A Distributed Algorithm for the Validation of Timed State Machines

Xavier Rebeuf, Gerardo Satriano, Françoise Simonot-Lion

A DISTRIBUTED ALGORITHM FOR THE VALIDATION OF TIMED STATE MACHINES

Xavier Rebeuf[2] Gerardo Satriano* Françoise Simonot-Lion+

Abstract: *In this paper, we propose an a priori validation technique for time critical systems. To model such system, we use a sub-class of timed automata formalisms, called Timed Input output State machine. We validate the model of a given system using a hybrid method combining exhaustive exploration and simulation. In order to avoid the combinatory explosion, we use the exhaustive analysis only for critical parts of the state machine. Such method allows obtaining deterministic proof for these parts. Furthermore, it produces a symbolic computation of the resulting clocks and constraints with regard to the initial ones. Therefore, it is possible to consider these parts as black boxes. We can perform simulations of the complete state machine substituting these critical parts with the corresponding black boxes. In this paper, we propose an algorithm to automatically partition the automata. All the obtained parts are independent, which implies we can exhaustively analyse each part. Thus, it is possible to parallelize the system analysis.*

Keywords: *TIOSM; A priori validation; exhaustive exploration; simulation; partitioning*.

1 Introduction

A priori validation of time critical systems requires a formalism that is able to represent characteristics such as relative or absolute instants or time intervals for event occurrences. Several formalisms are available for this purpose. Among them, we find extended Process Algebras (Timed CSP [RR87], ACSR [BCL93]), temporal logics (for example, RTTL [Ost89]), temporal extensions of Formal Description Techniques (for example [BGK+01]) or timed state machines as Time Petri Nets in the sense of Merlin and Farber [MF76] or Timed Automata [AD94]. For timed state machine formalisms, it exists some tools for the analysis of models (e.g. Kronos [BDM+98], UPPAAL [PL00]).

In this paper we focus on a sub-class of this last formalism called Timed Input Output State Machine (TIOSM [KC95]). In TIOSM formalism, the form of transition firing time constraints is a *bounded interval*. Moreover, this formalism allows specifying the application as a set of communicating state-machines. In a Timed Input Output State Machine, two main attributes can act on the transition firing: the reception/emission of messages and timed constraints expressed on different clocks. In order to validate a time critical system, an analysis of its model must be done. Note that the level of abstraction or the accuracy of a model is inverse ratio to its capacity to be treated in a bounded time. Two ways can be used for a model analysis:

- By *exhaustive exploration* of the model,
- By *simulation* of the model.

The first approach leads to deterministic results thanks to several approaches based either on the graph of regions ([ACH+95], [AD94a]) or on enumerative method ([KSK00], [KS00]) as they are used for Time Petri Nets ([BD91], [TST97]). If deterministic proof can be obtained, the combinatory explosion is a bar to exhaustive method application. As opposite, the last approach allows getting results quickly, but the pertinence of these results is strictly related to the scenario and the simulation duration. A method was proposed in [KS01] and [Kai01] to take profit of the advantages of these two approaches. It consists of applying exhaustive analysis to critical parts of a system and using the obtained results for the simulation of the whole system.

In this paper we propose to extend this hybrid method, combining simulation with exhaustive analysis. The validation needs a unique representation of the complete system in term of one TIOSM (possibly obtained by a synchronised product of the set of automata representing subsystems). The complete system is decomposed in parts. The set of parts realizes a partition of the whole automata and the properties of this partition allow the parallelization of exhaustive analysis of the different parts.

[2] LORIA (UMR CNRS 7503)/INPL, 2 avenue de la Forêt de Haye, 54516 Vandoeuvre-les-Nancy CEDEX, France.
Email : rebeuf@loria.fr, simonot@loria.fr
* Université La Sapienza,via Eudossiana 18, 00184 Rome, Italy.
Email : gsatriano@yahoo.it

This paper will present successively in section 2 the TIOSM formalism and, in section 3, an overview of the hybrid validation method. The properties of the partition and how to compute it are defined in section 4. In section 5 and 6 we recall how to build an abstraction of each part of the model (named "black box") and how to integrate these "black boxes" in the complete model in order to simulate it. Finally we conclude in section 7.

2 Timed Input Output State Machines

Formally, a TIOSM is defined by a tuple $T=(S,L,C,s_0,\kappa)$ [KC95].

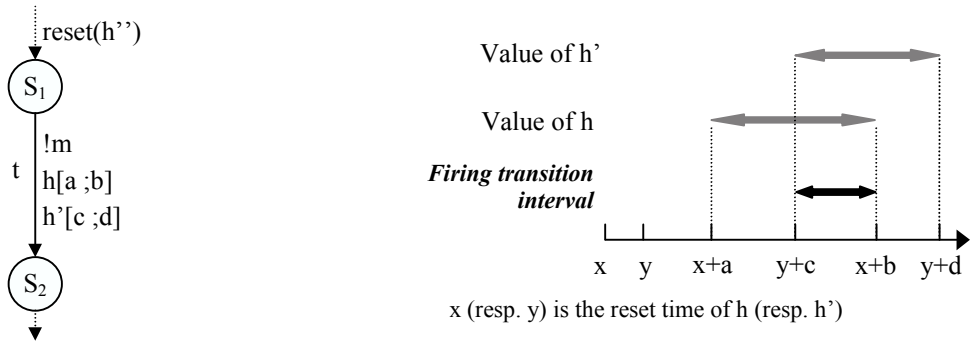
Definition 1:

- *S is a non empty finite set of states; $s_0 \subset S$ is the initial state of T*,
- *L is a non empty finite set of messages*,
- *C is a non empty finite set of clocks*,
- *κ is a non-empty finite set of transitions*.

Any $t \subset \kappa$ has the form $t=(s,\sigma,D,Z,d)$ where:

- *$s \subset S$ is the origin state of t, $d \subset S$ is the target state of t*,
- *$\sigma = \{!,?\} \times L \times \{\vartheta\}$; ϑ is an internal action, !a, ?a is an output or input of message a, where $a \subset L$*,
- *D is a non empty, finite list of temporal properties having the form $(c,[m,M])$ $(c \subset C;\ m, M \subset Q_+,\ m\ \Omega\ M)$*,
- *Z is a finite set of clocks to be reset after t is been fired*,

Figure 1- (a) illustrates an example of transition characterization.



(a) TIOSM T

(b) Firing interval of t for TIOSM T

Figure 1 : Example of a TIOSM T and of transition firing rules

Definition 2: *We define the following firing times:*

- *$\chi(t,c)$ firing time of t respecting constraints on clock c where $((c,[a;b]) \subset D(t))$*,
- *$\chi(t)$ firing time of t respecting constraints on each clock c: $\chi(t) = \chi(t,c_1)\ \sim\chi(t,c_2) \sim \ldots\ \sim\chi(t,c_n)$, $(c_i,[a;b]) \subset D(t)$, $i = 1, \ldots, n$*. Obviously *i* represents the clock index, and *n* is the number of clocks associated to the transition t.

In order to avoid non-determinism, two hypotheses are added:

- every transition t = (s,σ,D,Z,d) may only send or receive a message,
- each pair of transitions (t,t') with s(t) =s(t') respect σ(t) (σ(t'). In the same state, two transitions can't use the same message.

The *Firing transition policy* for the transition characterized in Figure 1-(a) is illustrated in Figure 1-(b).
Some previous works are based on this formalism. In particular, the construction of an accessibility graph similar to class graph obtained for Timed Petri Nets ([MF76], [BD91]) was specified in [Kai01]. An extension of the "on the fly" approach, introduced in [FJJV96], was developed for real-time properties verification in [KSK00]; an adaptive tester, minimizing the number of inconclusive verdicts, is the purpose of [KS00], while [Lau99] proposed a canonical tester construction used for real systems. Finally, the analysis of a TIOSM by a hybrid method can be found in [Kai01], [KS01] and a demonstration of equivalence between TIOSM and Time Petri nets was demonstrated in [HSK02].

3 Validation method

As introduced previously, we use for this methodology both exhaustive analysis for parts of the system and simulation for the whole system. For this purpose we must define four kinds of model:

- The *complete model* is the initial model of the whole system in terms of one TIOSM.
- A *partial model* is a part of the complete model; the set of partial models is a partition of the complete model.
- A *black box* is the abstraction of the partial model obtained after its validation;
- The **result** of the integration of all the black boxes in the complete model is called *derived model*.

The hybrid validation method manipulates these models in six steps, which are divided in two main stages (three steps each). The first one concerns the validation of each part, while the second one is about the validation of the whole system.
Figure 2 illustrates the construction and validation of the *partial models*:

- *Modelling* the whole system in TIOSM formalism (complete model),
- *Identification* of the partition of complete model and extraction of partial models from the complete model; each result is a particular TIOSM,
- *Exhaustive analysis* of each partial model; this step consists in validation of partial model by application of model checking techniques; thanks to the properties of the partition, this step can be parallelized.

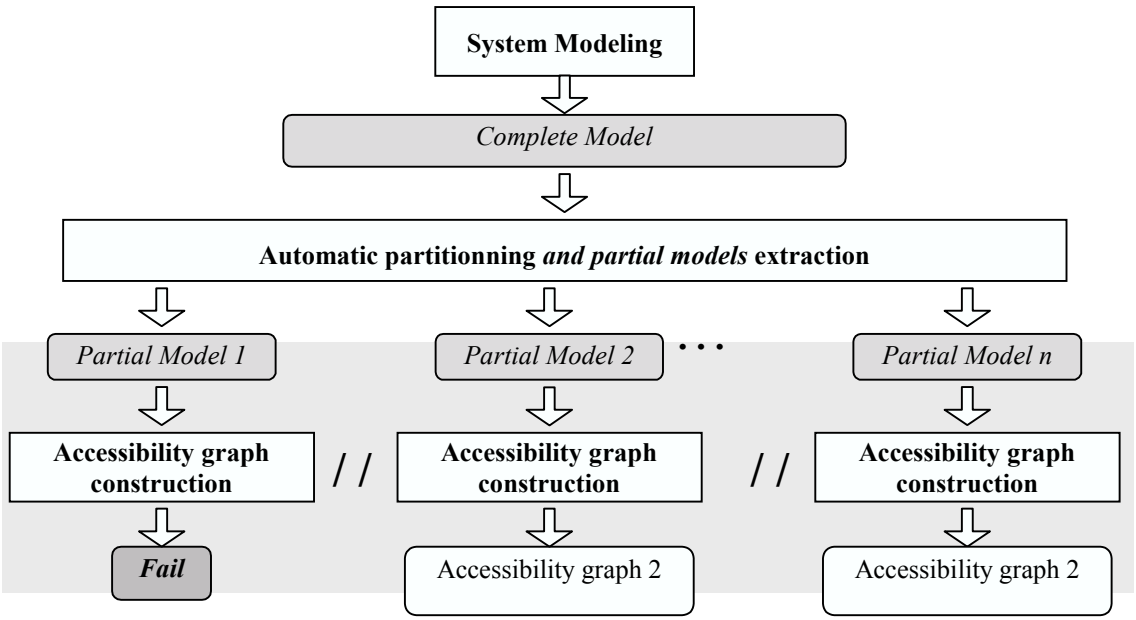


Figure 2 : Partial models construction and validation

The next steps (see Figure 3) require that every *partial model* must be validated; the role of these steps is to build *black boxes* and *derived model* and to validate this *derived model* by simulation:

- For each partial model, specification of the corresponding black box; these activities can be parallelized,
- Integration of each black box in the complete model in order to obtain the derived model,
- Finally, simulation of the derived model.

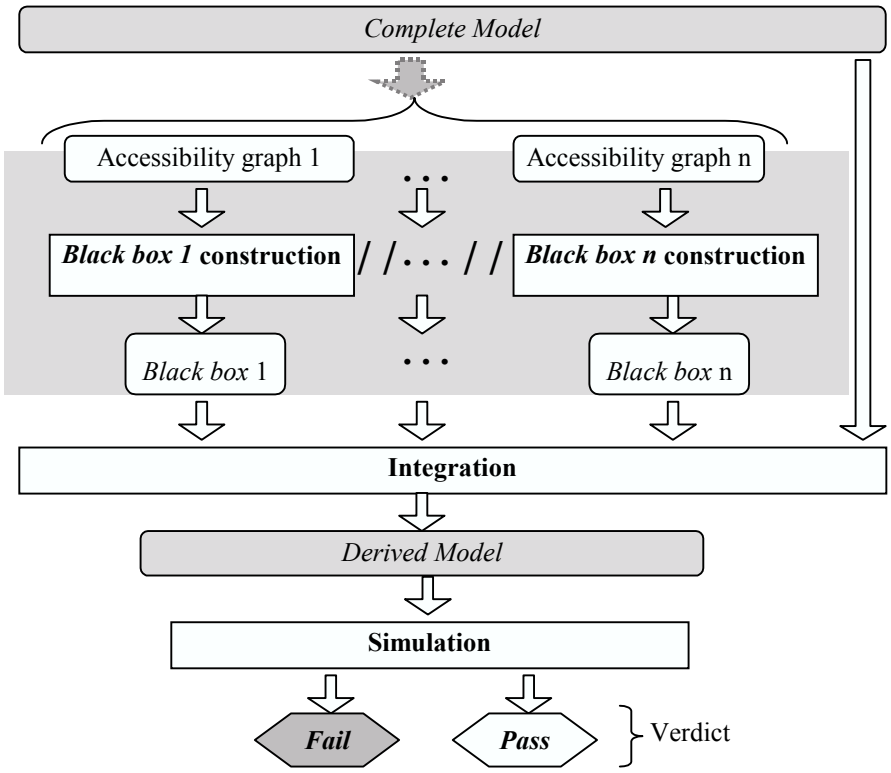


Figure 3 : Black boxes and derived model construction

4 State machine partition

This activity is divided in two steps. The first one takes into account the structure of the automata and realizes the real partition. This step must respect some rules, ensuring on the one hand that the set of partial models is a partition and on the other hand that different partial models are "independent". The second step consists in including temporal information in the obtained *partial models*.

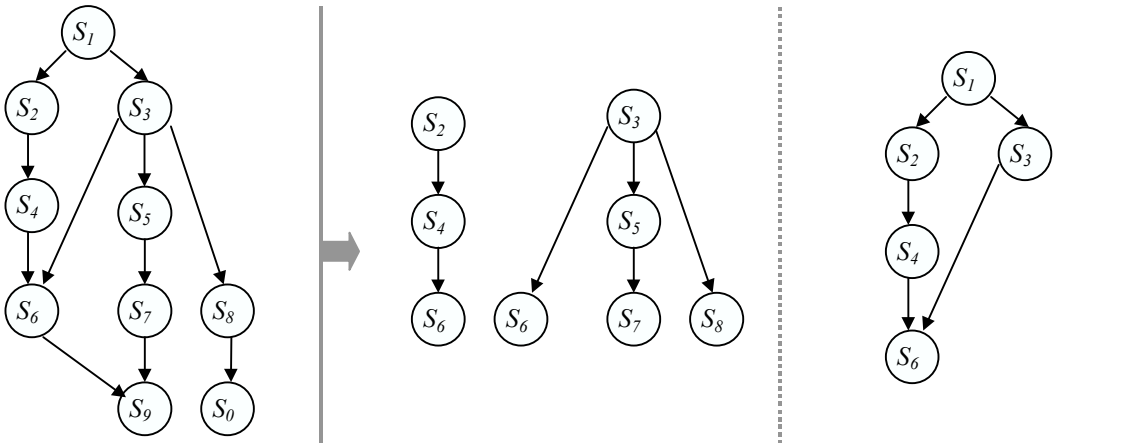
4.1 Independence properties

Let T_S be the TIOSM modelling the whole system and T_{Si} ($i \subset [1;n]$) the TIOSMs modelling the *partial models*. The "independence" of T_{Si} is expressed by the following rules (these rules are illustrated on Figure 4) called **black box criteria** [KS01]:

- Each partial model T_{Si} must have one and only one "entry" state s (for example, in Figure 4, S_2 is the "entry" state of partial model T_{S1} and S_3 is the "entry" state of partial model T_{S2}). It is noted as $enter(T_{Si})$.
- Each partial model must have at least one "exit" state. We introduce the set $exit(T_{Si})$. For example, in Figure 4, we have $exit(T_{S1})=\{S_6\}$ and $exit(T_{S2}) = \{S_6, S_7, S_8\}$.
- An "entry" state s (resp. an "exit" state s') of a partial model T_{Si} can belong to another partial model T_{Sj} only if: $s \subset exit(T_{Sj})$ (resp. $s' \subset exit(T_{Sj})$ or $s' = enter(T_{Sj})$; *exit* states can be shared by more than one

part). As a consequence, an entry state of a partial model can't be an entry state for another partial model.

- A state s that is neither an "entry" state nor an "exit" state can belong only at one partial model.



(a) - Complete model T_S *(b) - Correct partial models* *(c) - Incorrect partial model*

Figure 4 : Example of partial models

4.2 Extraction algorithm

We propose here an algorithm to automatically build a partition of a given model. Between the possible partitions that can be computed, this algorithm observes the following criteria:

- *Maximization* of the number of the partial model (in order to maximize the parallelization),
- Each part contains *at least* one transition.

The temporal attributes of the TIOSM are not taken into account in this step.

Definition 3:

- ***Z** the set of states of the complete model (**Z**=S)*,
- ***T** the set of transitions of the complete model (**T**=κ)*,
- ***T_k** the set of transitions that have not been used up to step **k**,*
- ***P** set of black boxes*,
- ***IQ** queue storing the entry states of T_k.*

A black box is defined thanks to a set of transitions. We first define elementary functions to manipulate such set.

Definition 4:

- *elem* computes the set of states occurring in a transitions set (T_R).
$$elem(T_R) = \{s \subset Z / (\,) s' \subset Z, (s's \subset T_R) \rangle (ss' \subset T_R))\};$$
- *entry* computes the set of entry states occurring in a transitions set
$$Entry(T_R) = \{s \subset elem(T_R) / \bigcup (\,) s' \subset elem(T_R), s's \subset T_R)\};$$
- *exit* computes the set of exit states occurring in a transitions set
$$exit(T_R) = \{s \subset elem(T_R) / \bigcup (\,) s' \subset elem(T_R), ss' \subset T_R)\}.$$

On the set of black boxes, we recursively define a partial order as follow:

- For $BB_1 \subset P, BB_2 \subset P$, we define $BB_1 <_0 BB_2$ if (exit(BB1) ~ entry(BB2) ∏ ∴);
- For $BB_1 \subset P, BB_2 \subset P$, we define $BB_1 <_n BB_2$ if ( ) $BB_3 \subset P$, $(BB_1 <_{n-1} BB_3) \langle (BB_3 <_0 BB_2)$.

Data : The given model of the system T
Result : The partition of the model T
// initialisation
$T_0 = T$
$P_0 = \therefore$
k=0
1. **while** (T_k ∏ ∴)
  push(IQ,entry(T)/IQ) //queue of the potential entry states
  2. **if** (IQ ∏ ∴)
    //construct a new black box
    i_k=**pop**(IQ) //the first element of the queue
    BB={i_ks ⊂ T_k} //The smallest black box
  else
    //extend an existing black box
    BB= **element of** {$B_b \subset P/(\ )$ s⊂exit(B_b) and s'⊂Z,ss'⊂T_k)}
    i_k= **element of** BB
    BB=BB≅{ss'⊂T_k/s⊂exit(BB) and s'⊂Z}
  endif
  $BB_B = \therefore$ //backup of the Black box
  //while the black box grows up
  3. **while** (BB∏ BB_B)
    BB_B =BB
    4. **foreach** s ⊂ elem(BB)/exit(BB)
      5. **if** (s⊂ entry(BB)⟨|entry(BB)|>1)⟩(s⊃entry(BB))
        //we include the transitions ending at s
        BB=BB≅{s's⊂ T_R /s'⊂Z}
        $T_R = T_R$ /BB
        6. **foreach** {$BB_T \subset P/(\ )$ s'⊂Z,s's⊂BB_T)}
          //we merge the two sets
          P=P/{BB_T}
          BB=BB≅ BB_T
          if ($BB_T <_k$ BB)
            I_k=element of entry(BB_T)
          endif
        endfor
      endif
      //we include the transitions starting from s
      BB=BB≅{ss'⊂ T_R /s'⊂Z}
      $T_R = T_R$ /BB
      7. **foreach** {$BB_T \subset P/(\ )$ s'⊂Z,ss'⊂BB_T)
        P=P/{BB_T}
        BB=BB≅ BB_T
      endfor
    endfor
  endwhile
  P=P≅{BB}
endwhile

Algorithm 1 : Partition in black boxes of the given system

Let us briefly describe the algorithm displayed above. Each black box is composed of transitions. The algorithm stops when each transition belongs to a black box.

- For each iteration, we first identify the set of possible entry states (i.e. the set of states which are not target of a transition in T_k). Such states are added to the queue IQ.
- We choose the first entry (denoted by i_k) in the queue. Note that using a queue imposes a width first search of the black boxes entry states with regard to the state machine.
- We construct the smallest set of transitions (denoted by BB) by adding the transitions starting from i_k.
- But such set of transitions is not necessary a black box. Therefore, we add transitions until the set BB satisfies the criteria we gave above.
    - For each state belonging to BB, if it is not an exit state, then we have to include in BB all the transitions starting from this state. If some of these transitions belong to an existing black box, then we merge the two sets in one black box.
    - For each state belonging to BB, if it is not an exit or an entry state, then we have to include in BB all the transitions ending at this state. If some of these transitions belong to an existing black box (denoted by BB_T), then we merge the two sets in one black box. In this case, the new set can have several entry states. If we have $BB_T <_k BB$, then the entry state i_k of the new set becomes the entry state of BB_T.
- After this loop, BB satisfies the black box criteria. Therefore, we can add it to the partition set P. Then, we choose another entry state from the queue in order to construct a new black box.
- If the queue IQ is empty and some transitions don't belong to any black box, then it means there are cycles in the remaining transition set. Each of them has to be included in an existing black box. Therefore, we choose the black box BB with an exit state source of one of the remaining transition. We add this transition to the BB set and we start the algorithm from this state.
- At the end of execution, the black boxes set P corresponds to a partitioning of the state machine.

An example presented below illustrates this algorithm. On the left hand side, Figure 5 presents the initial complete model. The right hand side represents the partitioning (in dotted boxes) after four iterations.

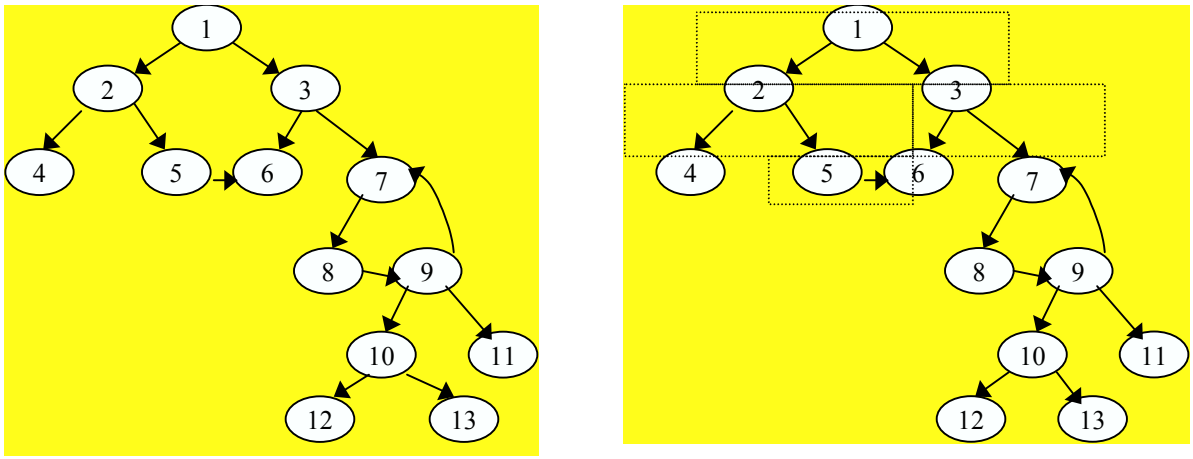


Figure 5 : Black box partition (1)

At this step, there is no entry state for the remaining transition set. Because this set is not empty, we merge one of these transitions (for example transition from state "9" to state "7") to the corresponding black box. Therefore, this set has to be completed in order to obtain another black box (displayed on the left hand side of the fig. 6). Then, the remaining set of transitions has a starting state (state 10). Therefore, the algorithm can construct another black box (displayed on the right hand side of the fig. 6).

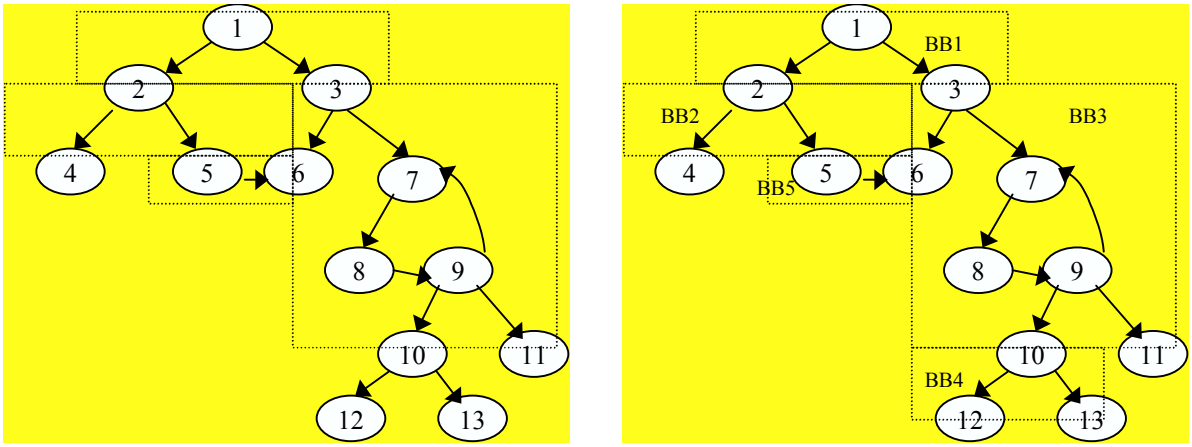
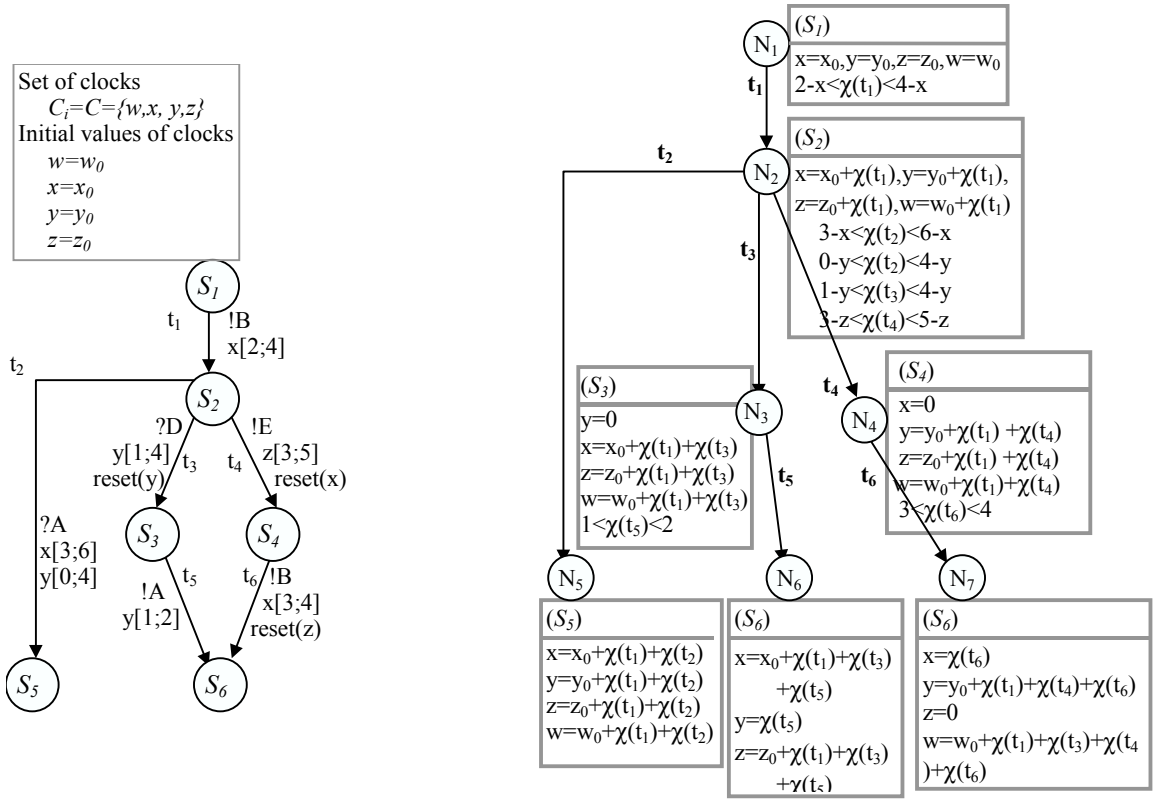


Figure 6 : Black box partition (2)

About the considered example, the algorithm gives as a result 5 black boxes.

4.3 Temporal attributes of partial models

Each partial model is a TIOSM T_{Si}. The set of clocks of each T_{Si} is equal to C, the set of clocks of the initial complete model. Even though some clocks are not considered in a *partial model* (neither reset nor any constraint), for the *complete model* simulation, any *partial model* must know the value of these clocks when reaching their "entry" states. This will allow us to build inequalities systems taking into account their values. Note that as T_{Si} is treated independently, the instant at which its initial state will be marked is not known. This instant depends on the trajectory that the whole system followed before reaching this state.
We associate to each partial model a set of variable representing the initial value of each clock when the system reaches the entry state of T_{Si}. Let us consider a complete model T where $C=\{w, x, y, z\}$, Figure 7-(a) represents a partial model T_{Si} of T.



(a) partial model

(b) accessibility graph of partial model

Figure 7 : Example of a partial model

5 Partial validation

Each TIOSM T_{Si} is exhaustively analysed. We developed an algorithm based on the resolution of inequalities systems (we can use for example the Simplex algorithm to this). An **accessibility graph**, similar to the state class graph proposed [BD91] for Timed Petri Nets model, is computed. Each node N_k (a state class) of this graph is described with:

- a **state** denoted by $st(N_k) \subset S(T_{Si})$
- an **inequalities system** denoted by $Q(N_k)$; this system contains information about the firing time of all transitions that will be fired later, and is built considering only the transitions whose origin is $st(N_k)$.

An edge e_k, between an origin and an extremity nodes (denoted *origin(e_k)* and *extremity(e_k)*), represents a transition $tr(e_k) \subset \kappa(T_{Si})$ whose firing is possible from the state *st(origin (e_k))*.
Below, we present the method and illustrate it on the example presented in Figure 7-(a). Figure 7-(b) is the accessibility graph.

5.1 Initial inequalities system

N_{init}, the initial node of the accessibility graph, is defined by the state $s_0=enter(T_{Si})$ and, for each clock $h \subset C_i$, by an equation $h=h_0$ (where h_0 is a variable representing the value of h when reaching s_0). In the case studied in Figure 7-(a), four clocks are used in T_{Si}, so the initial system is $Q_{init}=\{w=w_0\ x=x_0,\ y=y_0,\ z=z_0\}$.

5.2 Computation of nodes

The construction of the graph is done from the initial node N_{init}:

- $st(N_{init}) = enter(T_{Si})$,
- $Q(N_{init}) = Q_{init}$

Then for each node N_j, the process is similar: let e_i be the edge whose extremity is N_j and consider $tr(e_i)$ and $st(N_j)$:

- In a first step, the different clocks $h \subset C(T_{Si})$, are updated.
- If h is reset after $tr(e_i)$ firing, then the equation becomes h=0; if not, the value of h is defined by adding $\chi(tr(e_i))$ to the previous value of h before $tr(e_i)$ firing; this previous value is obtained from the inequalities system associated to $origin(e_i)$. For example, on Figure 7, $Q(N_2)$ contains the equation $x=x_0+\chi(t_1)$, … where $\chi(t_1)$ is the firing time of $t_1$and $Q(N_4)$ contains the equation x=0, because clock x is reset while firing t_4.
- Then, for each transition t_k whose origin is $st(N_j)$, we consider its clock constraints: for each constraint expressed on a clock h, $h \subset [a,b]$, the inequality $a-h<\chi(t_k)<b-h$ is added to the system. In the presented example, the inequalities $3-x<\chi(t_2)<6-x$, $0-y<\chi(t_2)<4-y$, … are added to $Q(N_2)$. So, we obtain the inequalities system characterizing the node.
- In order to determine if a transition t_k can be fired from $st(N_j)$, we evaluate if the inequalities system admits at least one solution. This is done thanks to Simplex method. Furthermore, we compute the minimum and the maximum of t_k firing time. Note that if there is no solution, we conclude that this transition will never be fired.

If the construction of the accessibility graph shows that no exit state can be reached, it demonstrates a “livelock” in the specification. So, the critical part modelled by T_{Si} is not correct. If not, this part is supposed to be correct (no deadlock, no “livelock”). Note that this step is important because the validation of each partial model is a necessary condition for the validation of the whole system. In this case, an abstraction of this partial model can be built. It reflects temporal properties of T_{Si}. In fact, this abstraction, called *black box* specifies how the time can be passed from the arrival of the *complete system* at the *enter(T_{Si})* to its arrival to a state belonging to *exit(T_{Si})*. The black box corresponding to the example is shown in Figure 8. Note that if one at least is not validated, we conclude that the system modelled by T is not correct.

This *black box* has one and only one initial state $S_{init}=st(N_{init})$ (S_1, in the presented example) and one terminal state S_k for each terminal node N_k characterized by $(st(N_k),Q(N_k))$. In the same example, nodes N_5, $N_{6,1}$, $N_{6,2}$ give 3 terminal states S_5, $S_{6,1}$, $S_{6,2}$. Notice that in this example two nodes refer the same state S_6 in T_{Si}; so, we create two different states in the *black box*. An inequalities system, Q'_k, is created and associated to each "black box" terminal state S_k that corresponds to a terminal node N_k; and contains:

- $Q(N_k)$,
- $Q_{init,k}$: all the inequalities that are associated to the transitions labeling edges on the path $\{N_{init}, N_k\}$ found in the accessibility graph.

For example, in Figure 8, Q'_5 is defined by $\{Q(N_5), Q_{1,5}\}$.

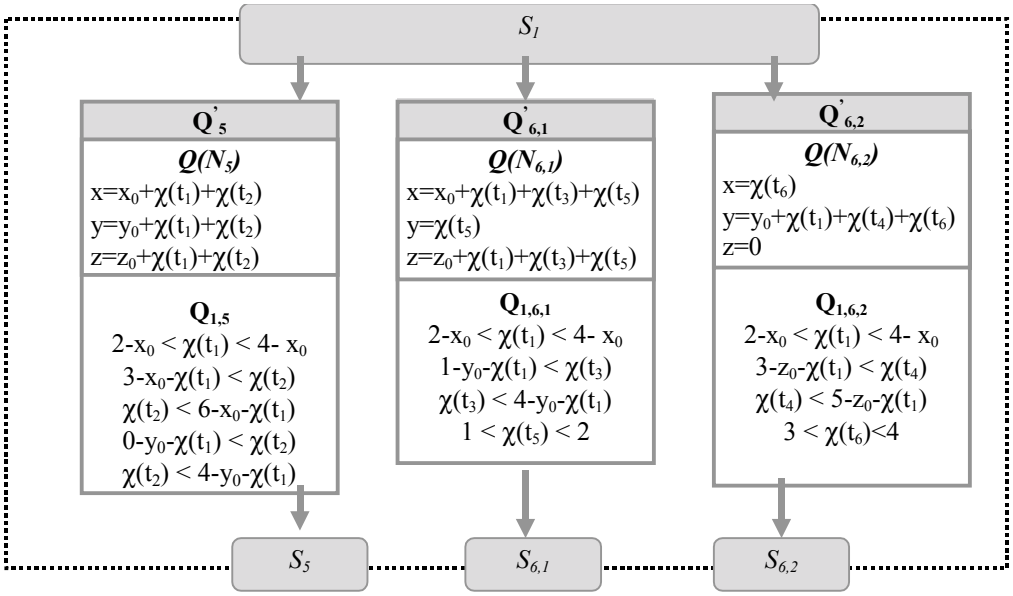


Figure 8 : Black box of partial model TSi

6 Global simulation

If each partial model is validated, a derived model can be built integrating the computed black boxes. To do this, the following process is observed: for each "*black box*" corresponding to T_{Si}:

- suppression in the complete model of each state and transition belonging to T_{Si},
- adding the "black box" to the obtained model
  - each transition t leading, in the complete model to enter(T_{Si}), is linked to S_{init}.
  - each transition t, whose origin state s belongs to exit(T_{Si}), are linked to each black box terminal state associated to s.

Then, along the simulation process, we verify that, when reaching an initial state of a "*black box*", the value of the different clocks allows an exit of the "black box". The simulation algorithm progresses from state to state and the calculus of transition firing time is based on the following rules:

- At initialisation, all the clocks used in the complete model are reset,
- Let S be the current state at a simulation step; the objective of the simulation is then to determine if there is at least a reachable state and, if yes, at which time this state can be reached. Two cases can be identified:

- S is not an entry state of any partial model; a transition t, from those that can be fired is chosen according to a given strategy (depth first, random choice, …) and an instant for this transition firing is determined (minimum, maximum, mean, random value); every clocks $h \subset C(T_S)$ are then updated or reset if they belong to Z(t).

- S is an entry state for a partial model; every clock variable representing partial model entry time must be updated (in example illustrated by Fig. 5 and 6, these variables are x_0, y_0 and z_0). Thanks to the Simplex method applied to the inequalities systems associated to the black box, it is possible to find which exit states are reachable and when they can be reached.

- Similarly to the precedent case, the simulation algorithm chooses a reachable exit state and its reaching time among the possibilities given by Simplex method.

This mechanism stops in two cases:

- When, from a current state, neither any transition can be fired nor any partial model exit state can be reached; if the system is not blocked in one of its terminal states, we identify a livelock and we prove that the modelled system is not correct.

- When the simulation duration is reached; in this case we conclude that the system is correct according to the simulation scenario. As we said about simulation, results are strictly related to the scenario simulation.

7 Conclusions and future trends

In this paper, we extend the hybrid method combining simulation with exhaustive analysis. We propose an algorithm, which perform a partition of the whole state machine. Each partition satisfies the black box criteria. Therefore, it is possible to realize a partial validation on each part. The result of this computation leads to an abstraction of the black box with regard to the whole state machine. Such validation can be parallelized in order to improve the execution time. Obviously, performances of the proposed algorithm depend on the given automate.
A program corresponding to the algorithm has been developed and it is under test into a tool prototype called XTIOSM. We plan to generalize the presented concepts in order to develop a library of re-usable model of partial systems that are exhaustively "pre-analysed". Using existing models, the partition algorithm can help us to identify the meaningful components and the inter-connection patterns. The use of temporal characteristics as criteria for partitioning is under study.

References

[ACH+95] Alur R., Courcoubetis C., Halbwachs N., Henzinger T.A., Ho P.H., Nicollin X., Olivero A., Sifakis J., Yovine, *The Algorithmic Analysis of Hybrid System*, Theorical Computer Science, 138(1), pp. 3-34, 1995.

[AD94] Alur R., D. A. Dill (1994). *The theory of Timed Automata*. In: Theorical Computer Science, vol.126, N°2.

[AD94a] Alur R., D. A. Dill (1994). *The theory of Timed Automata*. In: Theorical Computer Science, vol.126, N°2.

[BCL93] Bremond-Gregoire P., Choi J.Y., Lee I. (1993), *The soundness and completeness of ACSR*, in Technical report MS-CIS-93-59, Univeristy of Pennsylvania, USA.

[BD91] Berthomieu, B., M. Diaz (1991). *Modeling and Verification of Time Dependant Systems Using Petri Nets*. In: IEEE Transactions of Software Engineering, 17(3) :259 :273.

[BDM+98] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. *KRONOS: a model-checking tool for real-time systems*. In Computer Aided Verification, CAV'98, 1998

[BGK+01] Bozga M., Graf S., Kerbrat A., Mounier L., Ober I.,Vincent D. (2001) *Timed Extensions for SDL* in LNCS "SDL Forum 2001"

[FJJV96] Fernandez, J-C., C. Jard, T. Jéron, C. Viho (1996). *Using on-the-fly verification techniques for the generation of test suites*. In : *CAV'96*, LNCS 1102, Springer Verlag.

[HSK02] Haar S., Simonot-Lion F., Kaiser L. *Equivalence of timed state machines and safe TPN*, In 6th International Workshop on Discrete Event Systems (WODES'02), Zargoza (Spain), October 2002

[Kai01] Kaiser, L., *Contribution à l'analyse des TIOSMs pour la vérification de propriétés temporelles de systèmes complexes*, Thèse de doctorat de l'Institut National Polytechnique de Lorraine, 2001.

[KC95] Koné, O., R. Castanet (1995). *Conformance with Time Extensions*, Technical Report, Université de Bordeaux

[KS00] Kaiser, L., F. Simonot-Lion (2000b). *Adaptative timed tests for temporal interoperability verification*. In 3rd IEEE International Workshop on Factory Communication Systems - WFCS2000, Porto, Portugal

[KS01] Kaiser L., Simonot-Lion F. *An Hybrid Method for the Validation of Real-time Systems*, in Proceedings 4th IFAC International Conference on Fieldbus systems an their Applications, FeT'2001,p. 210-217, Nancy (France), November 2001

[KSK00] Kaiser, L., F. Simonot-Lion, O. Kone (2000a). *Verification method of interoperability for real time systems*. In : S4th IFAC International Symposium on Intelligent Components and Instruments for ControlApplications - SICICA'2000, Buenos Aires, Argentine

[Lau99] Laurençot P., *Intégration du temps dans les tests de protocoles de communication*, Thèse de doctorat de l'Université de Bordeaux,1999.

[MF76] Merlin, P.M., D.J. Farber (1976). *Recoverability of Communication Protocols – Implications of a Theorical Study*. In: IEEE Transactions on Communications, 24(9): 1036-103.

[Ost89] Ostroff J. (1989), *Temporal Logic for Real Time Systems*, in Research Studies Press LTD, England.

[PL00] Paul Pettersson, Kim G. Larsen, *UPPAAL2k*, in Bulletin of the European Association for Theoretical Computer Science, volume 70, pages 40-44, 2000

[RR87] Reed G., Roscoe A. (1987), *Metric spaces as models for real-time concurrency*, In Proc. Mathematical Foundations of Computer Science, New York, USA.

[TST97] Toussaint J., Simonot-Lion F., Thomesse J.-P., *Time Constraint Verification Methods Based on Time Petri Nets*, in Proceedings 6th Workshop on Future Trends in Distributed Computing Systems, FTDCS'97, p. 262-267, Tunis (Tunisie), Octobre 1997.