International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 4– No2., September 2012 – www.ijais.org

Durational Actions Timed Automata: Determinization and Expressiveness

Ilham Kitouni
MISC Laboratory,
Mentouri University
Constantine, 25000
Algeria.

Hiba Hachichi
MISC Laboratory,
Mentouri University
Constantine, 25000
Algeria.

Kenza Bouaroudj
MISC Laboratory,
Mentouri University
Constantine, 25000
Algeria.

Djamel-Eddine Saidouni
MISC Laboratory,
Mentouri University
Constantine, 25000
Algeria.

ABSTRACT
In this paper we present durational actions timed automata, DATA*, as a sub class of timed automata. In the contrast of T.A, the underling semantic of DATA* is the maximality semantics which claim that actions have durations and true concurrency is captured differently from choice. DATA* model is in one hand useful for modeling and validating reel aspects of systems. In the other hand, it is determinizable and closed under all Boolean operations. As result, the language inclusion problem is decidable. Then, we compare a durational actions timed automata to event recording automata, which is a determinizable sub class of the classical timed automata. Next, we propose a simple framework to aggregate region of DATA* for reducing its space state. This study is based on an aggregation region automata procedure to reduce the combinatorial explosion of regions. Finally, we discuss equivalence and validation of systems.

General Terms
Real time systems, Formal models, Characterization of models.

Keywords
Formal timed models, durational actions timed automata, determinizable models, expressiveness, reduction approach.

1. INTRODUCTION
Distributed applications, such as communication protocols, which apprehend temporal aspects, are characterized by their big complexity. Transition systems model have been mainly used. Modeling systems by means of a finite transition system permits effective constructions procedure for analyzing its qualitative properties, and leading to automatically manipulating system behavior.

When real-time aspects of systems are considered, quantitative timing properties have to be treated. So, the correctness of the complete system behavior may depend on the different delays. For these systems, the specification by means of classical finite transition systems is no adequate. This inadequacy can be adjusted naturally by extending classical model with suitable notion of time.

In practice the precise knowledge of the different types of timing constraints permits a quick identification of the constraints that may exist in the problems in addition to the good understanding of system behavior. Indeed it facilitates the exact specification of real time systems. A classification of the different types of timing constraints is important.

Behavioral constraints ensure that the environments of a system as well as the system are well behaved. They can further be classified into three types: *Delay Constraint*, *Deadline Constraint*, and *Duration Constraint*. The most significant example is the timed automata model. They have emerged as a standard theoretical model for real-time systems, proposed by Alur and Dill. Consequently their formal properties have been well studied.

Timed automata have been extensively studied from different facets [4],[15],[17], particularly for their use in the verification of real-time aspects. Timed automata are represented by finite state structures augmented with a finite set of real-variables named clocks. An edge is labeled by a symbol which represents action to be performed when the edge is executed.

The underlying semantic used in timed automata, is said interleaving semantics [21], so actions are instantaneous. Time can elapse in states. A set of clocks can be reset to zero on edge. At any point of time, the value of a clock is equal to the time elapsed since the last reset on it. An edge may be executed only if the current value of clocks satisfies its constraint. Constraints are temporal formula on clocks which used to manage execution of edges.

In real world systems, actions are not instantaneous, but have durations. This realistic characteristic is important in many cases. The durational actions timed automata model was proposed as an extension, which expand the timed automata model by maximality semantics. Thus it permits to drop the assumption that actions are instantaneous. In DATA* model an action can take some time to be completed.

Interleaved interpretations of concurrency are justified by assumption that all actions are atomic a direct consequence is that no two actions can occur simultaneously. In the opposite, models based on maximality semantics present concurrent actions differently from choice [21]. As an illustration, consider the following simple example depicted by Fig.1.In fact, information {x,y} on locations S_2 and S_4, in Fig.1.b make the difference and inform about the concurrent execution of actions (*a* and *b*).

1.1 Contribution
In this paper we present a durational actions timed automata model (DATA*) [3],[13],[14], which claim avoiding the assumption that actions are instantaneous. The durational actions timed automata model is constructed on classical timed automata and augmented by maximality semantics [12]. This later allows us i) *to carry durations of actions*, which is

realistic assumption for specifying in a natural way systems and ii) *to handle true concurrency*.

To model duration actions, every edge of the automaton is equipped with constraints on clocks. Implicitly constraints on the edges enclose the durations of actions, those that are already started. In addition, other real time aspects are considered (delay and deadlines). A single clock is reset on every edge in transition. When clock is reset it corresponds to the beginning. The termination of action will be capture by information on locations of the automaton, precisely on the destination location of transition, a set of temporal formula identifies current execution of actions. In reality, the duration of an action is either in the constraint of the following edge, if there is dependence between the following actions, otherwise it is in the next locations and that means: action is not over yet. This elegant way to capture the durations is the impact of the maximality semantics.

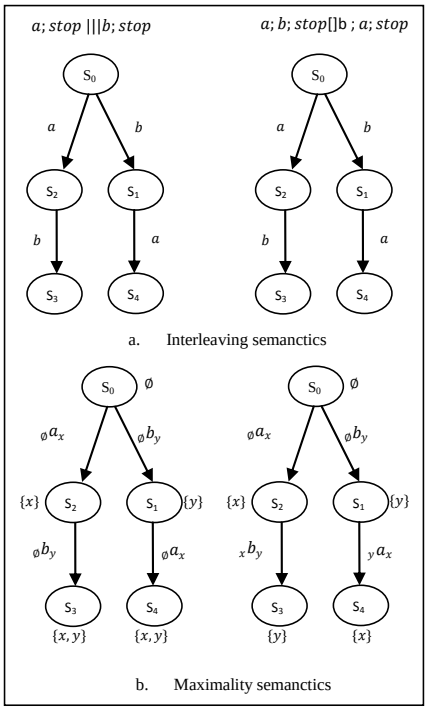

Fig 1: Representation of concurrency and choice in maximality semantics and interleaving semantics.

In this paper, we mainly focus on the decision problems and closure properties of DATA*. Hence, our aim is to characterize timed languages recognized by DATA* in terms of some suitable deterministic class of timed automata. We also show that DATA* are closed under Boolean operations.

We investigate the expressive power of DATA*, we can show that the DATA*$_0$, which are a durational actions timed automata with null durations, are expressively equivalent to Event recording automata [2]. However, the known strict inclusion of DATA*$_0$ in DATA* seems to result into a new map for inclusion in the class of timed languages.

After we give a technique of reducing regions automaton of DATA* in an aggregated regions automaton to avoid explosion state space. We show that there is a homomorphism on the behaviors of the two automata.

1.2 Related Work

Determinization is a key for implementability and indispensable in validation problems like observation, fault diagnosis and test generation [4],[19],[24]. Often, works in such domains are restricted to determinizable subclasses of timed automata, for instance, the called event-clock automata [2].

Also complementation is important for capturing the negation of logical specification by automata, particularly in automata theoretical based verification.

In this brief exploration of related work, we focus on two main concepts. First, we consider determinization and closure proprieties of timed automata classes. Second, discuss T.A class which considers duration of actions.

An interesting quest has been done to find determinizable and Boolean closed sub-classes of timed automata. In [2] it was shown that a subclass of timed automata called Event Recording Automata (ERA) can indeed be determinized with one exponential blowup in the automaton size.

The [11] work propose the Integer Reset Timed Automata with τ-transitions named τ-IRTA (τ is the internal action) which is a syntactic subclass of timed automata; It restricts clock resets to integral time points, in addition τ-transitions are allowed. τ-IRTA and ERA are expressively incomparable.

The work [7] as well as [5] have defined 1-clock alternating timed automata which are Boolean closed. Moreover the emptiness of these automata is decidable but with Non primitive-recursive complexity. In another work [4] it have been shown that emptiness checking for 1-clock TA is remain highly complex although these automata are not closed under complementation. The language inclusion and universality of 1-clock timed automata has also been studied [6],[10] and shown to be decidable with Non-primitive recursive complexity.

Our studies concern Durational actions timed automata, and their properties with large hypothesis on clocks number and constraints size.

The consideration of actions which elapse in time is not a new idea; effectively it is explored in [22]. In our knowledge it is the unique tentative to considerer explicitly of actions which elapse in time, however the meaner to materialize those quantities of time in automata is not very recent. In fact like untimed automata case, when considering non-instantaneous actions, every edge must to be divided in two edge one for the action initiation and the second for the action completion.

Timed automata with non-instantaneous actions are based on the same approach; every edge is decorated by a initiation constraint and a completion constraint. An edge can be taken when its initiation constraint is satisfied by the current clocks valuation, and it can be completed only when its completion constraint is satisfied. Analogously, every edge is associated with a set of clocks which are reset to zero when it is taken (initiation reset) and a set of clocks which are reset to zero when the action is completed (completion reset).

An action with duration is modeled, using timed automata, by two sub-actions corresponding to the initiation and the completion of this action. The resulting automaton has a different alphabet from the original one, and the information that the event is unique is lost.

Even the notion of timed language accepted by a timed automaton is revisited in this context; different notions of language acceptance are defined. The difference related to action occurrence, either on its beginning or completion point.

The investigation concern the expressiveness of the new model comparatively with classical T.A, it seems that timed automata with non-instantaneous actions are more expressive than timed automata. Indeed this good result is interesting but it comes at the cost of a complex theory and practice of the model.

What we think is heavy:

1 - The semantic used in model is based on dividing all actions into two sub actions.

2-Treatment of the timed automaton with non-instantaneous actions require the use of two identical automaton over two different sets of actions and finally,

3 - The proposal of a timed automaton simulator; which is the materialization of timed automaton with non-instantaneous actions; to simulate the behavior of the proposed model is heavy task (in practical point of view).

Our approach is more natural, whether in the modeling process or validating systems. The basic principle is consistent with standards knowledge and practitioners do not need to change conception approaches. Also it is important to note that our results are valued by the natural that provides maximality semantics.

1.3 Outline

In section 2, we propose certain preliminaries for enfolding our subject; we define durational actions timed automata in section 3; section 4 is devoted to the determinism and determinization method of DATA*. Section 5 deals with expressiveness of the DATA* model and propose a new classification of language classes; the decidability of the reachability problem is treated in section 6 by an algorithm which allows the minimization of state space. Finally, section 7 handles equivalence and validation systems, it shows that the behavior of a DATA* is a homomorphic image of the behavior of its aggregated regions automaton. The last section concludes the paper and gives some perspectives.

2. PRELIMINARIES

We present in this section *Timed Automata*. In the following R^+ is the set of nonnegative real numbers. A *clock* takes values from R^+ or it is undefined, denoted by $\perp$. Without loss of generality, we write $R_\perp^+ = R^+ \cup \{\perp\}$ where the set of nonnegative real numbers is extended with the special value $\perp$

Given a set *X* of clocks, a *clock valuation* over *X* is a function assigning a nonnegative real number to every clock.

The set of valuations of *X*, denoted V_X, is the set of total function from *X* to R^+. A valuation $v \in V_X$ is a mapping on *X* to R^+. The valuation $v+d$ maps every clock *y* to $v(y)+d$ $(d \in R^+)$.

Given a set λ of clocks, a *reset* λ is a subset of *X*. Given a valuation *v* and a reset λ, we define the valuation $v[\lambda \leftarrow 0]$ by $v(x) = 0$ for all *x* in λ and $v(x)$ if $x \notin \lambda$.

The set C_X of *clock constraints* over *X* are defined by the following grammar:

$C_X := \{false, true, C_X \wedge C_Y, C_X \vee C_Y, \overline{C_X}, (x * t), (t * x)\}$, where $x \in X$, $t \in R_\perp^+$, and * is a binary operator and $* \in \{<, >, \leq, \geq\}$. Clock constraints are evaluated over clock valuations. The satisfaction with respect to clock valuation function $v \in V_X$, the expression $\perp * \perp$ evaluates to true and all other comparisons that involve $\perp$ evaluate to false. We write $v \models C_X$ to denote that according the valuation function *v*, C_X evaluates to true. Let *ACT* be a set of label symbols,

In general way timed automata are finite state machines whose transitions are decorated by clocks constraints.

They are widely studied as model in which the control of real time systems is finite. In fact timed automata are construction of both finite set of locations and finite set of clock variables. Each edge is specified by a label name which contains action (that is going to be executed) and clocks formula. Those are considered as a guard of the edges and set of clocks which are going to be reset. Clock variables, in fact capture the time elapsed since the last clocks rest.

The execution (control) of automaton proceeds along an edge only when the valuation on clocks satisfies the corresponding constraint.

A finite timed word over *ACT* is defined as $\overline{(a,t)} = (a_0,t_0)(a_1,t_1)(a_2,t_2).....$, where $a = a_1 \ldots a_n$ is a finite sequence of symbols in *ACT* and $t = t_1 \ldots t_n$ is a finite monotone sequence of non-negative real numbers. t_i represents the time stamp of the occurrence of the event a_i.

For convenience we assume the initial time stamp $t_0 = 0$, prefixed to any sequence of time stamps *t*.

A run *r* of a timed automaton is a sequence of timed transitions:

$$r = (l_0, v_0) \xrightarrow{(a_0,t_0)} (l_1, v_1) \xrightarrow{(a_1,t_1)} .... \\ (l_n, v_n) \xrightarrow{(a_n,t_n)} (l_{n+1}, v_{n+1}) \qquad (1)$$

With l_0 an initial location and v_0 is such that $v_0(x) = 0$, $\forall x \in X$. The run *r* is accepting iff l_{n+1} is a final or terminal location. A timed word $\overline{(a,t)}$ is accepted by a timed automaton A iff, it exist a path in the automaton, labeled by $a_0 a_1 a_2 ... a_n$, and an accepting run over A. The timed language $L(A)$ accepted by A is defined as the set of all finite timed words accepted by A. We say that two automata are equivalent iff the timed languages accepted by them are the same.

3. DURATIONAL ACTIONS TIMED AUTOMATA

The Durational Actions Timed Automata model DATA*[3] is a timed model defined as timed automata over an alphabet representing actions to be executed. This model takes into account the duration of actions based on an intuitive idea: temporal and structural non-atomicity of actions.

Each clock in durational actions timed automaton is real value variable that records the duration of associated action. According to this sense, it exist an association between label names and clock variables, during its life.

Illustrate the model with the example above (Fig.2):

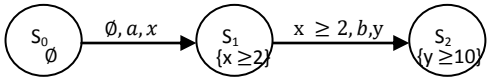

Fig 2: DATA*.

The durations associated to the actions are represented by constraints on the transitions and in the states targets of each of them. In this sense, any enabled transition represents the beginning of the action execution. On the target state of transition, a timed expression means that the action is possibly under execution.

From operational point of view, each action is associated to a clock which is rested to 0 at the start of the action. This clock will be used in the construction of the temporal constraints as guard of the transitions.

Fig.1 presents a system of two consecutive actions *a* and *b*, the clock *x* is associated to the action *a*, on the locality s_1 the temporal formula *{x≥2}* represents the duration of the action *a* (*which is important to distinguish from invariant in timed automata*).

The end of the execution of an action is deduced implicitly in the case of an action that it is causally dependent. The action *b* depends on *a*, so the transition is guarded by constraint formed by duration of *a* and more time *(x≥2).*

Starting from this, we can express different possibilities of real time systems behavior like delaying execution of action or limiting its offering time by manipulation of clock constraints.

Consider the following example describing the behavior of an automatic light switch from. The Light Switch can be specified by a durational actions timed automaton *A*, with

- $S = \{s_0, s_1\}$
- $S0 = \{s_0\}$
- $\Sigma = \{on, off\}$ *and* $dr(on)=1\ dr(off)=0$
- $C = \{x\}$
- $E = \{(s_0, true, on, \{x\}, s_1), (s_1, 1<=x < 6, on, \{x\}, s_1), (s_1, x>=6, off, \emptyset, s_0)\}$

Its behavior is as follows: The state of the system in which the light is off is represented by s_0, and the state s_1 represents the situation where the light is on. The light can be turned on by pushing the *on* button which elapse 1 unit of time to be executed. After five time units the switch turns itself *off*. Before that happens, the *on* button may be pushed again which will leave the light on, in this sense on is offered in the interval *[1, 6]* of time only (Fig. 3).

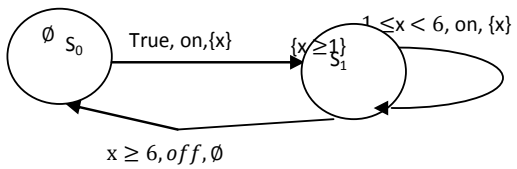

Fig 3: Light switch example.

Indeed, using durational actions timed automata model is very suitable to capture the true concurrency in systems behavior. Since each locality detain the information about current execution of action; when more than one action are under execution then associated temporal formula are found on locality. With this simple technique, the true concurrency is finely captured without heavy artefact. The model is interesting and attracting increasing research efforts because of how it extends timed automata with maximality semantics [7][9][13][14][25].

3.1 Formalization

Definition 1 : a DATA* A is a tuple (L, l_0, X, T_D, L_S) over ACT a finite set of actions, *L* is a finite set of locations, $l_0 \in L$ is the initial location, *X* is a finite set of variables named clocks and T_D is a set of edges. A subset of *L* noted L_f for terminal locations.

An edge *e=(l,G,a,x,l')* represents a transition from location *l* to location *l'* on input symbol *a*, *x* is a clock to be reset with this transition. *G* is the corresponding guard which must be satisfied to launch this transition.

Finally, $L_S : L \to P(C_X)$ is a maximality function which decorates each location by a set of timed formula named actions durations. Those concern overlapping execution of actions (C_X set of *clock constraints* over *X* defined above).

The function used to isolate the clock names about timed constraints, named Dom is defined as:

$$Dom: C_X \to X \ such that, Dom(G) = \{x_i | (x_i * t) \in G\} \quad (2)$$

Definition 2 : The semantic of a DATA* A is defined by the Timed transitions system(TTS) $(S_A, s_0, \to)$, over $ACT \cup R^+$. A state of S_A (or configuration) is a pair (l, v) such that l is a location of A and v is a valuation function over X , with initial configuration $(l_0, \perp)$. A terminal (accepting) configuration of *TTS* is a pair (l, v) with *l* in L_f .

The transitions on S_A are labeled either by a real number representing the elapsed time (Time steps), or by an action in *ACT* (Discrete steps). The rules to derive the transitions on S_A are the following:

$$R1: \frac{d, d' \in R^+,\ v + d' | \neq G : \forall (s, G, a, x, s') \in T_D, d' \leq d}{(s, v) \xrightarrow{d} (s, v + d)} \quad (3)$$

$$R2: \frac{(s, G, a, x, s') \in T_D\ v' | = G}{(s, v') \xrightarrow{a} (s', v'[\{x\} \leftarrow 0])} \quad (4)$$

As defined above, a run is manner to capture the behavior of automata. A run records the states and the values of all the clocks at the end-point of transitions.

Definition 3: a run *r* of DATA* A over a timed word $(\overline{a,t}) = (a_0, t_0)(a_1, t_1)(a_2, t_2).....(a_n, t_n)$, is a finite sequence:

$$r = (l_0, v_0) \xrightarrow{(a_0, t_0)} (l_1, v_1) \xrightarrow{(a_1, t_1)} ....(l_n, v_n) \xrightarrow{(a_n, t_n)} (l_{n+1}, v_{n+1}) \quad (5)$$

Of states (l_i, v_i) about locations $l_i \in L$ an edge $e_i = (l_i, G_i, a_i, x_i, l_{i+1}) \in T_D$ such that l_0 is the initial location and for all $i, 0 \leq i \leq n, v_i | = G_i$ and $v_i [x_i \leftarrow 0]$.

4. DETERMINISM OF DATA*

It is well established that, in the untimed automata (i.e finite state machines), every nondeterministic automaton can be determinized; consequently the both automata define equivalent languages.

Nevertheless, in timed automata the deterministic variety is strictly less expressive than the nondeterministic one [1].

We prove hereafter, that durational actions timed automata are a subclass of timed automata which are determinizable. The determinism property of automaton ensures that at each point of execution of the system the next step is controlled, in that case, merely by the current location in the automaton and label name (offered action).

We consider the definition of determinism that was proposed for timed automata in [1] as reference.

Definition 4: The DATA* $A=(ACT,L,l_0,X,T_D,L_s)$ is deterministic iff :

- It has at most one start location, and
- Two edges with the same location and the same label name have mutually exclusive clock constraints; that is, if $(l,G_1,a,x,l')\in T_D$ and $(l,G_2,a,x,l'')\in T_D$ then for all clock valuation functions *v:* $v\models G_1\wedge G_2$.

Theorem 1: A durational actions timed automata, is a determinizabe timed model.

Proof: To proof determinization of DATA* structure, we demonstrate that the DATA* have one start location verified by definition of DATA* model and at every stage of the execution of timed trace, the choice of transition to be executed is determined uniquely by the timed trace itself and it doesn't depend on the automaton. In other words the timed traces are determined uniquely by the offered action and the clock valuations.

Relying on that, the clock valuation function determines value of clocks which are significant for launching a next transition. In the durational actions timed automata model, we claim that at all levels of the automaton; the clock valuation function is determined by the current state and the actions that have already start their executions. Consider two determinative aspects of durational actions timed automata model:

1- In DATA* model, information about actions under execution (i.e transitions already launched) are stored in the target locations of edges, noted by .

2- Due to the underlying semantics of DATA* model, the guard of the transition depends uniquely on the clocks associated to actions potentially in execution.

Formalized by:

$$\forall G_i,\ Dom(G_i)\subseteq Dom(L_s(s_i)) \tag{5}$$

For a run *r* of DATA* A over a timed word:

$\overline{\omega}=\overline{(a,t)}=(a_0,t_0)(a_1,t_1)(a_2,t_2).....(a_n,t_n)$, it was

$$\begin{cases}(t_0=0 \text{ and } \forall x\in X, v_0(x)=\perp)\\ \forall j>0, v_j^{\overline{\omega}}\models G_j \text{ and } v_j^{\overline{\omega}}[x_j\leftarrow 0]\end{cases}$$

$$\begin{cases}\text{and for all } x\in X,\\ v_j^{\overline{\omega}}(x_a)=\begin{cases}\left(t_j-t_i \text{ if } (\exists i|0\le i<j \text{ and } a_i=a)\ (1)\right)\\ \left(\text{and}\left(x_a\in Dom(L_s(l_j))\right)\quad (2)\right)\end{cases}\\ \qquad\qquad\begin{cases}\left(\perp \text{ if } (\forall k|0\le k<j\ a_k\neq a)\quad (3)\right)\\ \left(\text{or}\left(x_a\notin Dom(L_s(l_j))\right)\quad (4)\right)\end{cases}\end{cases} \tag{6}$$

Explanation of different parts of formula "Equ.6" is as follows: Part (1) states that action a_i was launched in step *i* before step *j* of the run and *x* was reset in the same time. Part (2) states that a_i was not terminated since *x* was still in $Dom(L_s(l_j))$.

Part (3) and (4) state that a_k was never launched or has already completed.

We can assert that all timed traces can be unique because it is determined by actions already undertaken, and therefore by the timed word. This follows from the formula that governed the valuations of the clocks.

4.1 Detreminization of DATA*

We propose a determinization method, inspired from the classical one named subset construction. *L* is a set of locations, $E(L)$ function which given the set of transitions from a location in *L*. With $\Gamma(E(l))$ we denote the set of guards within $E(l)$. Let $A=(ACT,L,l_0,X,T_D,L_S)$ be a DATA*, L_f a final localities subset of *L*. *D*eterminized structure of A is $Det(A)$ (*Algorithm in subsection below*).

Note that the maximality function (L_s) ensuing is not altered by the construction method, nor the clocks set.

Properties:

1) For every durational actions timed automaton A and the deterministic timed automaton $Det(A)$: $L(A)=L(Det(A))$. This is ensured by the computation operation of the deterministic automaton, which based on timed words of initial automaton so it doesn't perturb the relation associating actions sequences to timed sequences in timed words.

2) Given a location l' of $Det(A)$, an input symbol a and a clock valuation function v there is exactly one edge (l',G,a,x,l'') in $Det(A)$ such that $v\models G$. This is proved by construction of deterministic automaton.

3) $Det(A)$, is a durational actions timed automaton. The construction algorithm of $Det(A)$, increases exponentially the number of locations, but doesn't changes the clocks to be reset nor the constants appearing in constraints. Maximality information on locations, given by $(L_s(l))$, is respected.

4.2 Algorithm

Consequently, the timed automaton $Det(A)$ agrees with the semantics of durational actions timed automata model. Those results are synthesized in the following theorem.

Theorem 2: For every durational actions timed automaton A, there exists a deterministic durational actions timed automaton $Det(A)$ that $L(A) = L(Det(A))$.

4.3 Closure Properties of Durational Actions Timed Automata

In this section we trait different proprieties of durational actions timed automata, in the aim to characterize the model. The closure properties are very important; it facilitates the use of the model in different fields of system validation.

The class of general timed automata is not closed under complement and the language inclusion problem for timed automata is undecidable [1],[26], the subclass durational actions timed automata is well behaved.

Lemma 1: DATA* are closed under union and intersection.

Proof: The construction techniques used in timed automata case are applicable in the case of durational actions timed automata as well.

Algorithm:

$Det(A) = (ACT, L_D, l_{D0}, X, T_{DD}, L_s)$, is constructed as follows:

1. $L_D \subseteq P(L)$: The locations set of $Det(A)$.
2. ACT, X, L_s: are respectively the sets of actions, clocks and the maximality function. They are the same of A.
3. $l_{D0} = \{l_0\}$: The initial location of $Det(A)$.
4. For any location $l \in L_D$ and action $a \in ACT$, Let $E' \subseteq T_D$ be a set of edges, where $E' = \{e_i | e_i = (l, G_i, a, x, l_i)\}$.
5. For every $E'' \in P(E')$
   - Create a location $l' \in L_D$ and $l' = \{\beta(e_i) \wedge e_i \in E''\}$
   - Compute a guard $G = \left(\underset{G \in \Gamma(E'')}{\wedge} G\right) \wedge \left(\underset{G \in \Gamma(E'/E'')}{\wedge} \neg G\right)$
   - Create an edge $(l, G, a, x, l') \in T_{DD}$
6. The set $L_{Df} \subseteq L_D$ is the set of terminal locations if L_{Df} I $P(L_f) \neq \phi$.

Lemma 2: Given a DATA* A, a deterministic DATA* $Det(A)$ can be constructed such that $L(A) = L(Det(A))$.

The detailed construction is given in subsection above.

Using determinization operation we get the following result:

Corollary 1: DATA* are closed under complementation.

Closure under complementation is obtained by determinization construction. The complement of DATA* structure A is obtained by constructing $Det(A)$ then complementing it by interchanging terminal and no terminal locations. $\neg Det(A)$, defines the complement language $\neg L(A) = L(\neg Det(A))$.

5. EXPRESSIVENESS OF DATA*

5.1 Introduction

We define $DATA_0$* as subclass of DATA* in which actions durations are null, this by returning to the assumption of instantaneous actions. With this hypothesis, the clocks of the $DATA_0$* inform about the start and completion of actions. In this sense clocks will implicitly record the date of the last occurrence of actions.

This recall a certain category of timed automata called Event Recording Automata (ERA). ERA represents subclass of timed automata which is a determinizable and studied in [2].

In the next we define both of models, $DATA_0$* and event recording automata (ERA), after what we prove the equivalence of languages defined by the two models.

*5.1.1 Durational Actions Timed Automata with Null Durations, DATA0**

Definition 5: A $DATA_0$* is a DATA* with the following restrictions:

1) Every action duration $d_a = 0$, for all a in *ACT*,

2) The maximality function L_s is redefined by $L_s : L \to \phi$.

The information about ending of actions are removed on localities of $DATA_0$*. By definition of DATA*, on every edge one clock at most is reset, in respect of the maximality semantics, this is for Tow reasons: first to capture duration of actions which are amount second to encode guards with those clocks names.

When we translate those restrictions on timed words of language defined by $DATA_0$* we can write:

Given timed word $\bar{\omega}$ from timed language of $DATA_0$*, and a run r over $\bar{\omega}$: At every step $j > 0$ of execution, clock valuations function is as follows:

$$\begin{cases} \text{for all } x \in X, \\ v_j^{\bar{\omega}}(x_a) = \begin{cases} (t_j - t_i \text{ if } (\exists i / 0 \le i < j \text{ and } a_i = a)) \\ (\perp \text{ if } (\forall k / 0 \le k < j \; a_k \neq a) \end{cases} \end{cases} \quad (7)$$

5.1.2 Event Recording Automata (ERA)

Like a timed automata, event recording automata (ERA) have a set of clocks to be reset when an action is taken and can be used in transitions guard. However, in ERA a unique clock is associated with each action. Whenever an action is executed, the associated clock is automatically reset. No additional resets are permitted.

Thus, unlike ordinary timed automata where clocks can be assigned to at will, ERA maintains a strict correspondence between actions and clocks. This ensures that the environment is in control over which clocks are reset, and this in turn gives the determinizability property. The event clock x_a associated with action a, thus records the amount of time passed since the last occurrence of *a*.

Definition 6: An Event Recording Automata are timed automata with the following restrictions:

1) For each action *a* there is an associated clock x_a, called the event clock of *a*. Thus, the set of clocks is:

$X = \{x_a / a \in ACT\}$.

2) The clock x_a, is automatically reset when action a is started. Thus, every edge labeled with action a is also implicitly labeled with the assignment $x_a := 0$. No further clock assignments are permitted.

When considering timed language recognized by event recording automaton and the set of timed word that it recognizes, the value of the clock x_a at the j-th position of timed word $\bar{\omega}$ is:

$$\begin{cases} \text{for all } x_a \in X, \\ v_j^{\bar{\omega}}(x_a) = \begin{cases} \left(t_j - t_i \text{ if } (\exists i / 0 \le i < j \text{ and } a_i = a) \text{and} \right. \\ \left. \forall k / 0 < k < j, a_k \neq a \right) \\ \left(\perp \text{ if } (\forall k / 0 \le k < j, a_k \neq a) \right) \end{cases} \end{cases} \quad (8)$$

For every x_a then $v(x_a) = t_j - t_i$, i is a position preceding *j-th* one such that a was occur, and the value of x_a is undefined if no occurrence of *a* precedes the current one (*j-th position*).

The event recording automata is a subclass of timed automata for which a deterministic for nondeterministic automata exists and both are language equivalent.

5.2 Expressiveness of DATA$_0$*

Lemma 3: DATA$_0$* and ERA are comparable. (i.e. (1) ERA ⊆ DATA$_0$*, and (2) DATA$_0$*⊆ ERA).

Proof: The timed languages accepted by the DATA$_0$* and ERA, are equivalent, indeed we define mapping application $\rho : X_0 \to X_E$ for all $\rho(x) = x_a$.

This application ensures the uniqueness of the clock name associated with an action, and also establishes the correspondence between the names of clocks and action labels.

1- To show that $L(DATA_0^*) \subset L(ERA)$ we give a construction that, given a DATA$_0$* A_0, builds an ERA E such that $L(A_0) = L(E)$.

For $A_0 = (ACT, L, l_0, X_0, T_D, L_S)$, the corresponding event recording automaton is the automaton $E = (ACT, Q, q_0, X_E, T_E)$, where:

- $q_0 = l_0$,
- $X_E = \rho(X_0)$ for each $x \in X_0$ and $a \in ACT$, $\rho(x) = x_a$,
- For each $\begin{cases} e = (l, G, a, x, l') \in T_D, \\ T_E \text{ contains } (q, G, a, q') \text{ and } \rho(x) = x_a \end{cases}$.

For every timed word $\bar{\omega}$ in $L(A_0)$, we have

$$\begin{cases} \text{for all } x \in X_0, \\ v_j^{\bar{\omega}}(\rho(x) = x_a) = \begin{cases} \left(t_j - t_i \text{ if } (\exists i / 0 \le i < j \text{ and } a_i = a) \right) \\ \left(\perp \text{ if } (\forall k / 0 \le k < j \; a_k \neq a) \right) \end{cases} \end{cases} \quad (9)$$

This corresponds exactly to the formula (8). Thus timed word $\bar{\omega}$ is also in $L(E)$,

All actions of A_0 are forced to be instantaneous. This is done by removing the set of maximal formula on locations and the matching of clocks set with the set of actions. So $|X_E| \ge |X_0|$, no possible reuse of clock variable in ERA.

So E operate as A_0 in all runs, and hence $L(A_0) = L(E)$ no matter, the given selection for A_0.

2- To prove the equivalence of timed languages accepted by the DATA*$_0$ and ERA, we show the other inclusion $L(ERA) \subset L(DATA_0^*)$. For this we construct a DATA$_0$* A_0, from an ERA E such that $L(E) = L(A_0)$.

Given $E = (ACT, Q, q_0, X_E, T_E)$, the corresponding DATA*$_0$ $A_0 = (ACT, L, l_0, X_0, T_D, L_S)$ is the automaton, where:

- $l_0 = q_0$
- $X_0 = X_E$ and
- For each $e = (q, G, a, q') \in T_E$, $(l, G, a, x, l') \in T_D$ and $x = x_a$.
- We define a maximality function L_s by $L_s : L \to \phi$, where $\forall l \in L, L_s(l) = \phi$.

This means that no action is underway simply because it has no duration.

With the same reasoning as in part 1of proof, we argue that $L(E) = L(A_0)$. The class of Durational actions timed automata with null durations is expressively equivalent to the class of Event recording automata. *End of proof.*

Hereafter we present our main theorem:

Theorem 3: DATA* is more expressive than ERA and less expressive than general timed automata. (i.e. (1) ERA ⊂ DATA* and (2) DATA* ⊂ TA).

Proof: Sketch, by lemma 3 and definition 5.

ERA=DATA$_0$* ⊂ DATA* therefore, ERA ⊂ DATA*.

To show that the class DATA* is a strict subset of TA, consider the timed automaton *T*, in which edge $e = (l, G, a, \lambda, l')$ resets $\lambda = \{x, y\}$. The language $L(T)$ cannot be selected accepted by any durational actions timed automaton.

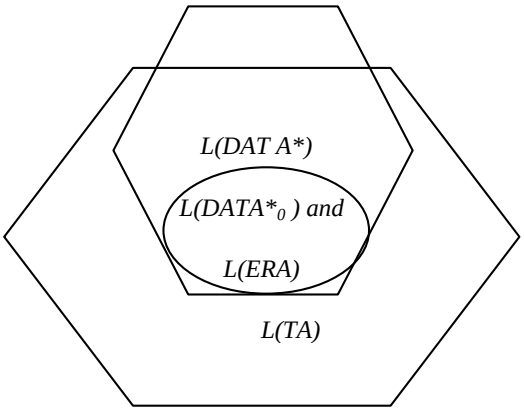

Fig 4: Inclusion among language classes.

6. REACHABILITY PROBLEM

Due to dense-time clocks, the transition system for a timed automaton has infinitely many states and operates on infinitely many symbols. Analysis of safety requirements of real-time systems can be formulated as reachability problems for timed automata. Since the transition system for a timed automaton is infinite, reachability analysis constructs a quotient called the region automaton by partitioning the uncountable state space into finitely many regions [26].

The clock valuations have to be synthesized in regions where valuations are grouped in classes of equivalent. This leads to construct region automaton for the initial automaton.

Region automata reproduce the infinite execution of timed automata by a finite set of transitions. 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 automata. Since the number of regions of timed automata is finite, it's used to solve reachability problems [20]. We first recall the notion of clock regions and region automata.

6.1 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. A region is a symbolic representation of a set of clock valuations, or formally, an equivalence class on clock valuations induced by an equivalence relation.

Definition 7: Let *A* be a durational actions timed automaton, the set of standard regions Ω of *A* is the set of equivalence classes' of relation $\equiv$. This relation is defined over the clocks valuations as follows:

$$v \equiv v' \text{ if } \forall (x,y) \in X \begin{cases} \lfloor v(x) \rfloor = \lfloor v'(x) \rfloor \text{ where} (v(x) > M_x \Leftrightarrow v'(x) > M_x) \\ \left(v(x) \le M_x \text{ and } v'(y) \le M_y \right) \Rightarrow \\ \left(frac(v(x)) \le frac(v(y)) \Leftrightarrow frac(v'(x)) \le frac(v'(y)) \right) \\ v(x) = \perp \Leftrightarrow v'(x) = \perp \end{cases} \quad (10)$$

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 X of two clocks x, y and $M_x = 2$ $M_y = 1$, so we have 28 regions (Fig.5).

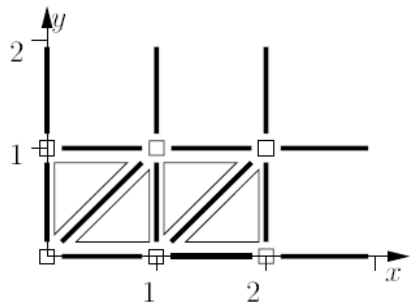

Fig 5: Standard regions.

6.1.1 Properties of Clock Regions

Two valuations of the same region must satisfy the same constraints.

The equivalence of the regions is compatible with the passage of time; it is therefore possible to define a successor function on all regions. We note $succ(r)$ the set of all successors of the region r by lapsing time.

Finally, we note that any clock region can be represented as follows: {*x = k: (k = 0,1 .., Mx) or k-1 <x < k: (k = 1, .. Mx) or k <x: (k = Mx)*}.

6.1.2 Region Automaton of Durational Actions Timed Automaton

The behavior of DATA* can be captured by a finite temporized finite state machine (named region automaton) where states are formed in a pair by locations and clock regions which are equivalent classes of clock valuations function.

Definition 8: Let $A = (ACT, L, l_0, X, T_D, L_S)$ be a DATA*, its region automaton $RA(A) = (S, s_0, T_R)$ over *ACT*, is defined as follows:

-The set of states is noted S, all of them are of the form $s_{ii} = (l_i, r_i)$ where l_i is a location and r_i is a clock region.. The initial state is $s_0 = (l_0, r_0)$. "r_0" is the initial region where every clock is initialized by zero.

-The set of transitions T_R is,

$$T_R = \left\{ t'/t' = (l,r) \xrightarrow{a} (l',r') \middle| \begin{cases} \exists l \xrightarrow{g,a,x} l' \in T_D \text{ and } \exists r'' \in succ(r) \\ \text{such as } r \subseteq g \text{ and } r' = r'' \left[x \leftarrow 0 \right] \end{cases} \right\} \quad (11)$$

$s_{ij}^f = (l_i, r_j)$ is a terminal state iff $l_i \in L_f$.

The example depicted by Fig.6 presents a DATA* and the its region automaton.

Theorem 4: For every durational actions timed automaton A, there exists an automaton $RA(A)$ recognizes untimed language of A, *untime* $(L(A)) = L(RA(A))$.

Nevertheless, the complexity of implementing region automata is exponential in the number of clocks and in the length of timing constraints [16][18][29].

In this last part, we define an aggregation operation on region automaton locations using a Bi- relation. This equivalence relation is used for grouping equivalent regions. Aggregation operation reduces the number of graph states. For this purpose we propose an algorithm implementing the aggregation relation starting from an initial partitioning of states. The aggregated regions automaton generated preserve the reachability property, thus the reachability question on locations of DATA* is reduced to reachability question about aggregated regions automaton. Therefore the language recognized by both automata is the same.

6.2 Aggregation Principles

In this section, we define an aggregation operation, which can be considered as a special determinization. The indeterminism considered here is due to the construction of the region automaton and not that inherent in the system.

The aggregation operation principle consists to find and regroup states witch verify those points:

-First, for any input transition of a state s_1, there exists an input transition to a state s_2, where they out coming from the same source state and having the same label. This second transition is called "*forward mirror*" of the first.

-Secondly for any output transition of a state s_1, there exists an output transition from a state s_2 where they outgoing to the same target state and having the same label. This second transition is called "*backward mirror*" of the first.

-Finally, those states are grouped if their sets "*forward mirror*" and "*backward mirror*" matches.

6.2.1 Notations and Definitions

Let t_i be a transition of a timed transition system, we note $\alpha(t_i)$ (resp. $\beta(t_i)$) the source state of t_i (resp. the target state of t_i), the label of the transition t_i is given by $\lambda(t_i)$.

The set of the input transitions in a state l_{ii} is: $in(l_{ii}) = \{ t_i \mid \beta(t_i) = l_{ij}\}$. The set of the output transitions from a state l_{ij} is: $out(l_{ij}) = \{ t_j \mid \alpha(t_i) = l_{ij}\}$.

Definition 9: "*forward mirror*" and "*backward mirror*" are a sets of states, defined respectively as follows:

$$FM(l_{ij}) = \left\{ \begin{array}{l} l_{ik} / \forall \tau \in in(l_{ij}), \exists \tau' \in in(l_{ik}) \text{ such as} \\ \lambda(\tau') = \lambda(\tau) \wedge \alpha(\tau') = \alpha(\tau) \end{array} \right\} \quad (12)$$

$$BM(l_{ij}) = \left\{ \begin{array}{l} l_{ik} / \forall \tau \in out(l_{ij}), \exists \tau' \in out(l_{ik}) \text{ such as} \\ \lambda(\tau') = \lambda(\tau) \wedge \beta(\tau') = \beta(\tau) \end{array} \right\} \quad (13)$$

Definition 10: A set of grouped localities with l_{ij}, noted $RL(l_{ij})$, is defined as follows:

$$RL(l_{ij}) = \left\{ l_{ik} / \begin{array}{|ll} l_{ik} \in FM(L_{ij}) & \text{if } l_{ik} \text{ is a terminal location} \\ l_{ik} \in FM(l_{ij}) \text{ I } BM(l_{ij}) & \text{else} \end{array} \right\} \quad (14)$$

Definition 11: IR is a grouping relation on states, it is defined as follows: $\forall x, y \in L / (x, y) \in IR$ iff $y \in RL(x)$.

The IR relation is an equivalence relation, which allows defining the equivalence classes of the grouped states. Indeed, IR is a reflexive relation: any state *s* is grouped with itself because it has the same "*forward mirror*" and "*backward mirror*" as *s*. If *s* is grouped with *s'* then *s* has the same "*forward mirror*" and "*backward mirror*" as *s'*: It is evident that *s'* has the same "*forward mirror*" and "*backward mirror*" as *s* according to definition 9. It is thus symmetrical. (*s,s'*) are grouped and (*s',s''*) are grouped, by application of definition 9, (*s,s''*) are grouped then the relation is transitive.

The summation of clocks regions is an operation defined on the set of regions as follows:

$$\hat{r} = \oplus(r_1, r_2, ... r_k) = \oplus_k(r_k) = \left\{ \begin{array}{ll} r & if \quad k = 1 \\ r_1 \cup r_2 \cup ... \cup r_k & if\ k > 1 \end{array} \right. \quad (15)$$

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

Definition 12: The Aggregated regions automaton $AR_A(A)=(S^A, s_0^A, T_{AR})$ of durational actions timed automaton *A* is a finite transition system over the alphabet *Act* defined as follows: The states of $AR_A(A)$ are of the form $(l, \hat{r})$ where $l \in L$ and $\hat{r}$ is a region, the initial state is of the form $(l_0, \hat{r}_0)$ 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 from aggregation operation.

6.2.2 Aggregation Algorithm

Given a region automaton $RA(A) = (S, s_0, T_R)$ over *ACT*, the following algorithm is used for grouping states according to the equivalence relation defined above. Hereafter, we consider following notations:

Π is a partition of S. Π_0 the initial partition, it is composed by singletons, elements of S. The function *frag* that fragments classes P in subclasses formed by singletons.

$$\delta_\alpha^{-1}(P) = \left\{ Q, Q \xrightarrow{\alpha} P \text{ and } \alpha \in ACT \right\} \quad (16)$$

$$\delta_\alpha(P) = \left\{ Q, P \xrightarrow{\alpha} Q \text{ and } \alpha \in ACT \right\} \quad (17)$$

$$\Phi_{P,Q} \left[\begin{array}{l} \text{for each } \alpha, \alpha' \in \Sigma \text{ and } P, Q \text{ Classes} \\ \text{if } \delta_\alpha^{-1}(P) = \delta_\alpha^{-1}(Q) \wedge \delta_{\alpha'}(P) = \delta_{\alpha'}(Q) \text{ then} \\ (P, Q) \text{ are grouped} \end{array} \right. \quad (18)$$

A function which joins Two classes Q and P :

$$Joi(P,Q) = K \text{ where } l_k = l_p = l_q \text{ and } \hat{r}_k = r_p \oplus r_q \quad (19)$$

The case of terminal locations is implicitly considered

$$\delta_\alpha(P) = \delta_\alpha(Q) = \phi \quad (20)$$

Algorithm:

$\Pi_0 = \text{initialpartition}, X = \Pi_0, \Pi = \phi$
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 $K := Joi(K, l_{ik})$
$\Pi := \Pi \cup \{K\}$
$X := (X / frag(K)) \cup K$
until marking all the elements of X

K is a set of grouped states at each step; in the end Π contains states of the aggregated regions automaton;

The termination property of algorithm is ensured by the finite number of regions in the initial graph. To implement this algorithm we only need an efficient representation for the regions and perform simple operations such as summation over the regions.

Example 1: The example below (Fig.6) presents a DATA* *A*. The associated regions graph is depicted by Fig.7. Fig.8 shows the region graph after applying the regions aggregation algorithm. The regions automaton is composed of 6 regions. The aggregated graph has 3 regions.

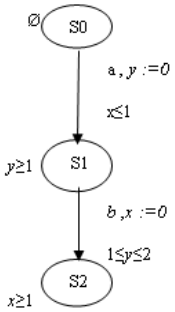

Fig 6: DATA* A.

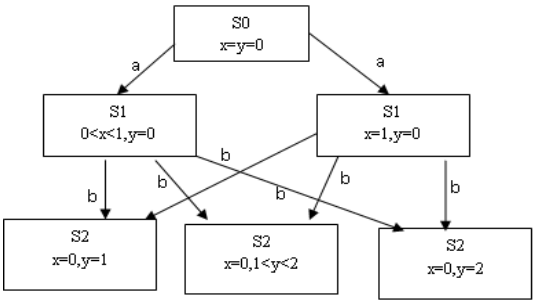

Fig 7: Region graph associated to A.

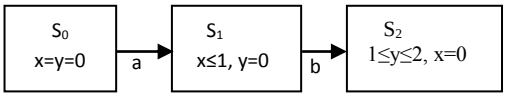

Fig 8: Aggregated region graph associated to A.

7. EQUIVALENCE AND VALIDATION OF SYSTEMS

In this section we show that the behavior of a durational actions 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 can sufficiently validate an implementation of the durational actions 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:

$H:(A,o)\to(B,*)$. Such that H(x) is an element of B and for any pair x1, x2 in A,

$$H(x_1 \,o\, x_2) = H(x_1) * H(x_2) \tag{21}$$

The Algorithm above suggests in natural way an homomorphism H from ARA(A) to A, mapping the states of ARA(A) to the locations of A while preserving the transitions relations over the same actions.

To detail this we define a projection over set of actions on transitions as:

$$\prod_{\Sigma} : T_A \to T'_A, \prod_{\Sigma}(\tau) = \prod_{\Sigma}(s,g,a,\gamma,s') = (s,a,s') = \tau' \tag{22}$$

$H: AR_A(A) \to A$, have the following properties: For all $l_{ij}^A \in L^A, H(l_{ij}^A) = s_i \in S$ such that $H(l_0^A) = s_0$ and for all transitions:

$$\lambda \in T_{AR}, \lambda = (l_{ij}^A, a, l_{kt}^A), H(\lambda) = (H(l_{ij}^A), a, H(l_{kt}^A)) \in \prod_{\Sigma}(T) \tag{23}$$

H maps action of λ to 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 location of the automaton. That is, H can be extended to map computations. $Exec(A)$, is the all computations set of durational actions timed automaton A.

$$H^+ : Exec(AR_A(A)) \to Exec(A) \tag{24}$$

$$H^+(\lambda_0\lambda_1\lambda_2 ..... \lambda_J .....) = H(\lambda_0)H(\lambda_1)H(\lambda_2)....H(\lambda_J).... \tag{25}$$

We recapitulate these results as follows: For every computational path in the aggregated regions automaton ARA(A) there is a computational path in the timed automaton A. The behavior of A is a homomorphic image of the behavior of its aggregated regions automaton ARA(A).

Since states of the aggregated regions automaton ARA(A) include clock regions (inherent from region automaton), which are the equivalence classes of grouped 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.

8. CONCLUSION AND OUTLOOK

In this paper we present durational actions timed automata, DATA*, as a subclass of timed automata. This model is determinizable and closed under all Boolean operations. After, we compare a DATA* with event recording automata which permits a new map of language classes. Finally, we propose an algorithm for reducing region automata. 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 states on the fly, which consists to build region graph without constructing the full one. Effectively it is possible since the proposed 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 [28][30][31].

9. REFERENCES

[1] R. Alur , D. L. Dill, A theory of timed automata, Theoretical Computer Science, 126(2):183-235,1994.

[2] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: a determinizable class of timed automata. Theoretical Computer Science, 211(1–2):253–273, 1999.

[3] N. Belala, Timed Models and Interest in Formal Verification of Real-Time Systems. PHD's theses, Mentouri University, 25000 Constantine, Algeria, Jun 2010.

[4] F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking timed automata with one or two clocks. In

Proceedings of the 15th international Conference on Concurrency Theory (CONCUR'04), volume 3170 of Lecture Notes in Computer Science, pages 387–401. Springer, August 2004.

[5] S. Lasota and I. Walukiewicz. Alternating timed automata. ACM Trans. Comput. Logic, 9(2):1–27, 2008.

[6] J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In LICS '04: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, pages 54–63, Washington, DC, USA, 2004.

[7] H. Hachichi, I .Kitouni, D. E. Saïdouni, "A Graph Grammar Approach for calculation of Aggregate Regions Automata", The International Arab Conference on Information Technology (ACIT), 2011, Naif Arab University for Security Science (NAUSS) Riyadh, Saudi Arabia (December 11-14, 2011).

[8] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In LICS '05: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science, pages 188–197, Washington, DC, USA, 2005.

[9] I. Kitouni, Determinization of Timed Automata With Action Duration for Formal Test. Aggregation Master's thesis, Mentouri University, 25000 Constantine, Algeria, Jun 2008.

[10] P.A. Abdulla, J. Ouaknine, K. Quaas, and J. Worrell. Zone-based universality analysis for single-clock timed automata. In Proc. FSEN'07, IPM International Symposium on Fundamentals of Software Engineering, Lecture Notes in Computer Science 1790, pages 98–112. Springer-Verlag, 2007.

[11] P. V. Suman, P. K. Pandya, S. Narayanan Krishna L. Manasa.G: Determinization of Timed Automata with Integral Resets. Technical Report TIFR-PPVS-GM-2007.

[12] D. E. Saïdouni , J. P. Courtiat, " Prise en Compte des Durées d'Action dans les Algèbres de Processus par l'Utilisation de la Sémantique de Maximalité" ,In CFIP.2003. Hermes, France, (2003).

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

[14] D. E. Saïdouni, I. Kitouni , H. Hachichi , Formalization of aggregated region automata associated to durational action timed automata. MISC REPORT 11001, Mentouri University, 25000 Constantine, Algeria, (2011).

[15] J. Springntveld, F. Vaandrager, P. D'Argenio , "Testing timed automata". Theoretical Computer Science, 254, (2001).

[16] R. Alur, C.Courcoubetis, and D. Dill, Model Checking in Dense Real-Time. Information and Computation, 104(1):2–34. (1993)

[17] B. Bérard, S. Haddad, and M. Sassolas. Interrupt Timed Automata: Verification and Expressiveness. Formal Methods in System Design, (2012).

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

[19] S. Tripakis. Fault diagnosis for timed automata. In FTRTFT'02. LNCS 2469, Springer, 2002.

[20] P.V Suman, P.K. Pandya, S.N Krishna, and L. Manasa: Timed automata with integer resets: Language inclusion and expressiveness. In FORMATS, pages 78–92. Springer, (2008). LNCS 5215.

[21] H. Bowman and R. Gomez: Concurrency Theory, Calculi and Automata for Modelling Untimed and Timed Concurrent Systems. ISBN-10: 1-85233-895-4 ISBN-13: 978-1-85233-895-4 Springer-Verlag (2006).

[22] R. Barbuti, N. De Francesco, L. Tesei : Timed Automata with non-instantaneous Actions. Fundamenta Informaticae 46 (2001) 1–15 1, IOS Press.

[23] R. Alur, C. Courcoubetis, Halbwachs, N., Dill, D.L. and Wong-Toi, H. Minimization of Timed Transition Systems. Proc. CONCUR 1992, Springer LNCS 630, 340–354.

[24] S. Tripakis, 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. (2004)

[25] I. Kitouni, H. Hachichi, D.E. Saidouni : A Simple Approach for Reducing Timed Automata. In: The 2nd IEEE International Conference on Information Technology and e-Services (ICITeS 2012). Sousse, Tunisia (March 24-26, 2012).

[26] R. Alur. Timed automata. In Nicolas Halbwachs and Doron Peled, editors, Proceedings of the 11th Internation Conference on Computer Aided Verification (CAV 1999), volume 1633 of Lecture Notes in Computer Science, pages 8–22. Springer-Verlag, 1999.

[27] B. Nielsen, , A. Skou,: Automated test generation from timed automata. In 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'01. LNCS vol 2031, pp 343-357. Springer (2001).

[28] H. Hachichi, I. Kitouni, D. E. Saidouni. Transforming DATA* with Dotty Format to Aggregate Region Automaton. International Journal of Computer Applications,vol 37(10), pp 35-42 (2012).

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

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

[31] M. Mao Zheng, V. Alagar and O. Ormandjieva, 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 (2008).