REDUCING TIMED AUTOMATA: A NEW APPROACH

Ilham KITOUNI, Hiba HACHICHI, Kenza BOUAROUDJ, Djamel Eddine SAIDOUNI

MISC Laboratory, University Mentouri, Constantine, 25000, Algeria
{kitouni,hachichi,bouaroudj,saidouni}@misc-umc.org

ABSTRACT

Today model checking is the most useful verification method for real time systems, so there is a serious need for improving its efficiency with respect to both time and resources. In this paper we present a new approach for reducing timed automata. In fact regions of a region automaton are aggregated according to a coarse equivalence class partitioning based on traces. We will show that the proposed algorithm terminates and preserves original timed automaton. Proposed algorithms are implemented by model transformation with Atom3 tool.

KEYWORDS

Real time systems, Timed automata, Region automata, model transformation, ATOM3

1. INTRODUCTION

Nowadays, technology is looking for distributed applications to develop and increase its domains (network, telecommunication…etc). This kind of applications is known by their complexity. Formal modelling based methods are the most used technique to deal with concurrent and distributed systems because of their ability to describe system behaviour without ambiguity.
It is well known that the quality of validation techniques and results depend on the quality of the models used for specification. A particular interest is given to these formalisms for their aptitude to be implemented in real world.

The model-checking consists to verify certain properties for the desired system. This later is modelled by timed automata. This technique has proven its efficiency over several years by validating protocols and circuits. In practice, due the consideration of real time quantitative aspect; verification and validation algorithms are hard to implement. These algorithms are based on region automata; they have been used for solving several problems like automaton emptiness, system supervising and system testing [9][11][18] [25][15][13].

Recall that the execution of a timed automaton is infinite. The idea of region automaton consists of partitioning the states space into finite regions, thus the graph which imitates the behavior of the initial automaton is constructed. The states of the automaton belonging to the same region are equivalent according to a well-defined relation (relation on clocks valuation).

Nevertheless, the complexity of implementing region automata is exponential with respect to both clocks number and length timing constraints [1][7][8][20][27].

DOI : 10.5121/ijist.2012.2402

In this context, we define an aggregation operation on region automaton localities using a Bi-relation which can exist between them. This equivalence relation is used for grouping equivalent regions. It is obvious that this aggregation reduces the number of graph localities. For this purpose we propose an algorithm implementing the aggregation relation starting from an initial partitioning of localities. The generated aggregated regions automaton preserves the reachability property. Consequently the reachability question on timed automaton states is reduced to the reachability question on the aggregated timed automaton. Therefore the language recognized by both automata is the same. In [2], the timed automata based model-checking of TCTL logic properties has been proven PSPACE complete. Since the aggregation algorithm reduces the region, the complexity doesn't increase. To experiment our approach, the proposed algorithm is implemented using model transformation based approaches.

The rest of the paper is organized as follows: after presenting related work in section 2, timed automata and region automata are presented in sections 3 and 4 respectively. In sections 5 we develop some intermediaries' results leading to the aggregation algorithm. Section 6 presents the algorithm. In section 7 we present the implementation and example to illustrate our approach. Finally we conclude the paper in section 8.

2. RELATED WORKS

Several works related to the minimization of timed automata have been proposed for avoiding the state space explosion problem. In [17], it has been shown that the formalization is able to identify symbolically timed properties from the timed automata. While in other work, the states space of the timed automata is reduced by a minimization during the construction of the regions graph, using a bissimulation relation on clock regions [2][26]. A minimization of timed automata resources, such as the clocks number or the constraints size by eliminating the inactive clocks is considered in [11]. It is based on a static analysis technique; however, it is judged to be not powerful for addressing the undecidability problem of timed automata. In [28], the reduction of the clocks number or constraints size can be made algorithmically. More recent work shows that the undecidability problem persists [14].

On the other hand, other works suggest the concept of grid. In fact, the grid covers the underlying dense time space of timed automata, mapping points in states space into a single representative of each grid region. The chosen grid size of a timed automaton is an integer number. The grid automaton G(A;d) is defined as a sub automaton of a timed automaton A that only contains clock valuations that are multiples of d (where $0<d<1$). G(A;d) represents a discrete version of A with discretization step d. The size of grid depends on the clocks number and the size of states set in the automaton. However, since the number of regions in real specifications is very large, it should be clear that the step size becomes small. Consequently, the algorithm which is theoretically exhaustive, is highly impractical [19][23]. The minimization principle of previous approaches is based on clock regions handling, to go off initial fine-grained regions, in order to achieve more compact regions. In contrast to this, our proposal is based on the actions labeling transitions, consequently reasoning on the traces and the reachable states of the Timed Automata.

Even the idea seems simple; it has the advantage to be compatible with previous approaches. Indeed, under certain conditions, reduction can be spectacular. Obviously, our approach may be used together with previous.

3. TIMED AUTOMATA

Timed automata have been introduced in 90's for modeling real-time systems. It is a finite untimed automaton to which is associated a finite set of positive real-valued variables said clocks [3][7].

Let H be a finite set of clocks, we assume the time domain be the set R^+ of positive real numbers. We note Φ a finite conjunction of constraints of the form $x \gg k$ where $x \in H$, $k \in Q^+$ and $\gg \in \{\leq, <, =, >, \geq\}$.

A valuation v over H is a mapping $v : H$ *to* R^+. Let h be a subset of clocks, the valuation $v[h \longleftarrow 0]$ resets each clock of h to 0, i.e. maps each clock x in h to 0, and each other clock x to $v(x)$. Let d be a positive real, the valuation $v+d$ maps every clock x to $v(x) + d$. The constraints are interpreted over valuations; we write $v \models g$ if the valuation v satisfies the clock constraint g. It is defined in a natural way by $v \models x \gg c$ whenever $v(x) \gg c$ and $v \models (g1 \wedge g2)$ whenever $v \models g1$ and $v \models g2$.

Definition 3.1. *Let $\sum$ be a finite alphabet, a timed automaton over $\sum$ is a tuple $A = (S, s_0, H, T, S_F)$ where S is a finite set of states, s_0 in S is the initial state, H is a finite set of clocks, $T \in S \times \Phi \times \Sigma \times 2^H \times S$ is a finite set of edges, S_F is the set of final states.*

A transition $(s,g,a,h,s') \in T$ represents a change of location from $s \in S$ to $s' \in S$ on symbol $a \in \sum$. The clock constraint $g \in \Phi$ (guard) specifies when the transition is enabled and the set $h \subseteq H$ gives the set of clocks to be reset by this transition.

The semantic of timed automata is given as timed transition systems. Let A be a timed automaton over $\sum$, the corresponding timed transition system is $SA = (Q, q_0, \rightarrow)$ where:

- $Q = S \times R^+$ is the set of states also called configurations,
- $q_0 = (s_0, 0)$ is the initial state,
- The transition relation is composed of the following moves: *Delay moves*: $(s,v) \xrightarrow{d} (s, v+d)$ for every d in R^+ and *Discrete moves*: $(s,v) \xrightarrow{a} (s', v')$ iff there exist some transition (s, g, a, h, s') $\in$ T such that $v \models g$ and $v' = v[h \leftarrow 0]$.

In practice, several models are based on timed automata. We reference the classical timed automata of Alur and Dill and some extensions of this model.

4. REGION AUTOMATA

The region automata are the automata which reproduce the infinite execution of timed automata by a finite set of transitions. Also it is well known that in the verification by model-checking, testing and supervision the region automata are very used because they allow de-timing the specification. We resume in this section the classical definition of region automata [1].

4.1 Clock Regions

The valuations of a finite set of clocks are a region, such as from two valuations of the same region, the same transitions are enabled.

Definition4.1. *Let A be a timed automaton, the set of standard regions, Ω of A is the set of equivalence classes' relation, noted≡. This relation is defined over the clocks valuations as follows:*

$$v \equiv v' \text{ if } \forall (x, y) \in H$$
$$\begin{cases} \lfloor v(x) \rfloor = \lfloor v'(x) \rfloor \text{ where } (v(x) > M_x \Leftrightarrow v'(x) > M_x) \\ \left(v(x) \le M_x \text{ and } v'(y) \le M_y \right) \Rightarrow \\ \left(frac(v(x)) \le frac(v(y)) \Leftrightarrow frac(v'(x)) \le frac(v'(y)) \right) \end{cases} \quad (1)$$

M_x is the maximum constants appearing in the constraints on clock x and for any real c, $\lfloor c \rfloor$ denotes the integral part of c, *frac(c)* denotes the fractional part of c.

For example, we consider a set H of two clocks x and y where $M_x = 3$ and $M_y = 1$. So we have 38 regions, Figure 1.

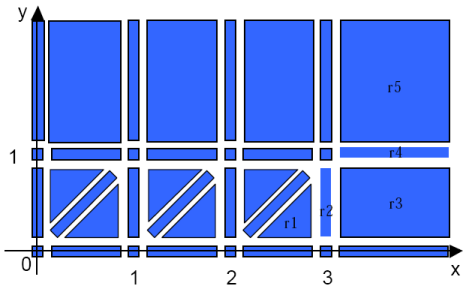


Figure 1. Standard regions.

Properties: Two valuations of the same region must satisfy the same constraints. The relation ≡ satisfies the following property:

$$v \equiv v' \text{ then for evry } g \text{ in } A : v \models G \Leftrightarrow v' \models G \text{ and} \quad (2)$$
$$\forall c \in R^+, \exists c' \in R^+ \text{such as } v + c \equiv v' + c'$$

The equivalence of regions is compatible with the variation of time; it is therefore possible to define a successor function on regions. We note succ(r), all successors of the region r by lapsing time. All of regions r satisfies the following property: $r' \in succ(r) \Leftrightarrow \exists v \in r. \exists c \in \mathrm{R}^+$ such as $v + c \in r'$. Finally, we note that any clock region can be represented as: {x = k: (k = 0,1, ... , M_x) or k-1 <x < k: (k = 1, ... , M_x) or k <x: (k = M_x)}.

Example: The successors of r1 [(2 <x <3),(0 <y <x <1)] are: r2 [(x=3), (0<y<1)], r3 [(x>3), (0<y<1)], r4 [(x>3), (y=1)], r5 [(x>3), (y>1)].

4.2 Region Automaton

Definition 4.2. *Let A = (S, s_0, H, T, S_F) be a timed automaton defined by a timed transition system. The region automaton RA(A) = (L, l_0, TR) over ∑corresponding to A is a finite automaton defined as follows: all localities of RA(A) are of the form l_{ij} = (s_i, r_j) where s_i is a state and r_j is a clock region. The initial locality is l_0= (s_0, r_0). The set of localities is noted L. The set of transitions TR is,*

$$T_R = \left\{ t'/t' = (s,r) \xrightarrow{a} (s',r') \middle| \left\{ \begin{array}{c} \exists s \xrightarrow{g,a,h} s' \in T \text{ and } \exists r'' \in succ(r) \\ \text{such as } r \subseteq g \text{ and } r' = r''[h \leftarrow 0] \end{array} \right\} \right. \quad (3)$$

$$l_{ij}^{F} = (s_i, r_j) \text{ is a terminallocalityiff } s \in S_F \quad (4)$$

5. AGGREGATION PRINCIPLE

In this section, we define an aggregation operation, which can be considered as a particular determinization. The indeterminism considered here is due to the construction of the initial region automaton [15], so it is not inherent to the system.

The principle of the aggregation operation consists to find and regroup localities which verify the following conditions:

- For any input transition of a locality l_1 there exist input transitions to a locality l_2, where their outgoing are coming from the same source locality and have the same label. This second transition is called "*forward mirror*" of the first.
- For any output transition of a locality l_1 there exist output transitions from a locality l_2 where their outgoing goes to the same target locality and have the same label. This second transition is called "*backward mirror*" of the first.
- Localities are grouped if their sets "*forward mirror*" and "*backward mirror*" matches.

5.1 Notations and Definitions

Let τ_i be a transition of a timed transition system. $\alpha(\tau_i)$ (resp. $\beta(\tau_i)$) denotes the source locality of τ_i (resp. the target locality of τ_i); the label of the transition τ_i is given by $\lambda(\tau_i)$.

Definition 5.1. *The set of the input transition in a locality l_{ij} is $in(l_{ij}) = \{\tau_j \mid \beta(\tau_i) = l_{ij}\}$. The set of the output transition from a locality l_{ij} is $out(l_{ij}) = \{\tau_j \mid \alpha(\tau_i) = l_{ij}\}$.*

Definition 5.2. *"forward mirror" is a set l_{ij} of localities defined as follows:*

$$FM(l_{ij}) = \left\{ l_{ik} / \forall \tau \in in(l_{ij}), \exists \tau' \in in(l_{ik}) \text{ such as } \begin{array}{l} \lambda(\tau') = \lambda(\tau) \wedge \alpha(\tau') = \alpha(\tau) \end{array} \right\} \quad (5)$$

Definition 5.3. *"backward mirror" is a set l_{ij} of localities defined as follows:*

$$BM(l_{ij}) = \left\{ l_{ik} / \forall \tau \in out(l_{ij}), \exists \tau' \in out(l_{ik}) \text{ such as } \begin{array}{l} \lambda(\tau') = \lambda(\tau) \wedge \beta(\tau') = \beta(\tau) \end{array} \right\} \quad (6)$$

Definition 5.4. A set of grouped localities with $l_{ii.}$ noted $RL(l_{ij})$, is defined as follows:

$$RL(l_{ij}) = \left\{ l_{ik} / \begin{array}{l} l_{ik} \in FM(L_{ij}) \text{ if } l_{ik} \text{ is a terminal locality} \\ l_{ik} \in FM(l_{ij}) \cap BM(l_{ij}) \quad \text{elsewhere} \end{array} \right\} \quad (7)$$

Definition 5.5. *IR is a grouping relation on localities; it is defined as follows:*

$$\forall x, y \in L / (x,y) \in IR \text{ iff } y \in RL(x). \quad (8)$$

The Bi-relation defined above is an equivalence relation, which allows defining the equivalence classes of the grouped localities. Indeed, IR is a reflexive relation: any locality *x* is grouped with itself because it has the same "forward mirror" and "backward mirror" as *x*. If *x* is grouped with *y* then *x* has the same "forward mirror" and "backward mirror" as *y*: It is evident that *y* has the same "forward mirror" and "backward mirror" as *x* according to definitions above, it is thus symmetrical. *(x, y)* are grouped and *(y, z)* are grouped, by definitions, *(x, z)* are grouped then the relation is transitive. □

Definition 5.6. *The summation of clocks regions is an operation defined on the set of regions as follows:*

$$\hat{r} = \oplus(r_1, r_2, ...r_k) = \oplus_k(r_k) = \begin{cases} r \quad if \quad k = 1 \\ r_1 \cup r_2 \cup ... \cup r_k \ if \ k > 1 \end{cases} \tag{9}$$

on a clock *x* specifying a clock region is extended by the following form: $\{\alpha \le x \le \beta : (\alpha < \beta < M_x$ The union operation $\cup$ is defined like the union on integer intervals. Note that the form of constraints)}. The new construction is a region; it follows the same semantic as region initially considered.

Definition 5.7. *The Aggregated regions automaton $AR_A(A)=(L^A, l_0^A, T_{AR})$ of the timed automaton A is a transition system over the alphabet $\sum$ defined as follows: The localities of ARA(A) are of the form $(s,\hat{r})$ where $s \in S$ and $\hat{r}$ is a region, the initial locality is of the form $(s_0,\hat{r}_0)$ such as $\hat{r}_0=r_0$ and the set of transitions T_{AR} is:*

$T_{AR} = T_R - T_{SP}$ where T_{SP} is the set of redundant transitions resulting after the aggregation operation.

6. AGGREGATION ALGORITHM

The aggregation operation consists to find localities to be grouped and their replacement by a new locality. This locality has the same name state; its region is the summation of regions of grouped localities; the entering and exiting transitions are substituted with new transitions linking this new locality to the graph.

Given a region automaton *RA(A)= (L, l_0, TR)* over $\sum$, the following algorithm is used for grouping localities according to the equivalence relation defined above.
Here after, the following notations are considered:

- Π is a partition of *L*.
- Π_0 the initial partition; it is composed by singletons (elements of L). The function *frag* that fragments classes P in subclasses formed by singletons.

$$\delta_\alpha^{-1}(P) = \{Q, Q \xrightarrow{\alpha} P \text{ and } \alpha \in \Sigma\} \tag{10}$$

$$\delta_\alpha(P) = \{Q, P \xrightarrow{\alpha} Q \ and \ \alpha \in \Sigma\} \tag{11}$$

- Function Φ checks if a class Q can be grouped with the class P.

$$\Phi_{P,Q}\left[\begin{array}{c} \text{for each } \alpha,\alpha' \in \Sigma \text{ and } P,Q \text{ Classes} \\ \text{if } \delta_\alpha^{-1}(P) = \delta_\alpha^{-1}(Q) \wedge \delta_{\alpha'}(P) = \delta_{\alpha'}(Q) \text{ then} \\ (P,Q) \text{ are grouped} \end{array}\right. \tag{12}$$

- The function $\text{Reg}(p,q) = k$ where $s_k = s_p = s_q$ and $\hat{r}_k = r_p \oplus r_q$ (13)
- The case of terminal locations is implicitly considered $\delta_\alpha(P) = \delta_\alpha(Q) = \phi$ (14)

6.1 Algorithm

About algorithm, K is a set of grouped localities at each step; at the end Π contains localities of the aggregated regions automaton; the termination property of the algorithm is ensured by the finite number of regions in the initial graph. To implement this algorithm we only need an efficient representation of regions and perform simple operations such as summation over regions.

Π_0 = initial partition, $X = \Pi_0, \Pi = \phi$
repeat
choose l_{ij} from X and mark it
$K \leftarrow l_{ij}$
for l_{ik} in X
if $\Phi_{l_{ij}, l_{ik}}$ then $K := \mathrm{Reg}(K, l_{ik})$
$\Pi := \Pi \cup \{K\}$
$X := (X / \mathrm{frag}(K)) \cup K$
until marking all the elements of X

Example 1: The example (Figure 2. and Figure 3.a) present a timed automaton *A* and the associated region graph.

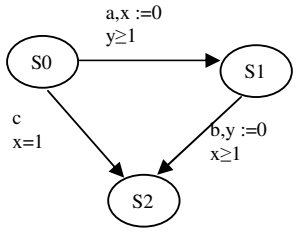
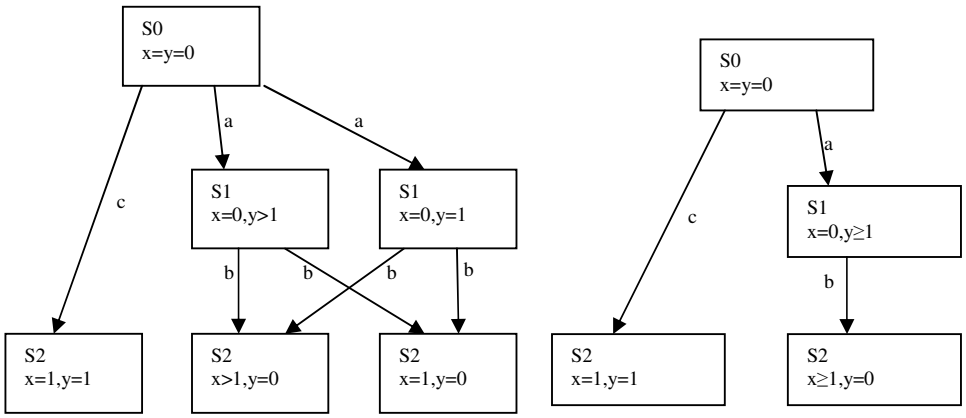


Figure 2. Timed automata A.



(a) (b)

Figure 3 (a) Region graph and (b) Aggregated regions graph associated to A.

Figure 3.b, shows the region graph after applying the regions aggregation algorithm.

6.2 Equivalence and Validation of Systems

In this subsection we show that the behavior of a timed automaton is a homomorphic image of the behavior of its aggregated regions automaton. With this result we claim that the validation operations, particularly the test, generated from the aggregated regions automaton, is sufficient for assessing an implementation of the timed automaton.

A homomorphism between two sets is a mapping of the elements of one set to the other such that their respective binary operations are preserved: $H:(A,\circ)\to(B,*)$. Such that *H(x)* is an element of *B* and for any pair x_1, x_2 in *A*, $H(x_1 \circ x_2) = H(x_1) * H(x_2)$. (15)

In fact, the proposed algorithm constructs an homomorphism H from $AR_A(A)$ to *A*, mapping the states of $AR_A(A)$ to the states of *A* while preserving the transition relations over the same actions. To detail this we define a projection over the set of actions on transitions as:

$$\prod_{\Sigma} : T_A \to T'_A, \prod_{\Sigma}(\tau) = \prod_{\Sigma}(s,g,a,\gamma,s') = (s,a,s') = \tau' \tag{16}$$

$H : AR_A(A) \to A$,has the following properties:

$$\text{For all } l_{ij}^A \in L^A, H(l_{ij}^A) = s_i \in S \quad \text{such that } H(l_0^A) = s_0 \tag{17}$$

$$\text{And for all transitions: } \lambda \in T_{AR}, \lambda = (l_{ij}^A, a, l_{kt}^A), H(\lambda) = (H(l_{ij}^A), a, H(l_{kt}^A)) \in \prod_{\Sigma}(T) \tag{18}$$

H maps the action of λ to the action of τ in *T*. The mapping *H* can be extended in the same way to map computational paths. A computational path in an aggregated regions automaton corresponds to sequence of transitions starting from the initial locality of the automaton. That is, *H* can be extended to map computations. $Exec(A)$ is a set of all computation of timed automaton *A*.

$$H^+ : Exec(AR_A(A)) \to Exec(A) \tag{19}$$

$$H^+(\lambda_0\lambda_1\lambda_2 ..... \lambda_J .....) = H(\lambda_0)H(\lambda_1)H(\lambda_2)....H(\lambda_J).... \tag{20}$$

These results are summarized as follows: For every computational path in the aggregated regions automaton $AR_A(A)$ there is a computational path in the timed automaton *A*. The behaviour of *A* is a homomorphic image of the behaviour of it's aggregated regions automaton $AR_A(A)$.

Since the localities of the Aggregated regions automaton $AR_A(A)$ include clock regions (inherent from region automaton and more), which are the equivalence classes of regrouped clock valuations, the timed behavior of *A* is simulated by the Aggregated regions automaton. Hence, we claim that the validation methods based on region automata and digitization of their state spaces are renewed by aggregation regions automata and optimized in the above sense.

Example 2: By changing clocks constraints in the previous example, the regions automaton is now composed by 7 regions (Figure 4.) The aggregated graph has 4 regions (Figure 5.).

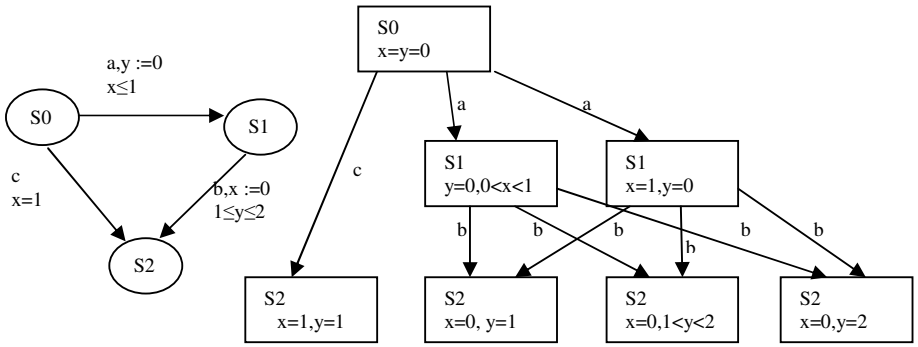


Figure 4. Timed automaton B and the associated region graph.

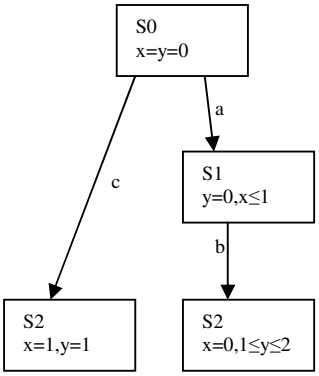


Figure 5. Aggregated region graph ARA(B).

7. IMPLEMENTATION

In this part we present an experimentation of our approach applied to particular kind of timed automata model named Durational Actions Timed Automata DATA*. The DATA* model [21][6][22] is a sub class of timed automata which takes into account the duration of actions. It's based on an intuitive idea: temporal and structural non-atomicity of actions. The example below presents the durational actions automaton model:

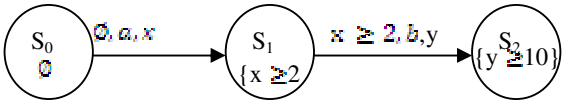


Figure 6. DATA* (a).

We present also, an approach to transform DATA* model into aggregate regions automata using graph transformation [16].

The graph transformation is a process that converts a model to another model. This task requires a set of rules that define how the source model has to be analyzed and transformed into other elements of the target model. Graph Grammars are used for model transformation [5][10]. They are composed of production rules; each having graphs in their left and right hand sides (LHS and RHS) (Figure 7). AToM3 [12] is a graph transformation tool among others. In this paper we use it.

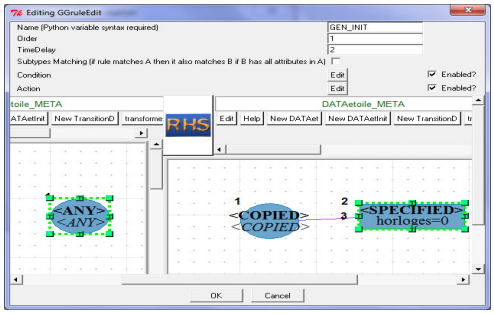


Figure 7. Example of grammar rule in AToM3 (LHS and RHS).

Example3: To illustrate our approach we propose the example of the ticket reservation system "TRS". This example supposes that to buy a ticket, we generally pass by two counters. The first counter R is for making a reservation and the second counter C is for paying and taking the ticket. This agency has one waiting room, three counters of type R and two of type C. On arrival, the client goes to the waiting room, when a counter of type R is free, he can make a reservation. Once the operation is complete, he waits until a counter C becomes free for paying and taking the ticket.

Figure 8. presents a DATA* of TRS for two clients with the graph editor dotty. The mapping of this DATA* with the graph editor dotty to the equivalent DATA* model in AToM3 syntax (Figure 9.) is performed using python program. We have applied our tool on the DATA* model and obtained automatically the aggregate region automaton of Figure 10. The result is saved in the text file of Figure 11.

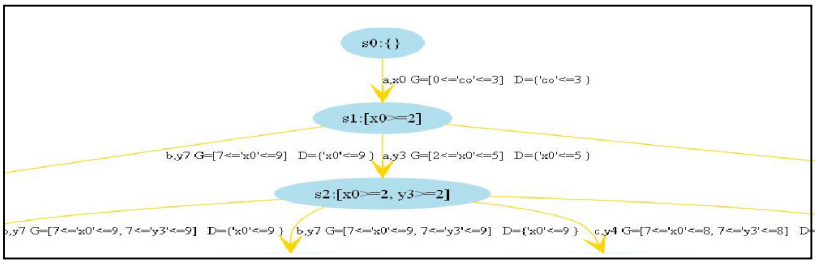


Figure 8. DATA* of TRS with the graph editor dotty.

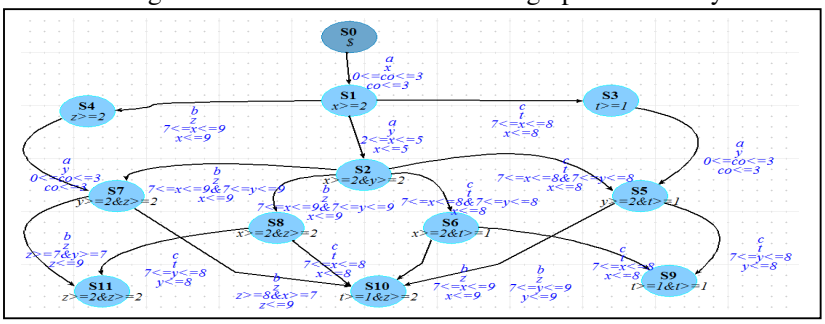


Figure 9. DATA* of TRS with AToM3.

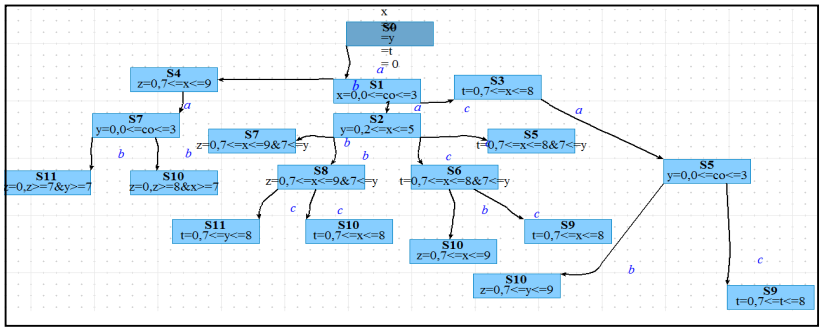


Figure 10. Aggregate region automaton.

```
******ARM_MDL******
les transition sont:
        S0--a-->S1
        S1--b-->S4
        S4--a-->S7
        S1--c-->S3
```

Figure 11. A textual aggregate region automaton.

8. CONCLUSION AND OUTLOOK

In this paper we proposed an algorithm for reducing region automaton. For this purpose we defined an equivalence relation among the regions. The proposed aggregation relation preserves language recognized by the original automaton. As perspectives it seems interesting to develop a theory of model-checking to verify properties specified in the TCTL logic on aggregated regions graph. An alternative to the proposed algorithm can be to aggregate localities on the fly, which consists for building such graph without constructing the full region graph (i.e. at the moment of generation of R.G associated to the automaton). It is possible because the algorithm requires only three levels of the initial regions graph. Finally this work can be applied in the test by a way of partial determinization of the automaton. This is in the same direction of recent promising work [24][4].

For illustration, we propose a method for generating an aggregated regions automaton from a Durational actions timed automata by the graph transformation approach and using the environment AToM3 in order to provide a finite abstraction of DATA* structures with a high number of states.

Firstly, we have proposed a program written in python language that transforms a DATA* structure, presented as a dotty file, to a DATA* structure written in the form of a python file respecting the syntax of AToM3. The meta-modeling tool AToM3 is used for this purpose. We have illustrated our approach through an example. In future work, we plan to implement our approach with other tools as AGG in order to compare performances. We plan also to Study the complexity of this transformation and its use in system testing.

REFERENCES

[1] Alur, R.and Dill, D. (1994): the theory of timed automata. Theoretical Computer Science, 126(2):183.

[2] Alur, R. Courcoubetis, C. Halbwachs, N. Dill, D. and Wong-Toi, H. (1992): Minimization of timed transition systems, in: 3rd Conference on Concurrency Theory CONCUR'92, in: Lecture Notes in Computer Science, vol. 630, Springer-Verlag, Berlin, pp. 340– 354.

[3] Alur, R. Courcoubetis C. and Henzinger, T.A. (1994): The Observational Power of Clocks : in the Proceedings of the International Conference on Concurrency Theory (CONCUR 94), Lecture Notes in Computer Science 836, Springer-Verlag, 1994, pp. 162-177.

[4] Baier, C. Bertrand, N. Bouyer, P. and Brihaye, T. (2009): When are timed automata determinizable? Seminaire LaBRI.

[5] Baresi, L., Hekel, R. (2004): Tutorial Introduction to graph transformation. A software Engineering perspective, Lecture Notes in Co0mputer Science, Volume 3256/2004, Springer Berlin, pp.431-433.

[6] Belala, N. (2010) : Modèles de Temps et leur Intérêt à la Vérification Formelle des Systèmes Temps-Réel. PHD's thesis, Mentouri University, 25000 Constantine, Algeria.

[7] Bouyer, P. (2004): Forward Analysis of Updatable Timed Automata, Formal Methods in System Design, vol. 24, n°3, p. 281–320.

[8] Bouyer, P. (2002): Modèles et Algorithmes pour la Vérification des Systèmes Temporisés, PhD thesis, Laboratoire Spécification et Vérification – CNRS UMR 8643 & ENS de Cachan 61, avenue du Président Wilson – 94230 Cachan – France.

[9] Bouyer, P. D'Souza, D. Madhusudan, P. and Petit, A. (2003): Timed control with partial observability, in: CAV'03.

[10] Czarecki, K., Helsen, S. (2006): Feature-based survey of model transformation approaches. IBM SYSTEMS JOURNAL, VOL 45, NO 3.

[11] Daws, C.and Yovine, S. (1996): Reducing the number of clock variables of timed automata, in: Proc. 17th IEEE Real-Time Systems Symposium, RTSS'96.

[12] De Lara, J., Vangheluwe, H. (2002): AToM3: A Tool for Multi-Formalism Modeling and Meta-Modeling. Proc. Fundamental Approaches to Software Engineering, FASE'02, Vol. 2306. LNCS. Grenoble, France, pp. 174-188.

[13] D'Souza, D. and Madhusudan, P. (2002): Timed control synthesis for external specifications, in: STACS'0 and Lecture Notes in Computer Science, vol. 2285, Springer, Berlin.

[14] Finkel, O. (2005): On decision problems for timed automata, Bulletin of the European Association for Theoretical Computer Science vol. 87 p. 185–190.

[15] Gouin, A. Libeaut, l. And ferrier, J.I. (1999): Supervisory Control of Timed Automata, Proc. ECC'99, Karlsruhe-Allemagne.

[16] Hachichi, H., Kitouni, I., Saïdouni, D. E.(2011) : A Graph Grammar Approach for calculation of Aggregate Regions Automata. The International Arab Conference on Information Technology (ACIT).

[17] Henzinger, T. Nicollin, X. Sifakis, J. and Yovine, S. (1992): Symbolic model-checking for real-time systems. In Proceedings of the Seventh IEEE Symposium on Logic in Computer Science, p. 394-406.

[18] Krichen, M. and Tripakis, S. (2004): Black-box conformance testing for real-time systems. In 11th International SPIN Workshop, Barcelona, Spain, LNCS No 2989. Springer.

[19] Mao Zheng, M. Alagar V.and Ormandjieva, O. (2008): Automated generation of test suites from formal specifications of real-time reactive systems In The Journal of Systems and Software vol. 81 p. 286–304.

[20] Ouaknine, J. and Worrell, J. (2007): On the decidability and complexity of metric temporal logic over finite words. Logical Methods in Computer Science, 3(1:8).

[21] Saïdouni, D. E., Belala, N. (2006) : Actions duration in timed models. The International Arab Conference on Information Technology (ACIT).

[22] Saïdouni, D. E., Kitouni, I., Hachichi, H. (2011). Formalization of aggregated regions timed automata whith durational actions. MISC REPORT 11001. Mentouri University, 25000 Constantine, Algeria.

[23] Springintveld, J. Vaandrager, F. and D'Argenio, P. (2001): Testing timed automata. Theoretical Computer Science, 254.

[24] Stainer, A. (2010) : Test d'automates temporisées, équipe VerTeCs, INRIA. Master en informatique à Rennes.

[25] Tripakis A. and Altisen, K. (2005): Implementation of timed automata: An issue of semantics or modeling. Technical Report Nr TR-2005, Verimag, Centre équation, 38610 Gières, France.

[26] Tripakis S.and Yovine, S. (2001): Analysis of timed systems using time abstracting bisimulations, Formal Methods in System Design vol. 18 (1) p. 25–68.

[27] Tripakis, S. (2004): Folk theorems on the determinization and minimization of timed automata, in: Formal Modeling and Analysis of Timed Systems (FORMATS'03), in: Lecture Notes in Computer Science, vol. 2791, Springer, Berlin.

[28] Wilke, T. (1994): Automaten und Logiken zur beschreibung zeitabhängiger Systeme, "Automata and logic to describe the time-dependent systems" Ph.D. thesis, Institut Für Informatik und Praktische Mathematik, Christian-Albrechts Universität, Kiel, (in German).

Authors

Ilham Kitouni obtained her BEng degree from University of Mentouri Constantine, Algeria, in 1992, after 15 years in different Algerian company as head of department of Computer Sciences, she recovers CFSC research group of MISC laboratory, Mentouri University of Constantine, Algeria. From October 2009, she prepares a PhD thesis. Her research domain is formal models for real-time systems specification and validation.

Hiba Hachichi received her master's degree in computing sciences from University of Mentouri Constantine, Algeria in 2009. Currently, she is a PhD student at CFSC research group of MISC laboratory, Mentouri University of Constantine, Algeria. Her research interests are graph transformation and formal methods for verifying and testing real time systems.

Kenza Bouaroudj received her master's degree in computing sciences from University of Mentouri Constantine, Algeria in 2010. Currently, she is a PhD student at CFSC research group of MISC laboratory, Mentouri University of Constantine, Algeria. Her research interests are system validation and testing real-time stochastic systems.

Djamel-eddine Saidouni received his PHD degree in theoretical computer science from the university Paul Sabatier of Toulouse, France in 1996. Actually he is a professor at the department of computer science, Mentouri University of Constantine, Algeria. Also, he is the head of the CFSC research group of MISC laboratory. His main research domain interests formal models for specifying and verifying critical systems, real time systems, true concurrency models and state space explosion problem.