4, t_5, t_6 \vdash I \otimes E$$

has a duration expression which only involves two paths as it can be seen in expression 29. These paths are those connecting A to I via C and G. The path connecting A to I via E has been deleted, and this is in relation with the fact that place E is not structurally safe. It must be remarked that there are no conflicting tokens because the two tokens which appear in place E (not necessarily in the same proof step) are totally ordered. Their labels are (0) and $(d_1 + d_2)$ (the length of the path connecting A to E). As the transitions are fired as soon as they are enabled in a completely specified scenario, transition t_6 will always be fired with the token arrived first in E.

The fourth scenario (end of section 4.3.2):

$$A \otimes F, t_1, t_2, t_3, t_4, t_5, t_6 \vdash I \otimes F$$

and the analysis of its duration expression 30 shows that the equivalent PERT graphs contains three paths, the two first are connecting A to I via C and E, the last one connects F to I. Exactly as in the preceding case, the tokens in place F have completely ordered labels ((0) and (d_1)).

In our approach, all these cases have been addressed exactly in the same way, without any graphical reasoning, just by constructing the proof of the sequent. The length of the proof, linearly depends on the number of transitions in the scenario (when it is completely specified).

5 Conclusion

We have shown that even when it was not possible to explicitly characterize the partial order and concurrency relations between the transitions of a Petri net by an algebraic formula, it was possible to derive an accurate algebraic formula of the scenario duration. This means that all concurrency relations are taken into account. An algorithm solving this issue in the case of completely specified scenarios has been presented. The fact that the scenario is completely specified is guaranted by restricting to event graphs scenarios. The Petri nets can be more complex, but at a first stage, we limit our approach to a class of completely specified scenarios for which the condition is easy to check.

We have pointed out examples in which the linear logic representation of Petri nets was fruitful. The current step is indeed not the current marking and the whole approach is based on the

fact that it is interesting to consider a list of tokens distributed in the places which is not a marking because they are not necessarily all considered on the same date.

On these examples, we have shown that it was possible to characterize the structural concurrency on one hand as well as the concurrency due to markings in other hand. Consequently, this algorithm can be used for non safe Petri nets or for cycles.

With respect to the class graph (for time Petri nets) we can handle more than one token in a place, we only construct a fragment of the classes because we have an explicit list of transitions to be fired and we exploit concurrency. However, the class graph and the approach presented here are complementary. Indeed, when contructing a canonical proof tree we only handle current steps and we have no information about reachability of intermediate markings.

This algorithm can be extended to deal with completely specified scenarios that are not event graphs but we are essentially interested in an exploration of incompletely specified scenarios. A proof tree will be generated for each schedule in the same way it has been elaborated for a completely specified scenario. The main issue is to introduce the minimal of supplementary partial order relations between the transition firings, and also to be able to avoid generating very bad schedule (schedule corresponding to completely specified scenarios which are always worse than some other examined ones).

Ackowledgements:

Many thanks to Yves Lafont (Laboratoire de Mathématiques discrètes, Marseille) for his fruitful comments.

References

[Be 92] B. Berthomieu, M. Diaz : Modeling and verification of time dependent systems using Time Petri nets, *IEEE Trans. on Software Engineering* 17, 1991, pp. 259-273.

[Br 89] C. Brown : Relating Petri Nets to Formulas of Linear Logic, *Edinburgh Tech. Report* ECS-LFCS-89-87, June, 1989.

[DE 97] J. Desel, T. Freytag, A. Oberweis, T. Zimmer : A partial-order-based simulation and validation approach for high-level Petri nets, *Proceedings of IMACS'97* 1997 p.361-366.

[EW 90] U. Engberg, G. Winskel : Petri nets as models of Linear Logic, *CAAP'90*, A. Arnold Ed., LNCS 431, Springer-Verlag, 1990, pp. 147-161.

[Ge 92] V. Gehlot : A proof theoretic approach to semantics of concurrency, PhD thesis, University of Pennsylvania, 1992.

[Gi 87] J.Y. Girard : Linear Logic, *Theoretical Computer Science*, n°50, 1987.

[Gi 95] J.Y. Girard : Linear Logic: A Survey, *Cahiers du centre de logique*, vol. 8, 1995, pp. 193-255.

[Gi 97a] F. Girault, B. Pradin-Chézalviel, R. Valette : A logic for Petri nets RAIRO-APII-JESA (Ed. Hermès, France), Vol 31, N 3, p.525-542,1997.

[Gi 97b] F. Girault : Formalisation en logique linéaire du fonctionnement des réseaux de Petri, Thèse de Doctorat, Université Paul Sabatier, Toulouse, 1997.

[Gu 89] C. Gunter, V. Gehlot : Nets as tensor theories, *10^{th} International Conference on Application and Theory of Petri nets*, Bonn, Germany,1989.

[Ku 97] L.A. Künzle : Raisonnement temporel basé sur les réseaux de Petri pour des systèmes manipulant des ressources, Thèse de Doctorat, Université Paul Sabatier,Toulouse, 1997.

[La 88] Y. Lafont : The Linear abstract machine, *Theoretical Computer Science*, n°59, 1988, pp. 157-180.

[Ma 91] M. Martí-Oliet and J. Meseguer : From Petri Nets to Linear Logic Through Categories: A Survey, *International Journal of Foundation of Computer Science*, vol. 2, n°2, 1991, pp. 297-399.

[Me 83] M. Menasche, B. Berthomieu : Time Petri nets for analysing and verifying time dependent protocols, Third international workshop on protocol specification, testing and verification, Zürich, June 1983

[Me 85] M. Menasche : PAREDE: an automated tool for the analysis of time Petri nets International workshop on timed Petri nets Torino July 1985, p. 162-169

[Me 76] P. Merlin, D.J. Farber : Recoverability of communication protocols; implementation of a theoretical study, IEEE Trans. on communications, Vol COM-24, No 9, Sept. 1976, p.1036-1043

[Pr 99] B. Pradin-Chézalviel, L.A. Künzle , F. Girault, R. Valette : Evaluation temporelle de scénario de réseaux de Petri incluant du parallélisme, *in* Congrès Modélisation des Systèmes Réactifs (MSR'99), Cachan France, 24-26 mars 1999, Ed. Hermès p.131-140.

[Ra 80] C.V. Ramamoorthy, G.S. Ho : Performance evaluation of asynchronous concurrent systems using Petri nets, IEEE trans. on Soft. Eng., Vol SE-6, n 5, p.440-449, Sept. 1990.

[Ra 73] C. Ramchandani : Analysis of asynchronous concurrent systems by Petri nets, PhD Thesis, MIT, 1973. Also in Project MAC, TR-120 MIT Cambridge 1974.

[Si 77] J. Sifakis : Use of timed Petri nets for performance evaluation, Third Int. Symp. Measuring, Modeling and Evaluating Comput. Syst., North Holland, Beilner and Gelenbe Eds. 1977, p.75-95.

[Si 90] M. Silva, R. Valette : Petri nets and Flexible Manufacturing, Advances in Petri nets 1989, Lecture Notes in Computer Science 424, Springer Verlag (1990), p.374-417.

ANNEX

RULES OF THE LINEAR INTUITIONIST LOGIC ILL USED IN THIS PAPER

F, G and H are formulas (not necessarily atomic ones)
Γ and Δ are blocks (with possibly the meta connective ,)

Identity group

$$\frac{}{F \vdash F}\ \text{id}$$

$$\frac{\Gamma \vdash F \quad \Delta, F, \vdash H}{\Gamma, \Delta \vdash H}\ \text{cut}$$

Structural group

$$\frac{\Gamma, F, G, \Delta \vdash H}{\Gamma, G, F, \Delta \vdash H}\ \text{exchange}$$

Logical group

$$\frac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H}\ \otimes\text{L}$$

$$\frac{\Gamma \vdash F \quad \Delta \vdash G}{\Gamma, \Delta \vdash F \otimes G}\ \otimes\text{R}$$

$$\frac{\Gamma \vdash F \quad \Delta, G \vdash H}{\Gamma, \Delta, F \multimap G \vdash H}\ \multimap\text{L}$$