ISSN 2047-3338

Timed Refusals Graph for Non-Deterministic Timed Systems

Ilham Kitouni, Hiba Hachichi, Kenza Bouaroudj and Djamel-Eddine Saidouni

MISC Laboratory, University Mentouri Constantine, 25000, Algeria

{kitouni, hachichi, bouaroudj, saidouni }@misc-umc.org

***Abstract*— In this paper we are interested in refusals based model for validating timed systems. We propose a new refusals graph named timed refusals regions graphs (TRRGs). In this case specifications are modeled by durational actions timed automata (DATA*) based on maximality semantics which claim that actions have durations. This latter model is in one hand useful for modeling and validating reel aspects of systems. In the other hand, it is determinizable. In TRRG, refusals could be temporary or permanent. Permanent refusals are provoked by the non-determinism in the specifications. However, temporary refusals are the result of the fact that actions elapse in time. We propose a framework for generating timed refusals regions graph. This framework is implemented by a combination of Meta-modelling and Graph Grammars, to transform a DATA* structure into a TRRG. This permits the automatic generation of a visual modeling tool. Finally, we argue the use of TRRG in formal test of timed systems.**

***Index Terms*— Formal Testing Models, Refusal Graphs, Timed Systems, DATA* and Maximality Semantics**

I. 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 big complexity. Formal validation methods are the most used technique to deal with concurrent systems safe requirements, because of formal methods ability to describe the system behavior without ambiguity; it offers several verification approaches for assessing systems behaviors. In this paper we are interested by formal testing approach of real time systems based on timed refusals. Formal testing techniques [8], [9], [21], [20], [23] provide systematic procedures to ensure implementations conformity and it allows also checking the correctness of systems and helps to ensure their quality.

In this paper, the system is represented by the "Durational Action Timed Automata model (DATA*)", which is a timed model, its semantic expresses the durations of actions and other notions for specifying the real-time systems such as urgency [3]. This model is based on the maximality semantics [19] and advocates the true concurrency; from this point of view it is well suitable for modelling real time, concurrent and distributed systems.

A. Contribution

In this paper firstly, we define a new structure named timed refusals graph that provides additional insight in the practice and theory of generating tests. Timed refusals graphs (TRGs) are generated from deterministic specifications graphs. In our case specifications are modeled by DATA*.

We proceed at first by determinization of automaton, after we calculate sets of refusals; which decorate each location in the specification graph. An important aspect considered at this level is the non-determinism which is captured by permanent refusals. Temporary refusals are induced by the fact that actions elapse in time. In the DATA* model the durations of actions are captured by temporal constraints on edges and on locations.

We propose a framework to generate the TRG's structure and we reduce its states space by an aggregation regions graph approach. Initially this approach was developed for reducing timed automata [10].

Finally, an implementation for this algorithm is proposed, using graph grammar and graph transformation for generating a visual tool. We argue the use of the proposed structure (TRRG) for testing timed systems.

The rest of this paper is organized as follows. Section 2 reviews durational actions timed automata and different concepts about this model. Section 3 introduces the testing framework named timed refusals graph also its generation and minimization of refusals sets. Section 4 extends a method presented as an alternative simple to reduce states space of TRG structure. Section 5 deals with a prototype implementation and illustrates the method on a small case study. Section 6 and section 7 we discuss the use of TRRG in test generation and we conclude through future work plans.

II. Durational Actions Timed Automata (DATA*)

A. *Intuition of Maximality Based Semantics and Maximality Based Timed Automata*

The semantics associated to the specification model allows the choice of an adequate representation. In the case of timed automata the underlying semantic used is the interleaving semantics [12], [24], [5]. In which concurrent executions of two actions are interpreted by their interleaved executions in time. Following this semantics, every action is supposed to be atomic (structural and temporal) i.e., actions are not divisible and may not elapse in time. These hypotheses make the associated theory simple and the validation tools relatively easy to build. In real world systems, actions are not instantaneous, but have durations. This realistic characteristic is important in many cases.

Maximality semantics based models respect the principles below: For every action is assigned an event name that materializes its effective execution. The event name will condition the execution of the following actions if they are dependent on this first one. There are sets of event names on states; they correspond to actions that are running simultaneously. When an action ends the event name which corresponds to it is released.

A relevant aspect of systems specification is the concurrency. In the opposite of interleaving semantics based models, the true concurrency is specified with elegant and natural way by models based on maximality semantics.

The example depicted by Fig. 1 shows those concepts. Fig. 1(a) illustrate the same representation of a system of two actions, a and b, at the left, they are in parallel, the right graph presents a choice on actions a and b.

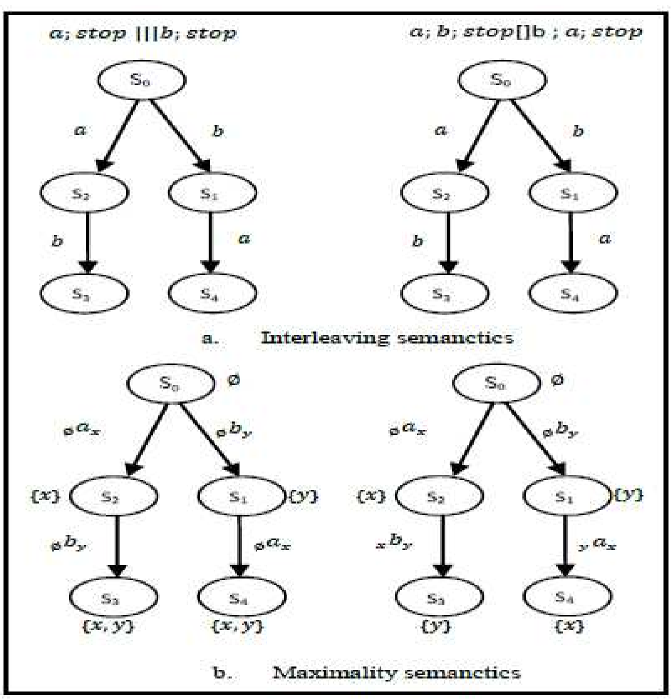

Fig. 1. Representation of concurrency and choice in maximality and in interleaving semantics

In fact, events name {x,y} concerning the run of *a* and *b* captured on locations S_2 and S_4, in Fig.1.b at the left makes the difference. It informs about the concurrent execution of actions (*a* and *b*).

Note that also at on the edges there exists a difference (Fig. 1(b) at the right), more information about dependence of actions. For example the atom *(x,b,y)* means that action *b* depend on event *x* (here x correspond to action a) and its own event name is *y*.

It is easy to see that the true concurrency and the auto concurrence are feasible.

B. *Refusals in Maximality Based Model*

This section defines new different kinds of refusals which are possible in models based on the maximality semantics.

Initially, refusals are defined as a set of actions which cannot be permitted from one state in systems behavior description. In our work those refusals are named forbidden actions to avoid ambiguity. At the time of test execution, forbidden actions lead to the failure verdict.

Other refusals are effectively possible in the same time. The permanent refusals are caused by the non-determinism in system behavior.

As illustrated by Fig. 2, after the system determinisation on action *a* (in left), the execution of the following actions (*b* and *d)* are uncontrolled and uncertain so each of them can be refused even they are offered after action *a* (in right).

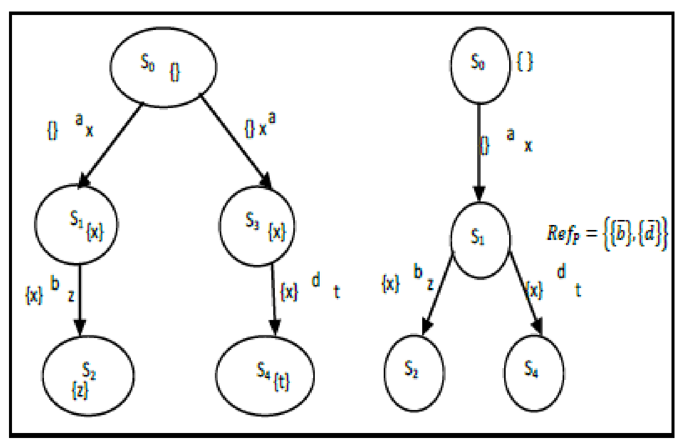

Fig. 2. Permanent refusals

The temporary refusals are provoked by actions which elapse in time; the elapsing property of action is associated to DATA* locations. Consequently certain executions are delayed until the termination of actions which they depend (in the sequential way). We show this in Fig.3. Temporary refusals on action (b), noted $\left(\bar{\bar{b}}\right)$ in refusal sets Ref_T.

In this paper, we are interested by those kinds of refusals only.

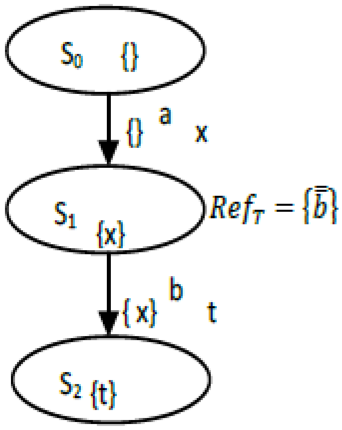

Fig. 3. Temporary refusals

C. Timed Automata with Duration of Actions

The durational actions timed automata (DATA*) model was proposed as an extension, which expand timed automata model by maximality semantics [13], [17], [18], [19]. Thus it permits to drop the assumption that actions are instantaneous. This allows specifying real time systems in a natural way.

The DATA* model 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. Each clock in DATA* is variable that records the duration of associated action. According to this, it exist an association between label names and clock variables, during action life. This clock will be released as soon as the action ends its run.

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

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

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. The Light Switch can be specified by a durational actions timed automaton *A,* with

- $S = \{s_0, s_1\}$
- $S_0 = S_f = \{s_0\}$
- $\Sigma = \{on, off\}$ *and* $dr(on)=1$ $dr(off)=0$
- $C = \{x\}$
- $E = \{(s_0, Ø, on, x, s_1), (s_1, 1<=x < 6, on,x,s_1), (s_1,x>=6, off,Ø,s_1\}$

Its behavior can be explained 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. 4).

On the location s_1 the temporal formula $\{x \geq 1\}$ represents the duration of the action *on*.

(*This is important to distinguish from invariant in timed automata).*

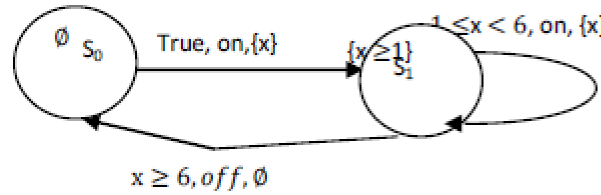

Fig. 4. Light switch example

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, the associated temporal formula are found in the following locations. With this simple technique, the true concurrency is finely captured without heavy artefact. As claimed above, this is inherited from the maximality semantics. Concurrent actions have different representation by transitions systems from choice on actions [4].

The model is interesting and increasing research efforts because of how it extends timed automata with maximality semantics [2], [17], [18], [20].

Definition 1 : a DATA D is a tuple* $(L, l_0, X, T_D, L_S, L_f)$ *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.* L_f *is a subset of L for terminal locations.*

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 $\perp$

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, $L_S : L \rightarrow P(C_X)$ 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 is a set of clock constraints over X.

The semantic of a DATA* *D* is given by the timed transitions system (*TTS*): $(S_D, s_0, \rightarrow)$ over $ACT \cup R^+_\perp$. A state of S_D (or configuration) is a pair (l, v) such that l is a location of *D* 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_D 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_D are the following:

$$R1:\frac{d\in R^{+}\ \forall d'\le d, v+d'|\ne G}{(s,v)\xrightarrow{d}(s,v+d)} \quad (1)$$

$$R2:\frac{(s,G,a,x,s')\in T_D\ \ v'|=G}{(s,v)\xrightarrow{a}(s',v[\{x\}\leftarrow 0])} \quad (2)$$

*D. Non-Deterministic DATA*s*

It's established [20], 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].

Definition 2: The DATA* D= (L,l_0,X,T_D,L_S,L_f) over *Act* is deterministic if and only if:

- It has at most one start location, and
- Two edges with the same source 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:* $v|\ne G_1\wedge G_2$.

*E. Refusals in Non-Deterministic DATA**

We define the extended permanent and temporary refusals sets in DATA* model as :

$\bar{\nabla}=\{\bar{a}(G)\}\in Ref_{TPR}(l)$: is a permanent refusal. It means that the action a may be refused permanently from the state *l*, this refusal is possible but not certain. This certitude will take place after the satisfaction of guard G.

$\bar{\bar{\nabla}}=\{\bar{\bar{a}}(G)\}\in Ref_{TPR}(l)$: is a temporary refusal and it means that actions are refused as much as the guard G is not satisfied.

$P\left(P\left(\bar{\nabla}\cup\bar{\bar{\nabla}}\right)\right)$ is partition of parts of refusal sets.

Definition 3: Let D be a DATA, for each location l in L, it is associated a refusal sets:*

$$Ref_{TPR}(l)\stackrel{\text{def}}{=}\{Ref_T(l)\cup Ref_P(l)\}:\ \begin{cases}Ref_T(l)\stackrel{\text{def}}{=}\text{set of temporary refusals}\\ Ref_P(l)\stackrel{\text{def}}{=}\text{set of permanent refusals}\end{cases} \quad (3)$$

III. Timed Refusals Graph

A timed refusals graph is a deterministic DATA* extended by refusals sets:

Definition 4: $T_{RG}=(D,Ref_{TPR})$ *with D=* (L,l_0,X,T_D,L_S,L_f) *is a deterministic DATA* over Act and* $Ref_{TPR}:L\rightarrow P\left(P\left(\overline{Act}\cup\overline{\overline{Act}}\right)\right)$ *is an application that associates for any* $l\in L$ *a set of refusals.*

$\overline{Act}=\{\bar{a}(G):a\in Act\}$ *and* $\overline{\overline{Act}}=\{\bar{\bar{a}}(G):a\in ACT\}$.

Since temporary and permanent refusals graph is deterministic, then:

$\forall l\in L$ *if* $\left(l\xrightarrow{G,a,x}l'\right)$ *and* $\left(l\xrightarrow{G',a,x'}l''\right)\in T_D$ *and* $G\wedge G'\ne\emptyset$ *then* $l'=l''$.

(4)

A. Timed Refusals Graphs Generation

In this section we propose a framework to create timed refusals graphs from DATA* specifications. The proposed framework consists in two steps:

1) determinization of DATA* and 2) decoration of automaton with refusals sets.

This requires computing $Ref_{TPR}=\{\cup_{l_i}(\{Ref_P(l_i)\cup Ref_T(l_i)\}):l_i\in L\}$. It's done as follows:

Let D=(L,l_0,X,T_D,L_S,L_f) be a DATA* over *Act*, the timed refusals graph of D, $TRG(Det(D))=(Det(D),Ref_{TPR})$ is a deterministic graph structure constructed as follows:

Step1: *Determinization :*

> $Det(A)=(ACT,L_D,l_{D0},X,T_{DD},L_s)$, 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 $L_{Df}\cap P(L_f)\ne\phi$

Step2: *Refusals decoration:*

For any location, $l\in L_D$,

a. If *e* is deterministic, $e=(l,G,c,x,l_i)$, let $E\subseteq T_{DD}$ be a set of edges, where $E=\{e_i|e_i=(l,G_i,c,x,l_i)\}$ and characterized by the size of $(l_i)=1$ so:

$$Ref_{TPR}(l) = \cup_{e \in E} \begin{Bmatrix} Ref_T(l) = \{\bar{c}(G) | c \in e \text{ and } G \neq \emptyset\} \\ \cup \ Ref_P(l) = \emptyset \end{Bmatrix} (5)$$

b. Otherwise, e is non deterministic (i.e $e = (l, G, c, x, l_i)$ and characterized by the size of $(l_i) \geq 2$) so,

For every E'' and $(e_i) \in E''$ from step 5 of determinization:

$$\begin{cases} E''_i = \{e_j | e_j = (\beta(e_i), G_j, c_j, x_j, l_j) \text{ and } e_i \in E''\} \\ E_i^c = \left\{e_k \middle| \begin{matrix} e_k = (\beta(e_i), G_k, c_k, x_k, l_k) \text{ and} \\ e_i \in E'' \setminus \{e_i\} \end{matrix} \right\} \end{cases} \quad (6)$$

Note : Equ.7 defines the set of E''_i and its complement on E''

$$Ref_{TPR}(l') = \cup_{e_i \in E''} \left\{ \left\{ \cup_{e_j \in E_i} \{(\bar{\bar{c}}_j(G_j))\} \right\} \cup \left\{ \cup_{e_k \in E_i^c} \{(\bar{c}_k(G_k))\} \right\} \right\} \quad (7)$$

Lemma 1: let D= $(L, l_0, X, T_D, L_S, L_f)$ be a DATA*, a timed refusals graph $T_{RG} = (D, Ref_{TPR})$ is a structure created by the previous framework, so it has the following properties:

1. The calculus of TRG terminates and verifies *Definition 4* of timed refusals graphs.
2. The DATA* and the corresponding TRG are trace equivalent.
3. The determinism of the TRG is insured.
4. After a timed trace σ, a set of actions which can be refused, is included in the set of refusals associated to the node g' where $g' = g_0\sigma$.

Sketch of proof: The termination property of the framework is insured by the finite number of state and determinizability property of DATA* model [20].

The timed refusals graph is inductively constructed by exploring all traces of the DATA*. This implies that they are trace equivalent.

The third and forth properties result from the algorithm by steps (1) and (2). ∎

B. Minimization of Refusals Sets

Minimizing refusals sets $Ref_{TPR}(l)$ allows minimization of timed refusals graphs. The minimization procedure of refusals sets eliminates redundant information about refusals at any location.

Definition 5: Let $T_{RG} = (D, Ref_{TPR})$ be a timed refusals graph, l an element of L and let A, B be elements of $Ref_{TPR}(l)$,

$A \Subset B : \forall(\bar{a}, \emptyset) \in A, (\bar{a}, \emptyset) \in B \text{ and } \forall(\bar{\bar{a}}, G) \in A \text{ either } (\bar{\bar{a}}, G) \in B \text{ or } (\bar{a}, \emptyset) \in B$

.

The minimization of refusals set A produces a new set A' calculated for any location $l \in L$ is as follows:

1. $\forall A \in Ref_{TPR}(l) \text{ if } (\bar{a}, \emptyset) \in A \text{ and } (\bar{\bar{a}}, G) \in A \text{ then remove } (\bar{\bar{a}}, G) \text{ from } A$
2. Minimize $Ref_{TPR}(l)$ with respect to the relation $\Subset$.

In fact, if a set A of refusals is an element of $Ref_{TPR}(l)$, and both permanent and temporary refusals on action a are in A. It means that a system may be in a state when action a is definitely refused or temporary refused.

Then, no way permits to ensure that action a will be offered after a laps of time. Which justifies the remove of temporary refusals of action a in the set A. In the second step, if $A \Subset B$ so A is removed. In fact, the refusals in B contain the refusals in A.

The timed refusals graph (TRG) is said minimal if the refusals set Ref_{TPR} remains unchangeable by the application of Step1 and Step2 for any locality $l \in L$.

IV. Reductions of Timed Refusals Graphs

The behavior of DATA* (respectively its TRG) can be captured by a temporized finite state machine, named regions automaton. In which states are formed in a pair by locations and clock regions. Regions are equivalent classes of clock valuations function. Nevertheless, the complexity of implementing regions automata is exponential in the number of clocks and in the length of timing constraints [16], [25].

A. Aggregate Regions Automata of DATA*

In previous work [10], we have defined an aggregation operation on regions automaton states, using an equivalence relation, and it serves to group regions. This aggregation reduces significantly the graph size. For this purpose we have proposed an algorithm implementing the aggregation relation starting from an initial partitioning of states. The generated aggregate regions automaton preserves the reachability property, thus the reachability question on DATA* model is reduced to reachability question about aggregate regions automata.

Definition 6: Let D= $(L, l_0, X, T_D, L_S, L_f)$ be a DATA*, its aggregate regions automaton $ARA(D) = (S, s_0, T_R)$ over *ACT*, is defined as follows:

All states of $ARA(D)$ are of the form $s_{ij} = (l_i, r_j)$ where l_i is a location and r_j is a clock region. The set of states is noted *S*. The initial state is $s_0 = (l_0, r_0)$.r_0 is the initial region where every clock is initialized by zero. $s_{ij}^f = (l_i, r_j)$ is a terminal state iff $l_i \in L_f$.

The set of transitions T_R is,

$$T_R = \left\{ t' / t' = (l, r) \xrightarrow{a} (l', r') \middle| \begin{cases} \exists l \xrightarrow{g,a,x} l' \in T_D \text{ and } \exists r'' \in succ(r) \\ \text{such as } r \subseteq g \text{ and } r' = r''[x \leftarrow 0] \end{cases} \right\} \quad (8)$$

While the regions graph associated to the DATA * was creating, the aggregation operation revealed symmetrical aspects of clock regions. Indeed, because of the causal dependence of actions when considering durations of actions, the guards of the transitions have a particular form also the clock reset on the start of action execution.

These two characteristics allow us deducting the form of regions and their successors which verify guards and clock rest at each point of time. To group localities (s, r_i) consists on

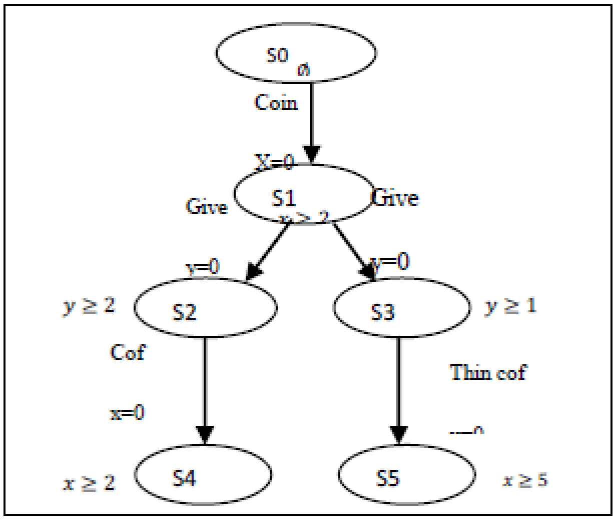

Fig. 5. DATA* M of coffee machine

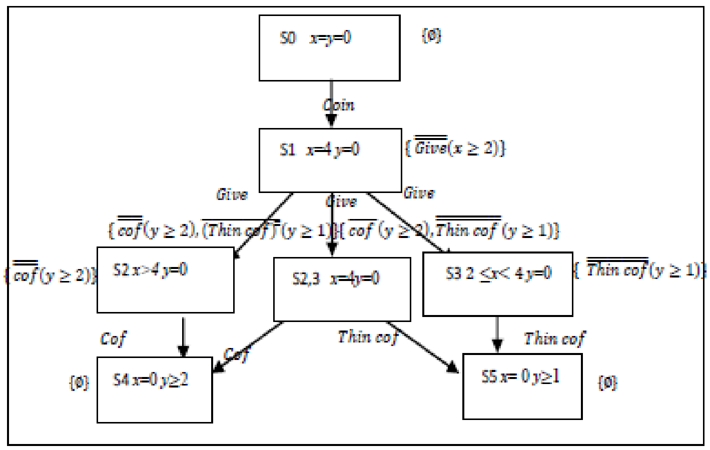

Fig. 6. TRRG M' associated to DATA* M

creating a new locality (s, R) such that R is a summation over regions on localities r_i. In fact this new region is none other than the guard G with the associated clock reset of the corresponding transition in DATA*.

For reduction of TRG structure we have to use an adapted form for algorithm proposed in [10]. We note that transformed TRG (in an aggregate regions automata augmented by refusals sets) will be named TRRG..

As illustration, let consider the DATA* M of coffee machine depicted by Fig. 5.

The timed refusals regions automaton M' associated to DATA* M is depicted by Fig. 6.

V. Implementation

The proposed approach was implemented using graph transformation [26]. 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 [27] are used for model transformation [11]. They are composed of production rules; each having graphs in their left and right hand sides (LHS and RHS)

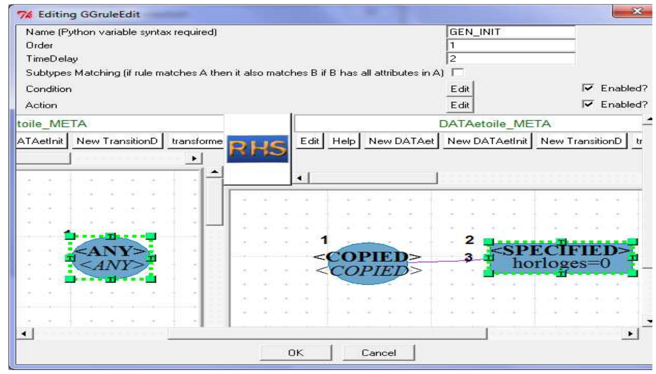

Fig. 7. Example of grammar rule in AToM3 (LHS and RHS)

(Fig. 7). AToM^3 [11] is a graph transformation tool among others. In this paper we use it.

To explain how the tool works, we propose the example of ATM system. This machine allows withdrawing money from account. Its behavior is as follows: Customer has to insert card in machine. After he has to type a code, if code is correct, the machine delivers money and card. If the code is wrong, the machine can keep card or reject it.

Fig. 8 presents a DATA* of ATM system. We have applied our tool on the DATA* model and obtained automatically the TRRG (Fig. 9).

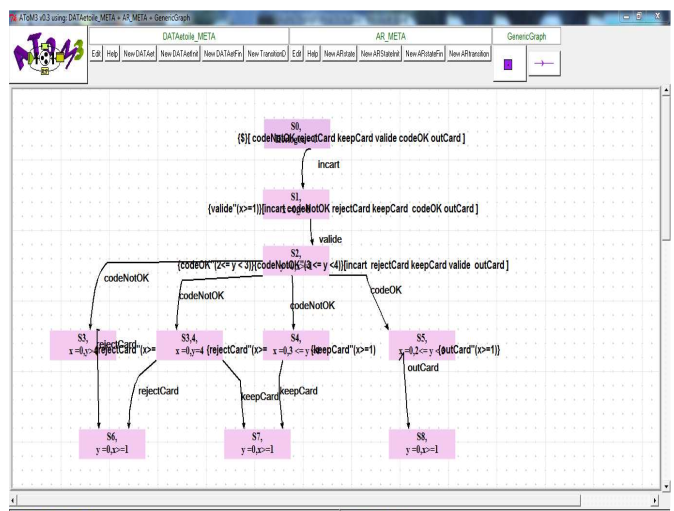

Fig. 9. TRRG associated to the DATA*

VI. Discussion

A. Related Work to Temporary Refusals

The consideration of temporary refusals in testing is a question that has been addressed in the literature since 1981 [15]. For instance, Langerak [14] considers system which may refuses some actions, however these refusals may disappear after applying extra events on it. In this theory, the origin of temporary refusals is unknown and extra events are needed to eliminate this lock.

Tretmans in [22] has defined the notion of quiescence in system behaviors, this situation may occur when a system executes a cyclic sequence of silent actions. To distinguish between quiescence situation and temporary refusals, Brinksma and al propose in [6] an extension of the conformance relation for real time systems and introduce a notion of quiescence parameterized by upper bound of duration for this lock. While this period has not expired, the refusal may be temporary, the system is considered in a quiescence location.

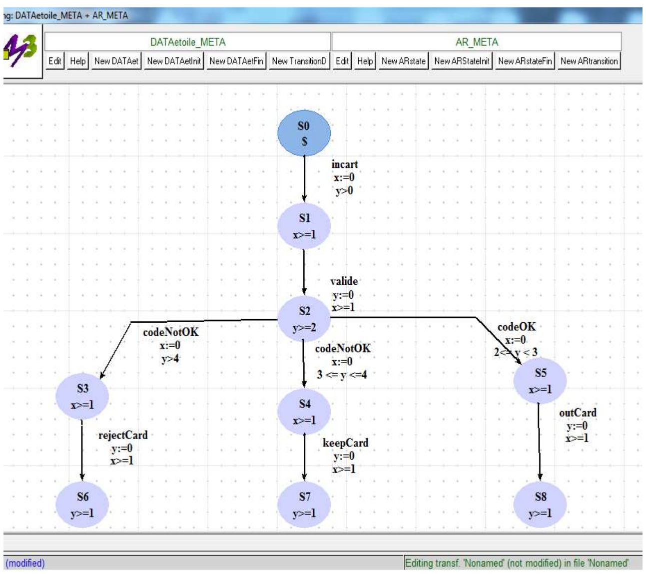

Fig. 8. DATA* of ATM system presented in AToM3 tool

B. Testing with TRRG

In our case we introduce the use of TRRG in testing timed systems specified by DATA*. It's known that the use of timed model introduces several technical difficulties in testing since the durations associated to the actions are represented by constraints. Therefore, the conformance relation must be reformulated and we have to take into account actions which elapse in addition to temporal requirements.

The use of the *TRRG* structure, permits to define an timed extension of conformance relation for *DATA**, based on *conf* relation defined in [28]. This relation was widely used in the practice of the test on Labeled Transitions Systems.

The TRRG considers two kinds of refusals, permanent and temporal refusals defined in section 2.4. Given the Fact that action have a duration represented by constraint in target state so the refusals are quantified.

We propose the following timed conformance relation named $conf_{tpr}$ defined as follows:

Definition 6:

σ is timed trace over Act,

$$I\ conf_{TPR}\ S \overset{\text{def}}{=} \forall \sigma \in TTraces(S) \begin{cases} \big(Forb(I,\sigma) \subseteq Forb(S,\sigma)\big) and \\ \big(Ref_{TPR}(I,\sigma) \subseteq Ref_{TPR}(S,\sigma)\big) \end{cases} \quad (9)$$

Intuitively, the relation presented in definition above holds between an implementation *I* and a specification *S*, if for every timed trace in the specification, the implementation does not contain unexpected deadlocks. It means that implementation and specification have the same timed traces in addition to forbidden actions (which are not allowed about current state noted *Forb* and the same sets of refusals (temporal or permanent).

The use of this notion of conformance makes *DATA** more expressive. And the implementation relation defined above can be refined and used explicitly for creating a tester for deriving test cases. As described in the beginning of the paper the canonical tester can be created on the structure of TRRG with respect of $conf_{TPR}$ for testing *DATA**.

VII. Conclusion

In this paper, we have introduced theory of refusal testing for a real-time systems modeled in Duration action time automata (DATA*). Next, we have proposed a timed refusals regions graph for DATA* specifications. In this variant of refusals graphs, temporary refusals are quantified, because actions have duration. Then, we have proposed a framework which generates timed refusals regions graph from DATA*. This framework is implemented by a combination of Meta-modelling and Graph Grammars, to transform a (DATA*) into a (TRRG) and to generate automatically a visual modeling tool. Finally, we discuss the use of the TRRG in testing theory. The timed refusal region graph is illustrated through an example.

As a perspective, we plan to use this result in order to construct a testing approach for real time systems by making explicit the conformance relation and to define how tests are applied to implementations. In this sense, we will construct a canonical tester based on TRRG; this will be done by deepening on the test selection method, in order to reduce the number of generated tests.

References

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

[2] M. Arous, D.E. Saidouni, J.M. Ilié, Maximality Semantics based Stochastic Process Algebra for Performance Evaluation.1st IEEE International Conference on Communications, Computing and Control Applications (CCCA'11) March 3-5, at Hammamet, Tunisia. IEEE Catalog Number: CFP1154M-ART, ISBN: 978-1-4244-9796-6 (2011)

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

[4] D. E. Saïdouni and N. Belala. Actions duration in timed models. In Proceedings of International Arab Conference on Information Technology (ACIT'2006). Al- Yarmouk University, Irbid, Jordan, December 19-21th 2006.

[5] 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 London Limited (2006)

[6] E. Brinksma, L. Brandan, and Briones. Test generation framework for quiescent real time systems. In J. Grabowski and B. Nielsen, editors, FATES, volume 3395 of LNCS, pages 64—78, Berlin Heidelberg, Springer-Verlag, (2005)

[7] E. Brinksma and J. Tretmans, Testing transition systems: An annotated bibliography. In 4th Summer School on Modeling and Verification of Parallel Processes, MOVEP 2000, LNCS 2067, pages 187–195. Springer, (2001)

[8] M. Clatin, R. Groz, M. Phalippou, R. Thummel, Two approaches linking test generation with verification techniques. In: A. Cavalli and S. Budkowski, editors, Eighth Int. Workshop on Testing of Communicating Systems. Chapman & Hall, (1996)

[9] K. Drira, P. Azema, P. de Saqui Sannes, Testability analysis in communicating systems. elsevier science B. V. computer networks 36, pages 671—693, (2001).Bravetti, M., Gorrieri R.: The theory of interactive generalized semi-Markov processes. Theoretical Computer Science, 282(1):5-32, (2002)

[10] 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)

[11] J. De Lara, H. Vangheluwe, AToM3: A Tool for Multi-Formalism Modeling and Meta-Modeling. Proc. Fundamental Approaches to Software Engineering, FASE'02, LNCS, vol 2306, pp. 174-188. Grenoble, France (2002)12. Lopez, N., Nuñez, M., Rodriguez I.: Specification, testing and implementation relations for symbolic-probabilistic systems. Theoretical Computer Science, 353(1-3):228-248,(2006)

[12] L. Lamport, Verification and specification of concurrent programs. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, A Decade of Concurrency, volume 803 of LNCS, pages 347—374. Springer-Verlag, (1994)

[13] R. Langerk, Bundle Event Structures: A non-interleaving Semantics for LOTOS. In Diez M. and Groz R (EDS), in proceedings of FORTE 92 , pp, 331-346 North Holland, (1993)

[14] R. Langerak, Transformations and Semantics for LOTOS. PhD thesis, University of Twente, Netherland, (1992)

[15] I. Phillips, refusal testing ; theoretical computer science : 241-284,1987. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195--197 (1981)

[16] 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)

[17] D.E. Saïdouni, N. Belala, M. Bouneb, Aggregation of transitions in marking graph generation based on maximality semantics for Petri nets. In Proceedings of the 2nd Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS'2008), University of Leeds, UK. BCS, July, 2-3rd (2008)

[18] D.E. Saïdouni,A. Benamira, N. Belala, F. Arfi, FOCOVE: Formal concurrency verification environment for complex systems. In Mediterranean Conference on Intelligent Systems and Application (CISA'2008). Annaba, Algeria, June 29-30th and July 1st 2008.

[19] D.E. Saïdouni, N. Belala, Using maximality-based labeled transition system model for concurrency logic verification. The International Arab Journal of Information Technology (IAJIT), 2(3):199—205, ISSN: 1683-3198, July (2005)

[20] D.E. Saidouni, I.kitouni, H. Hachichi and K. Bouarroudj, Durational Actions Timed Automata: Determinization and Expressiveness. Research report 12002, Misc laboratory, mentouri university Constantine, 25000, Algeria

[21] J. Tretmans, Testing Concurrent Systems: A Formal Approach. Jos C.M. Baeten, Sjouke Mauw (Eds.): CONCUR'99, LNCS 1664, pp. 46_65,. Springer-Verlag , (1999)

[22] J. Tretmans, Test generation with inputs,outputs and repetitive quiescence. In Software-Concepts and Tools,17(3) (1996)

[23] J. Tretmans, An overview of OSI conformance testing. Formal Methods Tools group University of Twente, January 25, 2001.

[24] E. Brinksma, Formal Methods for Conformance Testing: Theory Can Be Practical : N. Halbwachs and D. Peled (Eds.): CAV'99, LNCS 1633, pp. 44-46, 1999. Springer-Verlag Berlin Heidelberg (1999)

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

[26] H. Hachichi, I. Kitouni, D.E. Saidouni, Transforming DATA* with Dotty Format to Aggregate Region Automaton. International Journal of Computer 27 Applications,vol 37(10), pp 35-42 (2012)

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

[28] E. Brinksma , theory for the derivation of tests. In S. Aggarwal and K. Sabnani, editors, Proceedings of the 8th IFIP Symposium on Protocol Specification, Testing and Verification (PSTV 1988). North-Holland, (1989).

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.