15th Triennial World Congress, Barcelona, Spain

RELATIONSHIPS BETWEEN PETRI NETS AND CONSTRAINT GRAPHS: APPLICATION TO MANUFACTURING

Catherine Mancel [*] **Pierre Lopez** [**] **Nicolas Rivière** [**]
Robert Valette [**]

[*] *IXI, 76 rue de la Colombette, F-31000 Toulouse and LAAS-CNRS, F-31077 Toulouse Cedex 4*
[**] *LAAS-CNRS, F-31077 Toulouse Cedex 4*

Abstract: The purpose of the paper is to compare timed and time Petri nets with constraint satisfaction problems and activity-on-arc graphs in the context of manufacturing. It is shown that constraints are not defined in the same way but that timed and time Petri nets could be translated into a set of activity-on-arc graphs. This translation is only straightforward for p-timed and p-time Petri nets.

Keywords: Constraint Satisfaction Problems, time Petri nets, timed Petri nets, Manufacturing

1. INTRODUCTION

Petri nets are commonly used to define resource allocation mechanisms for flexible manufacturing systems. They can be enriched by temporal inscriptions in different ways (t-timed, t-time, p-timed and p-time Petri nets) allowing thus the specification of complex constraints involving time and resources. Constraint Satisfaction Problems (CSP) and activity-on-arc graphs (AOA graphs) are also commonly used for the analysis of temporal constraints encountered in scheduling issues for manufacturing. The purpose of the paper is to compare the two approaches and to show that it is possible, from a Petri net model, to derive scenarios (partial orders on a set of transition firings), each one corresponding to an AOA graph when time constraints are taken into account.

2. ORDINARY PETRI NET AND CSP

2.1 *Some definitions*

Let us first recall what are CSPs (Tsang, 1993) and Petri nets (Murata, 1989).

Definition 1. A constraint satisfaction problem (CSP) is defined as a triple (X,D,C). $X = \{x_1, \ldots, x_n\}$ is a set of n variables. $D = \{d_1, \ldots, d_n\}$ is the set of the domains of the variables. $C = \{c_1, \ldots, c_e\}$ is a set of e constraints, each constraint c_i being defined by: the set of variables $X(c_i) \subset X$ involved in constraint c_i and a relation $R(c_i)$ on the variables of $X(c_i)$.

Definition 2. A Petri net $\mathcal{P}$ is defined as a 4-tuple (P,T,Pre,$Post$). P is a finite set of places p_i. T is a finite set of transitions t_j. Pre is an application $P \times T \rightarrow \mathbb{N}$ ($\mathbb{N}$ is the set of natural numbers), it defines the input arcs of the transitions. $Post$ is an application $P \times T \rightarrow \mathbb{N}$, it defines the output arcs of the transitions.

Definition 3. A marking M is a distribution of tokens in the places, it is an application $P \rightarrow \mathbb{N}$.

In a Petri net a transition t can only be fired if it is enabled by the current marking M that is if $Pre(.,t) \leq M$. Let ${t_i}^j$ be the j^{th} firing of transition t_i. Let us consider an unordered list of transition firings to be fired sc (a list of event that should occur). Let $\overline{sc}$ be the firing vector corresponding to sc

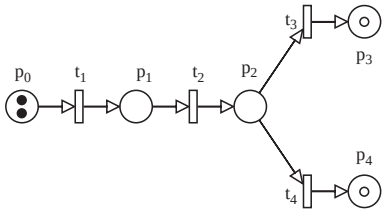

Fig. 1. Petri net fragment

($\overline{sc}(t_i) = k$ if sc contains k firings of t_i). If we consider a Petri net $\mathcal{P}$ between the two markings M and $M + (Post - Pre).\overline{sc}$, it defines precedence relations on the list of transition firings sc. These precedence relations have to satisfy a set of constraints. Is it possible to express this set of constraints under the form of a CSP?

We underline here the fact that if the Petri net represents the resource allocation policies of a shop, any short term scheduling issue can be expressed under the form of the computation of the optimal firing dates of a set of transitions transforming an initial marking (current state of the shop) to a final marking (desired state at the end of the horizon).

2.2 *Constraints on precedence relations defined by Petri nets*

If we want to derive a CSP defining the same constraints as the Petri net with a marking and a list of transitions, we have first to define the variables. Precedence constraints can be characterized by means of Boolean variables $x_{i,j,k,l}$. If $x_{i,j,k,l} = 1$ this means that ${t_i}^j$ must precede ${t_k}^l$. If $x_{i,j,k,l} = 0$ then no precedence relation is imposed and ${t_i}^j$ may occur before or after ${t_k}^l$, unless a precedence relation can be derived by combining two or more existing ones (for example $x_{i,j,m,n} = 1$ and $x_{m,n,k,l} = 1$).

Let us consider for example the Petri net in figure 1 for a marking M such that there are two tokens in place P_0 and for the list of firings:

$$sc = {t_1}^1, {t_1}^2, {t_2}^1, {t_2}^2, {t_3}^1, {t_4}^1$$

A search for possible firing sequences shows that the fact that place p_1 connects transitions t_1 and t_2 entails that $x_{1,1,2,1} = 1$.

Each place of $\mathcal{P}$ defines constraints between its input and output transition firings (for each M and sc). For example, place p_1, in the above example, generates two unary constraints (a constraint involving a unique variable): $x_{1,1,2,1} = 1$ and $x_{1,2,2,2} = 1$.

When a place has more than one input transition or more than one output transition, the generated constraint is no longer unary. In the above example, place p_2 generates the following binary constraint:

$$x_{2,1,3,1} = 1 \text{ and } x_{2,1,4,1} = 0$$
$$\text{or } x_{2,1,3,1} = 0 \text{ and } x_{2,1,4,1} = 1$$

Let us now consider the precedence relations involving the second firing of t_2. If t_3 has already been fired

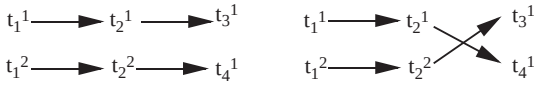

Fig. 2. The two partial orders

(a consequence of $x_{2,1,3,1} = 1$) then it is necessary to fire t_4. Otherwise, it is necessary to fire t_3. This means that the constraint is:

$$\text{if } x_{2,1,3,1} = 1 \text{ then } x_{2,2,4,1} = 1 \text{ else } x_{2,2,3,1} = 1$$

The constraint has a *procedural* form, this means that it is necessary to know which decisions have been made in the past in order to verify the preceding constraints and to write the constraint at hand in a CSP-like form.

2.3 *Partial order among the set of transition firings*

When $x_{i,j,k,l} = 1$ the j^{th} firing of t_i must occur before the l^{th} firing of t_k. If the marking $M + (Post - Pre).\overline{sc}$ is reachable from M in Petri net $\mathcal{P}$, this means that there exists at least one set of variables $x_{i,j,k,l}$ verifying all the constraints. This set defines a partial order among the transitions firings of sc which can be represented by a directed graph. The nodes of the graph are the elements ${t_i}^j$ of sc, there is an arc between ${t_i}^j$ and ${t_k}^l$ if and only if $x_{i,j,k,l} = 1$. For example, if we consider the Petri net in figure 1 (M and sc as defined above) two partial orders are possible. They are represented in figure 2.

Since the expression of the constraints requires the knowledge of the past decisions, the unique way to translate them under a CSP is to list all the possible partial orders. A unique constraint c_1, involving all the variables is then derived. For the above example, the relation $R(c_1)$ is:

$$(x_{1,1,2,1} = x_{2,1,3,1} = x_{1,2,2,2} = x_{2,2,4,1} = 1)$$
$$\text{or } (x_{1,1,2,1} = x_{2,1,4,1} = x_{1,2,2,2} = x_{2,2,3,1} = 1)$$

2.4 *Conclusion*

It is therefore possible to translate the set of constraints generated by a Petri net, an initial marking and a list of transition firings under the form of a CSP. However it is a degenerated form with a unique constraint which is the disjunction of the set of solutions. Is there any mathematical tool to compute this set of solutions (the possible partial orders)? Recent developments about Petri nets and Linear logic offer a solution.

3. USING LINEAR LOGIC TO FIND A PARTIAL ORDER

Typically, in Petri net theory, reachability analysis is based on firing sequences in which transition firings

occur in a total order. In the framework of Linear logic, it is possible to derive partial orders. We will just give here the main points in order to understand the approach. For more detail about Linear logic the reader can refer to (Girard, 1987) and for more detail about Petri nets and Linear logic to (Gehlot, 1992) and (Pradin-Chézalviel *et al.*, 1999).

3.1 *Basic concepts on Linear logic*

Linear logic has been defined in the framework of sequent calculus. A sequent (the left part is a list of hypotheses and the right one a list of conclusions) is proved if and only if it is syntactically correct, that is if it can be proved that each connective can be introduced by using a set of rules. Linear logic can be seen as a restriction of classical logic in order to deal with resources as logical propositions. Atoms corresponding to logical propositions may be counted, produced and consumed exactly like tokens in Petri net places. It is the reason why there is an equivalence between reachability in a Petri net and the provability of some Linear logic sequents.

In this paper we will just use two Linear logic connectives. The connective $\otimes$ represents the simultaneous availability of some resources. The connective $\multimap$ corresponds to the linear implication and represents causality because the atoms on the left side of $\multimap$ have to be consumed in order to produce those on the right part.

3.2 *Translating Petri nets into Linear logic*

A marking M is a monomial in $\otimes$ (for instance $p_0 \otimes p_0 \otimes p_1$ corresponds to a marking for which place p_0 contains two tokens and place p_1 one token). A transition t is a formula of the form $M_1 \multimap M_2$ where M_1 and M_2 are partial markings. The monomial M_1 is another notation for the column of matrix Pre corresponding to t (denoting the input places of t) and M_2 for the corresponding column of $Post$.

The sequent $M, sc \vdash M'$ where sc is an unordered list (separated by commas) of transition formulas, represents the reachability of M' from M by firing the corresponding transitions (if a transition t_i is fired n times, the sc contains n times the formula corresponding to t_i). The condition

$$M' = M + (Post - Pre).\overrightarrow{sc}$$

is a necessary but not sufficient condition. On the contrary, the proof of this sequent is equivalent to that of the reachability. This means that during the proof all the constraints generated by the Petri net and the initial marking are checked.

3.3 *Deriving a partial order*

The first step of the proof consists in removing the connective $\otimes$ within the initial marking. This means that in place of operating on a marking (the token location at a given time point), it is possible to operate with a list of tokens (logic atoms) which are logically independent and thus not necessarily simultaneously present. During the proof, these tokens correspond to precedence relations. If a token is produced by a transition firing ${t_i}^j$ and is consumed by ${t_k}^l$, this means that $x_{i,j,k,l} = 1$. At the end of a proof it is sufficient to collect all the tokens which have appeared, each one corresponds to one precedence relation and the whole set exactly defines the partial order.

The proof can be seen as a rewriting process, each step allows the elimination of one transition formula. At each step, a transition formula ${t_i}^j$ is removed from sc; the tokens consumed by ${t_i}^j$ are removed from M and the produced one added; M' remains unchanged. The sequent is proved when sc is empty and M is the list of the tokens in M'.

If at a given step more than one elimination is possible and if the concerned transitions and tokens are all disjoint, this means that the resulting proof will be the same whatever the order of the eliminations. If transitions or tokens are shared (token conflicts or transition conflicts as defined in (Pradin-Chézalviel *et al.*, 1999)), then one proof tree for each order has to be derived because each one characterizes a different partial order.

3.4 *Example 1*

Let us consider the example in figure 1 again. The initial marking is $p_0 \otimes p_0$ and the final marking $p_3 \otimes p_4$. List sc contains two formulas corresponding to t_1 ($p_0 \multimap p_1$), two ones corresponding to t_2, and one for t_3 and for t_4.

During the proof two tokens are inevitably produced in place p_1. The first is produced by ${t_1}^1$ and consumed by ${t_2}^1$. The second is produced by ${t_1}^2$ and consumed by ${t_2}^2$. This means that $x_{1,1,2,1} = 1$ and $x_{1,2,2,2} = 1$.

When the first token, produced by ${t_2}^1$, appears in p_2 there is a conflict because the elimination of ${t_3}^1$ and of ${t_4}^1$ are both candidates. In consequence two proofs have to be done. In the first case $x_{2,1,3,1} = 1$ is derived and in the second one $x_{2,1,4,1} = 1$. In the first case ${t_4}^1$ is eliminated by consuming the second token in p_2 (${t_3}^1$ in the second case). The elimination rewriting rule is used 6 times because sc initially contains 6 firings. Finally (derivation details are skipped), the two partial orders in figure 2 are obtained.

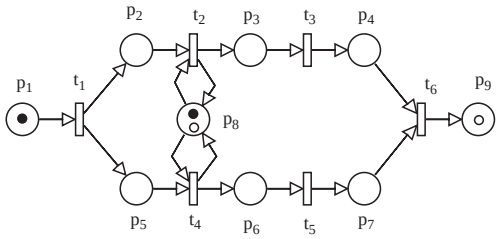

Fig. 3. An example

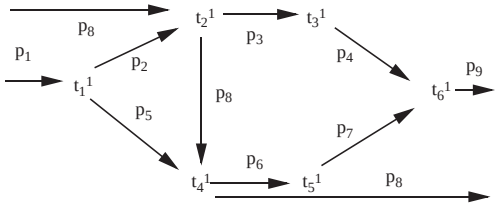

Fig. 4. A first partial order

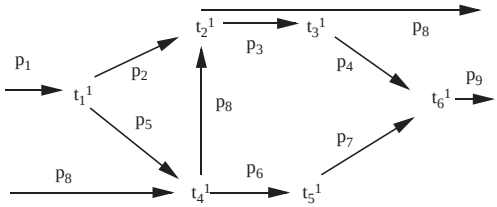

Fig. 5. A second partial order

3.5 *Example 2*

Let us now consider an elementary case of a manufacturing system with two part routes to be executed and a shared resource (see figure 3). Initial and final events are $t_1{}^1$ and $t_6{}^1$. The first part route consists in two operations ($t_2{}^1$ and $t_3{}^1$) and the second one in ($t_4{}^1$ and $t_5{}^1$). Operations t_2 and t_4 share the same resource. The initial marking is $p_1 \otimes p_8$ and the final one is $p_8 \otimes p_9$.

There is a conflict after the elimination of $t_1{}^1$ when the list of tokens p_2, p_5, p_8 is reached because both $t_2{}^1$ and $t_4{}^1$ are candidates and both consumes p_8. The two partial orders in figures 4 and 5 are then derived. It can be noted that label p_8 appears three times. It is because there is initially one token in p_8 which is consumed by $t_2{}^1$ (in the first partial order) and produced again. It is then consumed and produced again by $t_4{}^1$.

4. INTRODUCING TIME CONSTRAINTS

4.1 *Petri nets and manufacturing*

In the context of scheduling in manufacturing, Petri nets are typically used when cyclic policies are required because the constraint corresponding to the cyclic behavior is easily represented by a Petri net. In the approach presented in this paper, Petri nets are used to represent the structure of the manufacturing system *i.e.* the part routes and how the machines are shared among the operations. Some deadlock avoidance policies may also be represented as well as some intermediate storage limitation.

The current state of the manufacturing system is the current marking and scheduling a set of operations is equivalent to find the best partial order among the transition firings corresponding to the operations to be done and their optimal firing date. The Petri net model has to be complemented by time constraints (operation durations, due dates etc..) and the set of variables $x_{i,j,k,l}$ by the set of the firing dates of the transitions ($x^f_{i,j}$ is the date of the j^{th} firing of transition t_i; f stands for firing). It has been shown above that a Petri net could be translated into a CSP for a list of transition firings. In the sequel this comparison includes timing considerations. In this paper only four propositions will be explored: p-timed and t-timed Petri nets, and p-time and t-time Petri nets.

4.2 *Temporal constraint satisfaction problem*

Definition 4. A temporal constraint satisfaction problem (TCSP) is a particular class of CSP where the set X of variables denotes a set of time entities (time points, time intervals, durations) and constraints (C) represent the possible logical or numerical temporal relations between variables. All TCSP constraints are binary (Schwalb and Vila, 1998).

Definition 5. An activity-on-arc graph (AOA graph) (Elmaghraby, 1977) is a graph such that the nodes represent the variables X and the edges the constraints. With each edge $i \rightarrow j$ a binary constraint c_{ij} is associated. An AOA graph can be associated with a TCSP. Nodes represent events or steps related to start or end of a set of activities. The length of an arc represents the duration of the activity attached to this arc. It is a time constraint between two events.

4.3 *p-timed Petri net*

Definition 6. A p-timed Petri net is a Petri net $\mathcal{P}$ with a function associating a duration d_i with each place p_i. Each token has to stay at least d_i when it arrives in p_i before being consumed by a transition firing.

It has been shown in the preceding section that from $\mathcal{P}$, M and sc a set of partial orders verifying the logical constraints could be derived. Each partial order can be represented by a graph whose nodes are the transition firings and whose arcs correspond to precedence relations between two firings. The arcs are labeled by places (see figures 4 and 5). If we replace each transition firing $t_i{}^j$ by the variable $x^f_{i,j}$ and each arc label p_k by its duration d_k then we have translated the set of constraints generated by the p-timed Petri net into a set of AOA graphs.

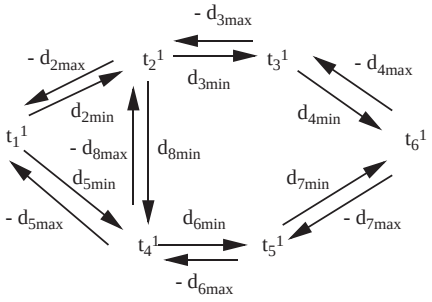

Fig. 6. The AOA graph for a p-time Petri net

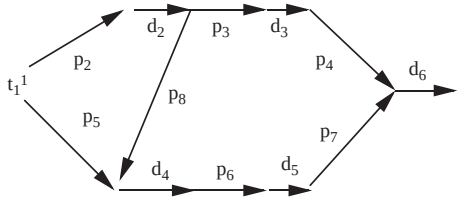

Fig. 7. The AOA graph for a t-timed Petri net

Since the structure of these graphs is that of a partial order, they are acyclic and it is always possible to find a solution. By means of p-timed Petri net it is indeed only possible to represent operation durations, due dates are not taken into account.

4.4 *p-time Petri net*

Definition 7. A p-time Petri net is a Petri net $\mathcal{P}$ with a function associating a minimal duration d_{imin} and a maximal duration d_{imax} with each place p_i. Each token has to stay at least d_{imin} in p_i and has to be consumed by a transition firing before d_{imax}.

With this model two constraints are associated with each precedence relation. One is for the minimal duration. The other one, in the reverse way, specifies the maximal duration. Figure 6 is the AOA graph corresponding to the first partial order of the Petri net in figure 3 (for simplicity we have deleted the arcs corresponding to the initial and final tokens). It is clearly possible to have positive circuits which means that the set of constraints is inconsistent. P-time Petri nets are indeed capable of taking into account due dates and are more general than p-timed Petri nets. In addition, the search of a solution may be done by means of linear programs as in (Bonhomme, 2001).

4.5 *t-timed Petri net*

Definition 8. A t-timed Petri net is a Petri net $\mathcal{P}$ with a function associating a duration d_i with each transition t_i. When firing t_i a time d_i elapses between consuming the input tokens and producing the output ones.

Before relabeling the nodes and the arcs, we have to transform the partial order graph because now transitions are no longer events: activities are attached to them. Each node is thus transformed into an arc, labeled by the corresponding transition firing. The nodes are now beginnings and ends of firings (the actual variables are firing beginnings). The result of such a transformation is represented in figure 7 for the fragment between $t_1{}^1$ and $t_6{}^1$. Then each place is replaced by 0 (it remains a simple precedence relation) and each firing $t_i{}^j$ by d_i. The obtained graph is acyclic and therefore there is always a solution to the set of constraints.

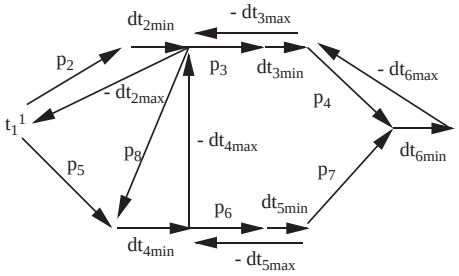

Fig. 8. The AOA graph for a t-time Petri net

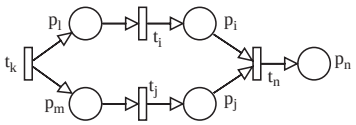

Fig. 9. Petri net with two parallel branches

4.6 *t-time Petri net*

Definition 9. A t-time Petri net is a Petri net $\mathcal{P}$ with a function associating a minimal duration d_{imin} and a maximal duration d_{imax} with each transition t_i. Transition t_i has to remain enabled at least d_{imin} and at most d_{imax} before being fired.

As for p-time Petri nets, arcs with negative length d_{imax} have to be introduced to bound the enabling duration of t_i. When t_i has more than one input place, the enabling event is not known: it is the end of the firing of the transition which produces the last token. Figure 8 represents the AOA graph (between $t_1{}^1$ and $t_6{}^1$ and for the first partial order) when t_6 is enabled by the firing of t_3.

Proposition 10. If $M + (Pre - Post).\overline{sc}$ is reachable in $\mathcal{P}$ then the set of constraints associated with any t-time Petri net built on $\mathcal{P}$ is always consistent for the initial marking M and the firing list sc.

This is a direct consequence of the fact that there are no positive circuit in the corresponding AOA graph. When translating a t-time Petri net into an AOA graph, the circuits generated by transitions having a unique input place are not positive (the length is $d_{imin} - d_{imax}$). More interesting circuits are generated if the Petri net contains two parallel

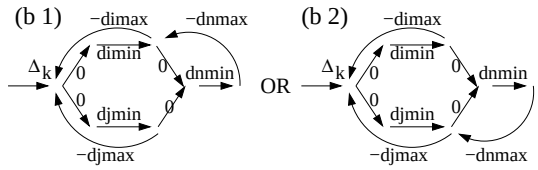

Fig. 10. Two possible AOA graphs

branches as shown in figures 9 and 10. In the graph fragment (b 1) of figure 10 there exists the circuit $(d_{jmin}, d_{nmin}, -d_{nmax}, -d_{imax})$. The negative arc attached to $-d_{nmax}$ connects the node corresponding to the end of firing of t_n to that of t_i. It is only correct if t_i has produced a token after t_j. Let θ be the production date of the token in p_i and θ' be the production date of the token in p_j. We have the relation: $\theta' \leq \theta$.

Let Δ_k be the date of the end of firing of t_k, we know that $\theta \in [\Delta_k + d_{imin}; \Delta_k + d_{imax}]$ and $\theta' \in [\Delta_k + d_{jmin}; \Delta_k + d_{jmax}]$. It can then be derived that $d_{jmin} \leq d_{imax}$.

The length of the interesting circuit in AOA graph (b 1) is $0 + d_{jmin} + 0 + d_{nmin} - d_{nmax} - d_{imax}$ and since $d_{nmin} \leq d_{nmax}$ and $d_{jmin} \leq d_{imax}$, then the length is negative or equal to zero.

A symmetric proof can be done for graph (b 2) in figure 10.

5. CONCLUSION

By showing how an ordinary Petri net could be translated into a CSP, this paper has pointed out that the constraints were specified in a procedural way by a Petri net and in a declarative one by a CSP. Linear logic allows computing all the partial orders among the transition firings satisfying the logical constraints generated by a Petri net for an initial marking and a list of firings.

The translation of Petri nets with time into AOA graphs has pointed out that it is more natural to associate the durations with places than with transitions because precedence constraints are generated by places. In addition, if an AOA graph can be associated with each partial order for p-time Petri nets, it is not the case for t-time Petri nets. Finally, it seems difficult to deal with due dates using t-time Petri nets because the inconsistencies which may be introduced will not be easily pointed out.

A scheduling approach may be derived from this study. If the logical constraints defined by the ordinary Petri net (deadlock avoidance and storage regulation) are sufficiently strong, or if a heuristic is available to get some good partial orders, then time can be introduced by means of p-time Petri nets and an optimal (for the corresponding partial order) solution can be derived by linear programming.

6. REFERENCES

Bonhomme, P. (2001). Réseaux de Petri p-temporels : contributions à la commande robuste. Phd thesis. LAMII/CESALP/ESIA, Annecy, Université de Savoie, July 12.

Elmaghraby, S.E. (1977). *Activity networks: project planning and control by network models*. John Wiley & sons. New York.

Gehlot, V. (1992). A proof theoretic approach to semantics of concurrency. Phd thesis. University of Pennsylvania.

Girard, J. Y. (1987). Linear logic. *Theoretical Computer Science* **50**, 1–102.

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

Pradin-Chézalviel, B., R. Valette and L.A. Künzle (1999). Scenario duration characterization of t-time petri nets using linear logic. In: *IEEE PNPM'99*. Zaragoza, Spain, September 6-10. pp. 208–217.

Schwalb, E. and L. Vila (1998). Temporal constraints: a survey. *Constraints: an International Journal* **2**, 129–149.

Tsang, E.P.K. (1993). *Foundations of constraint satisfaction*. Academic Press Ltd. London.