Maximality-based Region Graph: a Novel Alternative

Riadh MATMAT Ilham KITOUNI Souad GUELLATI D-Eddine SAIDOUNI,
MISC Laboratory, Constantine 2 University, 25000, Algeria
{matmat; kitouni; guellati; saidouni}@misc-umc.org

Abstract – Timed automata with durational actions (daTA) are a form of timed automata that admit a more natural representation of durational actions and capturing true concurrency, with those additional benefits the kinds of properties to be verified on real-time systems will be enlarged. We present a novel approach to construct a region graph, based on the maximality semantics and preserving that specificity. More precisely, we renew the maximality capabilities on the region graph; we propose the MRG construction algorithm. We also describe an implementation of this construction (TaMaRG tool). The approach is illustrated by means a case study.

Keywords – Real time systems, Durational Action Timed Automata, Maximality semantics, Maximality-based region graph.

1. INTRODUCTION

Nowadays, formal verification becomes increasingly used in the industry as a part of the design process. There is a growing need for efficient tools dealing with real aspects of systems.

Model Checking [11] is one of the most automated and powerful technique, used when designing and debugging critical reactive systems. It has been extended to real-time systems in which quantitative (timed) requirements are essential.

In general, the system is firstly described using a high level specification models (as timed Petri net [12], timed automata [2], the specification is then translated to an abstract representation (as region graph [2]). This latter is used in several validation methods.

Timed Automata (TA) [2] was proposed to specify quantitative requirements expressed by timed constraints [2], they are an extension of finite state machine with a finite number (but arbitrary) clocks in continuous time. TA is very suitable for modeling and verifying real-time systems. They stick a good balance between expressiveness and tractability and they are supported by different verification tools (e.g., KRONOS [19] and UPPAAL [10]).

In spite of this, the model suffers from many problems (decidability, state space explosion...) which is impeding its scalability.

Many studies have pointed the interest of TA, its underlying semantic is said the interleaving semantics [21].

Decidability proof of the reachability problem in TA model is based on a finite abstraction of the set of the clock valuations. The result of this abstraction is the region graph [2]; it is obtained by the construction of a finite partition of all valuations, with respect of clock constraints. Partitions are also compatible with the progression of time and reset.

It's obvious that in real world systems, actions are not instantaneous and have durations. This realistic characteristic is important in many cases.

Instead of the interleaving semantics, maximality semantics has been proved necessary and sufficient for carrying both the refinement process and action durations [19]. Accordingly,

models based on maximality semantics present concurrent actions differently from choice [11], because of non atomicity of actions. These models advocate modeling durational actions without splitting them.

Durational actions timed automata model (daTA) was proposed as a timed automata coated by the maximality semantics. daTA model allows to carry durations of actions and to handle true concurrency; which are realistic assumptions for specifying timed systems.

Indeed, to model duration of actions, every edge of the automaton is annotated by constraints on clocks. Implicitly constraints on the edges contain the durations of actions, those that are already started. In addition, other real-time aspects are considered (like delaying the start and deadlines). A single clock is reset on every edge; it corresponds to the beginning of event (action). The termination of an action will be captured by information on locations of the automaton, precisely on the destination location of transition.

In reality, the action duration is represented by both in the constraints (guards) of the following edges and on the next locations and that means action is not over yet. To better understand these principles, consider two actions, if their executions are dependent, then one should expect the termination of the other, this will be realized by the use of the duration of the first action in the guard of the second one. Otherwise they are in concurrency (parallel executions). See the example depicted Figure 1.

Another important aspect of real-time systems is the urgency (i.e., actions whose execution cannot be delayed beyond a certain time bound). In daTA model urgency is represented by deadlines as proposed in [17]. The original proposition establishes results on timed safety automata [8]. This notification of action's urgency is more natural and has the advantage to avoid a lot of cases of time locks [28]. Indeed deadlines replace invariants as time progress conditions (TPC). Deadlines are clock constraints associated directly with edges in the automaton. Thus, in daTA, every state either allows time to pass or allows actions to be executed (i.e., daTA are time-reactive).

Unfortunately, the region graph of TA doesn't contain information about durations or parallel execution of actions; those limitations motivate the development of Maximality based Region Graph (MRG) from durational action timed automata.

Our contribution: In order to preserve all achievements of the maximality semantics namely duration of actions and true concurrency, we will use the data model and information concerned by the maximality when developing the region graph. We are interested by the preservation of the maximality semantics even on the abstracted graph of executions of daTA. The fundamental interest of this approach is to propose a structure based on classical regions for model checking and all other validation requirements of concurrent real-time systems (possibly distributed).In this paper we expand a theory of region abstraction to capture novel aspects which are relevant for verifying true concurrency and resulting from the maximality semantics.

In fact for the model checking based on daTA model, the range of proprieties to be verified is enlarged to capture actions incompatibility, auto concurrency and the concurrency degree specification [9].

For this purpose we propose an algorithm for generating Maximality-based Region Graph (MRG). We also describe an implementation of this algorithm, which allows the construction of the MRG and its transformation into dotty grammar. Finally, the tool is experimented by means a case study in order to illustrate its functionalities and to show its capabilities to deal with real systems.

Paper Outline: duration actions timed automata are recalled in Section 2. Definition and formalization of Maximality based Region Graph are proposed in Section 3. Section 4 presents the construction algorithm of MRG with an illustrative example. Section 5 describes the associated tool (TaMaRG) and the major functionalities. Also, a case study is presented and results are commented in this section, namely sender parts of alternative bit protocol. Section 6 concludes the paper and gives some perspectives on future work.

2. TIMED AUTOMATA WITH DURATION OF ACTIONS

2.1. daTA model

The daTA model (durational actions timed automata) [24] is a timed model defined by a timed transition system over an alphabet representing actions to be executed. This model

takes into account in the specification, the duration of actions based on an intuitive idea: temporal and structural non-atomicity of actions. This model seems interesting and funneling more and more research because it coats the timed automata model by the maximality semantics [19].
Illustrate this model by an example (Figure 1):

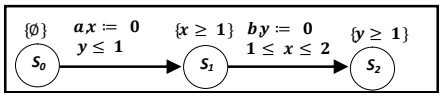

Figure 1: daTA(A)

The duration associated to the action is represented by constraints on the transitions and in the target states of each one. 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 clock is associated to an action. This clock is reset to 0 at the start of the action and will be used in the construction of the temporal constraints as guard of the transitions.
Fig.1 presents a system of two consecutives actions a and b, the clock x is associated to the action a, on the locality s_1 the temporal formula $\{x \geq 1\}$ represents the duration of the action a (which is important to distinguish from invariant in timed automata).
The end of an action execution 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 $\{1 \leq x \leq 2\}$
Starting from this, we can express different possibilities of real time systems behaviour like delaying execution of action or limiting its offering time by manipulation clocks constraints.
Consider another example depicted by Figure 2; the system P consists of two concurrent processes $P1$ and $P2$ synchronizing on an action d. The process $P1$ executes the action a followed by d, while $P2$ executes b then d, and suppose that actions a, b and d have respectively 10, 12 and 4 units of time as durations.
In daTA of Fig.2, from the state s_3, the actions a and b can comply in parallel, and each one can finish only if its clock reaches a value equal to its duration, from where duration condition set $\{x \geq 10 y \geq 12\}$.

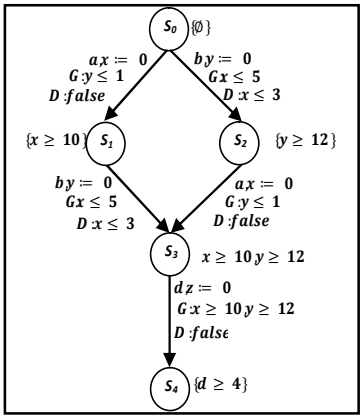

Figure 2: daTA(B)

The guard of the action d becomes $x \geq 10 \wedge y \geq 12$. Once this latter is satisfied, d can start at any time in the enabling open interval $x \in [10, +\infty[$, $y \in [12, +\infty[$, the so called enabling domain.
Another concept of real time systems is urgency in our proposal it's represented as deadlines [8][17].
We also assume that $D \Rightarrow G$, this condition guarantees that if time cannot progress, at some state at least one action is enabled from this state.

2.2. Formalization

In the following $\mathbb{R}^+$ is the set of nonnegative real numbers. A clock takes values from $\mathbb{R}^+$. Given a set $\mathcal{H}$ of clocks, a clock valuation over $\mathcal{H}$ is a function assigning a non negative real number to every clock. The set of valuations of $\mathcal{H}$ noted $V(\mathcal{H})$ is the set of total function from $\mathcal{H}$ to $\mathbb{R}^+$. A valuation v is a mapping on $\mathcal{H}$ to $\mathbb{R}^+$. Let x be a clock, the valuation $v[x \leftarrow 0]$ resets clock x to 0 and each other clock y to $v(y)$. The valuation $v + d$ maps every clock y to $v(y) + d, d \in \mathbb{R}^+$.
Clock constraints over $\mathcal{H}$ are defined by the following grammar: $C ::= true \mid false \mid x \sim c \mid C \wedge C$, where $x \in \mathcal{H}$, $c \in \mathbb{N}$ and $\sim \in \{<, >, \leq, \geq\}$. The satisfaction with respect to clock valuation function written $v \vDash C$ means that according the valuation function v , C evaluates to true.
We use $C(\mathcal{H})$ to denote the set of clock constraints, ranged over by G and also by D.
We also use a subset of constraints where only the atomic form of clocks comparison is allowed. This set is defined by $C_d(\mathcal{H})$ by the grammar: $C ::= x \geq c$, where $x \in \mathcal{H}$ and $c \in \mathbb{N}$. This subset represents condition duration over a finite set of actions noted Act .

Definition2.1. A daTA is a tuple $A = (S, s_0, \mathcal{H}, L_S, T)$ over ACT, where S is a finite set of states, $s_0 \in S$ is the initial state, $\mathcal{H}$ is a finite set of variables named clocks. $T \subseteq S \times C(\mathcal{H}) \times C(\mathcal{H}) \times Act \times \mathcal{H} \times S$ is a finite set of transitions.

Given a transition $t = (s, G, D, a, x, s') \in T$, represents an edge from location s to s' that launch the execution of action a whenever guard G becomes true. In addition, deadline D imposes an urgency condition: the transition cannot be delayed whenever D is satisfied, x is a clock to be reset at this transition.

Finally, $L_S: S \to 2^{Cd(\mathcal{H})}$ is a maximality function which decorates each state by a set of timed formula named action durations; these actions are potentially in execution on it. $L_s(s_0) = \emptyset$ means that no action is yet started.

We define Clock Label Occurrence $CLO: C(\mathcal{H}) \to 2^{\mathcal{H}}$, as a function which gives clock names occurred in a given timed formulas, recursively by:

$$\begin{cases} CLO(true) = CLO(false) = \emptyset \\ CLO(\{x \sim c\}) = \{x\} \\ CLO(F_1 \wedge F_2 \wedge \ldots \wedge F_n) = \bigcup_{i=1..n} CLO(F_i) \end{cases}$$

Such as:

$F_i \in C(\mathcal{H}), x \in \mathcal{H}, \sim \in \{<, >, \leq, \geq\}$ and $c \in \mathbb{N}$.

It is a function which gives clock names occurred in a given timed formulas.

Definition2.2. The semantics of a daTA A is a timed transition system $TTS_A = (Q, q_0, \to)$ over $ACT \cup \mathbb{R}^+$. A state of TTS_A (or configuration) is a pair (s, v) such as s is a state of A and v is a valuation over $\mathcal{H}$, where:

- $Q = \{(s, v) \mid s \in S \text{ and } v \in V_{\mathcal{H}}\}$;
- $q_0 = (s_0, v_0)$ such that $\forall x \in \mathcal{H}, v_0(x) = 0$;
- $\to \subseteq Q \times (Act \cup \mathbb{R}^+) \times Q$ consist of the discrete and continuous transitions.

The discrete transition is defined for all $t \in T$ by $R1 = \frac{(s, G, D, a, x, s') \in T \; v \models G}{(s, v) \xrightarrow{a} (s', v[x := 0])}$.

The continuous transition is defined for all $d \in \mathbb{R}^+$ by $R2 = \frac{d \in \mathbb{R}^+ \; \forall d' < d, v + d' \models TPC(s)}{(s, v) \xrightarrow{d} (s, v + d)}$, where $TPC(s) = \neg \forall (D \mid \exists t \in T: t = (s, G, D, a, x, s'))$ is the time progress condition in s [7] [13].

Rule $R1$ states that an edge $s \xrightarrow{G, D, a, x} s'$ defines a discrete transition from current location s whenever the guard holds in current valuation v and clock x is reset to 0. According to $R2$, time can progress in s only when $TPC(s)$ is true, that is as long as no deadline of an edge leaving s becomes true.

3. DEFINITION OF MAXIMALITY-BASED REGION GRAPH

Characterization of equivalent behavior using the temporal maximality semantics and verification by enumeration (model-checking) must pass through abstractions of durational actions timed automata, that both:

- Generate a finite graph (finite system transitions);
- Preserve the degree of parallelism;
- Preserve the system properties.

Several verification problems such as reachability analysis, untimed language inclusion, language emptiness as well as timed bisimulation can be solved by techniques based on the region abstraction [2].

3.1. Clock regions

An infinite number of distinct valuations can verify exactly the same guards in daTA. This observation will serve us to analyze the daTA. The restriction on clock constraints that define the guards and deadlines are used to define a finite number of equivalence classes of clock valuations (called regions) that can unfold a duration actions timed automata in a finite automaton, called Maximality-based Region Graph (MRG). This property allows algorithmic assessments of the MRG. These valuations correspond to those of the corresponding infinite automaton.

Notation: for $x \in \mathcal{H}$, for any $v(x) \in \mathbb{R}^+$, $fract(v(x))$ denotes the fractional part of v, and $\lfloor v(x) \rfloor$ denotes the integral part of v.

Definition3.1. Let $A = (S, L_S, s_0, \mathcal{H}, T)$ be a daTA. For $x \in \mathcal{H}$, let c_x be the largest integer c such that $(x \leq c)$ or $c \leq x$ is a sub-formula of some clock constraint appearing in T.

The equivalence relation $\cong$ is defined over the set of all clock interpretations for $\mathcal{H}$; $v \cong v'$ iff all the following conditions hold:

i. For all $x \in \mathcal{H}$, either $\lfloor v(x) \rfloor$ and $\lfloor v'(x) \rfloor$ are the same, or both $v(x)$ and $v'(x)$ are greater than c_x.
ii. For all $x, y \in \mathcal{H}$ with $v(x) \leq c_x$ and $v(y) \leq c_y$, $fract(v(x)) \leq fract(v(y))$ iff $fract(v'(x)) \leq fract(v'(y))$.
iii. For all $x \in \mathcal{H}$ with $v(x) \leq c_x$, $fract(v(x)) = 0$ iff $fract(v'(x)) = 0$

The equivalence relation $\cong$ is defined over the set of all clock interpretations for $\mathcal{H}$. We will use $[v]$ to denote the equivalence classes of $V(\mathcal{H})$ to

which v belongs. It also clock regions denoted by: $r = [v] = \{v' \in V(\mathcal{H}) | v \cong v'\}$.

Example 3.1. Consider a daTA A with two clocks x and y with $c_x = 2$ and $c_y = 1$.

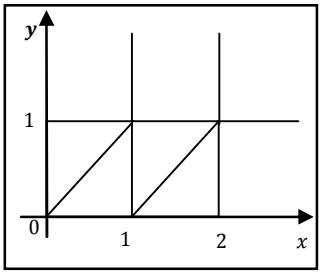

Figure 3: Clock regions

The clock regions shown in Figure 3 are:

- 14 open line segments: e.g. $[0 < y = x < 1]$;
- 8 open regions: e.g. $[0 < y < x < 1]$;
- 6 corner points: e.g. $[x = 1, y = 1]$.

We call end-region the region satisfying the following condition: $\forall x, x \in \mathcal{H} \Rightarrow x > c_x$. The end region is open to infinity on all clocks region. It does not have a successor region, in Figure 3 the end-region is $[x > 2, y > 1]$.

3.2. Successor function

Let r and r' be regions. The region r' is a successor of r, noted $succ(r) = r'$ iff $\forall v \in r \Rightarrow \exists t \in \mathbb{R}^+$ and $v + t \in r'$.

Now we are ready to define the Maximality-based Region Graph associated to a daTA $A = (SL_S, s_0, \mathcal{H}, T)$.

3.3. Regions Graph Coated of Maximality

Let $MRG(A)$ be a Maximality-based Region Graph of a daTA A. Similar to region automaton [2], a state of $MRG(A)$ records the state of the timed transition of A, and the equivalence class of the current values of the clocks, but also it conserve the information of actions under execution. It is of the form (s, r) with $s \in S$ and r is a clock region. The Maximality-based Region Graph starts in some state (s_0, r_0).

According to the maximality semantics and the daTA structure, each state of Maximality-based Region Graph contains a set of clocks which corresponds to events for actions under executions. Those events are captured by timed formulas initially on states of daTA (delivery by L_S function). The function $U_s(s, r)$ presented in definition 3.2 ensures this feature.

The transitions relation of $MRG(A)$ is an extension of transitions relation defined for a region automaton. It has an edge from (s, r) to (s', r') , labeled with $CLO(G_i), a_i, x_i$, where x_i is a clock associate to the action a_i and $CLO(G_i)$ the set of clock names corresponding to the actions conditioning the execution of a_i. This is effective if in daTA a transition of the form $(s, G_i, D_i, a_i, x_i, s') \in T$. As usual the transitions relation can be defined using a successor function over the clock regions and some maximality functions.

Definition3.2. Let $A = (SL_S, s_0, \mathcal{H}, T)$ be a daTA. The Maximality-based Region Graph associated to A is $MRG(A) = (L, l_0, U_s, T^r)$ where:

i. L, is a set of states regions having the form (s, r) with $s \in S$ and r is a region.
ii. $l_0 = (s_0, r_0)$, is the initial state region, where s_0 is the initial state for A and $r_0 = [v_0(x)]$ such as $\forall x \in \mathcal{H}, v_0(x) = 0$.
iii. $U_s : S \times R \to 2^{\mathcal{H}}$, is a maximality function assigning to each state region (s, r) a finite set of clocks representing maximal event names. It's defined as: $U_s(s, r) = \{x | x \geq c \in U_s(s)\}$.
iv. The transition relation T^r is defined by two kinds of transitions:

- *Passage of time*. Each region (s, r) , where r is not an end class, has an edge to its successor $(s, succ(r))$. Formally we write:

$$\frac{r \vDash TPC(s) \wedge succ(r) \vDash TPC(s)}{(s, r) \xrightarrow{d} (s, succ(r))}$$

- *Transition of A* . Given a region (s, r) , for each transition $t = (s, G, D, a, x, s')$, there is an edge to (s', r') where: $r' = r[x := 0]$, $r \vDash G$ and $r \vDash TPC(s')$. This is formalized by the rule:

$$\frac{s \xrightarrow{G, D, a, x} s' \wedge r \vDash G \wedge r[x := 0] \vDash TPC(s')}{(s, r) \xrightarrow{CLO(g), a, x} (s', r[x := 0])}$$

4. Generating Maximality-based Region Graph

In this section we propose an algorithm which constructs the Maximality-based Region Graph for durational actions timed automata. It is based on the successor function and maximality semantics to create new regions by using the daTA structure.

The Algorithm: Initially the daTA is traversed so as to find c_x for each clock $x \in \mathcal{H}$. This is required for abstraction and ensuring finite number of states regions in the MRG. After creating initial state region l_0 of MRG, the algorithm calculates the successor regions. Regions are specified by one of three forms (type_1, type_2, type_3).

The function $succ$ returns a successor of current region. In this step the algorithm checking under the successor region guard and TPC of daTA for creating regions states and transitions, the regions states are creating with keeping all actions under execution, this is performed by the function U_s, the transitions are created either by passage of time(delay transitions) or by actions (discrete transitions).

Delay transition is created if the current region and its successor satisfies the TPC, while discrete transitions is created if the region satisfies the guard and satisfies the TPC, in this case, we keep (on the transition) clock associated with the action and clocks corresponding to actions under execution. Recall that those actions condition the launching of current transition. This is performed by the function CLO.

On states regions the function $U_s(sr)$ is constructed, to capture the maximal events and to ensure maximum information on concurrency. The MRG generated by this algorithm correspond to daTA of Figure 1, is represented by Figure 4.

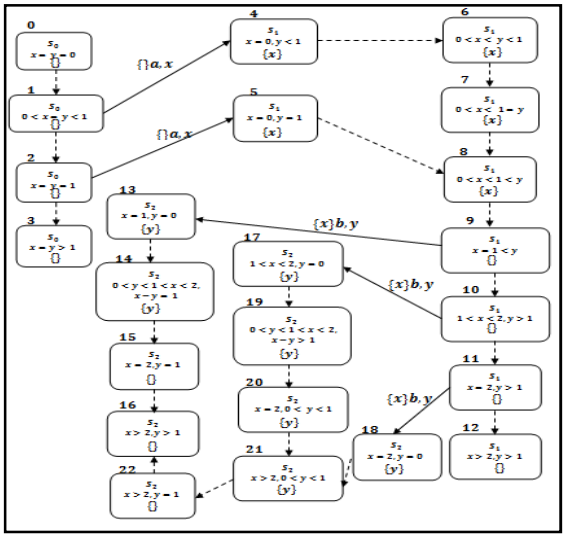

Figure 4: MRG correspond to daTA (A)

Algorithm 1 Construction of Maximabased Re gion Graph
Input : durational actions timed automata A
Output : Maximal-based Region Graph correspondi ng to A

1: calculate c_x for each clock $x \in \mathcal{H}$.
2: let l_0 be the initial state region of MRG
3: $P \leftarrow \emptyset$;
4: enqueue(P; l_0); // $l_0 = (s_0 r_0)$
5: **while** P not empty **do**
6: l = dequeue(P); // $l = (sr)$
7: **if** r is not end region **then**
8: **Switch** type_region(r)
9: **case** type_1 $r' = succ(r, type\_1)$;
10: **case** type_2 $r' = succ(r, type\_2)$;
11: **case** type_3 $r' = succ(r, type\_3)$;
12: **if** $r \vDash TPC(s) \wedge succ(r) \vDash TPC(s)$ **then**
13: $l' = (sr')$
14 $U_s(l') := \{x | x \geq c \in L_s(s)\}$
15: add l' to L ;
16: add t_a^r to T^r;
17: $t_a^r : l \xrightarrow{a} l'$;
18: enqueue(P; l');
19: **end if**
20: **for all** $t_i = (sG_i D_i a_i x_i s_i)$ **do**
21: $r' \vDash G \wedge r'[x := 0] \vDash TPC(s_i)$ **then**
22: $l_i = (s_i, [x_i := 0]r')$;
23 $U_s(l_i) := \{x | x \geq c \in L_s(s_i)\}$
24: add l_i to L;
25: $t_a^r \xrightarrow{CLO(G_i) a_i x_i} l_i$;
26: add t_a^r to T^r;
27: enqueue(P; l_i);
28: **end if**
29: **end for**
30: **end if**
31: **end while**

5. Implementation and case studies

5.1. Implementation

TaMaRG is a tool for generating Maximality-based Region Graph from durational actions timed automata. This tool has a graphical editor to draw and edit daTA structure Figure 6. The MRG-generator generates a maximality-based region graph, and MRG-Dotty adaptor produces a dot file type as showed in Figure 7. Its functional view is sketched in Figure 5.

The input is a daTA of the intended behavior of the system under verification, the output is an MRG.

The entire tool was implemented using java as programming language.

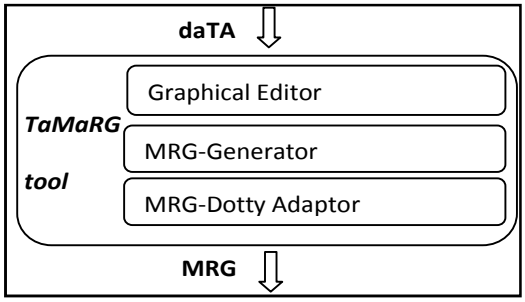

Figure 5: Functionalities of TaMaRG tool

5.2. Case studie

ABP Protocol: ABP (Alternating Bit Protocol) is a connection-less protocol for transferring messages in one direction between a pair of protocol entities. It is a simple form of the Sliding Window Protocol with a window size of 1 [23], [16] .The name of this protocol stems from the fact that each message is augmented with an additional bit. This protocol uses one-bit sequence number (which alternates between 0 and 1) in each message and an acknowledgment to determine whether the message must be retransmitted.

The aim of this section is to illustrate the use of daTA tool; we will focus on sender part.

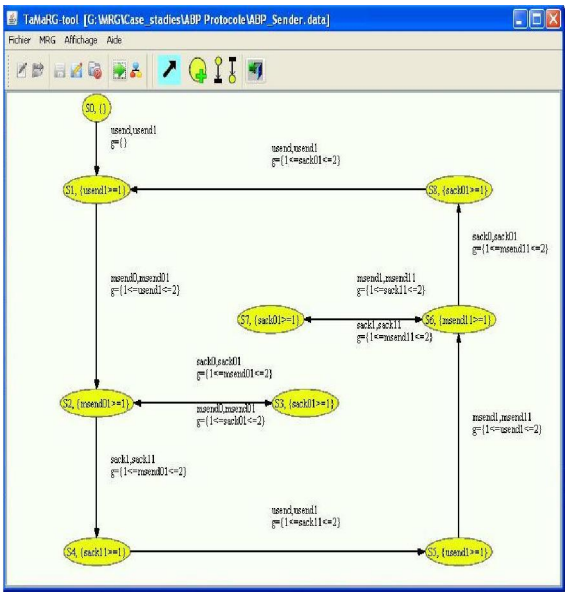

Figure 6: daTA of sender part of ABP protocol

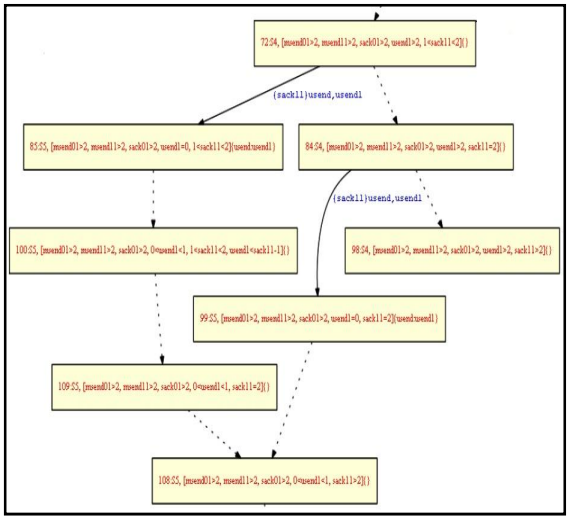

Figure7: Part of MRG of daTA of sender part of ABP

The Model: The alternating bit protocol consists of a sender S, a receiver R, a channel K from S to R and a channel L from R to S, its architecture is as follow:

The sender S and receiver R communicate via the same lossy communications mediums (L and K). So messages may be lost. The basic principle is to stamp messages with a one bit sequence number. When a protocol entity sends message (either Data or Acknowledgment) with sequence number b. The next message it receives should be $\neg$b. If the sequence number is not as expected, the protocol entity concludes that the message has been lost and retransmits.

In this paper, we will focus on the sender part of ABP protocol. To explain how the tool work.

The Sender Part of ABP Protocol: is responsible for accepting messages from the application and sending them via the Medium to the Receiver. The daTA of sender is presented in Figure 6.

Table 1: MRG Result for ABP Protocol

	S	R	K	L
State	151	224	276	152
Transition	195	261	361	198

6. Conclusion

In this paper, we defined Maximality-based Region Graph, also an algorithm generating

MRG structure from durational actions Timed Automata (Timed Automata which are coated of Maximality semantics) is presented. The interest of MRG model comes from the fact that it contains information about parallel execution of actions. This allows verification of properties related to true concurrency, in addition to those associated to real-time systems.

As perspective, we will complete this work by the characterization of equivalent behaviors using the temporal maximality bisimulation. Likewase we aim at the reduction of the MRG size for efficient verification by enumeration (model-checking).

7. References

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

[2] R. Alur, and D. Dill,"A Theory of Timed Automata",In Theoretical Computer Science (TCS),vol. 126(2), pp. 183–235, 1994.

[3] C. Baier, J-P. Katoen,"Principles of Model Checking"; The MIT Press Cambridge, Massachusetts, London, England: ISBN 978-0-262-02649-9, 2008.

[4] H. Bowman, Time and action lock freedom properties for timed automata. In Proceedings of FORTE 2001, 119–134, 2001.

[5] A. Burns, "how to verify a safe real time: the application of model checking and timed automata to the production cell case study", In Real time Systems journal, vol. 24(2), pp. 135-152, 2003.

[6] 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.

[7] 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.

[8] T.A. Henzinger, X. Nicollin, J. Sifakis, & S.Yovine, "Symbolic model checking for realtime systems". Inf. & Comp., 111(2):193–244, 1994.

[9] S. Guellati, I. Kitouni, R. Matmat and D.E. Saidouni, "Timed Automata with Action Durations - From Theory to Implementation", In the 20th International Conference on Information and Software Technologies (ICIST 2014), Kaunas, Lithuania, G. Dregvaite and R. Damasevicius (Eds.): ICIST 2014, CCIS 465, pp. 94-109, 2014. Springer International Publishing Switzerland 2014. (Accepted to be published).

[10] G. Behrmann, A. David, and K. Larsen. "A tutorial on Uppaal". In SFM-RT 2004, LNCS 3185,pages 200–236. Springer, 2004.

[11] 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.

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

[13] J.F. Groote, J. Springintveld,"Focus points and convergent process operators". A proof strategy for protocol verification, In Journal of Logic and Algebraic Programming, 49(1/2):31{60}, 2001.

[14] Holton, D. R. W., "A PEPA Specification of an Industrial Production Cell", In The Computer Journal, vol. 38 (7), pp. 542-551, 1995.

[15] K. Bouaroudj, DE. Saidouni, and I. Kitouni, "Testing Stochastic Systems Using MoVoS Tool: Case Studies". T. Skersys, R. Butleris, and R. Butkiene (Eds.): ICIST 2013, CCIS 403, pp. 310–321, Springer-Verlag Berlin Heidelberg, 2013.

[16] A. Hessel, K. Larsen, B. Nielsen, P. Pettersson, andA. Skou, "Time-optimal real time test case generation using UPPAAL", In FATES'03, Montreal, October 2003.

[17] R. Gómez, "Model-checking timed automata with deadlines with Uppaal". Formal Aspects of Computing, vol 25 (2), pp 289-318, 2013.

[18] S. Guellati, I. Kitouni, and D.E. Saidouni, "Verification of durational action timed automata using UPPAAL", In International Journal of Computer Applications (IJCA), vol. 56 (11), pp. 33-41, Published by Foundation of Computer Science, New York, USA, October 2012.

[19] S. Yovine, " Kronos:A verification tool for real-time systems", InInternational Journal of SoftwareTools for Technology Transfer, 1(1-2), pp. 123–133, 1997.

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

[21] 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.

[22] S. Tyszberowiez, "How to implement a safe real-time system : The OBSERV implementation of the production cell case study" In Real time Systems, vol. 15(1), pp. 61-90, 1998.

[23] K.H. Talukder," Formal verification of the Alternating Bit Protocol", In 6th International Conference on computer & Information Technology (ICCIT), 2003.

[24] I. Kitouni, H. Hachichi, K. Bouaroudj, and D.E. Saidouni, "Durational Actions Timed Automata:Determinization and Expressiveness", In (IJAIS), vol. 4(2), pp. 1-11,Published by Foundation of Computer Science, New York, USA, September 2012.