Reachability search in timed Petri nets using constraint programming

Olfa BELKAHLA DRISS\(^1\)  Pascal YIM\(^2\)
\(^1\)Stratégie d’Optimisation de l’Ingénierie des Informations et de la connaissance, ISG de Tunis, Cité Bouchoucha, Bardo, 2000 Tunisia.
Olfa.Belkahla@isg.rnu.tn
Pascal.Yim@ec-lille.fr

Ouajdi KORBAA\(^2\)  Khaled GHEDIRA\(^1\)
\(^2\)Laboratoire d’Automatique, Génie Informatique & Signal (LAGIS, anciennement LAIL), Ecole Centrale de Lille, BP48, 59651 Villeneuve d’Ascq cedex, France.
Ouajdi.Korbaa@ec-lille.fr  Khaled.Ghedira@isg.rnu.tn

Abstract – This paper presents a logical abstraction of the reachability graph of a timed Petri net using constraint programming. We apply it to the scheduling of transient inter-production states for cyclic productions in Flexible Manufacturing System.
So, we propose to adapt the approach of Benasser and Yim (1999) based on the search of the accessibility by means of constraints using concepts of partial marking and partial step which allow a logical abstraction of the reachability graph of a Petri net. Having the timed Petri net (where a duration is associated to each transition), we propagate time to the obtained steps. In fact, we associate, to each marking extracted from a step, a timestamp vector: each timestamp corresponds to the date of the last token produced in a place at a step. Then, under temporal constraints, we solve scheduling problems, using constraint programming.

Keywords: timed Petri nets, reachability graph, scheduling, constraint programming.

1 Introduction

In this paper, we present a logical abstraction of the reachability graph of a timed Petri net using constraint programming by adapting the approach of Benasser and Yim [4] based on the search of the accessibility by means of constraints using new concepts of partial marking and partial step. These two concepts allow a logical abstraction of the reachability graph of a Petri net [4] [10]. This abstraction is reduced to a sequence. The search of firing sequence between two markings consists then in a constraint solving problem. Since, constraints are linear, the problem can be efficiently solved.

Our interest is about cyclic scheduling problem for Flexible Manufacturing Systems (FMS), we apply it to the scheduling of transient inter-production states for cyclic productions in FMS.

In fact, the Petri net paradigm [11] is suited to the modelling of flexible manufacturing systems. The schedule comes down to the reachability problem (to the search of all transition sequences which lead from an initial marking of the Petri net to a final one). Korbbaa and Gentina, 1997 [9] and Benasser et al., 1996 [2] have proposed techniques of scheduling using Petri nets.

The contribution of Petri net modelling used for the computation of the scheduling of FMS is demonstrated by Benasser et al., 1999 [3].

In order to solve scheduling problem, it is necessary to introduce time to the Petri net. The Timed Petri Net of Ramchandani (1974) [12] allows to model easily scheduling problem. The inconvenient of this model is that the evolution of the current state is controlled by an external clock. The TB nets, defined in [6], associates a timestamp to each token. This timestamp progresses when the transitions fire but this approach makes a distinction of the tokens. Benasser and Yim (2001) [5] introduced a new model called Autonomous Timed Petri Net where a timestamp is associated to each place of the Petri net. This timestamp is the date of the last token produced in the place. The behaviour is then autonomous since the date is not controlled by an external clock but progresses when the transitions fire. They applied it to the scheduling of a flexible manufacturing system. But this approach introduces a loss of information.

In fact, a step [7] [8] is a non empty multi-set over the set of transitions. A step s is enabled at a marking m if the elements of s are concurrently enabled, i.e. the transitions of s can get their own tokens without having to share them with other transitions of s. Steps represent a set of equivalent transition sequences which represent the same process in the sense that the same conflict-resolutions are applied. So a few step sequences cover the whole set of transition sequences. It is well-suited to express the solutions of a reachability problem.

The number of states of a system can grow more than exponentially, even for a small system. So the enumeration of the full state space is often intractable. In order to reduce the size of the reachability graph, nodes can represent a set of markings rather than single markings. Based on constraint programming, nodes and edges are respectively annotated by partial markings and partial steps. Partial markings and partial steps can be seen as markings and steps containing variables not totally bounded. These
variables are linked by a constraint. Partial markings and partial steps represent all objects obtained by a valuation of the variables such that the constraint holds.

The aim is to deduce all step sequences from partial step sequences. A partial step is enabled at a partial marking if there exists a valuation satisfying a constraint which indicates that a marking has enough tokens to enable a step. When a partial step sequence is enabled at the initial partial marking and leads to another partial marking, some firable step sequences and some reachable markings can be extracted from the valuation of the partial step sequence and the partial marking.

The reachability algorithm tests successively the satisfiability of the constraint (called finality constraint). As soon as this constraint is satisfiable, step sequences which lead to the final marking can be extracted from the partial marking sequence.

So, in this study, having the timed Petri net (where a duration is associated to each transition), we propose to propagate time to the obtained steps. In fact, we associate, to each marking extracted from a step, a timestamp vector: each timestamp corresponds to the date of the last token produced in a place at a step. Then, under temporal constraints, we solve scheduling problems using constraint programming.

This paper is composed as following: firstly, we will present basic definitions about Petri net, partial marking, partial step, reachability problem and an illustrative example. Then, we present our model followed by an illustrative example and experimentation. Finally, a conclusion is given.

2 Basic definitions

2.1 Petri nets

A net [11] is a triple (P, T, W) where:

- P and T are finite and disjoint sets of respectively places and transitions.
- The weight function W is a mapping of \((P \times T) \cup (T \times P)\) into the set of non negative integers.

A marking \(m: P \to \mathbb{N}\) is a mapping of \(P\) into \(\mathbb{N}\). If \(p \in P\), \(m(p)\) is called the number of tokens of the place \(p\).

A Petri net is a quadruple \((P, T, W, m_0)\) where \((P, T, W)\) is a net and \(m_0\) is a marking called the initial marking.

Let \(PN = (P, T, W, m_0)\) be a Petri net where \(P = \{p_1, p_2, \ldots, p_n\}\) and \(T = \{T_1, T_2, \ldots, T_m\}\). A marking \(m\) will be represented by the column vector \((m(p_1), m(p_2), \ldots, m(p_n))^t\).

2.2 Steps

A step is a non empty multi-set over the set of transitions \(T\) [7] [8]. A step \(s\) is enabled at a marking \(m\) if the elements of \(s\) are concurrently enabled, i.e. the transitions of \(s\) can get their own tokens without having to share them with other transitions of \(s\). Steps represent a set of equivalent transition sequences which represent the same process in the sense that the same conflict-resolutions are applied. So a few step sequences cover the whole set of transition sequences.

A step \(s\) is enabled at a marking \(m\) (denoted by \(m[s>\)) if:

\[
\forall p \in P, \sum_{i \in s} W(p, t)s(t) \leq m(p)
\]

If a step \(s\) is enabled at a marking \(m\), then it may occur or fire, leading to another marking \(m'\) (denoted by \(m[s>m']\)) defined by:

\[
\forall p \in P, m'(p) = m(p) - \sum_{i \in s} W(p, t)s(t) + \sum_{i \not\in s} W(t, p)s(t)
\]

2.3 Partial markings and partial steps

The number of states of a system can grow more than exponentially, even for a small system. So the enumeration of the full state space is often intractable. In order to reduce the size of the reachability graph, nodes can represent a set of markings rather than single markings. Based on constraint programming, nodes and edges are respectively annotated by partial markings and partial steps. Partial markings and partial steps can be seen as markings and steps containing variables not totally bounded. These variables are linked by a constraint. Partial markings and partial steps represent all objects obtained by a valuation of the variables such that the constraint holds.

2.3.1 Partial markings

A partial marking is a triple \((M, C, V)\) where:

- \(M\) is an application which associates to each token an expression as \(k_0 + k_1 x_1 + \ldots + k_n x_n\) where the \(k_i\) are integer constants and \(x_i\) are integer variables in \(V\).
- \(C\) is a linear constraint containing the variables included in the \(M\) expression.
- \(V\) is the set of variables involved in \(M\) and \(C\).

Let \(0\) be an assignment of the variables such that it satisfies \(C\) and that:
∀p ∈ P, (PMθ)(p) = (Mθ)(p) = (M(p))θ

This marking is called instance of PM under the valuation θ. A partial marking PM = (M, C, V) represents the set of its instances. This set is denoted by PM. The initial partial marking of a Petri net PN = (P, T, W, m₀) is the partial marking defined by PM₀ = (M₀, Ø, Ø) where ∀p ∈ P, M₀(p) = m₀(p). We have PM₀ = {m₀}.

2.3.2 Partial steps

A partial step is a triple (S, C, V) where:

- S is a mapping of the set of transitions T into variable symbols.
- C is a constraint containing the variables included in the S expression.
- V is the set of variables involved in S and C.

A partial step PS represents the set of steps (called instances of PS) obtained by any valuation θ which satisfies the constraint C such that:

∀t ∈ T, (PSθ)(t) = (Sθ)(t) = (S(t))θ

This step is called instance of PS under the valuation θ and the set of instances of PS is denoted by PS.

The aim is to deduce all step sequences from partial step sequences. A partial step PS = (S, C, V) is enabled at a partial marking PM₁ = (M₁, C₁, V₁) (denoted by PM₁[PS]) if there exists a valuation (satisfying a constraint which indicates that a marking has enough tokens to enable a step) such that it satisfies the following constraint, called the firing constraint:

∀p ∈ P, M₁(p) ≥ W(p) ⊆ M₀(p) \cup \bigcup_{t ∈ T} \{PS(t) ≥ 0\} \cup C₁ \cup C

Then the firing yields to another partial marking PM₂ = (M₂, C₂, V₂) defined by:

- ∀p ∈ P, M₂(p) = m₀(p) - \bigcup_{t ∈ T} W(p, t)PS(t) + \bigcup_{t ∈ T} W(t, p)PS(t)
- C₂ = \bigcup_{p ∈ P} \{PS(t) ≥ 0\} \cup C₁ \cup C
- V₂ = V₁ \cup V

This is denoted by PM₁[PS] > PM₂.

2.4 Reachability problem

When a partial step sequence is enabled at the initial partial marking and leads to another partial marking, some firable step sequences and some reachable markings can be extracted from the valuation of the partial step sequence and the partial marking.

Let PN = (P, T, W, m₀) be a Petri net and T = {T₁, T₂, ..., Tᵣ} be the set of transitions. Let PM = (M, C, V) be a partial marking. For i ∈ [1, +∞], let PSᵢ = (Sᵢ, Cᵢ, Vᵢ) be partial steps. Let PSS* = PS₁⁺PS₂⁺PS₃⁺... be the infinite partial step sequence defined by:

- ∀i ∈ [1, +∞], Vᵢ = {V₁, V₂, ..., Vᵢ}
- ∀i ∈ [1, +∞], Cᵢ = Ø
- \left( \bigcup_{i ∈ [1, +∞]} Vᵢ \right) \cap V = Ø

Let PM₀⁺ be the initial partial marking and PMᵢ⁺ (i ∈ [1, +∞]) be the partial markings defined by PM₀⁺[PS₁⁺PS₂⁺PS₃⁺...] > PMᵢ⁺. The sequence PM₀⁺PS₁⁺PM₁⁺PS₂⁺PM₂⁺...⁺PMᵢ⁺⁺... is called a partial marking sequence of PN. All reachable markings and all firable step sequences are represented by the partial marking sequence.

Let PN = (P, T, W, m₀) be a Petri net and m be a marking of PN. Let PMᵢ⁺ = (Mᵢ, Cᵢ, Vᵢ) (i ∈ IN) be partial markings and let PSᵢ⁺ = (Sᵢ, Cᵢ, Vᵢ) (i ∈ [1, +∞]) be the partial steps such that PMᵢ⁺PS₁⁺PM₁⁺PS₂⁺PM₂⁺...⁺PMᵢ⁺⁺... is a partial marking sequence of PN. The marking m is in the reachability set if and only if one the constraint (called finality constraint):

∀p ∈ P, Mᵢ⁺(p) = mᵢ⁺(p)

Then, for any valuation θ such that Cᵢ \bigcup_{p ∈ P} \{Mᵢ⁺(p) = mᵢ⁺(p)\} holds, m₀⁺[PS₁⁺PS₂⁺...⁺PSᵢ⁺⁺0] > m.

So the reachability algorithm tests successively the satisfiability of the finality constraint. As soon as this constraint is satisfiable, step sequences which lead to the final marking can be extracted from the partial marking sequence.

2.5 Illustrative example
Let PN be the Petri net depicted in figure 1. Let PM₀ be the initial partial marking and let PM₁ = (M₁, C₁, V₁) the partial marking defined by:

- \( M₁(P₁) = 3 - N₁ - N₂ \)
- \( M₁(P₂) = N₁ - N₃ \)
- \( M₁(P₃) = 1 + N₂ - N₃ \)
- \( M₁(P₄) = N₃ \)
- \( C₁ = \{N₁ \geq 0, N₂ \geq 0, N₁ + N₂ \leq 3, N₃ \leq 0, N₃ \leq 1, N₁ \geq 1\} \)
- \( V₁ = \{N₁, N₂, N₃\} \)

Thus, the set of instances of PM₁ is the set \( PM₁ = \{m₁, m₂, m₃, m₄, m₅, m₆\} \) where:

- \( m₁ = M₁[P₁] \)
- \( m₂ = M₁[P₂] \)
- \( m₃ = M₁[P₃] \)
- \( m₄ = M₁[P₄] \)
- \( m₅ = M₁[N₃] \)
- \( m₆ = M₁[N₄] \)

Suppose that we want to reach the final marking \( m₇ = (0, 0, 2, 1, 1, 0) \). The partial marking sequence starts with the initial partial marking \( PM₀ \). The occurrence of the first partial step \( PS₁ \) yields to partial marking \( PM₁ \). The finality constraint \( C₁ \cup \{3-N₁-N₂=0, N₁-N₃=0, N₂-N₃=0, N₃=2\} \) is not satisfiable. So the partial marking \( PM₁ \) is produced by the partial step \( PS₁ \). The finality constraint \( C₁ \cup \{3-N₁-N₂=0, N₁-N₃=0, N₂-N₃=0, N₃=2\} \) is not satisfiable.

The algorithm is able to find all step sequences which lead to a given reachable marking.

In the following section, we introduce our model.

### 3 Our model

We present a logical abstraction of the reachability graph of a timed Petri net using constraint programming. We adapt the approach of Benasser and Yim [4] based on the search of the accessibility by means of constraints using concepts of partial marking and partial step which allow a logical abstraction of the reachability graph of a Petri net and this by introduction of time concept.

So, in this study, having the timed Petri net (where a duration is associated to each transition), we propose to propagate time to the obtained steps. In fact, we associate, to each marking extracted from a step, a timestamp vector: each timestamp corresponds to the date of the last token produced in a place at a step.

The behaviour is then autonomous since the date is not controlled by an external clock but progresses when the transitions fire.

We associate, to each marking extracted from a step, a timestamp vector \( dp_i \) where each timestamp \( dp_i,j \) is associated to each transition, we propose to propagate time to the obtained steps. In fact, we associate, to each marking extracted from a step, a timestamp vector: each timestamp corresponds to the date of the last token produced in a place at a step.

The following diagram shows the Petri net PN, the initial marking PM₀, and the reachable markings PM₁ and PM₂.

![Figure 1. A Petri net](image-url)
\((M_0,d_0) [S_1> (M_1,d_1)]\) where:
- \(d_0\) denotes an initial timestamp vector before firing
- \(d_1\) denotes a timestamp vector at step \(S_1\) after firing the transitions of \(S_1\).

So,
- Tokens created by the firing of a transition \(t\) become available after \(duration\) units of time after firing of \(t\).
- For all transitions, the timestamp of a place \(P_j\) can not decrease from a step to the next step: the date of the last token produced in a place \(P_j\) at a step \(S_i\) is greater than the date of the last token produced in a place \(P_j\) at the previous step \(S_{i-1}\):
  \[ d_{i,j} \geq d_{i-1,j} \quad \forall i,j \]
- For a transition \(t\) fired at a step \(S_i\), the date of the last token produced in all output places \(P_j\) at a step \(S_i\) is greater than the date of the last token produced in input places \(P_k\) at the previous step \(S_{i-1}\) plus duration of \(t\):
  \[ d_{i,j} \geq d_{i-1,k} + d_{k,j} \quad \forall i \in \text{step}, j \in t_n^*, n / S(n) \neq 0, k \in \text{°}t_n \]

The example that we propose here is based on the Petri net depicted in figure 2.

\[ \text{Figure 2: A Petri net} \]

Let \(M_0 = (2 \ 1 \ 1 \ 0)^t\) be the initial marking and \(M_f = (0 \ 0 \ 0 \ 2)^t\) be the final marking.

The resolution performed by a reachability search in the Petri net \([4]\) gives the following result: \(M_0 [t_1t_3t_2 > M_1 > M_f]\)

Now, we associate to the initial marking \(M_0\), a timestamp vector \(d_0\). Then, we associate to the marking \(M_1\) obtained by the first step constituted by \(t_1\) and \(t_3\), a timestamp vector \(d_1\). Finally, we associate to the marking \(M_2\) obtained by the second step constituted by \(t_1\), a timestamp vector \(d_2\).

Therefore, we can note \((M_0,d_0) [t_1t_3t_2 > (M_1,d_1)] [t_2 > (M_2,d_2)]\) where:
- \(d_0 = (d_{0,1} \ d_{0,2} \ d_{0,3} \ d_{0,4})^t\)
- \(d_1 = (d_{1,1} \ d_{1,2} \ d_{1,3} \ d_{1,4})^t\)
- \(d_2 = (d_{2,1} \ d_{2,2} \ d_{2,3} \ d_{2,4})^t\)

Each timestamp \(d_{i,j}\) is the date of the last token produced in a place \(P_{i,j}\) at a step \(S_i\).

In order to fire the first step, the following constraints must be satisfied:
- \(d_{1,1} \geq d_{0,1}\)
- \(d_{1,2} \geq d_{0,2}\)
- \(d_{1,3} \geq d_{0,3}\)
- \(d_{1,4} \geq d_{0,4}\)

Then,
- \(d_{1,2} \geq d_{0,1} + 3\)
- \(d_{1,3} \geq d_{0,1} + 4\)
- \(d_{1,4} \geq d_{0,2} + 2\)
- \(d_{1,4} \geq d_{0,3} + 2\)

In order to fire the second step, we must satisfy the following constraints:
- \(d_{2,1} \geq d_{1,1}\)
- \(d_{2,2} \geq d_{1,2}\)
- \(d_{2,3} \geq d_{1,3}\)
- \(d_{2,4} \geq d_{1,4}\)

Then,
- \(d_{2,4} \geq d_{1,2} + 2\)
- \(d_{2,4} \geq d_{1,3} + 2\)

\[ \text{4 Experimental results} \]

Now, we are able to solve the previous example. For this, we suppose that \(d_0 = (0 \ 0 \ 0 \ 0)^t\).

In order to fire the first step, we have the following constraints to satisfy:
- \(d_{1,1} \geq 0\)
- \(d_{1,2} \geq 0\)
- \(d_{1,3} \geq 0\)
- \(d_{1,4} \geq 0\)

Then,
- \(d_{1,2} \geq 3\)
- \(d_{1,3} \geq 4\)
- \(d_{1,4} \geq 2\)

We have:
- \(d_{1,1} = 0\)
- \(d_{1,2} = 3\)
- \(d_{1,3} = 4\)
- \(d_{1,4} = 2\)

In order to fire the second step, we have to satisfy the following constraints:
- \(d_{2,1} \geq 0\)
- \(d_{2,2} \geq 3\)
- \(d_{2,3} \geq 4\)
- \(d_{2,4} \geq 2\)

Then,
- \(d_{2,4} \geq d_{1,2} + 2 = 5\)
- \(d_{2,4} \geq d_{1,3} + 2 = 6\)

so, we have:
- \(d_{2,1} = 0\)
- \(d_{2,2} = 3\)
- \(d_{2,3} = 4\)
- \(d_{2,4} = 6\)

This means that transitions \(t_1\), \(t_2\) and \(t_3\) are fired at time 0 and then transition \(t_3\) is fired at time 4.
After finalizing this resolution method based on timed Petri nets and constraint programming, we think to apply it to cyclic scheduling problems for Flexible Manufacturing Systems and in particular to the scheduling of transient inter-production states. The Transient Inter-Production State is considered as the end of the current production and the beginning of the next one. The goal is to schedule this state and to minimize its duration. In fact, we have already developed a Multi-Agent model for Transient Inter-production Scheduling, called MATIS, for sequencing many cyclic productions while computing the transient states between each production pair [1]. Indeed, the cyclic productions are firstly determined, in the planning phase, to respect the initial demand and secondly sequenced each one in relation to the others by means of the multi-agent model aiming to minimize the global makespan while reducing temporal complexity. The model involves cooperating agents (an agent created to each transient inter-production state).

The determination of a cyclic deterministic command of FMS interests us because cyclic behavior reduces the complexity of the general scheduling problem. It avoids the schedule of all the operations by choosing a set of parts to constitute a cycle and optimize it at best.

So, our model is able to solve cyclic problem by considering the initial marking equal to the final marking.

5 Conclusion

The partial marking and the partial step concepts allow a more efficient accessibility search. The complexity of the search is then transferred to the constraint resolution after all constraints have been posted. This technique allows cutting the branch which does not yield to the final marking.

We present a logical abstraction of the reachability graph of a timed Petri net using constraint programming. So, we adapt the approach of Benasser and Yim based on the search of the accessibility by means of constraints using concepts of partial marking and partial step which allow a logical abstraction of the reachability graph of a Petri net and this by introduction of time concept.

Our model is able to solve cyclic problem by considering the initial marking equal to the final marking.

We are now adapting this approach to the transient inter-production scheduling problem.

References


