POLYPUBLIE
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Contraction of the ITCPN state space
Title:

Auteurs: Hanifa Boucheneb, & G. Berthelot
Authors:

Date: 2002

Type: Communication de conférence / Conference or Workshop Item

Référence: Boucheneb, H., & Berthelot, G. (avril 2002). Contraction of the ITCPN state space [Communication écrite]. Theory and Practice of Timed Systems (Satellite Event of ETAPS 2002), Grenoble, France. Publié dans Electronic Notes in Theoretical Computer Science, 65(6). https://doi.org/10.1016/s1571-0661%2804%2980466-3
Citation:

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: https://publications.polymtl.ca/4802/
PolyPublie URL:

Version: Version officielle de l'éditeur / Published version
Révisé par les pairs / Refereed

Conditions d'utilisation: CC BY-NC-ND
Terms of Use:

Document publié chez l'éditeur officiel
Document issued by the official publisher

Nom de la conférence: Theory and Practice of Timed Systems (Satellite Event of ETAPS 2002)
Conference Name:

Date et lieu: 2002-04-06 - 2002-04-07, Grenoble, France
Date and Location:

Maison d'édition: Elsevier
Publisher:

URL officiel: https://doi.org/10.1016/s1571-0661%2804%2980466-3
Official URL:

Mention légale:
Legal notice:

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

Electronic Notes in Theoretical Computer Science 65 No. 6 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume65.html 15 pages

Contraction of the *ITCPN* state space

H. Boucheneb [1]

Department of Computer Engineering
École Polytechnique de Montréal
P.O. Box 6079, Station Centre-ville, Montréal, Québec

G. Berthelot [2]

Institut d'Informatique d'entreprise
Conservatoire National des Arts et Métiers
18, Allée J. Rostand, 91025 Evry Cedex, France

Abstract

We show here how to contract the *ITCPN* state space. We distinguish three levels of contraction that translate the *ITCPN* state space into one well timed and coherent timed automaton. We consider here only equivalence based on delays. To achieve more contractions, the equivalence based on delays can be completed with equivalence based on colours as shown in [4].

1 Introduction

Behaviours of almost systems depend not only on order in which events occur but also on times at which events occur. To perform, on the same model, both correctness and performance analysis, non-timed models have been extended with time. Several timed models have been developed such timed automata [1] and various timed Petri nets (*Ramchandani* [13], *Sifakis* [14], *Merlin* [12], *André* [2], v*an der Aalst* [15] and *Diaz* [8]).

Integrating time parameter in models increases their modelling power but it complicates their analysis. Effectively, because of time density, states of timed models are infinite and then their state space are infinite too. Several analysis approaches, based on state space contractions, have been developed for timed automata ([7,9,10]) and for some timed Petri nets ([3,4,5,6,15]). Nevertheless, these approaches are either complex, or their basic models are not enough expressive.

[1] Email: hanifa.boucheneb@polymtl.ca
[2] Email: berthelot@iie.cnam.fr

©2002 Published by Elsevier Science B. V. Open access under CC BY-NC-ND license.

We consider here the van der Aalst's model *(INCPN)* that completes the coloured Petri net by associating time intervals with outgoing arcs. This model allows to describe large and complex real-time since its underlying model is a coloured Petri net. To analyse this model, van der Aalst has proposed, in [15], an approach that generates a state graph which is "sound", but not "complete" i.e: the obtained state graph is not necessarily equivalent to the state space of the model. Moreover, for nets allowing infinite occurrence sequences, it would lead to infinite state graphs. We have proposed, in [4], an approach that generates a reduced state space which have not these drawbacks. This paper is devoted to simplify and improve this approach to generate a more contracted state space. We propose here one bisimulation relation that realizes more contractions and simplifies the building of the contracted state space. The contracted state space is one timed automaton whose summits are the *ITCPN* reachable markings. This translation releases an attractive contraction of the state space and allows taking advantage of methods and tools well developed for timed automata (*Hytech* [9], *KRONOS* [7], *SGM* [10]). Moreover, the obtained timed automaton is well timed and coherent (i.e.: all summits are reachable and it is always possible to leave any non final summit). These characteristics allow proving some none timed properties using its underlying automaton. We consider here only equivalence based on delays. To achieve more contractions, the equivalence based on delays can be completed with equivalence based on colours as shown in [4]. So, the *ITCPN* models allow to generate condensed timed automata.

Firstly, we give, in sections 2 and 3, some definitions related to the *ITCPN* model and its behaviour. We show afterwards how to contract the *ITCPN* state space. We will distinguish three levels of contraction. Sections 4, 5 and 6 are devoted to these levels of contraction. Finally, we end with a simple example.

2 Interval Timed Coloured Petri Nets

We will introduce here only necessary definitions and notations. For further details, we refer to [11] for coloured Petri nets and to [15] for Interval timed coloured Petri nets.

Definition 2.1 [time domain and multi-sets]

- The time domain is the set of all non-negative real numbers, i.e.: $\mathbb{R}^+$.
- Let A be a set. A multi-set over the set A is a function N which associates with each element of set A, an integer number. It is represented by the following formal sum: $\sum_{a \in A} N(a) \bullet a$.
  We denote by A_{MS} the set of all multi-sets over set A.

An Interval Timed Coloured Petri Net (*ITCPN*) is a Coloured Petri Net completed with time intervals attached to output arcs. From the semantic point of view, each created token has a time stamp which can be any value

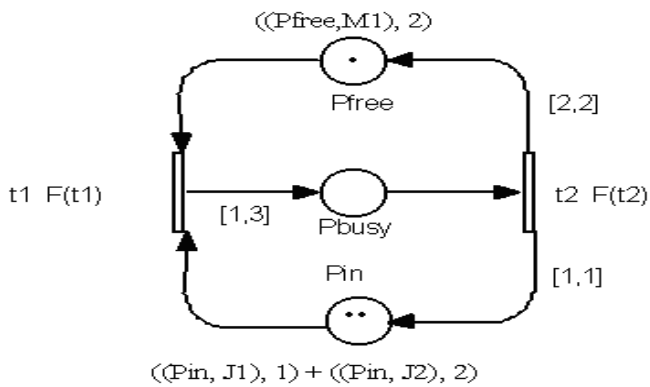

Fig. 1. An *ITCPN* model

inside the interval associated with its creating arc. The time stamp of a token indicates the delay required for the token to become available (i.e.: useful).

Definition 2.2 [an *ITCPN* model] An *ITCPN* is a five tuple $(\triangle, P, T, C, F)$ where

- $\triangle$ is a finite set of types, called colour sets.
- P is a finite set of places.
- T is a finite set of transitions.
- $C \in P \rightarrow \triangle$. $C(p)$ is a finite set which specifies the set of allowed values (or colours) for any token of place p.
- Let CT be the set of all possible coloured tokens, i.e.: $CT = \{(p, c) | p \in P \wedge c \in C(p)\}$ and INT the set of all intervals such that their bounds are rational numbers, i.e.: $INT = \{[y, z] \in Q^+ \times Q^+ \mid y \leq z\}$.
  F is the transition function over the set T: $F(t) \in CT_{MS} \longrightarrow (CT \times INT)_{MS}$
  $F(t)$ specifies which tokens are consumed and produced by firing transition t and also the interval in which their time stamps must be chosen.
  Each transition is supposed to produce a finite set of tokens.

3 *ITCPN* behaviour

We will first explain the behaviour of an *ITCPN*, using an example given in [15] and reported here in Figure 1.

Figure 1 is the graphic representation of an *ITCPN* model, which is composed of three places $p_{in}, p_{busy}, p_{free}$, two transitions t_1, t_2 and three colour sets:
$M = \{M_1, M_2, ..., M_s\}$ attached to the place p_{free},
$J = \{J_1, J_2, .., J_r\}$ attached to the place p_{in}, and
$M \times J$ attached to the place p_{busy}.

It represents a jobshop, where jobs of place p_{in} are executed repeatedly. The jobshop is composed of a number of machines. Each machine is represented by a token, which is either in place p_{free} or in place p_{busy}.

Tokens consumed and produced by firing transitions t_1 and t_2 are specified by functions $F(t_1)$ and $F(t_2)$:

- $\forall j \in J, \forall m \in M$,
  $F(t_1)((p_{in}, j) + (p_{free}, m)) = ((p_{busy}, (m, j)), [1, 3])$.
  Which means that transition t_1 consumes two tokens one from place p_{in} and one from place p_{free} and produces one token for place p_{busy}. When transition t_1 occurs, the time stamp of the created token is one value inside interval $[1, 3]$.
- $F(t_2)(p_{busy}, (m, j)) = ((p_{free}, m), [2, 2]) + ((p_{in}, j), [1, 1])$.
  This means that transition t_2 consumes one token from place p_{busy} and produces two tokens one for place p_{in} and one for place p_{free}. When transition t_2 occurs, the time stamp of the token created in place p_{free} is inside interval $[2, 2]$ while this of the token created in place p_{in} is inside interval $[1, 1]$.

3.1 States of an ITCPN

To characterise the model state, we associate with each token one variable, called delay, which is initialized, at its creation, with its time stamp. Afterwards, the delay decreases synchronously with time until its associated token is consumed. The state can be defined by a multi-set of timed tokens (i.e.: tokens completed with values of their delays).

Definition 3.1 [timed token and timed marking]

- A timed token is a 3-uple $((p, c), d)$ where p is its place, c is its colour and d is the value of its delay.
- A timed marking TM is a multi-set of timed tokens. A state of an *ITCPN* is a timed marking.

Consider the previous model (Figure 1). Its initial timed marking, named TM_0, can be written as follows:

$$((p_{free}, M_1), 2) + ((p_{in}, J_1), 1) + ((p_{in}, J_2), 2).$$

At the starting time, there are three tokens which are not yet available. The first and the third one will become available after two time units while the second one will become available exactly after one time unit.

3.2 State evolution

Initially, the model is in its initial timed marking. Afterwards, its state evolves either by time progressions (delays decrease with time) or by occurrences.

Definition 3.2 events enabled for a timed marking

- An event is a pair composed with one transition and all tokens required for

its occurrence.

- Let e be an event. We denote by $Jin(e)$ and $Jout(e)$ multi-sets of tokens used and produced by event e.
- Let TM be a timed marking. An event e is enabled for the timed marking TM if and only if, all required tokens are present (created but not necessary available) in TM, i.e.: $Jin(e) \leq TM$.

In this model, an event should occur as soon as possible (i.e.: when all required tokens become available). Its occurrence takes no time but it leads to a new marking: consumed tokens disappear and eventually new tokens are created.

Definition 3.3 [time progression and event occurrence] Let TM be a reachable state of an $ITCPN$ model, $ES(TM)$ the set of events enabled for the timed marking TM, $e*$ an event and dh a non negative real number.

- The occurrence delay (i.e.: the firing delay) of event $e*$, denoted by $FD(e*)$, is the delay required for all tokens of $Jin(e*)$ to become available, i.e.: $FD(e*) = max_{j \in Jin(e*)}(d_j)$.
- A time progression of dh units can occur, from state TM, before any event occurrence, if and only if, dh is less or equal to the occurrence delays of all events of $ES(TM)$, i.e.: $dh \leq min_{e \in ES(TM)}(FD(e))$.

  After this time progression, the delay of each token of TM decreases of dh time units. We denote by $\lfloor TM \rfloor_{-dh}$ the obtained timed marking.
- Event $e*$ can occur from TM, before other events, if and only if, it is enabled and its occurrence delay is not greater than those of other enabled events, i.e.: $(Jin(e*) \leq TM) \wedge (FD(e*) \leq min_{e \in ES(TM)}(FD(e)))$.
- If event $e*$ can occur from state TM, it occurs instantaneously exactly after $FD(e*)$ time units. The obtained state TM' is:

$$\lfloor TM - Jin(e*) \rfloor_{-FD(e*)} + Jout(e*)$$

  . Where $Jout(e*)$ is obtained from $F(t)(Jin(e*))$ by replacing each interval with one value chosen inside it.

Consider the model given in *Figure 1* and its initial state TM_0:
$((p_{free}, M_1), 2) + ((p_{in}, J_1), 1) + ((p_{in}, J_2), 2)$.
For the initial state TM_0, we have two enabled events:

$$e_0 = (t_1, ((p_{free}, M_1), 2) + ((p_{in}, J_1), 1))$$

and

$$e_1 = (t_1, ((p_{free}, M_1), 2) + ((p_{in}, J_2), 2)).$$

Their firing delays are: $FD(e_0) = max(2, 1)$ and $FD(e_1) = max(2, 2)$.

Both events can occur from the initial state after two time units. For example, if event e_0 occurs, its occurrence leads to the state TM_1 such that:

$$TM_1 = ((p_{in}, J_2), 0) + ((p_{busy}, (M_1, J_1)), d)$$

with $d \in [1, 3]$.

It appears that, because of time density, generally, the *ITCPN* model has infinite reachable states and each state has infinite successors (i.e.: infinite state space). After we have defined the operational semantic of the *ITCPN* model, we will, in the following, show how to contract its infinite state space.

4 States reached by the same sequence

To contract the *ITCPN* state space, we first agglomerate, into one state group, all states reached by the same sequence of occurrences independently of the occurrence dates.

Definition 4.1 state group

- A state group is the set E of states reached by the occurrence of the same event sequence independently of instants at which events have occurred. It can be characterised by a pair (SM, FT) where:
  - SM is obtained from any timed marking of E by replacing values of delays by variables representing delays.
  - FT is a logical formula which indicates all valuations of delays associated with tokens.
- Let E be a state group and $e*$ an event. Event $e*$ can occur from state group E if and only if, there is a state TM of group E such that event $e*$ can occur from it. This possibility of event occurrence is denoted by $E[[e* \gg$.

  If event $e*$ can occur from group E, its occurrence leads to the state group E' which consists of all states reached by the occurrence of event $e*$ from any state of group E. This occurrence is denoted by $E[[e* \gg E'$.

The initial state group of the model of *Figure 1* is $E_0 = (SM_0, FT_0)$ where:

- $SM_0 = ((p_{free}, M_1), d_1) + ((p_{in}, J_1), d_2) + ((p_{in}, J_2), d_3)$.
- $FT_0 = (d_1 = 2 \wedge d_2 = 1 \wedge d_3 = 2)$.

The enabled event $e_0 = (t_1, ((p_{free}, M_1), d_1) + ((p_{in}, J_1), d_2))$ can occur from state group E_0 if and only if, the following formula is consistent:

$$FT_0 \wedge (max(d_1, d_2) \leq min(max(d_1, d_2), max(d_1, d_3))).$$

The reached group by the occurrence of event e_0 from the group E_0 is $E_1 = (SM_1, FT_1)$ where:

- $SM_1 = ((p_{in}, J_2), d_3) + ((p_{busy}, (M_1, J_1)), d_4)$ and
- $FT_1 = (d_3 = 0 \wedge 1 \leq d_4 \leq 3)$.

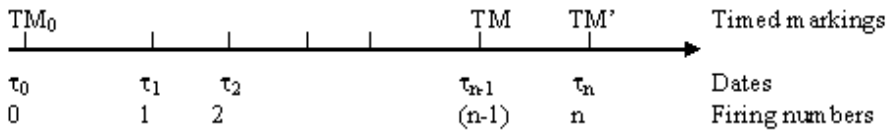

Fig. 2. Evolution of an ITCPN

5 Agglomeration of state groups

This first contraction is not enough since the set of reachable groups can be infinite even if the model has a finite number of reachable non timed markings. This occurs, for example, if there is a token that is never used (its delay decreases infinitely). To contract more the state space and to simplify the building of the contracted state space, we have worked to define a simple bisimulation relation over the set of reachable state groups. Our relation is based on the following theorem shown in [4].

Consider the group E reached from the initial state group by firing an event sequence of length $(n-1)$ with $n > 0$ (see *Figure 2*).

$\forall (k,l) \in [0, n-1]^2$, we denote by $TE_E(l,k)$ the maximal distance between dates of the lth and the kth firings, i.e.: $TE_E(l,k) = max_{TM \in E}(\tau_k - \tau_l)$.

Let j be one created token. We denote by f_j the number of the firing that has created the token j, and by $[a_j, b_j]$ the interval associated with its creating arc.

Theorem 5.1 *Let $e*$ be an event enabled for the state group $E = (SM, FT)$.*

- *Event $e*$ can occur from state group E if and only if:*
  $\forall i \in Jin(e*), \forall e \in ES(SM), \exists j \in Jin(e),$
  $a_i \leq max(b_j + TE_E(f_i, f_j), TE_E(f_i, n-1))$
- *Suppose that $e*$ can occur from the state group E and its occurrence leads to the state group E'. Distances between previous firings in state group E' can be computed using those of state group E as follows:*
  $\forall (l,k) \in [0, n-1]^2$
  $TE_{E'}(l,n) = min_{e \in ES(M)}(max_{j \in Jin(e)}(max(b_j + TE_E(l, f_j), TE_E(l, n-1))))$
  $TE_{E'}(n,l) = min(TE_E(n-1, l), min_{i \in Jin(e*)}(TE_E(f_i, l) - a_i))$
  $TE_{E'}(l,k) = min(TE_E(l,k), TE_{E'}(l,n) + TE_{E'}(n,k))$

5.1 Simplification of the firing rule

We will define here a simple firing rule which avoids solving systems of inequalities or using constraint graphs.

Definition 5.2 Let EE be the set of all reachable state groups of an *ITCPN*

model. Let A, B and D be functions which associate with each state group $E = (SM, FT)$ of EE, functions A_E, B_E and D_E defined as follows:

- $A_E : SM \to Q$, $\forall i \in SM, A_E(i) = min(0, TE_E(f_i, n-1) - a_i)$.
- $B_E : SM \to Q, \forall j \in SM$, $B_E(j) = max(0, b_j + TE_E(n-1, f_j))$.
- $D_E : SM^2 \to Q$, $\forall (i,j) \in SM^2$,
  $D_E(i,j) = max(A_E(i), min(B_E(j), b_j + TE_E(f_i, f_j) - a_i))$

For the initial state group E_0, the associated functions A_{E0}, B_{E0} and D_{E0} are defined as follows: $\forall (i,j) \in SM_0^2$,
$A_{E0}(i) = -a_i$; $B_{E0}(j) = b_j$ and $D_{E0}(i,j) = b_j - a_i$.

The value of $D_E(i,j)$ is intuitively the maximal distance, in group E, between delays of tokens i and j. The values of $-A_E(i)$ and $B_E(i)$ are the minimal and maximal delays of token i, in state group E.

Proposition 5.3 *Let $E = (SM, FT)$ be a state group of EE, $e*$ an event, and $ES(SM)$ the set of events enabled for SM. Event $e*$ can occur from state group E if and only if:*
$e* \in ES(SM) \wedge$
$\forall i \in Jin(e*), \forall e \in ES(SM), \exists j \in Jin(e), D_E(i,j) \geq 0$.

Proof. Suppose that:
(1) $\exists i \in Jin(e*), \exists e \in ES(SM), \forall j \in Jin(e)$, $D_E(i,j) < 0$.
From expression of $D_E(i,j)$, $D_E(i,j) < 0$ if and only if:

$$b_j + TE_E(f_i, f_j) - a_i < 0 \wedge TE_E(f_i, n-1) - a_i < 0.$$

Relation (1) is equivalent to: $\exists i \in Jin(e*), \exists e \in ES(SM), \forall j \in Jin(e)$,
$a_i > max(b_j + TE_E(f_i, f_j), TE_E(f_i, n-1))$
Using theorem 5.1, we deduce That: event $e*$ can occur from E, if and only if,

$$\forall i \in Jin(e*), \forall e \in ES(SM), \exists j \in Jin(e), \quad D_E(i,j) \geq 0$$

. □

The following proposition shows how to compute, using function D_E, the function $D_{E'}$ of any successor group E'.

Proposition 5.4 *Let $E = (SM, FT)$ be one group of set EE and $e*$ an event which can occur from E. Let $E' = (SM', FT')$ be the state group reached by firing event $e*$ from state group E. The functions $A_{E'}, B_{E'}$ and $D_{E'}$ of the state group $E' = (SM', FT')$ can be computed using function D_E as follows:*
$\forall i \in SM'$,

- *if i is not created by event $e*$:*
  $A_{E'}(i) = min_{e \in ES(SM)}(max_{j' \in Jin(e)}(min(0, D_E(i, j'))))$
  $B_{E'}(i) = max_{i* \in Jin(e*)}(max(0, D_E(i*, i)))$
- *if i is created by event $e*$: $A_{E'}(i) = -a_i$ and $B_{E'}(i) = b_i$.*

$\forall (i,j) \in SM'^2$,

- *if i and j are not created by event $e*$:*
  $D_{E'}(i,j) = max(A_{E'}(i), min(D_E(i,j), A_{E'}(i) + B_{E'}(j)))$
- *if i or j is created by event $e*$:* $A_{E'}(i) + B_{E'}(j)$

Proof. Recall definitions of $A_{E'}, B_{E'}$ and $D_{E'}$:
$\forall i \in SM'$, $A_{E'}(i) = min(0, TE_{E'}(f_i, n) - a_i)$
$\forall j \in SM'$, $B_{E'}(j) = max(0, b_j + TE_{E'}(n, f_j))$.
$\forall (i,j) \in SM'^2$,
$D_{E'}(i,j) = max(A_{E'}(i), min(B_{E'}(j), b_j + TE_{E'}(f_i, f_j) - a_i))$.
We consider here only the complex case, i.e.: tokens i and j are not new ($f_i < n$ and $f_j < n$). The other cases are more simple and can be shown similarly. It suffices to show and use progressively the following relations:
(1) $max(X, min(Y, Z)) = min(max(X, Y), max(X, Z))$
(2) $min(X, max(Y, Z)) = max(min(X, Y), min(X, Z))$
(3) $A_E(i) \leq 0$,
(4) $B_E(j) \geq 0$
(5) $min(0, D_E(i,j)) = min(0, max(TE_E(f_i, n-1), b_j + TE_E(f_i, f_j)) - a_i)$
(6) $max(0, D_E(i,j)) = max(0, b_j + min(TE_E(n-1, f_j), TE_E(f_i, f_j) - a_i))$
(7) $TE_{E'}(f_i, f_j) = min(TE_E(f_i, f_j), TE_{E'}(f_i, n) + TE_{E'}(n, f_j))$
(8) $B_{E'}(j) = min(B_{E'}(j), B_E(j))$
(9) $A_{E'}(i) = max(A_{E'}(i), A_E(i))$
(10) $A_{E'}(i) = min_{e \in ES(SM)}(max_{j' \in Jin(e)}(min(0, D_E(i, j'))))$
(11) $B_{E'}(j) = max_{i* \in Jin(e)}(max(0, D_E(i*, j)))$
Using relations (7), (8), (9) and (1) we show that:
(12) $D_{E'}(i,j) = min(A_{E'}(i) + B_{E'}(j), max(A_{E'}(i), D_E(i,j)))$
Finally, by applying relations (2) and (4), we deduce that:
$D_{E'}(i,j) = max(A_{E'}(i), min(A_{E'}(i) + B_{E'}(j), D_E(i,j)))$ □

5.2 Bisimulation relation over the set of state groups

We have rewritten the occurrence rule by using only the marking and the function D. Consequently, all state groups that share the same marking and the same function D are bisimilar and then can be agglomerated into one summit.

Definition 5.5 Let EE be the set of state groups and $B \subseteq EE^2$ one binary relation over EE defined as follows:

$$\forall E_1 \in EE, \forall E_2 \in EE,$$

(let SM_1 and SM_2 be symbolic markings of state groups E_1 and E_2)
$(E_1, E_2) \in B$ if and only if, it is possible to rename delays in SM_2 so as to obtain: $SM_1 = SM_2$ and $D_{E1} = D_{E2}$.

Theorem 5.6 *Relation B is a bisimulation.*

Proof.

To show that relation B is a bisimulation, it suffices to prove the following:
$\forall (E_1, E_2) \in B, \forall e*$,
(i) $E_1[[e* \gg E_2[[e* \gg$
(ii) $(E_1[[e* \gg E_{1'} \wedge E_2[[e* \gg E_{2'}) \Rightarrow (E_{1'}, E_{2'}) \in B$
Since, it is possible to rename delays in SM_2 so as to obtain: $SM_1 = SM_2$ and $D_{E1} = D_{E2}$,
relations (i) and (ii) are deduced from propositions 5.4 and 5.5 □

5.3 Graph of reachable state classes

All bisimilar state groups are agglomerated into one state class. The state classes of the model are the equivalence classes of relation B.

Definition 5.7 [state class] Let $E = (SM, FT)$ be a state group of EE. The state class of group E is the pair (SM, D_E).

The building of the state class graph is obtained by applying the firing rule given in Propositions 5.4 and 5.5, to the initial state class and to each new state class. The following theorem establishes one necessary and sufficient condition to have a finite class graph. The proof of this theorem uses the following proposition shown in [3].

Proposition 5.8 *Let Y be a finite linear combination i.e.: $Y = n_1 \times y_1 + n_2 \times y_2 + ... + n_r \times y_r$ where $n_1, n_2, ..., n_r$ are integer numbers and $y_1, y_2, ..., y_r$ are rational constants. If Y is bounded by rational constants (i.e.: $a \leq Y \leq b$) then the number of different linear combinations is finite. In other words, the value domain of Y is finite.*

Theorem 5.9 *An* ITCPN *has a finite state class graph if and only if, it is bounded.*

Proof. $\Rightarrow$) is obvious.

$\Leftarrow$) If the model is bounded, it has a finite set of reachable markings. Since the number of different markings is finite, it suffices to prove that for a given marking, we have a finite number of different classes that share the same marking. More precisely, we shall show that for any reachable marking SM the number of possible D_E is finite. For each pair(i, j) of SM^2, $D_E(i, j)$ is a finite combination of rational constants with integer coefficients (finite combination because the number of different time intervals in the model is finite).
$D_E(i, j)$ is bounded: $-a_i \leq D_E(i, j) \leq b_j$.
From proposition 5.8, its value domain is finite. □

This necessary and sufficient condition to have a finite graph may be difficult to use since we have not a general procedure to decide whether or not any *ITCPN* has a finite number of different markings. However, we have a straightforward sufficient condition using the underlying coloured Petri net (*CPN*),

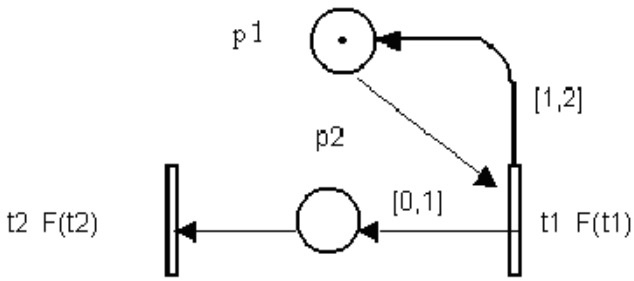

Fig. 3. Bounded ITCPN but unbounded CPN

and we know several methods to decide this property on *CPN*, namely the invariant method: An *ITCPN* has a finite number of markings (i.e.: bounded), if its underlying *CPN* has a finite number of different markings (i.e.: bounded). The reverse is not true. Consider the *ITCPN* given in *Figure 3* and suppose that its initial marking is: $M_0 = (p1, prod)$ and its transition functions are defined as follows: $F(t_1)(p_1, prod) = ((p_1, prod), [1, 2]) + ((p_2, mess), [0, 1])$ and $F(t_2)(p_2, mess) = 0$.

This model has three reachable markings: M_0, $M_1 = (p_1, prod) + (p_2, mess)$ and $M_2 = (p_1, prod) + 2(p_2, mess)$. But its underlying coloured Petri net is unbounded (place p_2 is unbounded).

If the state class graph is finite and its size is not too big, we can build it and then prove properties of the model by exploring the graph. It would be interesting to reduce the size of the graph so as to be able to analyse bigger models. To reduce the size of the class graph, we propose to agglomerate into one summit, all classes sharing the same marking. This operation reduces considerably the size of the graph (by half if every marking is shared by two classes). The obtained graph is translated into one timed automaton. This translation is needed to preserve equivalency between the *ITCPN* state space and its reduced state space.

6 Translating *ITCPN* into one timed automaton

6.1 The timed automaton model

A timed automaton is an automaton completed with clocks and temporal formulas associated with nodes and arcs. Clocks evolve synchronously with time progression but each one can be reset individually to zero.

Each arc is labelled with one event, one formula (named firing condition) and one list of initialisations which must be performed when the event occurs. The firing condition characterises the set of clock valuations for which the event may occur. Formulas associated with nodes (one per node) are named validate conditions and specify the sojourn time in each node.

The state of a timed automaton is defined by the current node and values of clocks. The model can remain in one node as long as values of clocks satisfy the validate condition of that node. A node must be left before that the validate condition becomes not consistent. From a node, the event of an outgoing arc can occur if and only if current values of clocks satisfy, both, validate condition of the node and firing condition of the arc.

6.2 Timed automaton of an ITCPN *model*

From its starting instant, an *ITCPN* model evolves as follows: first, its remains in its initial marking until at least one of enabled transitions becomes ready to occur. At this time, one of the ready transitions occurs and then leads to another marking. The *ITCPN* model remains in the reached marking until at least one of enabled transitions becomes ready to occur, and so on.... Recall that in the *ITCPN* model, transitions occur as soon as possible (i.e.: when all required tokens become available).

The timed automaton of an *ITCPN* model is its marking graph completed by associating one clock and two parameters with each token, one validate condition with each marking, one firing condition and one list of initialisations with each arc. For each token j, we denote by h_j, a_j and b_j its clock and its two parameters. When a token j is created, its clock h_j is initialised with zero and other time parameters are initialised with bounds of time interval associated with its creating arc. Afterwards, the value of its clock increases with time until it is consumed but values of other parameters do not change.

6.3 Validate condition of a marking

Let SM be a symbolic marking and $ES(SM)$ the set of events enabled for the symbolic marking SM. The model remains in symbolic marking SM until at least one of enabled transitions becomes ready to occur (i.e.: all required tokens become available). In other words, it remains in symbolic marking SM as long as for each enabled event there is at least one of its consumed tokens which is not yet available or, at last, which becomes available:

$$\forall e \in ES(SM), \exists j \in Jin(e), h_j \leq b_j$$

Which is equivalent to: $min_{e \in ES(SM)}(max_{j \in Jin(e)}(b_j - h_j)) \geq 0$.

6.4 Firing condition of an enabled event

Let SM be a symbolic marking and $e*$ an event enabled for the marking SM. The firing condition of event $e*$ indicates all possible clock valuations at which the event can occur. For the $ITCPN$ model, an event should occur when all required tokens become available. The firing condition of event $e*$ from the symbolic marking SM is defined as follows: $\forall j \in Jin(e*)$, $a_j \leq h_j$.
Which is equivalent to: $max_{j \in Jin(e*)}(a_j - h_j) \leq 0$.

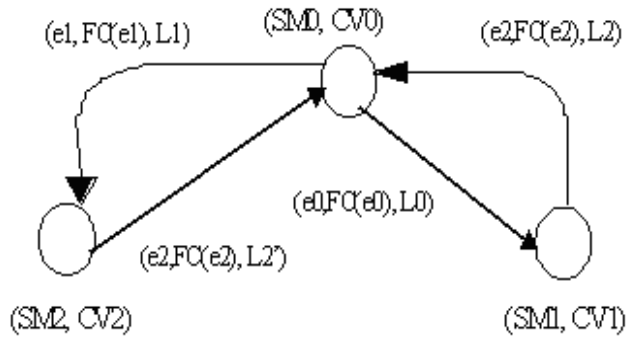

Fig. 4. A timed automaton

For example, consider the model of *Figure 1* and its initial symbolic marking SM_0: $((p_{free}, M_1), h_1, [a_1, b_1]) + ((p_{in}, J_1), h_2, [a_2, b_2]) + ((p_{in}, J_2), h_3, [a_3, b_3])$ with $[a_1, b_1] = [2, 2], [a_2, b_2] = [1, 1]$ and $[a_3, b_3] = [2, 2]$.
There are two enabled events for symbolic marking SM_0:
$e_0 = (t_1, ((p_{free}, M_1), h_1, [a_1, b_1]) + ((p_{in}, J_1), h_2, [a_2, b_2]))$ and
$e_1 = (t_1, ((p_{free}, M_1), h_1, [a_1, b_1]) + ((p_{in}, J_2), h_3, [a_3, b_3]))$.
The validate condition of the initial marking SM_0 is:
$min(max(b_1 - h_1, b_2 - h_2), max(b_1 - h_1, b_3 - h_3)) \geq 0$.
The firing conditions of events e_0 and e_1 are:
$FC(e_0) = max(a_1 - h_1, a_2 - h_2) \leq 0$ and
$FC(e_1) = min(a_1 - h_1, a_3 - h_3) \leq 0$

An *ITCPN* and its associated timed automata have the same behaviours, since the starting points and rules for evolutions are the same in both cases. The timed automaton of an *ITCPN* is finite if and only if, the *ITCPN* model has a finite set of reachable markings.

7 Application

For example, applying our approach for the *ITCPN* of *Figure 1*, produces the timed automaton shown in *Figure 4*.

Where:
$SM_0 = ((p_{free}, M_1), h_1) + ((p_{in}, J_1), h_2) + ((p_{in}, J_2), h_3)$
$SM_1 = ((p_{in}, J_2), h_3, [a_3, b_3]) + ((p_{busy}, (M_1, J_1)), h_4, [a_4, b_4])$
$SM_2 = ((p_{in}, J_1), h_2, [a_2, b_2]) + ((p_{busy}, (M_1, J_2)), h_4, [a_4, b_4]))$
$CV_0 = (min(max(b_1 - h_1, b_2 - h_2), max(b_1 - h_1, b_3 - h_3)) \geq 0)$
$CV_1 = (b_4 - h_4 \geq 0)$.
$CV_2 = (b_4 - h_4 \geq 0)$.
$e_0 = (t_1, ((p_{free}, M_1), h_1, [a_1, b_1]) + ((p_{in}, J_1), h_2, [a_2, b_2]))$
$e_1 = (t_1, ((p_{free}, M_1), h_1, [a_1, b_1]) + ((p_{in}, J_2), h_3, [a_3, b_3]))$

$e_2 = (t_2, ((p_{busy}, (M_1, J_1)), h_4, [a_4, b_4]))$
$FC(e_0) = (max(a_1 - h_1, a_2 - h_2) \leq 0)$
$FC(e_1) = (max(a_1 - h_1, a_3 - h_3) \leq 0)$
$FC(e_2) = (a_4 - h_4 \leq 0)$
$L_0 : h_4 = 0; a_3 = 2; b_3 = 2; a_4 = 1; b_4 = 3.$
$L_1 : h_4 = 0; a_2 = 1; b_2 = 1; a_4 = 1; b_4 = 3.$
$L_2 : h_1 = 0; h_2 = 0; a_1 = 2; b_1 = 2; a_2 = 1; b_2 = 1.$
$L_{2'} : h_1 = 0; h_3 = 0; a_1 = 2; b_1 = 2; a_3 = 2; b_3 = 2.$
Initially, we have: $a_1 = b_1 = 2; a_2 = b_2 = 1$ and $a_3 = b_3 = 2$.

8 Conclusion

We have shown how to contract the state space of the *ITCPN* model. We have distinguished three levels of contraction. In the first level, we have agglomerated, into one group, all states reached by the same occurrence sequence. Afterwards, we have established an attractive bisimulation relation, over the set of groups, that allows both agglomerating, into one class, all bisimilar groups, and simplifying the building of the class graph. Finally, all classes that share the same marking are agglomerated into one summit. The obtained marking graph is translated into one time automaton which is coherent and well timed. Moreover, since there is a finite reachable classes that share the same marking, we think that all these characteristics can be exploited to simplify the proof of some properties.

Furthermore, the obtained timed automaton can be more contracted with a colour based equivalence as shown in [4].

References

[1] R. Alur, D. Dill, *Automata for modeling real-time systems*, 17ème ICALP, LNCS 443, Springer-verlag, 1990.

[2] C. André, *Synchronized Elementary Net Systems*", Advances in Petri Nets, Grzegorz Rosenberg, editor, LNCS 424, Springer-Verlag, 1989.

[3] B. Berthomieu, M. Diaz, *Modeling and verification of time dependent systems using time Petri nets*, IEEE Transactions on Software Engineering, **17**:3, March 91.

[4] G. Berthelot, H. Boucheneb, *Occurrence graphs for interval timed coloured nets*, 15th International Conference on Application and Theory of Petri Nets, Zaragoza (Spain), LNCS 815, Springer-verlag, June 1994.

[5] H. Boucheneb, G. Berthelot, *Towards a simplified building of time Petri Net Reachability graphs*, in proc. of Petri Nets and Performance Models PNPM'93, Toulouse (France), IEEE Computer Society Press, October 1993.

[6] S. Christensen, L.M. Kritensen, T. Mailand, *Condensed state spaces for timed Petri Nets*, 22nd International Conference on Application and Theory of Petri Nets and 2nd International Conference on Application of Concurrency to System Design, June 2001, Newcastle Upon Tyne.

[7] C. Daws, A. Olivero, S. Tripakis and S. Yovine, *The tool Kronos*, In Hybrid Systems III, Verification and Control, LNCS 1066, Springer-Verlag, 1996.

[8] M. Diaz, P. Sénac,*Time Stream Petri Nets a model for Timed Multimedia Information*, 15th International Conference on Application and Theory of Petri Nets, LNCS 815, Springer-Verlag, Zaragoza (Spain), June 1994.

[9] T. A. Henzinger, P-H. Ho, H. Wong-Toi, *HyTech: A Model Checker for Hybrid Systems* Software Tools for Technology Transfer 1: 110-122, 1997.

[10] Pao-Ann Hsiung, Farn Wang, *A State-Graph Manipulator Tool for Real-Time System Specification and Verification*, In proc. of the 5th International Conference on Real-Time Computing Systems and Applications, RTCSA'98, October 1998.

[11] K. Jensen, *Coloured Petri Nets: Basic concepts, Analysis Methods and Practical use*, volumes 1 and 2, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1982.

[12] P. Merlin, D.J. Farber, *Recoverability of communication protocols*, IEEE Trans. on Communications 24, 1976.

[13] C. Ramchandani, *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*, Project MAC, TR 120, MIT, 1974.

[14] J. Sifakis *Use of Petri Nets for Performance Evaluation*, Measuring, Modeling and Evaluating Computer Systems, H.Beilner and E.Gelenbe editors, North-Holland, 1977.

[15] W.M.P. van der Aalst, *Interval Timed Coloured Petri Nets and their Analysis*, 14th International Conference of Application and Theory of Petri Nets, Chicago, 1993.