Actions Duration in Timed Models

Djamel Eddine Saïdouni and Nabil Belala

LIRE Laboratory, University of Mentouri, 25000 Constantine, Algeria
saidounid@hotmail.com, nbelala@gmail.com

ABSTRACT

This paper introduces a real-time model based on a true-concurrency semantics, expressing parallel behaviors and supporting at the same time timing constraints, explicit actions durations, structural and temporal non-atomicity of actions and urgency. This model is called Durational Action Timed Automata. As an application, we propose translating rules from D-LOTOS language specifications to DATA*'s structures.*

Keywords: *Real-time systems, Actions duration, Maximality-based semantics, DATA*'s*

1. INTRODUCTION

Specification of real-time systems is a quite difficult process since these systems are known to be complex and critical. Formal models are usually used to specify behaviors and verify some expected properties; one can cite timed extensions of Petri nets [24,26], process algebras as TCCS, ET-LOTOS, RT-LOTOS, D-LOTOS [14,15,21,25,31] and state-transition models like *timed automata* [1,2] which extend state-transition graphs with timing constraints using a set of real-valued clocks. In this context, many questions have been raised and studied in the literature; an important one is how to give semantics to a specification model to be able to express concurrent and parallel behaviors in a natural way, i.e. to distinguish between sequential and parallel runs of actions. This is not the case of the interleaving semantics: to use this latter advisedly, actions must be temporally and structurally atomic (actions are indivisible and of null duration).

Another question concerns the expression of non-null duration actions. To do this, most of works consider actions as two instantaneous events: their start and their completion, in addition to the wait between these events. Although this approach seems to be attractive, it may contribute toward graph explosion in state-transition models. Timed automata model and most of their sub-classes and extensions opt for splitting actions up into start and completion events. Among timed automata sub-classes, we can quote *Timed Safety Automata* [20] in which a state of an automaton can contain local timing clock constraint called *invariant*, *Event-Recording Automata* [3] in which a corresponding clock x_a is reset automatically with each occurrence of an action a, *Dynamic Timed Automata* [13,22] including a set of clocks no longer global in all the system but local in each automaton state, *Timed Automata with Deadlines* of [8,9] allowing the expression of urgency at the level of transitions by a left-closed deadline constraints, and *Updatable Timed Automata* [10,11,12] which are more expressive than original timed automata and allow, besides clock reset, assigning non-null values to clocks.

By taking these models in consideration, we will show some points that may lead to difficulties in expressing non-null duration actions as well as concurrent and parallel behaviors. The first is certainly splitting actions up into start and completion events that is inherent to these models. To get round the posed problem of graph explosion, an alternative consists in representing actions of non-null duration as non-instantaneous transitions, following the example of *timed automata with non-instantaneous actions* model [4]. According to the semantics of this latter, transitions are indivisible requiring that actions are structurally atomic, which prevents the execution of parallel actions. To observe clearly, consider the example of a process P executing two concurrent actions a and b. If the respective durations of a and b are 10 and 12 units of time, the underlying behavior can be expressed, as shown in Figure 1, by respectively a labeled transitions system, a timed automaton or a timed automaton with non-instantaneous actions. $a\uparrow$ and $a\downarrow$ express respectively the start and the completion of an action a. When actions a and b are of non-null duration, their simultaneous execution is included implicitly in the state s in the timed automaton of Figure 1.(b). This information is lost in Figure 1.(c) because of structural atomicity of actions.

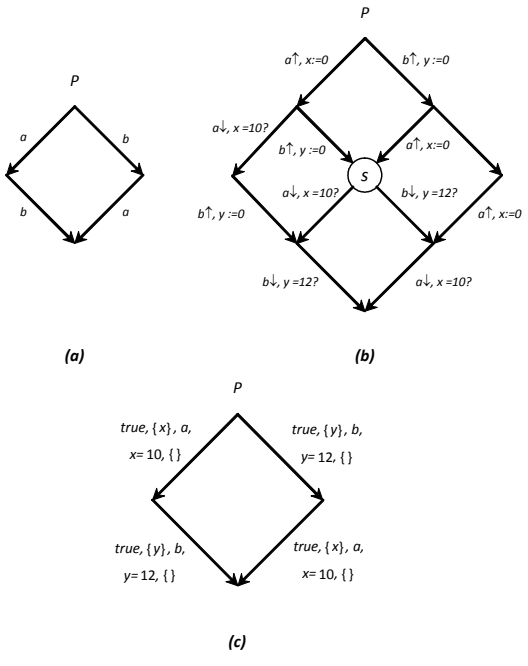

Figure 1: Concurrent actions a and b.

Another point is that concerning event-recording automata. It is the case where one has several actions of the same name complying in parallel (auto-concurrency). Those must share by definition only one clock, which puts some difficulties in expressing such cases. To make out closely this problem, let us consider the example in which two concurrent actions have the same name *a* and the same duration 5. In the event-recording automaton of Figure 2, the clock x_a corresponds to action *a*. This association defined in the beginning between x_a and *a* prevents the expression of auto-concurrency if an action starts its execution whereas the other did not finish yet. In other words, one clock cannot inform us about the evolution of each action since we lose any information on the execution of an action *a* just after the launching of another of the same name.

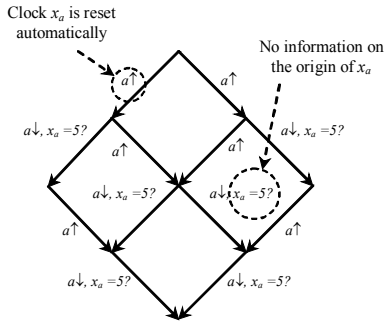

Figure 2: Auto-concurrency in event-recording automata.

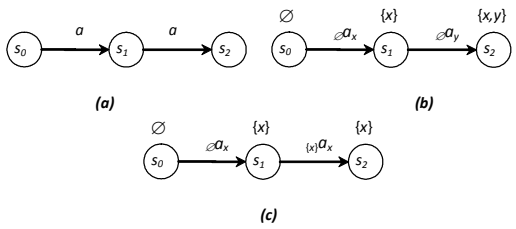

Figure 3: Maximality semantics.

A second alternative to express non-atomic actions is to avoid splitting actions. It consists in using true-concurrency semantics like the *maximality semantics* [17,18]. To implement such semantics, Maximal Trees and Maximality-based Labeled Transition Systems (MLTSs) have been defined [16,28] and used in work relating to the specification and the verification of reactive systems [5,29,30,32]. To have an idea of the maximality semantics, consider the Basic LOTOS [7] behavior expression *E=a;stop* ||| *a;stop* representing the concurrent execution of two actions with the same name *a*. Figure 3.(a) gives the labeled transition system obtained by the interleaving semantics. This transition system is exactly the same of the expression *F=a;a;stop* representing the sequential execution of two actions *a*. The application of the maximality-based operational semantics for Basic LOTOS language produces the transition system of Figure 3.(b) [16,28]. *x* and *y* are events identifying respectively the first start and the second start of the action *a*. In state s_0, no action is complying, which is denoted by an empty set. The set $\{x\}$ in state s_1 indicates that the occurrence of *a* identified by *x* is possibly in execution (the duration is implicit). The set $\{x,y\}$ in state s_2 stipulates that the two runs of the action *a* can take place simultaneously. All of this could be expressed by considering each transition labeled by *a* of the MLTS as only the start of the action *a*.

The completion of an action is identified *implicitly* when another one starts and this latter awaits the completion of the first action. We can observe this in the case of the behavior expression *F=a;a;stop* in which the completion of the first action *a* is detected implicitly as soon as the second action *a* starts. In Figure 3.(c) representing the behavior of *F*, from the state s_1, the second action *a* can only start after the completion of first action *a*. This information is expressed by the event *x*, identifying the start of the first action *a*, present in brackets at the level of the second transition, i.e. the event *x* is bound to the first transition. Once the first action finishes its execution, the event *x* becomes free at the level of the second transition. Thus, the start of the second *a* may be identified by *x* since the first action *a* has finished its execution.

Maximality semantics has been also adopted for the real-time language D-LOTOS [31] which is very close syntactically to ET-LOTOS. D-LOTOS extends Basic LOTOS with timing constraints and urgency constraints, in addition to a function giving to each action a duration. The syntax of D-LOTOS is defined as follows:

$$E ::= stop \mid exit\{d\} \mid \Delta^d E \mid X[L] \mid g@t[SP];E \mid i@t\{d\};E \mid E[]E \mid E|[L]|E \mid hide\ L\ in\ E \mid E>>E \mid E[>E$$

Let *a* be an action (observable or internal), *E* a behavior expression and $d \in \mathbf{D}$ a value in a countable time domain (for example, $\mathbf{D}$ is $\mathbf{Q}^+$). Intuitively, $a\{d\}$ means that the action *a* must start its execution in the temporal domain $[0,d]$. $\Delta^d E$ means that no evolution of *E* is allowed before a time delay equal to *d*. In $g@t[SP];E$ (resp. $i@t\{d\};E$), *t* stores the time passed since the enabling of the action *g* (respectively *i*) and which will be substituted by zero when this action finishes its execution.

In [5], action duration has been explicitly taken into account; this led to the definition of *Durational Action Timed Automata* (DATA's) semantic model. Translating rules of DATA's have been proposed for Basic LOTOS specification language extended with a function which assigns a duration to each action. Explicit timing constraints are not taken into account in DATA's model. The question is how to give semantics to real-time specification languages which support explicit actions duration like D-LOTOS.

This work extends DATA's model in order to take into account timing constraints and urgency constraints present in real-time systems. The following section introduces and defines formally DATA*'s model for the specification of real-time systems. Translating rules to this extension (DATA*'s) are given for D-LOTOS language in Section 2. Some discussions on related work are exposed in Section 3.

2. DATA*'s MODEL

DATA's model was introduced with an aim of expressing temporal and structural non-atomicity of actions. The idea to model (not necessarily null) durations associated with actions can be inspired by the maximality semantics in which a transition represents the start of an action. In the resulting state one says that the action is possibly complying, no conclusion can be drawn with regard to its completion; however, this information can be deduced in a later state in which an action which is dependent to the first one is executed. The association of explicit durations to actions will enable us to express the beginning and the end of execution of actions.

To get an idea of DATA's structures, consider the example of a system P which consists on two concurrent processes P_1 and P_2 synchronizing on an action d. The process P_1 executes the action a followed by d, while P_2 executes b then d, and suppose that actions a, b and d have respective durations 10, 12 and 4. The behavior of P is given by the DATA of Figure 4.(a). Since we can have the case where a and b comply at the same time, we will assign to each one a clock, x and y respectively, to distinguish their occurrences. Therefore, starting from state s_0, the two following transitions are possible: $s_0 \xrightarrow{a,x:=0} s_1$ and $s_0 \xrightarrow{b,y:=0} s_2$. A transition labeled with a indicates the start of the action a, the associated clock counts the evolution in the time of this action. Following the same reasoning, the two following transitions are possible: $s_1 \xrightarrow{b,y:=0} s_3$ and $s_2 \xrightarrow{a,x:=0} s_3$.

Starting from state s_3, the action d can obviously comply only if the two actions a and b finished their execution. Therefore, the transition d can be drawn only if a condition relating to the executions of a and b is satisfied. This condition, called *duration condition*, is built according to the durations of a and b. Initially, we show the construction of duration conditions for s_0, s_1 and s_2 before that of s_3. After launching the transition $s_0 \xrightarrow{a,x:=0} s_1$, we need information on the possible execution of the action a in state s_1. One is sure that the action a finishes its execution when the corresponding clock x reaches 10, therefore, one adds to state s_1 the duration condition of a, $\{x \geq 10\}$, which expresses that if the value of x is higher or equal to 10 then one is sure that the action finished execution. The same thing for the state s_2 which will be labeled by $\{y \geq 12\}$. In state s_0, no action is complying, which implies that duration conditions set is empty. In the state s_3, 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 conditions set $\{x \geq 10, y \geq 12\}$. The execution condition of the action d becomes $x \geq 10 \wedge y \geq 12$. Once this latter 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*. In state s_3, duration condition of actions a and b implies the possibility of their *parallel* evolutions.

In general, real-time systems cannot be completely specified if one does not regard concepts as urgency, deadlines, constraints, etc. To take into account these new concepts, we need to pass towards the DATA*'s model that we introduce in this section.

2.1. INTRODUCTION OF TIMING CONSTRAINTS

The type of constraints which we want to express is that implying restrictions on the enabling domain. In a context of real-time systems, these restrictions cannot be due solely to the durations of former actions, forming so open time domains as for $x \geq 10 \wedge y \geq 12$, but can limit enabling domains regardless of the durations of other actions by delaying for example an action of a certain quantity of time or by limiting the time during which an action is offered to its environment (temporal restriction).

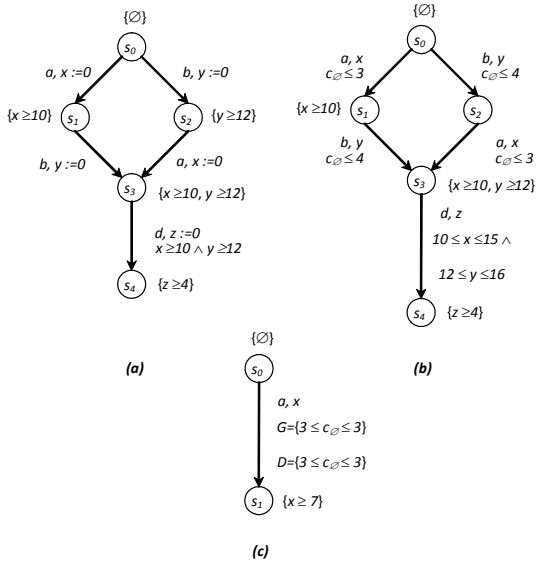

Figure 4: P behavior in term of DATA and DATA*.

Thus, let us suppose that the action a can start only in the first three units of time, i.e. in the domain [0,3]. An action can possibly start if the value of a certain clock belongs to its enabling domain. Action a can start only if a particular clock did not reach value 3 yet. This clock is initialized at the enabling moment of the system P (i.e. at time 0). Given that the action a does not await the end of any other action, let this clock be $c_\varnothing$. Consequently, the transition a in the resulting DATA* will be labeled by the constraint $c_\varnothing \leq 3$ (i.e. $c_\varnothing \in [0,3]$). This constraint, called *guard*, will have to be satisfied so that the action a can comply. If moreover, a is delayed by a lapse of time equal to 1 (as makes the operator Δ^d of D-LOTOS), the guard on transition a will be $1 \leq c_\varnothing \leq 4$ (i.e. $c_\varnothing \in [0+1,3+1]$).

The same reasoning can be applied on the other actions. By admitting that actions a and b of the system P are enabled for respectively 3 and 4 units of time, and that the action d is enabled in the process P_1 (resp. P_2) in the first 5 (resp. 4) units of time, the global behavior of P is represented by the DATA* of Figure 4.(b).

From state s_3 of Figure 4.(b), P_1 and P_2 can be synchronized on action d provided that actions a and b

finished their executions. The start of *d* is conditioned by the constraint over the durations of *a* and *b*: x≥10∧y≥12, and in addition by the temporal restriction of the enabling domain of action d by 5 and 4 units of time respectively according to the source of *d* (from P_1 or P_2). Action *d* coming from P_1 awaits the completion of *a* (which has *x* as clock), i.e. it waits for reaching value 10 by clock *x*. Once this value reached, the expiry time of the offer of action *d* of P_1 starts, and finishes after 5 units of time, i.e. after *x* reaches value 15. Therefore, the enabling domain of this action is $x\in$ [10,15]. The same thing for the other action *d* having as enabling domain $y\in[12,16]$.[1]

2.2. EXPRESSING URGENCY

We noted that the new model of DATA*'s is able to express timing constraints due to restrictions on the enabling domain of an action. Let us observe at present the expression of urgent actions in this model. An urgent action must comply as soon as it is enabled, while time progression is stopped.

It is necessary to distinguish between urgent actions and actions whose enabling domain is made of only one moment in time. Let us consider the example of an action *a* with duration 7 and having as enabling domain $x\in[3,3]$. This action can comply only if clock *x* reached the value 3; beyond this domain (for example in the case of a refusal of the environment), action *a* cannot comply any more. If on the other hand, *a* is an urgent action having always as enabling domain $x\in[3,3]$, once *x* is equal to 3, *time progression is stopped* until *a* starts. Therefore, the *urgency domain* of *a* is $x\in[3,3]$. By considering the assumption of time monotonicity, at the *moment of satisfaction* of the condition corresponding to the urgency domain of an action, the latter must occur immediately.

In our example, the domain $x\in[3,3]$ indicates at once enabling domain and urgency domain except that its semantics differs in the two contexts. This observation lets us to introduce the urgency domain at the level of DATA*'s transitions. Thus, all transitions will be labeled in addition to the enabling constraint *G* (for *guard*) by the urgency constraint *D* (for *deadline*). In the case of system *P* behavior represented by Figure 4.(b), all the transitions will be labeled by *D*={false} because of the absence of urgent actions. The urgency constraint *D* can be hidden without any ambiguity if *D*={false}.

The behavior of the preceding example in which the action *a* is urgent is illustrated by Figure 4.(c). The clock $c_\varnothing$ is used instead of *x* if *a* is the first action which the system can execute, which is indeed the case in our example. In fact, clock *x* will be associated to action *a*, the constraint over the duration of *a* placed on the state s_1 is written thus according to *x*.

[1] $x\in$[min,max], min≤*x*≤max, *x*≥min∧*x*≤max or {*x*≥min,*x*≤max} means the same thing. The purpose of this remark is to ensure that the functions on domains, which will be thereafter defined, can be applied on all preceding forms of domains, even if they will be explicitly defined by using only one form.

Note that the actions may be eager, delayable or lazy as in Bornot et. al model [9]. An action is *eager* when one has *D*=*G* in the corresponding transition, *delayable* if $D=\vee_{(\min\sim i\sim\max)\in G}$ (*i*=max) with $\sim\in\{<,\leq\}$, and *lazy* if *D*={false}.

2.3. FORMALIZATION

Definition 2.1: Let *H*, ranged over *x*, *y*... be a set of clocks with nonnegative values (in a time domain **T**, like $\mathbf{Q}^+$ or $\mathbf{R}^+$). The set $\Phi_t(H)$ of temporal constraints γ over *H* is defined by the syntax γ::=*x*~*t*, where *x* is a clock in *H*, $\sim\in\{=,<,>,\leq,\geq\}$ and $t\in\mathbf{Q}^+$. F_x will be used to indicate a constraint of the form *x*~*t*. A *valuation* (or *interpretation*) *v* for *H* is a function which associates to each $x\in H$ a value in **T**. One says that a valuation *v* for *H* satisfies a temporal constraint γ over *H* iff γ is true by using clock values given by *v*. For $I\subseteq H$, $[I\rightarrow 0]v$ indicates the valuation for *H* which assigns value 0 to each $x\in I$, and agrees with *v* over the other clocks of *H*. The set of all valuations for *H* is noted Ξ(*H*). The satisfaction relation |= for temporal constraints is defined over the set of valuations for *H*, by $v\models x\sim t \Leftrightarrow v(x)\sim t$ such as $v\in\Xi(H)$. $2^{\mathbf{T}}_{fn}$ is used to note the set of finite subsets of a set **T**. □

Definition 2.2: A *DATA* A* is a tuple (S,L_S,s_0,H,T) where:

1. *S* is a finite set of states,
2. $L_S : S \rightarrow 2^{\mathbf{T}}_{fn}$ is a function which corresponds to each state *s* the set F of ending conditions (duration conditions) of actions possibly in execution in *s*,
3. $s_0\in S$ is the initial state,
4. *H* is a finite set of clocks, and
5. $T\subseteq S\times 2^{\mathbf{T}}_{fn}\times 2^{\mathbf{T}}_{fn}\times Act\times H\times S$ is the set of transitions. A transition (s,G,D,a,x,s') represents switch from state *s* to state *s'*, by starting execution of action *a* and resetting clock *x*. *G* is the corresponding guard which must be satisfied to fire this transition. *D* is the corresponding deadline which requires, at the moment of its satisfaction, that action *a* must occur. (s,G,D,a,x,s') can be written $s \xrightarrow{G,D,a,x} s'$. □

Definition 2.3: The semantics of a DATA* $A=(S,L_S,s_0,H,T)$ is defined by associating to it an infinite transitions system S_A over $Act\cup\mathbf{T}$. A state of S_A (or *configuration*) is a pair $\langle s,v\rangle$ such as *s* is a state of *A* and *v* is a valuation for *H*. A configuration $\langle s_0,v_0\rangle$ is initial if s_0 is the initial state of A and $\forall x\in H$, $v_0(x)=0$. Two types of transitions between S_A configurations are possible, and which correspond respectively to time passing (rule *RA**) and the launching of a transition from *A* (rule *RD**).

$$RA^*\ \frac{d\in\mathbf{R}^+ \quad \forall d'\in[0,d],\ v+d'\models D}{\langle s,v\rangle \xrightarrow{d} \langle s,v+d\rangle} \qquad RD^*\ \frac{(s,G,D,a,x,s')\in T \quad v\models G}{\langle s,v\rangle \xrightarrow{a} \langle s',v[x:=0]\rangle}$$

□

According to the maximality semantics, the *label a* in *RD** rule implies the *start* of the action *a* and not the

whole execution of a. In RA^* rule, $D = \bigvee_{i \in I} D_i$ where $\{(G_i, D_i, a_i, x_i, s_i)\}_{i \in I}$ is the set of all transitions stemming from state s. Indeed, whenever a D_i holds, time cannot progress regardless of the other D_i. The function L_S, which gives the duration conditions at the level of a state is useful to construct timing and urgency constraints. By construction, if D_i or G_i are true then causal actions have finished their executions. The use of L_S will be shown afterwards in translating process between D-LOTOS expressions and DATA*'s structures in Section 3.

Note that if one wants to guarantee that at least a transition could be drawn starting from a state if time cannot progress any more within this state, one requires that the formula Di(Gi be satisfied. Let us observe another problem in which an action a must be drawn as soon as possible at one moment higher strictly than 1 (i.e. in the interval]1,+([). This moment cannot obviously be known, which lets us require that urgency domains do not have to be left open.

REMARK 2.1: URGENCY DOMAINS ARE ALWAYS LEFT CLOSED, I.E. THEY ARE ON THE FORM [.] OR [.[.

3. Operational construction of DATA*'s

The approach of generating DATA*'s from D-LOTOS expressions is very close with that of [16,28] for the generation of MLTS structures from Basic LOTOS expressions. From now on, the set M of maximal events will be used to indicate the set of clocks. This is justified by the dynamic choice of clocks corresponding to the occurrence of events. To formalize generation process, functions (, \ and the substitution one introduced in [16,28] spread directly on the configurations relating to DATA*'s states. Configurations corresponding to DATA*'s states belong to the set C*. The function (of extraction of the maximal events set from a configuration is redefined on C* in the same way in order to specify used clocks in a DATA* configuration, except that the equation is replaced by , with given by Definition 3.1.

Definition 3.1: The function $\chi_H : 2_{fn}^{\ldots} \to 2_{fn}^{H}$, which determine the set of used clocks in a duration conditions set, is defined recursively by:

$$\chi_H(\emptyset) = \emptyset$$
$$\chi_H(\{x \sim t\}) = \{x\}$$
$$\chi_H(F_1 \cup F_2) = \chi_H(F_1) \cup \chi_H(F_2)$$

such as

$F_1, F_2 \in 2_{fn}^{\ldots}$, $x \in H$, $\sim \in \{<, \leq, =, \geq, >\}$ and $t \in Q^+$ □

In order to determine duration conditions set of a configuration C_*, we use the function $\phi_{DC} : C_* \to 2_{fn}^{\ldots}$.

If E is a DATA configuration; $E\backslash K$, which indicates the configuration obtained by the suppression of the set K of duration conditions written according to clocks used by the set F of E remains the same one, except the equation $({}_F[E])\backslash K = {}_{F\backslash K}[E]$ which replaces $({}_M[E])\backslash N = {}_{M-N}[E]$, with $F\backslash K$ given in Definition 3.2.

Definition 3.2: Let F be an ending conditions set; $F\backslash K$ indicates the set obtained by the suppression, starting from F, of all the duration conditions written according to the clocks of K ($K \subseteq H$). $F\backslash K$ is recursively defined on F as follows:

$$\emptyset\backslash K = \emptyset$$
$$(F_1 \cup F_2)\backslash K = F_1\backslash K \cup F_2\backslash K$$
$$\{x \sim t\}\backslash K = \begin{cases} \emptyset & \text{if } x \in K \\ \{x \sim t\} & \text{otherwise} \end{cases}$$

such as

$F_1, F_2 \in 2_{fn}^{\ldots}$, $K \subseteq H$, $x \in H$, $\sim \in \{<, \leq, =, \geq, >\}$ and $t \in Q^+$ □

Simultaneous substitution remains unchanged aside from the equation $({}_M[E])\sigma = {}_{M\sigma}[E]$ which will be replaced by $({}_F[E])\sigma = {}_{F\sigma}[E]$. If $x, y \in H$ and $F \in 2_{fn}^{\ldots}$ then $F\sigma = \bigcup_{F_i \in F} F_i\sigma$ with $\{x \sim t\}\sigma = \{\sigma(x) \sim t\}$, and σ is the substitution function given in [16,28].

The specification of delays and timing constraints requires the definition of a function *shift(G,t)* dealing with the move of a temporal domain by a time t, i.e. $shift(G, t) \overset{déf}{=} \{\min - t \sim i \sim \max - t \mid \min \sim i \sim \max \in G\}$ with $\sim \in \{<, \leq\}$.

In order to restrict a temporal domain on its upper bound, we use function *maxval(G)* $\overset{déf}{=} \bigcup_{\min \sim i \sim \max \in G} \{\max\}$ with $\sim \in \{<, \leq\}$.

Definition 3.3: The transition relation of DATA*'s $C_* \times 2_{fn}^{\ldots} \times 2_{fn}^{\ldots} \times Act \times H \times C_*$ is defined as being the smallest relation satisfying the following rules:

$$\frac{}{{}_{\emptyset}[exit\{u\}] \xrightarrow{G=\{c_{\emptyset} \leq u\}, D=\{\mathtt{false}\}, \delta, x}_* {}_{\{x \geq 0\}}[stop]} \; x = get(\mathcal{M}) \qquad (1a)$$

$$\frac{}{{}_{F}[exit\{u\}] \xrightarrow{G=\{\cup_{i \in \mathcal{H}}\{shift(0 \leq i \leq u, t)\} \mid F_i = \{i \geq t\}\}, D=\{\mathtt{false}\}, \delta, x}_* {}_{\{x \geq 0\}}[stop]} \; x = get(\mathcal{M}) \qquad (1b)$$

$$\frac{}{{}_{\emptyset}[a\{u\};E] \xrightarrow{G=\{c_{\emptyset}\le u\},D=\{\mathtt{false}\},a,x}_{*} {}_{\{x\ge\tau(a)\}}[E]}\; x=get(\mathcal{M}) \quad (2a)$$

$$\frac{}{{}_{F}[a\{u\};E] \xrightarrow{G=\{\cup_{t\in\mathcal{H}}\{shift(0\le i\le u,t)\}|F_t=\{i\ge t\}\},D=\{\mathtt{false}\},a,x}_{*} {}_{\{x\ge\tau(a)\}}[E]}\; x=get(\mathcal{M}) \quad (2b)$$

$$\frac{}{{}_{\emptyset}[i\{u\};E] \xrightarrow{G=\{c_{\emptyset}\le u\},D=\{c_{\emptyset}=u\},i,x}_{*} {}_{\{x\ge 0\}}[E]}\; x=get(\mathcal{M}) \quad (3a)$$

$$\frac{}{{}_{F}[i\{u\};E] \xrightarrow{G=\{\cup_{t\in\mathcal{H}}\{shift(0\le i\le u,t)\}|F_t=\{i\ge t\}\},D=maxval(G),i,x}_{*} {}_{\{x\ge 0\}}[E]}\; x=get(\mathcal{M}) \quad (3b)$$

$$\frac{SP=\{\min\le t\sim\max\} \qquad \sim\in\{<,\le\}}{{}_{\emptyset}[a@t\,[SP]\,;E] \xrightarrow{G=\{[c_{\emptyset}/t]SP\},D=\{\mathtt{false}\},a,x}_{*} {}_{\{x\ge\tau(a)\}}[[val\,(c_{\emptyset})+\tau\,(a)\,/t]\,E]} \quad (4a)$$

with $x=get(\mathcal{M})$

$$\frac{SP=\{\min\le t\sim\max\} \qquad \sim\in\{<,\le\}}{{}_{F}[a@t\,[SP]\,;E] \xrightarrow{G=\{\cup_{t\in\mathcal{H}}\{[i/t]shift(SP,d)\}|F_t=\{i\ge d\}\},D=\{\mathtt{false}\},a,x}_{*} {}_{\{x\ge\tau(a)\}}[[val\,(j)+\tau\,(a)-\theta/t]\,E]} \quad (4b)$$

with $x=get(\mathcal{M})\wedge F_j=\{j\ge\theta\}\wedge F_j\in F$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}'}{\mathcal{F}\;[]\;\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad \mathcal{E}\;[]\;\mathcal{F} \xrightarrow{G,D,a,x}_{*} \mathcal{E}'} \quad (5)$$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad a\notin L\cup\{\delta\}}{\mathcal{E}\;|[L]|\;\mathcal{F} \xrightarrow{G,D,a,y}_{*} \mathcal{E}'[y/x]\;|[L]|\;\mathcal{F}\backslash G} \quad (6a\text{-}6b)$$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad a\notin L\cup\{\delta\}}{\mathcal{F}\;|[L]|\;\mathcal{E} \xrightarrow{G,D,a,y}_{*} \mathcal{F}\backslash G\;|[L]|\;\mathcal{E}'[y/x]}$$

with $y=get(\mathcal{M}-((\psi(\mathcal{E})\cup\psi(\mathcal{F}))-\psi_{\mathcal{H}}(G)))$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad \mathcal{F} \xrightarrow{G',D',a,y}_{*} \mathcal{F}' \qquad a\in L\cup\{\delta\}}{\mathcal{E}\;|[L]|\;\mathcal{F} \xrightarrow{G\cup G',D\cup D',a,z}_{*} \mathcal{E}'[z/x]\backslash G'\;|[L]|\;\mathcal{F}'[z/y]\backslash G} \quad (6c)$$

with $z=get(\mathcal{M}-((\psi(\mathcal{E})\cup\psi(\mathcal{F}))-(\psi_{\mathcal{H}}(G)\cup\psi_{\mathcal{H}}(G'))))$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad a\notin L}{hide\ L\ in\ \mathcal{E} \xrightarrow{G,D,a,x}_{*} hide\ L\ in\ \mathcal{E}'} \quad (7a\text{-}7b)$$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad a\in L}{hide\ L\ in\ \mathcal{E} \xrightarrow{G,D:=G,i,x}_{*} hide\ L\ in\ \mathcal{E}'}$$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad d\in\mathcal{D}}{\Delta^{d}\mathcal{E} \xrightarrow{shift(G,d),shift(D,d),a,x}_{*} \mathcal{E}'} \quad (8)$$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad a\ne\delta}{\mathcal{E}\gg F \xrightarrow{G,D,a,x}_{*} \mathcal{E}'\gg F} \quad (9a\text{-}9b)$$

$$\frac{\mathcal{E} \xrightarrow{G,D,\delta,x}_{*} \mathcal{E}'}{\mathcal{E}\gg E \xrightarrow{G,D,i,x}_{*} {}_{\{x\ge 0\}}[E]}$$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad a\ne\delta}{\mathcal{E}\;[>\;\mathcal{F} \xrightarrow{G,D,a,y}_{*} \mathcal{E}'[y/x]\;[>\;\mathcal{F}\backslash G} \quad (10a)$$

$y=get(\mathcal{M}-((\psi(\mathcal{E})\cup\psi(\mathcal{F}))-\psi_{\mathcal{H}}(G)))$

$$\frac{\mathcal{E} \xrightarrow{G,D,\delta,x}_{*} \mathcal{E}' \qquad \forall z\in\psi(\mathcal{F})}{\mathcal{E}\;[>\;\mathcal{F} \xrightarrow{G,D,\delta,y}_{*} \mathcal{E}'[y/x]\;[>\;{}_{\psi_{DC}(\mathcal{F})-G_z}[stop]} \quad (10b)$$

$y=get(\mathcal{M}-((\psi(\mathcal{E})\cup\psi(\mathcal{F}))-\psi_{\mathcal{H}}(G)))$

$$\frac{\mathcal{F} \xrightarrow{G,D,a,x}_{*} \mathcal{F}' \qquad \forall z\in\psi(\mathcal{E})}{\mathcal{E}\;[>\;\mathcal{F} \xrightarrow{G,D,a,y}_{*} {}_{\psi_{DC}(\mathcal{E})-G_z}[stop]\;[>\;\mathcal{F}'[y/x]} \quad (10c)$$

$y=get(\mathcal{M}-((\psi(\mathcal{E})\cup\psi(\mathcal{F}))-\psi_{\mathcal{H}}(G)))$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad a\notin\{a_1,\ldots,a_n\}}{\mathcal{E}[b_1/a_1,\ldots,b_n/a_n] \xrightarrow{G,D,a,x}_{*} \mathcal{E}'[b_1/a_1,\ldots,b_n/a_n]} \quad (11a)$$

$$\frac{\mathcal{E} \xrightarrow{G,D,a,x}_{*} \mathcal{E}' \qquad a=a_i\ (1\le i\le n)}{\mathcal{E}[b_1/a_1,\ldots,b_n/a_n] \xrightarrow{G,D,b_i,x}_{*} \mathcal{E}'[b_1/a_1,\ldots,b_n/a_n]} \quad (11b)$$

$$\frac{P:=E \qquad {}_{F}[E] \xrightarrow{G,D,a,x}_{*} \mathcal{F}}{{}_{F}[P] \xrightarrow{G,D,a,x}_{*} \mathcal{F}} \quad (12)$$

such as, $c_{\emptyset}$ is a particular clock which is created and initialized at the enabling time of the system, and D is the underlying countable time domain for D-LOTOS language. □

Notation 3.1: We have chosen the following notation:

For the observable actions, we have

- *a@t[t≤d];E=a{d};E* (the selection predicate *SP* of the form *t≤d* is replaced by a temporal restriction), with *t* not free in *E*.
- *a@t[d≤t≤d'];E=*Δ^{d}*a{d'-d};E*, with *d≤d'* and *t* is not free in *E*.

If *i* is a non observable action, then

- *i@t{0};E=i@t;E* (If {*d*} is omitted, then *d*=0).
- *i@t{d};E=i{d};E* if *t* is not free in *E*.

By convention, if the enabling domain $\{d\}$ is omitted in $a\{d\};E$, then $d=\infty$. In the same way, if SP is omitted, then $SP=$`true`. If the deadline constraint D is omitted in any transition, then $D=\{$`false`$\}$.

For the sake of presentation, we focus our explanation only on rules 1, 2, 3 and 4 which give the main idea of the operational semantics. The meaning of the other rules should be easy after explanations below.

In rules 1a and 2a, an action offered in a lapse of time equal to u and depending on the end of no other one can comply provided that the value of clock $c_\varnothing$ does not exceed the u value. If the start of an action a depends on the end of at least another action, the enabling constraint G is built on the one hand starting from durations of clocks corresponding to actions of which a depends, and on the other hand by the offer interval u of a, which is expressed by the rules 1b and 2b.

In rules 1a, 1b, 2a and 2b of Definition 3.3, all actions are not urgent, which explains why the constraints D are always put at `false`.

Rules 3a and 3b express the firing of an internal action. This action becomes urgent at the end of enabling lapse of time d. In these rules, the urgency constraint D is built starting from enabling constraint G. As we already mentioned, the semantics of the two constraints differs: at the moment of satisfaction of D, the action must be launched. The same reasoning will be applied to hidden actions where the urgency is expressed through the rule 7b.

We will explain the use of operator @ through D-LOTOS expression $E=a@t1[t1\leq 3];\ b@t2[t2\leq t1];\ stop$. Let us suppose that the respective durations of a and b are 20 and 7. As soon as the expression $a@t1[t1\leq 3];F$ is equivalent to $a\{3\};[d/t1]F$ (d is the interval between the enabling and the end of execution of a), and the start of action a does not depend on the end of any other action, we will just substitute the variable $t1$ on the level of the predicate $t1\leq 3$ by the clock $c_\varnothing$ to have the enabling constraint $c_\varnothing\leq 3$. In the remaining expression $b@t2[t2\leq t1];stop$, all free occurrences of variable $t1$ receive the lapse of time between the enabling ($c_\varnothing=0$) and the end of execution of a (current value of $c_\varnothing$, which will be noted $val(c_\varnothing)$, plus the duration of a, $\tau(a)$). Therefore, this interval is equal to $val(c_\varnothing)+\tau(a)-0 = val(c_\varnothing)+\tau(a)$. It is expressed by the rule 4a. The following transition becomes possible:

$$\underbrace{\emptyset\ [E]}_{config_0} \xrightarrow{c_\emptyset \leq 3, a, x} \underbrace{\{x\geq 20\}\ [b@t_2\ [t_2 \leq val\,(c_\emptyset) + 20]\ ;\ stop]}_{config_1}$$

In the configuration *config1*, enabling domain of b is $t2\in[0,val(c_\varnothing)+20]$, and as the action b depends on the end of a which lasts 20 units of time and which has x as clock, enabling domain of b becomes $x\in[0+20,val(c_\varnothing)+20+20]$. Generally speaking, if one has several clocks in duration constraint F (given by L_S function) of the source configuration, the shift of the interval implies all these clocks, which is expressed by the rule 4b. In our example, enabling constraint of b only implies clock x with a shift of 20 units of time. The following transition becomes possible:

$$\underbrace{\{x\geq 20\}\ [b@t_2\ [t_2 \leq val\,(c_\emptyset) + 20]\ ;\ stop]}_{config_1} \xrightarrow{20\leq x\leq val(c_\emptyset)+40, b, x} \underbrace{\{x\geq 7\}\ [stop]}_{config_2}$$

Now, if instead of *stop* we have the expression $c@t3[t3\leq t1+t2];stop$ (i.e. the expression E becomes $a@t1[t1\leq 3];b@t2[t2\leq t1];c@t3[t3\leq t1+t2];stop$), with c of duration 18, the transition representing the beginning of execution of c will be guarded by the domain $x \in \left[\ 0, \underbrace{val(\cdot)+20}_{t_1}\ \underbrace{val(\cdot)+7}_{t_2}\ \right]$, shifted by 7 units, i.e. $7 \leq x \leq \underbrace{val(\cdot)+20}_{t_1}\ \underbrace{val(\cdot)+7}_{t_2}$. In other words, $t1$ always contains the value $val(c_\varnothing)+20$, while the variable $t2$ contains $val(x)+7-20$ because the action b is enabled when $x=20$, and the end of execution of b is stored in $\underbrace{val(\cdot)}_{\text{start of } b}+7$. In a more general way, if the transition b is guarded by a constraint built of several clocks such as $x\in[\min_x,\max_x]\wedge y\in[\min_y,\max_y]\wedge\ldots$, the value of $t2$ in the remaining expression $c@t3[t3\leq t1+t2];stop$ can be expressed using only one clock among $\{x,y,\ldots\}$, such as j, since the lapses of time are equal regardless of time scale (i.e. $val(x)+7-20=val(y)+7-d1=val(z)+7-d2=\ldots$ and $d1,d2,\ldots\in D$). This is expressed by the rule 4b, hence the following transition:

$$\underbrace{\{x\geq 7\}\ [c@t_3\ [t_3 \leq val\,(c_\emptyset) + val\,(x) + 7]\ ;\ stop]}_{config_2} \xrightarrow{7\leq x\leq val(c_\emptyset)+val(x)+14, c, x} \underbrace{\{x\geq 18\}\ [stop]}_{config_3}$$

Regarding the use of the operator @ with internal actions, the rules 3a and 3b will be applied in the same way on expression $i@t\{d\};E$ with t not free in E.

The rule 8 relates to delaying an action a with a certain quantity of time d. One can notes that if the delaying operator Δ is applied to a configuration E, only the first action drawn from the configuration E will be delayed.

Remark 3.1: Without loss of generality, we used D-LOTOS language in which actions durations are fixed once and for all at the enabling moment of the system. The case where actions have durations chosen from an interval $[m,M]$ is not considered.

4. DISCUSSION

The study of timed automata and timed automata with non-instantaneous actions models showed us that the state space combinatorial explosion problem and the difficulty of expressing certain parallel behaviors are respectively inherent in the structure of each one of these models. The DATA*'s model enabled us to overcome each one of these problems.

In dynamic timed automata [13,22], the automaton is known as *dynamic* because clocks number can vary from a state to another. Note that DATA*'s model, in addition to its aptitude to take into account non-atomic actions, allows the use of *dynamic* clocks of which construction is done in one step by, in our case, the function *get*. However, in DTA's, clocks construction is made in two steps [13], the first one is mainly intended to build a virtual clock for each concurrent component of the RT-LOTOS specification, while the second one is devoted to the mapping between these virtual clocks and current clocks of the generated DTA.

Now, regarding the class of updatable timed automata [10,11,12], and given that the transitions between DATA*'s states imply an automatic reset of the corresponding clocks, some behaviors expressed by updatable timed automata could not be expressed using a DATA*. To specify protocols based on updates as ABR (*Available Bit Rate* [6]), an alternative consists in modeling updates by the use of more general operations on variables, as it is implemented in HYTECH tool [19].

We can deduce that a theoretical comparison between DATA*'s model and the existing ones would be of a capital significance to be able to surround *the least expressive model but most sufficient to meet the needs of real-time applications with durations associated with actions.*

5. CONCLUSION AND PERSPECTIVE

This paper proposes a real-time extension to DATA's model which is very near syntactically to timed automata, taking into account structural and temporal non-atomicity of actions, timing constraints and urgency. The goal has been to define a semantic model which makes more natural expressing concurrent and parallel behaviors of real-time systems. The idea is based on the principle of the maximality semantics in which only the starts of actions are modeled; ends of execution are captured by the corresponding durations.

As perspective, we hope that additional information in maximality-based labeled transition systems can allow us reducing the number of states and transitions in the graph without loss of information, and then escaping from the explosion of the underlying graph. For instance, the concurrency of two actions may be expressed as $\emptyset \xrightarrow{\Downarrow a_x, \Downarrow b_y} \{x, y\}$. Work in this direction can be found in [23,27,34,35]. This should spread over DATA*'s model.

Regarding formal verification of real-time systems, we think that we can adapt model checking algorithms by taking as semantic model DATA*'s structures. This can express, as for MLTS model [30,33], desired properties in a more natural way reasoning directly on actions.

REFERENCES

[1] Alur R. and Dill D., "Automata for Modeling Real-Time Systems," *in Proceedings of ICALP'90*, vol. 443 of LNCS, pp. 322-335. Springer-Verlag, 1990.

[2] Alur R. and Dill D., "A Theory of Timed Automata," *TCS*, vol. 126, pp. 183-235, 1994.

[3] Alur R., Fix L., and Henzinger T. A., "Event-Clock Automata: A Determinizable Class of Timed Automata," *in Proceedings of CAV'94*, pp. 1-13, Springer-Verlag, 1994.

[4] Barbuti R., De Francesco N., and Tesei L., "Timed Automata with Non-Instantaneous Actions," *Fundamenta Informaticae*, vol. 46, pp. 1-15, 2001.

[5] Belala N. and Saïdouni D. E., "Non-Atomicity in Timed Models," *in Proceedings of ACIT'2005*, Al-Isra Private University, Jordan, December 2005.

[6] Bérard B. and Fribourg L., "Automatic Verification of a Parametric Real-Time Program: The ABR Conformance Protocol," *in Proceedings of CAV'99*, vol. 1633 of LNCS, Springer-Verlag, 1999.

[7] Bolognesi T. and Brinksma E., "Introduction to the ISO Specification Language LOTOS," *Computer Networks and ISDN Systems*, vol. 14, pp. 25-59, 1987.

[8] Bornot S. and Sifakis J., "On the Composition of Hybrid Systems," *in Proceedings of HSCC'98*, vol. 1386 of LNCS, pp. 69-83. Springer-Verlag, 1998.

[9] Bornot S., Sifakis J., and Tripakis S., "Modeling Urgency in Timed Systems," *in Proceedings of COMPOS'97*, vol. 1536 of LNCS, Springer-Verlag, 1997.

[10] Bouyer P., *Modèles et Algorithmes pour la Vérification des Systèmes Temporisés*, PhD thesis, Ecole Normale Supérieure de Cachan, France, April 2002.

[11] Bouyer P., Dufourd C., Fleury E., and Petit A., "Are Timed Automata Updatable?," *in Proceedings of CAV'2000*, vol. 1855 of LNCS, pp. 464-479, Springer-Verlag, 2000.

[12] Bouyer P., Dufourd C., Fleury E., and Petit A., "Updatable Timed Automata," *Theoretical Computer Science*, vol. 321(2-3), pp. 291-345, 2004.

[13] Courtiat J.-P., and de Oliveira R. C., "A Reachability Analysis of RT-LOTOS Specifications," *in Proceedings of FORTE'95*, Chapman and Hall, London, 1995.

[14] Courtiat J.-P., Santos C. A. S., Lohr C., and Outtaj B., "Experience with RT-LOTOS, a Temporal Extension of the LOTOS Formal Description Technique," *Computer Communications*, vol. 23, pp. 1104-1123, 2000.

[15] Courtiat J.-P., de Camargo M. S., and Saïdouni D. E., "RT-LOTOS: LOTOS Temporisé pour la Spécification de Systèmes Temps-Réel," *CFIP'93*, pp. 427-441, Hermes, 1993.

[16] Courtiat J.-P. and Saïdouni D. E., "Relating Maximality-based Semantics to Action Refine-ment in Process Algebras," *in Proceedings of FORTE'94*, pp. 293-308, Chapman and Hall, 1995.

[17] Devillers R., "Maximality Preservation and the ST-Idea for Action Refinement," *Advances in Petri Nets*, vol. 609 of LNCS, pp. 108-151, Springer-Verlag, 1992.

[18] Devillers R., "Maximality Preserving Bisimulation," *TCS*, vol. 102, no. 1, pp. 165-184, 1992.

[19] Henzinger T. A., Ho P.-H., and Wong-Toi H., "HYTECH: A Model Checker for Hybrid Systems," *Journal of Software Tools for Technology Transfer*, vol. 1(1-2), pp. 110-122, October 1997.

[20] Henzinger T. A., Nicollin X., Sifakis J., and Yovine S., "Symbolic Model-Checking for Real-Time Systems," *Information and Computation*, vol. 111, pp. 193-244, 1994.

[21] Léonard L. and Leduc G., "An Introduction to ET-LOTOS for the Description of Time-Sensitive Systems," *Computer Networks and ISDN Systems*, vol. 29, pp. 271-292, 1997.

[22] Lohr C., *Contribution à la Conception de Systèmes Temps-Réel S'appuyant sur la Technique de Description*

Formelle RT-LOTOS, PhD thesis, LAAS-CNRS, Toulouse, France, 2002.

[23] Magniette F., Pilard L., and Rozoy B., "Model Checking et Produit Synchronisé," *in Proceedings of MSR'03*, pp. 213-224, Metz, France, October 2003.

[24] Merlin P., *A Study of the Recoverability of Computer System*, PhD Thesis, Dept. Computer Sciences, University of California, Irvine, 1974.

[25] Moller F. and Tofts C., "A Temporal Calculus of Communicating Systems," *CONCUR*, vol. 458 of LNCS, pp. 401-415, Springer-Verlag, 1990.

[26] Ramchandani C., *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*, PhD Thesis, MIT, Cambridge, Feb. 1974.

[27] Ribet P. O., *Vérification Formelle de Systèmes: Contribution à la Réduction de l'Explosion Combinatoire*, PhD thesis, LAAS-CNRS, Toulouse, France, 2005.

[28] Saïdouni D. E., *Sémantique de Maximalité: Application au Raffinement d'Actions en LOTOS*, PhD Thesis, LAAS-CNRS, Toulouse, France, 1996.

[29] Saïdouni D. E. and Belala N., "Straightforward Adaptation of Interleaving-Based Solutions for True Concurrency-Based Logic Verification Approaches," *in Proceedings of CISC'2004*, University of Jijel, Algeria, September 2004.

[30] Saïdouni D. E. and Belala N., "Using Maximality-Based Labeled Transition System Model for Concurrency Logic Verification," *The International Arab Journal of Information Technology (IAJIT)*, vol. 2(3), pp. 199-205, July 2005. ISSN:1683-3198.

[31] Saïdouni D. E. and Courtiat J.-P., "Prise en Compte des Durées d'Action dans les Algèbres de Processus par l'Utilisation de la Sémantique de Maximalité," *in Proceedings of CFIP'2003*, France, Hermes, 2003.

[32] Saïdouni D. E. and Ghenaï A., "Intégration des Refus Temporaires dans les Graphes de Refus," *in Proceedings of NOTERE'2006*, Hermes, Toulouse, France, 2006.

[33] Saïdouni D. E. and Labbani O., "Maximality-Based Symbolic Model Checking," *in Proceedings of ACS/IEEE*, Tunisia, July 2003.

[34] Vernadat F., Azéma P., and Michel F., "Covering Step Graph," *in Proceedings of Application and Theory of Petri Nets 96*, vol. 1091 of LNCS. Springer-Verlag, 1996.

[35] West C. H., "Protocol Verification by Random State Exploration," *in PSTV VI*, pp. 233-242, 1986.