Verification of Durational Action Timed Automata using UPPAAL

Souad GUELLATI
MISC Laboratory, Mentouri University, Constantine, 25000, Algeria

Ilham KITOUNI
MISC Laboratory, Mentouri University, Constantine, 25000, Algeria

Djamel-Eddine SAIDOUNI
MISC Laboratory, Mentouri University, Constantine, 25000, Algeria

ABSTRACT

The increasing complexity of software is incessant, this phenomenon is even more accentuated when temporal aspects are introduced, hence the need for rigorous verification methods.

The main purpose of this paper is to propose a quantitative verification approach based on model checking. Their properties are expressed in TCTL (*Timed Computation Tree Logic*) on real-time systems. The system behavior is expressed by temporal labeled systems; namely Durational Action Timed Automata model (DATA* model). This model supports the expression of the parallel behavior, the temporal and structural non-atomicity of actions and urgency.

Our approach is to interpret the behavior described by DATA* to Timed Safety Automata. The environment UPPAAL allows us verifying quantitative temporal properties, especially the bounded liveliness.

Keywords

Formal verification, Model Checking, TCTL, DATA*'s model, Timed Safety Automata, Bounded Liveliness, UPPAAL.

1. INTRODUCTION

Computer applications become increasingly involved in critical and real-time systems (e.g., automotive, avionic and robotic controllers, mobile phones, communication protocols and multimedia systems). These systems are known by their high complexity.

One approach for designing such systems is to use formal specification models, such as Petri nets, process algebras and formal description techniques, historically this work are based on searches similar to [5, 6, 12, 16]. In fact the specification may contain mistakes, so, formal validation and verification approaches are needed.

In this work, we are interested by *model checking* [8] based verification methods. It is used to verify some behavioral properties specified, by the temporal logic *TCTL(Timed Computation Tree Logic)*, on the system's state space, in order to detect precocious errors of conception.

It is well known that the quality of validation techniques and verification depend on the quality of models used for specification. In this context the model of *Durational Actions Timed Automata (DATA*)* has more interest. DATA* is a timed model, its semantic express the durations of actions and other notions for specifying the real-time systems such as urgency and deadlines [4]. This model is based on a maximality semantic [19] and advocates the true concurrency; from this point of view it is well suitable for modeling real time, concurrent and distributed systems. But, the consideration of a dense time domain make the generation of space state infinite from which the impossibility to use model checking [8] as methods for verification.

The IF toolset [24] and UPPAAL [15] offered different modeling and verification environments. For instance, the IF toolset verifies requirements on TAD (Timed Automata with Deadlines) models that are expressed in the alternation free μ-calculus [25], or are expressed as observers (safety properties). Instead, our translation allows requirements to be expressed in the fragment of TCTL [3] that is supported in UPPAAL, and safety properties. Bornot et al [26] suggested a way to translate TAD to Timed Automata, whereas Barbuti and Tesei [27] proposed an extension of TA with urgent transitions. [28]

In this paper, we will propose an approach for translating DATA*'s to Timed Safety Automata. However, we will keep the advantage offered by DATA*s model, in which only the beginnings of the actions are considered. This translation allows us to use the *UPPAAL* tool for verification.

This paper is organized as follows: Section 2 presents the timed automata with durational actions, we describe the syntax and semantic of Timed Automata and DATA*. Section 3 presents the Timed Safety Automata model. A translation approach of DATA* to TSA is given in Section 4. Section 5 describes the verification of DATA* using the UPPAAL tool. Section 6 presents the results of applying our approach to the case study. The last section concludes the paper and gives some perspectives.

2. TIMED AUTOMATA WITH DURATIONAL ACTIONS

2.1 Notations

In the following $\mathbb{R}^+$ is the set of non-negative real numbers. A *clock* takes values from $\mathbb{R}^+$ or it is undefined, denoted by $\perp$. Without loss of generality, we write $\mathbb{R}_\perp^+ = \mathbb{R}^+ \cup \{\perp\}$ where the set of non-negative real numbers is extended with the special value $\perp$.

Given X a set of clocks. A *clock valuation* over X is a function assigning a non-negative real number to every clock. The set of valuations of X, denoted V_X, is the set of total function from X to $\mathbb{R}^+$. A valuation $v \in V_X$ is a mapping on X to $\mathbb{R}^+$. The valuation $v+d$ maps every clock y to $v(y)+d$ $(d \in \mathbb{R}^+)$.

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

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

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

2.2 Timed Automata [1, 2]

In general way Timed Automata are finite state machines whose transitions are decorated by clocks constraints.

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

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

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

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

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

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

With l_0 an initial location and v_0 is such that $v_0(x) = 0$, $\forall x \in X$. The run *r* is accepting iff l_{n+1} is a final or terminal location.

2.3 Durational Actions Timed Automata

The Durational Actions Timed Automata model DATA* [4] is a timed model defined as Timed Automata over an alphabet representing actions to be executed. This model takes into account the duration of actions based on an intuitive idea: temporal and structural non-atomicity of actions.

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

Illustrate the model with the example above (see Figure 1):

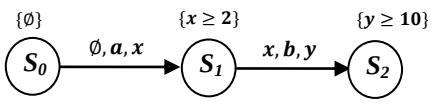

Fig 1: DATA* (1).

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

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

Figure 1 presents a system of two consecutive actions *a* and *b*, the clock *x* is associated to the action *a*, on the locality s_1 the temporal formula *{x≥2}* represents the duration of the action *a*.

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

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

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

Indeed, using Durational Actions Timed Automata model is very suitable to capture the true concurrency in systems behavior. Since each locality detain the information about current execution of action; when more than one action are under execution then on locality, associated temporal formula are found. 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 [7].

This model seems interesting and funneling more and more research because it coated model of Timed Automata by maximality semantics [10, 13, 18, 20, 14].

The DATA* model, as the timed models, takes in charge the notion of urgency; expressed with deadlines as temporal constraints of the system.

2.3.1 Formalization

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

An edge *e=(l,G,D,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. And *D* specifies the deadline.

Finally, $L_S : L \rightarrow 2^{C_X}_{fn}$ is a maximality function which decorates each location by a set of timed formula named actions durations, about actions potentially in execution on it.

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

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

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

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

$$R1 \quad \frac{d \in \mathbb{R}^+ \quad \forall d' \leq d, v + d' \not\models D}{(s,v) \xrightarrow{d} (s, v+d)}$$

$$R2 \quad \frac{(s,G,D,a,x,s') \in T_D \quad v \models G}{(s,v) \xrightarrow{a} (s', [\{x\} \leftarrow 0]v)}$$

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

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

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

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

$$R1 \quad \frac{d \in \mathbb{R}^+ \quad \forall d' \leq d, v + d' \not\models D}{(s,v) \xrightarrow{d} (s, v+d)}$$

$$R2 \quad \frac{(s,G,D,a,x,s') \in T_D \quad v \models G}{(s,v) \xrightarrow{a} (s', [\{x\} \mapsto 0]v)}$$

In the following example we consider the execution of two actions in parallel, each of them has a specific time execution and offered delay during which it can start its execution. Figure 2 illustrates this behavior.

The transition from the state s_0 to the state s_1 is labelled by the action a, indicating the beginning of its execution while respecting a offered delay 3 units of time expressed by the condition $c_\phi \leq 3$ (The clock c_ϕ serves to count the offer delay of the first action, in this case the action a, $c_\phi \in [0,3]$) ; the symbol ϕ meaning that the beginning of a waits for the end of no other action and the clock x counting the evolution of time from the crossing of this transition.

The label $\{x \geq 10\}$, associated to the state s_1 is an information on the action execution time of the action a and mean that the action a is currently executed, as the clock x has not overtaken the value 10 yet.

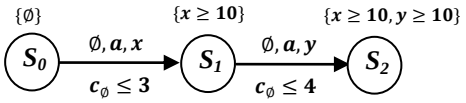

Fig 2: DATA* (2).

In the state s_2, the two actions of the same name a can be executed in parallel. It is the auto-concurrency, and each of them cannot finish that if his clock reaches a value equal to its duration. Hence the set of durational conditions $\{x \geq 10, y \geq 10\}$. Given that the second action a is offered for 4 units of time, she can start only if her beginning respects the condition $c_\phi \leq 4$.

3. Timed Safety Automata Model

Timed Safety Automata (TSA) [11] has been proposed for the specification of real-time systems. TSA has been adopted in several verification tools for timed automata e.g. UPPAAL[1] [15] and Kornos[2] [21].

Definition 4: A Timed Safety Automaton $\mathcal{A}$ is a tuple $(\Sigma, L, l_0, X, E, I)$ where:

- Σ is a set of actions,
- L is a set of locations,
- $l_0 \in L$ is the initial location,
- X is the set of clocks,
- $E \subseteq L \times L \times \Sigma \times 2^X_{fn} \times \Phi(X)$ is a set of edges. An arc (l, l', a, λ, r) represent a transition from the state l to the state l' by reading symbol a. The set $r \subseteq X$ contains the clocks to be reset by this transition, and λ is a temporal constraint on X. (l, l', a, λ, r)can be written $l \xrightarrow{a,\lambda,r} l'$, and
- $I: L \to \Phi(X)$ Assigns invariants to locations. Invariants are predicates ρ defined by the following grammar:

  $$\rho ::= x \# c \mid true \mid \rho \wedge \rho$$

  Where $x \in X, c$ is an integer and $\# \in \{\leq, <\}$.

A way to model this behavior by Timed Safety Automata is by splitting each action in two instantaneous actions corresponding to the *beginning* and the *end*, due to the supposition of the temporal atomicity of the actions (the execution time is zero. The system starts the execution of the action when the constraint $x \leq 3$ is verified. The invariant $x \leq 10$ forces the residence of the system to the state s_1, once this condition is violated, the system has to execute the transition End (a) when the guard $x \geq 10$ is verified.

In the example bellow we consider the action a with its two parameters, duration of execution and delay of offer have respectively 10 and 3 units of time. The Timed Safety Automata is illustrated by the Figure 3.

[1] www.uppaal.com
[2] http://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos/

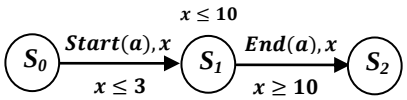

Fig 3: Timed Automaton.

Although this solution allows the correct modelling of an action with durations, it cannot be accepted by the reason of the combinatorial explosion problem which it engenders for the state space and the set of alphabet.

4. TRANSLATION OF DATA* TO TSA

Our objective is to verify quantitative properties expressed by temporal logic TCTL on system behavior witch is expressed by DATA*'s model passing by Timed Safety Automata in order to use the tool UPPAAL.

Let's consider the behavior expressing the sequential execution of two actions a and b have respective durations 10 and 12 and respective offered delays 3 and 4 the behavior of S is represented by DATA* of the Figure 4.

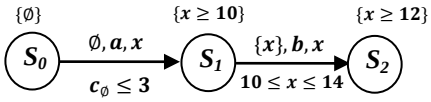

Fig 4: Example of a sequential behaviour.

From initial state s_0 , action a can begin its execution while respecting the condition $c_\phi \leq 3$. a can begin its execution since a delay of 3 units of time from enabling the system has not expired.

The system passes in the state s_1 and cannot leave it before the end of action a, the same reasoning applies for the action b.

Let us try now to express this behavior with timed automaton.

The first passage from s_0 to s_1 can be described by the transition $s_0 \xrightarrow{Start(a), c_\phi \leq 3, x} s_1$ (the transition from the state s_0 to s_1 begin the action , the condition represents the *garde* and x is the clock to be reset in zero by this transition to count the execution time of the action a). Once the system is in the state s_1 , he cannot leave it neither before ending of a nor after the expiration of the offered delay of b, the action b is enable at the time of the ending of a, i.e. the offer delay of b begin its expiration once a is ended. The passage from s_1 to s_2 can be described by the transition $s_1 \xrightarrow{Start(b), 10 \leq x \leq 14, x} s_2$.

Clearly it is possible to express the duration without being forced to consider each action as two events: the *beginning* and *end*.

Therefore, our approach is to model an action, which has two optional parameters: *offered delay* and *time execution*, with a Timed Automaton including:

- A transition to express the beginning of action (with guard if there is, offered delay to respect and reset the clock appropriate to capture the instant of beginning of action),
- A state, where the system resides during the execution of action (the system is not forced to leave this state once the action is complete).

Therefore, the guard of the next transition that will serve us capture both, the end of action ($x \geq 10$) and beginning of next action ($x \leq 14$) . Finally we obtain the automata representing the global behavior of S in Figure 5.

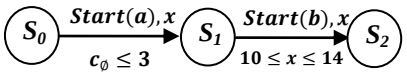

Fig 5: Timed Automaton obtained after translation.

Comparing the two models illustrated in Figure 4 and Figure 5, we able to observe that both express the same behavior except that in Timed Safety Automata we have lost information about end of the action .

It is clear that it is impossible to capture the end of last action without adding another transition. For that we consider all processes expressed by DATA*'s necessarily use particular atomic action δ to marked its ends. And thus we obtain the behavior illustrated by Figure 6.

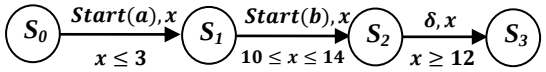

Fig 6: The Timed Automaton corrected.

Effectively, the passage from DATA* to Timed automaton, in the case of sequential processes, is naturally performed.

Now we want to know if this approach supports the *urgency* and the *parallelism* (*concurrency*).

4.1 Expressing urgency

Take the example of Figure 6, action a (respectively b) must occur when the clock x (respectively y) attained the value 3 (respectively 4).

The expression of urgency in Timed Safety Automata is done by using invariants. Therefore the timed automaton describing this behavior is illustrated by Figure 7.

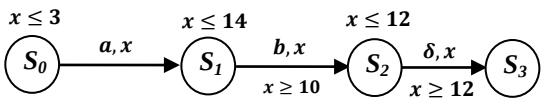

Fig 7: Timed Automaton expressing the urgency.

4.2 Concurrency

Let us consider the example of a system S which consists of two concurrent subsystems S_1 and S_2synchronizing on an action d. The subsystem S_1 executes the action a followed by d, while S_2 executes b then d.

Now suppose that actions a, b and $d$have respective durations $10, 12$ and 4. The temporal restriction of the domain of sensibilization of the action d is 5 and 4 units of time following from d of S_2 or S_2. The global behaviour of S is represented by the DATA* in Figure 8.

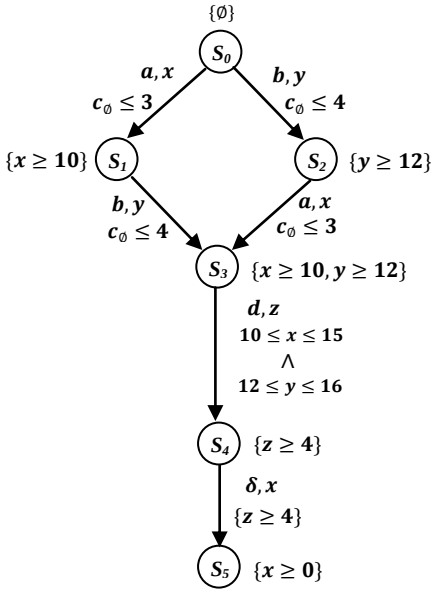

Fig 8: DATA* of behavior .

Applying the previous reasoning, we obtain the timed automaton illustrated by Figure 9**.**

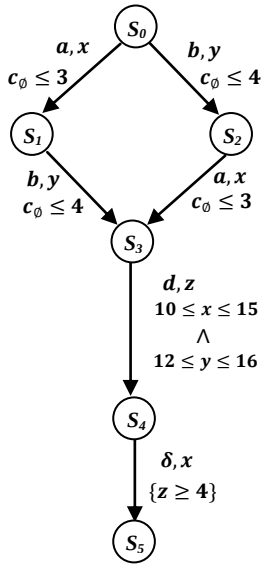

Fig 9: The Timed Automaton generated.

4.3 Algorithm of translation

The translation of *Durational Action Timed Automaton* $\mathcal{D} = (Act, S, s_0, L_S, \mathcal{H}, T_D)$ in *Timed Safety Automata* $\mathcal{A} = (\Sigma, L, l_0, X, E, I)$ equivalent is made while applying the following algorithm:

Begin

1. *The set of locations* $\boldsymbol{L \cup l_{End} = S}$
2. *The initial control state* $\boldsymbol{l_0 = s_0}$
3. *The set of the clocks* $\boldsymbol{X = \mathcal{H}}$
4. *For all transition* (s, G, D, a, x, s')*of* T_D*, we constructs the transition* (l, l', a, λ, r)*of Timed Safety Automata where:*
   (a) *The source control state (location)* $\boldsymbol{l = s}$
   (b) *The target location* $\boldsymbol{l' = s'}$
   (c) *The label of transition is* $\{\boldsymbol{a}\}$
   (d) *The guard of transition* $\boldsymbol{\lambda = G}$
   (e) *The initialization of clocks* $\boldsymbol{r = x}$
5. *For all state* l *of the set of states* $\boldsymbol{L \setminus l_{End}}$ *, the invariant* $I(l)$ *is calculated by the following algorithm :*
   (a) *For all transition labelled by a and therefore of state l do:*
   If D is not empty (void) then
   $I(l) \leftarrow I(l) \wedge not(D(s))$
   Else $I(l) \leftarrow true$

The global invariant associated to the state l *is the conjunction of all deadlines calculated for all transitions starting from l.*

End.

5. UPPAAL FOR VERIFICATION OF DATA*

5.1 Temporal logics

Temporal logic has been proposed as a formalism to *specify* and *verify* the correctness of computer programs in 1977 by Pnueli [22]. It supports the formulation of properties on timed behavior systems. Different interpretations of temporal logics exist; linear and branching temporal logics. The most successful propositional temporal logics are LTL (*Linear Temporal Logic*) [22], CTL (*Computation Tree Logic*) [23] and TCTL (*Timed Computation Tree Logic*) [3]. TCTL is a real-time extension of CTL for which model checking algorithms and several tools exist.

The temporal operators:

- ***X*** (ne***X***t),
- ***U*** (**U**ntil),
- ***G*** (always or ***G***lobally),
- ***F*** (eventually or ***F***uture),
- ***E*** (for some path) and
- ***A*** (for all paths).

The formula:

- $\boldsymbol{X_{\varphi}}$ means $\boldsymbol{\varphi}$ is verified in the next state,
- $\boldsymbol{G_{\varphi}}$, to all future states,
- $\boldsymbol{F_{\varphi}}$, to some future state, and
- $\boldsymbol{U}$**,** to all future states until a certain condition becomes valid.

Example of TCTL formula: every transmission of a message is followed by a reply within 5 units of time. Formally:

$$\boldsymbol{AG}[\boldsymbol{send}(\boldsymbol{m}) \Rightarrow \boldsymbol{AF}_{<5}\, \boldsymbol{receive}(\boldsymbol{r_m})]$$

In the following, we focus only in verification of the *bounded liveliness property*. The formula $\boldsymbol{\varphi \cup_{\leq t} \psi}$ is true, for

execution ρ, iff there exists a state s, located within t time units from the initial state verifying ψ and such as all previous states along ρ verify φ.

5.2 Verification

Let's consider the actions a, b and d has respective durations $10, 12$ and 4 and are offered for the following durations $3, 4$ and 4.

The timed automaton generated from DATA* is described by Figure 10.

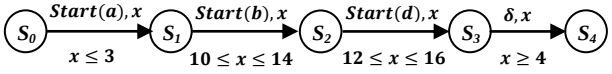

Fig 10: Example of Timed Automaton to simulate.

First, we will proceed with the transformation of bounded liveness property by a simple safety property for reasons of performance. This step is inspired by the works [17].

For this purpose, we increase the model by two variables:

- A clock *time*, to overcome the problem of lost information of time past, following the reuse clocks. Therefore *time* is a reference clock.
- A boolean variable *bool*.

We must first examine time zones generated using the UPPAAL simulator in order to explain the usefulness of *time* and *bool*.

A zone associated with a state s, defined by a clock constraint, describes the maximum time that the system can stay in this state. These zones are a symbolic representation and its construction is on the fly algorithms (For more details, the reader is referred to the work on graph regions [2]).
Suppose we want to verify that the time between the beginnings of the two actions a and d illustrated in the Figure 10 respect an interval $[i, j]$.

First initialize the clock *time*. If the initialization is made on the first transition s_0 to s_1, the duration for which action a is offered will not be detected by the clock *time*, for this initialization *time* is made before crossing the transition s_0 to s_1.

The zones calculated by UPPAAL are as follows:

$$\begin{cases} time \geq 0 \\ x \geq 0 \\ time = x \end{cases} \xrightarrow{Stat\ (a)} \begin{cases} time \geq 0 \\ x \geq 0 \\ time - x \ \ in\ [0,3] \end{cases}$$

$$\xrightarrow{Stat\ (b)} \begin{cases} time \geq 10 \\ x \geq 0 \\ time - x \ \ in\ [10,17] \end{cases}$$

$$\xrightarrow{Stat\ (d)} \begin{cases} time \geq 22 \\ x \geq 0 \\ time - x \ \ in\ [22,33] \end{cases}$$

$$\xrightarrow{Stat\ (\delta)} \begin{cases} time \geq 26 \\ x \geq 0 \\ time - x \ \ \leq -26 \end{cases}$$

The zone Z_0 (corresponds to statess_0) described by the constraints $time \geq 0, x \geq 0$ allow the system to stay indefinitely and it corresponds to the semantics model. About the constraint$time = x$, it expresses as time passes the same way for both clocks namely $time$ and x. Now see the constraint $time - x \ in\ [0,3]$ of Z_1. We know that action a can start if it respects the interval $[0,3]$. Observing other zones, the interval given by $time - x$ always corresponds to earliest and later start dates. Effectively b begins at the earliest at time 10(end of a) and at least $3 + 10 + 4 = 17$ (3and 4 delays are committed by the actions a and b, 10 is the execution time of a). Action d begins after the end of b so at the earliest after $10 + 12 = 22$ and at least $3 + 10 + 4 + 12 + 4 = 33$ using the same reasoning before.

This information is very useful for verification, it suffices to initialize the variable *bool* by 1 (*true*) the entrant transition to the state s_3, triggering the start of d, and 0 (*false*) on outgoing transition. Thus we can use the formula of safety $\boldsymbol{A}\,[](\boldsymbol{b\ imply\ i \leq time - x \leq j})$ to verify that the beginning of d meets an interval after the beginning of a.

The automaton increased by *time* and *bool* is illustrated by Figure 11.

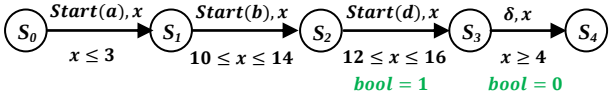

Fig 11: The Timed Automaton increased.

In the case of parallelism, this approach remains valid. The verification of the same property i.e. the beginnings of the two actions a and d illustrated in the Figure 9 respects an interval $[i, j]$.

The only constraint we are interested among those that constitute the zone corresponding to the state $s_4$is: $time - z\ in\ [12, 18]$ the lower bound represents the maximum between duration of action a and b namely 10 and 12 which represents the earliest start of action d. For the upper bound, this represents the maximum delay that can be done and this corresponds to the minimum (due to synchronization) between $10 + 3 + 5$ and $12 + 4 + 4$. So the property to verify is formulated as follows: $\boldsymbol{A}\ [](\boldsymbol{b\ imply\ i \leq time - z \leq j})$.

6. CASE STUDY

The purpose of this section is to show how to specify a system behavior in DATA* model. Then, we present our verification approach of quantitative properties based on DATA* model.

We take as example the gas burner, which is a classic example widely used as case study for the specification of real-time systems. This example was introduced in [9] where the specification is written in the formal specification Duration Calculus [9]. A simplified schema is given in Figure 12.

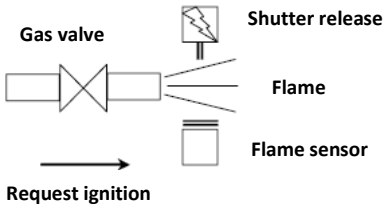

Fig 12: Primitive scheme of gas burner.

6.1 Verification

In our case, we adopt an approach of verification by model checking [8]. Using our approach we verify the quantitative properties of a reactive system amounts to the model by a DATA* as described in Figure 13 then transformed into timed automaton.

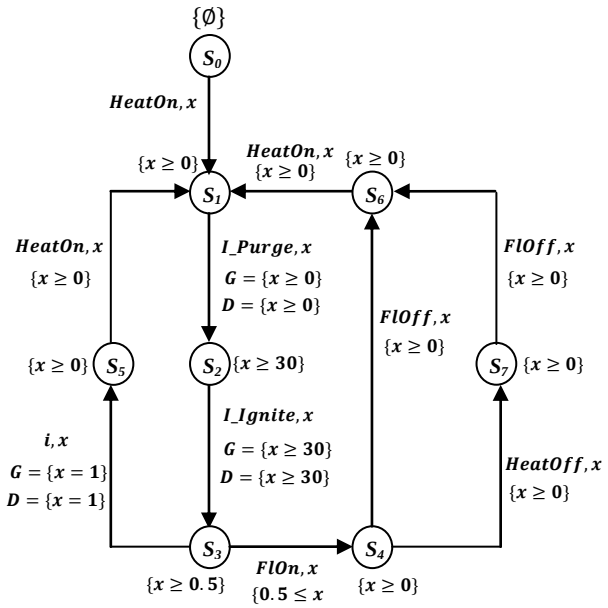

Fig 13: DATA* of gas burner.

6.2 Bounded liveness property verified

1. «The burner should trigger a spark ignition after a request from the user in 30 seconds».

We begin by increased timed automaton result by two variables *time* and *b* (see Figure 14).

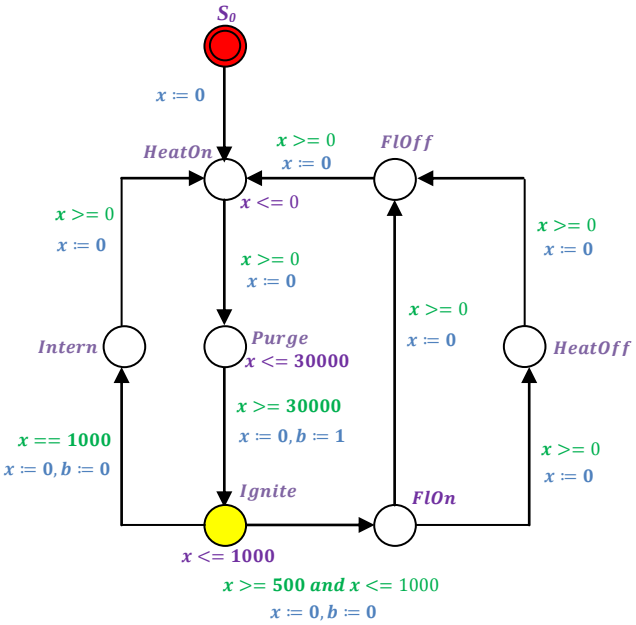

Fig 14: The timed automaton increased with *time* and *b*.

The property to verify:

$$\boldsymbol{A}\ [](\boldsymbol{b\ imply\ time - x \geq 30000})$$

2. «A request ignition can always be satisfied in interval of 30.5 seconds to 31 seconds»

We will verify the property:

$$\boldsymbol{E}[](\boldsymbol{c\ imply\ 30500 \leq time - x\ and\ time - x \leq 31000})$$

after increasing the timed automaton by two variables *time* and *c* (see Figure 15).

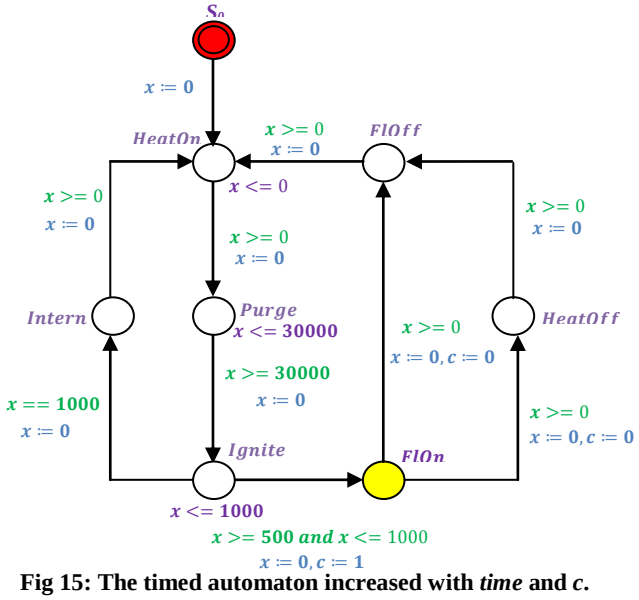

Fig 15: The timed automaton increased with *time* and *c*.

The results obtained using the tool UPPAAL are represented in Figure 16.

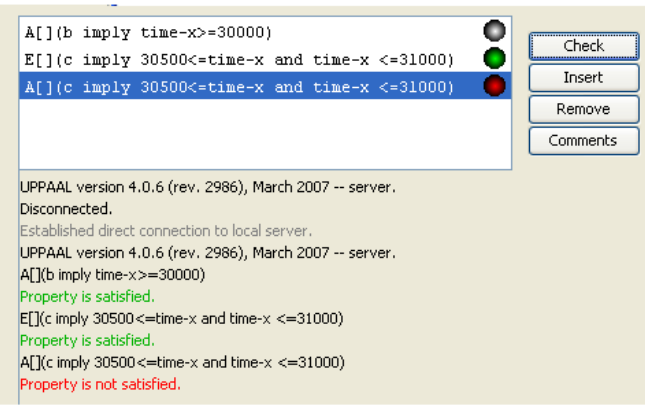

Fig 16: The results of the verification.

7. CONCLUSION AND PERSPECTIVES

In this paper, we have proposed an approach for verifying real time systems. The systems are modelled by *Durational Action Timed Automata* (DATA*). Properties are expressed in temporal logic TCTL*(Timed Computation Tree Logic)*.

Our approach is based on the translation of DATA*'s in Timed Safety Automata.

As we have demonstrated in this work, the interest of this translation is twofold. First, the advantage offered by DATA*s model in which only the beginnings of the actions are considered. Second, this translation allowed the use of UPPAAL tool. Since the model resulting is Timed Safety Automata.

As perspective, we plan to apply our study on other systems, such as real-time communication protocols, examples of "academic" as the media stream (Multimedia Stream) etc... . Also the use of translation approach for testing real time system.

8. REFERENCES

[1] Alur, R. and Dill, D. L. 1990. Automata for modeling real-time systems. In *ICALP*, pages 322–335.

[2] Alur, R. and Dill, D. 1994. A theory of timed automata. Theoretical Computer Science 126, 183-235.

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

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

[5] Bergstra, J. A. and Klop, J. W. 1985. Algebra of communicating processes with abstraction. TCS, 37:77—121.

[6] Bolognesi, T. and Brinksma, E. 1987. Introduction to the ISO specification language LOTOS. Computer Networks and ISDN Systems, 14:25—59.

[7] Bowman, H. and Gomez, R. 2006. 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.

[8] Clarke, E.M., Emerson, E.A. and Sistla, A. P. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems, 8(2): 244-263, (January 1986).

[9] Chaochen, Z., Hoare, C. A. R. and Ravn, A. P. 1991. A Calculus of Durations. Information Processing Letters, 40(5):269—276.

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

[11] Henzinger, T., Nicollin, X., Sifakis, J. and Yovine, S. 1994. Symbolic model checking for real-time systems. Information and Computation 111(2), 193-244.

[12] Hoare, C. A. R. 1985. Communicating Sequential Processes. Prentice Hall.

[13] Kitouni, I. 2008. Determinisation des automates temporises avec durees d'actions pour le test formel , Aggregation Master's thesis, Universite Mentouri, 25000 Constantine, Algerie, (Juin 2008).

[14] Kitouni, I., Hachichi, H. and Saïdouni, D.E. 2012. 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).

[15] Larsen, K.G., Pettersson, P. and Yi, W. 1997. UPPAAL in a nutshell. Springer International Journal of Software Tools for Technology Transfer, 1(1+2).

[16] Petri, C. 1962. Kommunikation mit Automaten. PhD thesis, Institut für Instrumentelle Mathematik, Bonn, Germany.

[17] Pettersson, P., Lindahl, M. and Yi, W. 1998. Formal Design and Analysis of a Gear Box Controller. In Proc. Of the 4th Workshop on Tools and Algorithms for the Construction and Analysis of Systems, number 1384 in Lecture Notes in Computer Science, pages 281.297. Springer-Verlag, (Mar 1998).

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

[19] Saïdouni, D. E., Belala, N. and Bouneb, M. 2008. 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.

[20] Saïdouni, D. E., Kitouni, I. and Hachichi, H. 2011. Formalisation du calcul de l'automate des regions agrege d'un automate temporise avec durees d'actions, MISC REPORT 11001, Universite Mentouri, 25000 Constantine, Algerie.

[21] Yovine, S. 1997. Kronos: A verification tool for real-time systems. International Journal on Software Tools for Technology *Transfer*, 1(1/2):123–133, (October 1997).

[22] [Pnu77] Pnueli, A. 1977. The temporal logic of programs. In *Proc. of the 18th IEEE Symposium on Foundations of Computer Sciences*, pages 46-77.

[23] [CE81] Clarke, E.M. and Emerson, E.A. 1981. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. of Workshop on Logic of Programs*, LNCS 131, pages 52 – 71. Springer-Verlag.

[24] [BGOOS04] Bozga, M., Graf, S., Ileana Ober, Iulian Ober, and Sifakis. J. 2004. The IF toolset. In *SFM-RT 2004*, LNCS 3185, pages 237–267. Springer.

[25] D. Kozen. 1983. Results on the propositional mu-calculus. Theoretical Computer Science, 27:333–354.

[26] Bornot, S., Sifakis, J. and Tripakis, S. 1998. Modeling urgency in timed systems. In *Proc. of COMPOS 1997*, LNCS 1536, pages 103–129. Springer.

[27] Barbuti, R. and Tesei, L. 2004. Timed automata with urgent transitions. *Acta Informatica*, 40(5), (March 2004).

[28] Gómez, R. 2009. Verification of Timed Automata with Deadlines in Uppaal. Technical Report No. 02-08. University of Kent at Canterbury, (July 8, 2009).