On the Verification by Approximation of Duration Systems

Narjes Berregeb and Riadh Robbana
LIP2 Laboratory, Tunisia

Abstract: *We consider the problem of verifying invariance properties for duration systems. Such systems are (extended) timed graphs with duration variables. They are especially suitable for describing real time schedulers. However, for this kind of systems, the verification problem of invariance properties is in general undecidable. We propose an over approximation method based on a particular extension of a given duration system, and we show that our over approximation includes all the digitization of all the real computations of the duration system. The over-approximated system can then be used to perform an interesting close analysis of invariance properties of the initial system, while other existing approaches fail.*

Keywords: *Approximation, digitization, duration systems, formal verification, real-time scheduler.*

Received March 1, 2003; accepted October 30, 2003

1. Introduction

Timed graphs constitute a powerful formalism widely adopted for modelling real-time systems [3, 7, 16, 18, 21]. A timed graph is a finite control locations graph, supplied with a set of *clocks* that can be tested and reset at each transition between locations. Each clock counts the elapsed time since its last reset; clocks values range over the positive reals and they are supposed to increase continuously. So, the clocks of a timed graph can be seen as continuous (real valued) linear variables running with rate 1 at every control location; the tests on these clocks allow one to explain the elapsed time between transitions executed by the modelled system.

However, it is often interesting to account for the accumulated times spent by computations at some particular locations. This corresponds to the concept of *duration* introduced in [12]. In particular, the whole time spent by some computation is simply the accumulation of the times spent at each visited location. For instance, consider a system where several tasks are executed in parallel. Suppose that we are interested in constraining the execution time of some particular task and assume that this task may be interrupted by other tasks of higher priority. Then, the constraint on the execution time of the considered task must be expressed using the accumulated times corresponding to its execution. Intuitively, to compute these accumulated times, we must use a clock that can be stopped (frozen) when the task is interrupted, and resumed when the task is active. This is typically the case of real-time schedulers with pre-emption [8]. Thus, a natural and interesting extension of timed graphs remains consistent considering *duration variables* that count accumulated times spent at some particular control locations.

Actually, duration variables are continuous linear variables with rates 0 or 1 at each location. Hence, we use the Duration Variables Timed Graphs (DVTG's) that have been introduced in [8] which are defined exactly as timed graphs except that they involve duration variables instead of clocks. DVTG are particular cases of the general models of hybrid systems proposed in [4, 6, 10, 18, 19]. Duration variables are called *integrators* or *stopwatch* in [4, 18]. It is proved in [13] that stopwatch automata [18] have the same expressivity as linear hybrid automata [5].

The problem we consider in this paper is the verification of invariance properties for timed systems modelled by DVTG's. Invariance properties correspond to safety requirements on the behavior of these systems, and thus, they constitute the major part of their specifications [20]. Invariance properties are the duals of reachability properties. Hence, the verification of invariance properties is equivalent to solving reachability problems.

It is well known that the reachability problem for timed graphs is decidable [3]. The decision procedure for these systems is based on the construction of a finite *region graph* obtained by partitioning the (non-countable) set of states into a finite set of regions such that all the states in a same region satisfy the same reachability properties [1, 2]. However, such a finite region graph does not exist in general when integrators are considered. Actually, it has been shown that the reachability problem is undecidable for timed graphs extended by one integrator [11].

This paper presents a technique extending a given DVTG system into another one containing the initial

computations as well as additional ones. Then we present a discretisation technique (called also digitization) allowing the translation from the continuous case to the discrete one. Using this digitization, we show that to each real computation in the initial system corresponds a discrete computation in the extended system. Thanks to this digitization technique, we can determine all the computations leading to locations falsifying the invariant. We show that our verification method can be used to verify duration systems represented by DVTG's with a finite number of integrators while all the known models for which reachability is decidable [6, 8, 18] or can be verified by approximation [22, 23] have strong constraints and have at most one integrator.

The paper is organized as follows. In the next section, we introduce our computational models, namely the duration variables timed graphs and their operational semantics. Section 3 defines the invariance and reachability properties and sets the verification problem that this paper deals with. Section 4 presents an example of the use of DVTG's to model timed systems and invariance formulas to express safety requirements on these systems. Then, in section 5, we present the notion of digitization [15] that we use and we define our approximation method. In section 6, we present the verification results. Concluding remarks are presented in section 7.

2. Duration Variables Timed Graphs

We introduce in this section models for timed systems, called *duration variables timed graphs*, which are extensions of the well-known timed graphs [3]. A Duration Variables Timed Graph (DVTG) is described by a finite set of locations and a transition relation between these locations. In addition, the system has a set of *duration variables* that are constant-slope continuous variables. Each of them changes continuously with a rate in {0, 1} at each location of the system (the rates of a same variable at different locations may be different). The transitions between locations are conditioned by *guards* that are arithmetical constraints on the values of the duration variables. The execution of any transition may reset some subset of the duration variables of the system. Duration variables can be seen as clocks that can be *stopped* (frozen) at some locations and then *resumed* at some other locations. This allows one to reason about the durations of some particular locations instead of the whole elapsed time in some computation. DVTG's are particular cases of the general models of hybrid systems proposed in [4, 9, 14, 18, 19]. We give hereafter their formal definition and their operational semantics.

2.1. Definition

First of all, let us introduce the notion of *guard*. Given a set of variables X, a guard on X is a boolean combination of constraints of the form $x \prec c$ where $x \in X$, c is an integer constant ($c \in \mathbb{N}$), and $\prec \in \{<, \leq\}$, the symbols < and ≤ representing the usual (strict and non-strict) ordering relations over the reals. Let G(X) be the set of guards on X. Clearly, we can assume without loss of generality that a guard is a union of conjunctions of constraints of the form $a \prec x$, $x \prec b$, $x \equiv_n c$ or $x ?_n c$, where the $\prec$'s are in {<,≤}, the a's in $\mathbb{N}$, the b's in $\mathbb{N}$, the c's in $\mathbb{N}$, and the n's in $\mathbb{N}-\{0\}$. We say that a guard is *closed* if all the $\prec$'s it contains are ≤'s (i.e., non-strict inequalities).

Now, let P be a set of atomic propositions and let $\Sigma = 2^P$. Then, a DVTG is a tuple $M = (\Sigma, L, d, ?, X, ?, a, \theta)$ where L is a finite set of *locations*, d is a set of transitions (edges) between locations, i.e., $d \subseteq L \times L$, $? : L ?$ S associates with each location ℓ the set of atomic propositions that hold at ℓ, X is a finite set of *duration variables*, $?: d ?$ G(X) associates with each transition a *guard* which should be satisfied by the duration variables whenever the transition is taken, $a: d ?$ 2^X gives for each transition the set of variables that should be reset when the transition is taken, and finally, $\theta: L \times X ?$ {0, 1} associates with each location $\ell \in L$ and each variable x in X the rate at which x changes continuously while the computation is at ℓ. This means that if the computation stays t amount of time at ℓ, the variation of x is $\theta(\ell, x) \cdot t$.

We say that a variable x is a *timer*, if for every location $\ell \in L$, $\theta(\ell, x) = 1$, otherwise we say that x is an *integrator*. Notice that the class of DVTG's such that all their variables are timers is the class of Timed Graphs (TG's) introduced in [3].

2.2. State Graph

We now give an operational semantics for the DVTG's. Consider a DVTG M. A state of the model M consists of a location and a valuation that assigns to each variable a real value, i.e., a state is a pair (ℓ, ν) such that $\ell \in L$ and $\nu \in [X \rightarrow \mathbb{R}]$. Let S_M be the set of states of the model M. A state (ℓ, ν) is called *integer state* if $\nu \in [X \rightarrow \mathbb{N}]$. We denote by $N(S_M)$ the set of integer states of S_M.

We associate with the DVTG M a state graph. For this, we define two transition relations $(\rightarrow)$ and (.) between the states of M. The relation $(\rightarrow)$ corresponds to transitions due to time progress at some location whereas (.) corresponds to moves between locations using transitions in d. Before giving the formal definition of these relations, let us first introduce some notations.

Given a valuation $\nu: X \to R$, a variable $x \in X$ and a real value $v \in R$, we denote by $\nu[x \leftarrow v]$ the new valuation which assigns v to x and coincides with ν for all the other variables. Moreover, for any $t \in R^+$, and every location $\ell \in L$, we denote by $[\nu+t]_\ell$ the valuation ν' such that for every $x \in X$, $\nu'(x) = \nu(x) + \theta(\ell, x) \cdot t$. Finally, given a valuation ν and a guard g, we denote by $\nu \models g$ the fact that the evaluation of g under the valuation ν is true.

Now, we define two families of relations between states $\rightarrow^t$ and $\rightarrow_d$ with $t \in R^+$ and $d \in d$. For every $t \in R^+$ and every $d \in d$, these relations are defined as the smallest relations included in $S_M \times S_M$ such that:

- $(\ell, \nu) \rightarrow^t (\ell, [\nu + t]_\ell)$,
- If $d = (\ell_1, \ell_2)$ and $\nu \models (d)$ Then $(\ell_1, \nu) \rightarrow_d (\ell_2, \nu[x \leftarrow 0]_{x \in \alpha(d)})$

We define $\rightarrow = \bigcup_{t \geq 0} \rightarrow^t$ and $\rightarrow = \bigcup_{d \in d} \rightarrow_d$, and we consider the relation $\rightarrow = \rightarrow \cup \rightarrow$. Then, the state graph associated with M is $(S_M, \rightarrow)$. We denote by $\rightarrow^*$ the reflexive-transitive closure of $\rightarrow$.

2.3. Computation Sequences and Trails

We define now the notion of computation sequence of a DVTG M. These sequences are defined as finite sequences of configurations. A configuration is a pair (s, τ) where s is a state in S_M and $\tau \in R^+$ is a time value. Intuitively, a computation sequence is a finite path in the state graph of an extension of M by an observation clock that records the global elapsed time since the beginning of the computation.

Formally, we extend the transition relations $\rightarrow^t$ and $\rightarrow_d$ from states to configurations. We denote these extensions by $(;^t)$ and $(;^d)$ respectively. Given two configurations (s,τ) and (s',τ'), these relations are defined by:

- $(s, t);^t (s', t')$ *iff* $s \rightarrow^t s'$ *and* $t' = t + t$
- $(s, t);^d (s', t')$ *iff* $s \rightarrow_d s'$ *and* $t' = t$

Let us denote by $(;)$ the union of all the $(;^t$'s) and the $(;^d$' s.) Then, a *computation sequence* of M starting from a state s is a finite sequence $(s_0, \tau_0); (s_1, \tau_1) \ldots ; (s_n, \tau_n)$ such that $s_0 = s$ and $\tau_0 = 0$. We denote by CS(M, s) the set of computation sequences of M starting from s.

Now, let us introduce the notion of *complete* computation sequences which is useful for the digitization issue. We say that a computation sequence $(\ell_0, \nu_O, \tau_0); (\ell_1, \nu_1, \tau_1); \ldots (\ell_n, \nu_n, \tau_n)$ is *complete*, if for each $u \in N$ such that $u \leq \tau_n$, there exists some rank $i \geq 0$ such that $\tau_i = u$.

It is clear that every computation sequence can be completed by adding intermediate configurations corresponding to the missing integer time values. Indeed, given a computation sequence, for each rank $i \geq 0$ such that $\tau_i \neq \tau_{i+1}$ (hence, necessarily $\ell_i = \ell_{i+1}$), let $\{u_1, \ldots, u_m\}$ be the set of integers between τ_i and τ_{i+1}. Then, we can insert between the configurations (ℓ_i, ν_i, τ_i) and $(\ell_{i+1}, \nu_{i+1}, \tau_{i+1})$ the following sequence:
$(\ell_i, \nu_i, \tau_i); {}^{t1}(\ell_i, \nu_i^1, u_1) \ldots; {}^{tm}(\ell_i, \nu_i^m, u_m); {}^{tm+1}(\ell_{i+1}, \nu_{i+1}, \tau_{i+1})$
where $t_1 = u_1 - t_i$, $t_{m+1} = t_{i+1} - u_m$, $\forall j \in \{2, \ldots, m\}, t_j = 1$, and

$$\forall j \in \{1, \ldots, m\}, \quad n_i^j = [n_i + \sum_{p=1}^{j} t_p]_{\ell_i}.$$

Now, we introduce the notion of *trail* which is also useful for the digitization issue. We define a trail as a sequence: $(\ell_0, \tau_0); (\ell_1, \tau_1); \ldots (\ell_n, \tau_n)$
where the τ's are positive real such that $\tau_0 = 0$, and for every $i \geq 0$, $\tau_i \leq \tau_{i+1}$.

Given a computation sequence of M

$$s = (\ell_0, n_O, t_0); (\ell_1, n_1, t_1); \ldots (\ell_n, n_n, t_n)$$

the trail *corresponding* to s is the sequence

$$r_s = (\ell_0, t_0); (\ell_1, t_1); \ldots (\ell_n, t_n).$$

Finally, we introduce the notions of *integer computations sequences* and *integer trails*. We say that a computation sequence (resp. trail) is an *integer computation sequence* (resp. *integer trail*) if all the τ_i's in its configurations are integers.

Notation 1: let M be a DVTG, we denote by Comp (M) the set of all the real computations of M, we denote by Digit (Comp (M)) the set of all the digitization of all the real computations of M.

3. Invariance Properties

Invariance properties correspond to safety requirements on the behaviors (computations) of some given system (DVTG). These properties are the duals of reachability properties. We define in this section formulas expressing invariance and reachability properties on the variables of DVTG's.

Let $M = (s, L, d, \Pi, X, \rightarrow, a, \theta)$ be a DVTG. Then, an *invariance formula* (resp. *reachability formula*) on M is written $\forall\Box\phi$ (resp. $\exists\Diamond\phi$) where ϕ is a boolean combination of atomic propositions ($p \in P$), and constraints of the form $x \prec c$ with $x \in X$, $c \in N$, and $\prec \in \{<, \leq\}$. The semantics of invariance and reachability formulas is defined by a *satisfaction relation* $\models$ between the states in S_M and these formulas. For every state $s = (\ell, \nu)$, the satisfaction relation $\models$ is inductively defined by:

$$s \models \forall\Box f \text{ iff } \forall s' \in S_M, \ s \rightarrow s' \text{ implies } s' \models f$$

$$s \models \exists\Diamond f \text{ iff } \exists s' \in S_M, \ s \rightarrow s' \text{ and } s' \models f$$

$$s^2 p \text{ iff } p \hat{I} P(\ell)$$
$$s^2 \emptyset f \text{ iff } s 2 f$$
$$s^2 f_1 \acute{U} f_2 \text{ iff } s^2 f_1 \text{ or } s^2 f_2$$
$$s^2 x \prec c \text{ iff } n ? x \prec c$$

Clearly, we have $\forall \square \phi = \neg \exists \Diamond \neg \phi$. Then, using standard laws of the Boolean connectives, together with the fact that any formula $\exists \Diamond (\phi_1 \vee \phi_2)$ is equivalent to $(\exists \Diamond \phi_1) \vee (\exists \Diamond \phi_2)$, it can be easily shown that every invariance formula is equivalent to the negation of a formula of the form

$$\bigvee_{i=1}^{n} \$ \grave{a} \left(P_i \wedge \bigwedge_{j1}^{m_i} x_i^j \prec_i^j c_i^j\right) \quad (1)$$

where the Π_i's are boolean combinations of atomic propositions, the x_i^j are in X, the c_i^j are integers, and the $\prec_i^j$ are in $\{<, >, \leq, \geq, \equiv_n, ?_n\}$. In this paper, we consider the verification of invariance formulas on DVTG's, i.e., deciding whether some given state s of some given DVTG satisfies some invariance formula ϕ ($s^2\phi$). As we have seen, this problem reduces to the verification of reachability formulas.

It is well known that this problem is decidable for timed graphs [3]. However, it has been shown that this problem is undecidable for DVTG's [11], and even for DVTG's with one integrator (non timer) [14]. In [4], the problem is shown to be semi-decidable.

4. A Real-Time Scheduler

In this section, we illustrate our framework through an example. This example cannot be treated by other existing formal verification methods given in [6, 8, 18]. Actually in [6, 8, 18] all the models for which reachability is decidable have strong constraints and have at most one integrator.

Example 1

We consider a real time scheduler with pre-emption which handles three (families of) tasks a and b and c using the following policy: The priority of the tasks are as follows: $a \prec b \prec c$, in other words the tasks c have the highest priority, the tasks a have the lowest priority and the tasks b are in the middle.

The timing assumptions are:

- The execution time of a is in the real interval]0, 2[.
- The execution time of b is in the real interval]0, 1[.
- The execution time of c is in the real interval]0, 3[.

The system is modelled in the following manner: we consider the DVTG represented in Figure 1. The model has eleven locations with the following interpretations:

- In S_0, a, b and c are not active.
- In S_1, a is active (b and c are not active). In this location a can be suspended.
- In S_9, a is active (b and c are not active). In this location a cannot be suspended.
- In S_6, b is active (a and c are not active). In this location b can be suspended.
- In S_7, b is active, a is suspended and c is not active. In this location b can be suspended.
- In S_8, b is active, a is suspended and c is not active. In this location b cannot be suspended.
- In S_{10}, b is active, (a and c are not active). In this location b cannot be suspended.
- In S_2, c is active (a and b are not active). c cannot be suspended.
- In S_3, c is active, a is suspended and b is not active. c cannot be suspended.
- In S_5, c is active, b is suspended and a is not active. c cannot be suspended.
- In S_4, c is active, a and b are suspended. c cannot be suspended.

In order to implement the timing assumptions given above, we use four variables x, y, z and t. The variables x serving to count the execution of a and y serving to count the execution of b are integrator because a and b can be interrupted. z serving to count the execution of c is a clock because c is never interrupted. t is a clock serving to count the time elapsed from the arrival of a, b or c from the initial location. x is stopped during the suspensions of a, in the locations S_3, S_4, S_7 and S_8. y is stopped during the suspensions of b that is in locations S_4 and S_5.

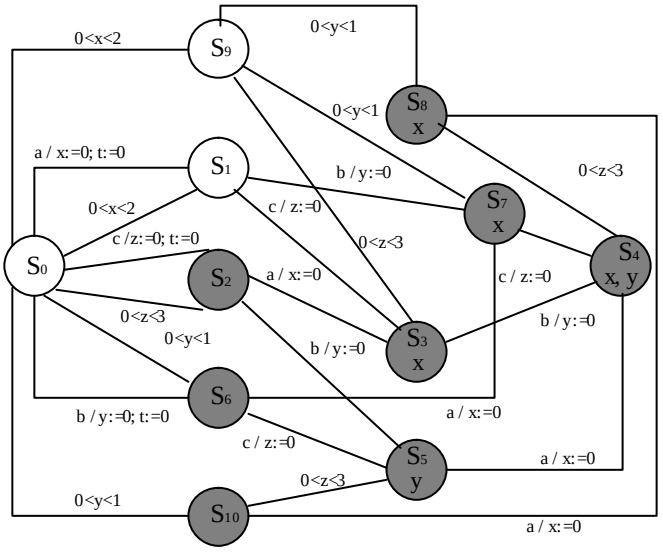

Figure 1. Real time scheduler.

We want to check the following property:

R: The time elapsed between the arrival of a, b or c from the initial location S_0 (Start event) and the end of a session when the initial location is revisited never reaches 6 time units.

The requirement R is expressed by the invariance formula:

$(at\_S_1 \vee at\_S_2 \vee at\_S_6) \Rightarrow \Box \neg (at\_S_0 \Rightarrow t<6)$

where at_S_i is a proposition true only in the location S_i.

5. Approximation

We consider in this paper the verification problem of reachability formulas. To get our detecting procedure, we prove that every real computation of a given duration system which starts from an integer state, has a discretisation which is also a computation of the approximate system obtained from the initial one. Then, we can do the verification problem of reachability formulas on duration system with discrete time.

5.1. Digitization

We present the notion of digitization introduced in [15] which is suitable for the systems we are interested in. Let us introduce some definitions and notations. Let $\tau \in R+$. For every $e \in [0, 1[$, we define the integer $[\tau]_e$ = if $\tau \leq (\lfloor \tau \rfloor + e)$ then

$\lfloor \tau \rfloor$

else

$\lceil \tau \rceil$.

Now, we recall the definition of digitization according to [15]. Given a trail $\rho = (\ell_0, \tau_0); (\ell_1, \tau_1); \ldots (\ell_n, \tau_n)$ and a digitization quantum $e \in [0,1[$, the digitization of ρ w. r. t. e is the *integer trail:*

$[\rho]_e = (\ell_0, [\tau_0]_e); (\ell_1, [\tau_1]_e); \ldots (\ell_n, [\tau_n]_e)$.

From a trail we can deduce the associated real computation by calculating at each step the valuation of the variables, this can be done using the following *definition*:

$$n^k(y) = \sum_{i=j_{k-1}}^{k-1} q(\ell_i, y) \times D^r(i)$$

where j_k denotes the greatest index j such that $j \leq k$, and the transition $;_j$ is of the form $;^d$, with $y \in a(d)$, i.e., y is reset by d. We take $j_k = -1$ if such an index does not exist, and from a digitizated trail we can deduce the associated digitizated computation by calculating at each step the valuation of the variables. This can be done using the following definition:

$$n_e^k(y) = \sum_{i=j_{k-1}}^{k-1} q(\ell_i, y) \times D_e^r(i)$$

We will use this notion of digitization in our approximation method. However in the following example we will see that we can have a DVTG with only one integrator for which there exists a real computation such that all its digitization are not computations of the system. Moreover, we can show that all the discretisations of this real computation are not computations of the system. So we cannot do the verification in the discrete model associated to a given DVTG M. Just before giving this example, we briefly define what we mean by discretisation.

Definition 1: Given a trail $\rho = (\ell_0, \tau_0); (\ell_1, \tau_1); \ldots (\ell_n, \tau_n)$ a discretisation $d(\rho)$ of ρ is defined as follows:

$d(r) = (\ell_0, d(t_0)); (\ell_1, d(t_1)); \ldots (\ell_n, d(t_n))$.
where $\forall i \in N$, $d(t_i) \leq d(t_{i+1})$ *and*

- $t_i \in N \Rightarrow d(t_i) = t_i$
- $u < t_i < u+1 \Rightarrow d(t_i) \in \{u, u+1\}$

Example 2

Let s be the real computation of the DVTG as given in Figure 2.

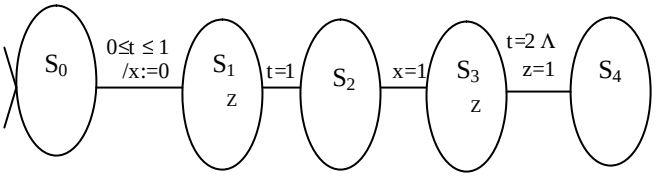

Figure 2. Example.

$s = \{<S_0, (0, 0, 0)>, <S_1, (0.5, 0, 0.5)>, <S_2, (1, 0.5, 0.5)>, <S_3, (1.5, 1, 1)>, <S_4, (2, 1.5, 1)>, \ldots\}$

where the first component of the valuation is the global time, the second is the clock x and the third is the integrator z.

There are four discretisations of ρ, two of them correspond to a uniform digitizations:

$d_1(s) = \{<S_0,(0, 0, 0)>, <S_1,(0, 0, 0)>, <S_2,(1,1, 0)>, <S_3,(1, 1, 0)>, <S_4,(2, 2, 0)>, \ldots\} = [s]_e$ *with* $0.5 < e < 1$

$d_2(s) = \{<S_0,(0, 0, 0)>, <S_1,(1, 0, 1)>, <S_2,(1, 0, 1)>, <S_3,(2, 1, 2)>, <S_4,(2, 1, 2)>, \ldots\} = [s]_e$ *with* $0 \leq e \leq 0.5$

$d_3(s) = \{<S_0,(0, 0, 0)>, <S_1,(0, 0, 0)>, <S_2,(1, 1, 0)>, <S_3,(2, 2, 1)>, <S_4,(2, 2, 1)>, \ldots\}$

$d_4(s) = \{<S_0,(0, 0, 0)>, <S_1,(1,0,1)>, <S_2,(1,0,1)>, <S_3,(1,0,1)>, <S_4,(2,1,1)>, \ldots\}$

5.2. Approximation Method

As we have seen in Example 2 (see Figure 2), some computations of a given DVTG M don't have any discretisation in M. Then we cannot do the verification of reachability in the discrete. Our idea consists of over-approximating the system M by a system M' such that Digit (Comp(M)) $\subseteq$ Comp (M'). In other words the discrete system associated to M contains all the digitization of all the computations of the initial system M.

So, let us first introduce some technical notions. Let e be a digitization quantum. Now, for a each trail ρ, for every $k \geq 0$, we define

$D^r(k) = t_{k+1} - t_k$ and $D_e^r(k) = [t_{k+1}]_e - [t_k]_e$

Intuitively, $\Delta^\rho(k)$ is the time taken by the transition $;_k$ in the real trail ρ (i.e., the time spent at location ℓ_k) whereas $\Delta_e^\rho(k)$ is the time taken by the same transition in the integer trail $[\rho]_e$.

Given $k \in \mathbb{N}$, we define $E_e^k(y)$ to be the difference between the values of a variable y in $[\rho]_e$ and ρ at position k, i.e., just after the transition $;_{k-1}$ (if it exists). Let us give the formal definition.

Let j_k denote the greatest index j such that $j \leq k$, and the transition $;_j$ is of the form $;_t$, with $y \in a(d)$, i.e., y is reset by d. We take $j_k = -1$ if such an index does not exist. Then, we recall

$$n^k(y) = \sum_{i=j_{k+1}}^{k-1} q(\ell_i, y) \times D^r(i)$$

and

$$n_e^k(y) = \sum_{i=j_{k+1}}^{k-1} q(\ell_i, y) \times D_e^r(i)$$

Intuitively, $\nu^k(y)$ (resp. $\nu_e^k(y)$) is the value of y at position k in ρ (resp. in $[\rho]_e$). Then, we have $E_e^k(y) = \nu_e^k(y) - \nu_e^k(y)$

Definition 2: We define a function $\beta: X \times d? \mathbb{N}$ that calculates for each variable $x \in X$ and each transition $e = (s, s')$ the maximum of restarts of x from the last reset of x until the location s in each way.

$\beta(x, e)$ = Maximum {number of restarts(x) from the last reset of x until the location s in each way}.

A restart of a variable x is the change of its rate from 0 to 1. And after a reset of a variable x, if the rate of a variable x in the current location is 1 then the access to this location is considered as a restart of x. For example if in the first location x has a rate equal 1 then the access to the initial state is considered as a restart. That is why for the clocks the function β is equal to 1 for each transition.

Remark 1: In the example given in the previous section (see Figure 1), for the integrators x and y we have $\beta(x, e) \in \{1, 2\}$ and $\beta(y, e) \in \{1, 2\}$ depending on the transition e. Actually, for x we have $\beta(x, (S_9, S_0)) = 2$ and $\beta(x, e) = 1$ for each $e \neq (S_9, S_0)$, and for y we have $\beta(y, (S_9, S_0)) = \beta(y, (S_8, S_9)) = 2$ and $\beta(x, e) = 1$ for each $e \notin \{(S_9, S_0), (S_8, S_9)\}$

Definition 3: A finite preemption DVTG (FP-DVTG for short) is a DVTG that obeys to the following rule:

For every variable $x \in X$ and for every transition e, $\beta(x, e)$ is bounded.

Remark 2: The example given in Figure 1 is an FP-DVTG. Actually, the DVTGs that obey to the following rule are FP-DVTGs:

For every variable $x \in X$ and for every loop, if the rate of x changes in this loop then x must be reset in this loop.

This rule is verified in Example 1.

Proposition 1: For every computation ρ of an FP-DVTG M, every digitization quantum e, every variable $y \in X$, every transition $e \in d$ and every $k \in \mathbb{N}$, we have

$$b(y, e) = 0 \Rightarrow E_e^k(y) = 0$$
$$b(y, e) > 0 \Rightarrow |E_e^k(y)| < b(y, e)$$

Proof: For every computation ρ and every variable $y \in X$ we have

$$E_e^k(y) = \sum_{i=0}^{k-1} q(\ell_i, y) \times (D_e^r(i) - D^r(i))$$

For a given index k, a given transition $e = (\ell_{k-1}, \ell_k)$ and a given variable y, if $\beta(y, e) = 0$ then the rate of y is 0 in the locations visited from the last reset of y. So $\nu_e^k(y) = \nu^k(y) = 0$, thus $E_e^k(y) = 0$.

Now, for a given index k, a given transition $e = (\ell_{k-1}, \ell_k)$ and a given variable y, if $\beta(y, e) > 0$ then the sequence ρ has a unique decomposition into a finite number of sets of indices $\{0,\ldots, j_0-1\}$, $\{j_0,\ldots, j_1\}$, $\{j_1+1,\ldots, j_2-1\}$, $\{j_2,\ldots, j_3\},\ldots\{j_p,\ldots, k\}$, where:

- j_0 is the first index i where $\theta(\ell_i, y) = 1$ after the last reset of y.
- $\theta(\ell, y) = 1$ in the following sets : $\{j_0,\ldots, j_1\},\{j_2,\ldots, j_3\},\ldots\{j_{2n},\ldots, j_{2n+1}\}$ and $\theta(\ell, y) = 0$ in the other sets.

We have $n+1 = \beta(y, e)$ by definition of β, because the number of restarts of y is equal to the number of subsets of indices on which $\theta(\ell, y) = 1$. Then,

$$E_e^k(y) = \sum_{i=j_0}^{j_1-1} (D_e^r(i) - D^r(i)) + \sum_{i=j_2}^{j_3-1} (D_e^r(i) - D^r(i)) + \cdots + \sum_{i=j_{2n}}^{j_{2n+1}-1} (D_e^r(i) - D^r(i))$$

Where j_0 is the first index i where $\theta(\ell_i, y) = 1$ after the last reset of y. Besides, for each $m \in \{0, 1,\ldots, n\}$, we have $-1 < \sum_{i=j_{2m}}^{j_{2m+1}-1} (D_e^r(i) - D^r(i)) < 1$. Thus,

$$-b(y, e) < E_e^k(y) < b(y, e)$$ □

Definition 4: The approximate model M'= App(M) is obtained from M by transforming each guard of a transition e of the form $u \prec y \prec w$ by the guard

If $u - b(y, e) \geq 0$ *then*
$u - b(y, e) \leq y \leq w + b(y, e)$
else
$0 \leq y \leq w + b(y, e)$

where $u, w \in \mathbb{N}$, $x \in X$ and $\prec \in \{<, \leq\}$.

6. Verification of FP-DVTG

We consider in this section the verification problem of reachability formulas for FP-DVTG's. To get our

detecting procedure, we prove that every real computation of a FP-DVTG which starts from an integer state, has a digitization (according to our definition given in the previous section) which is also a computation of the approximate system obtained from the initial one. Then, we show that the verification problem of reachability formulas on FP-DVTG's with discrete time, reduces to the reachability problem in finite state graphs.

6.1. Digitization Result

Let M be a FP-DVTG. Along this subsection, we refer to the following trail

$$r = (\ell_0, t_0);(\ell_1,t_1); \dots (\ell_n,t_n)$$

To simplify some definitions given later, we assume that the value of the variables are initially 0. The consideration of any other integer values does not present any difficulty. Under these assumptions, we prove that ρ has a digitization which also corresponds to a computation sequence of the extending model M'= App(M). We can see that any digitization preserves the satisfaction of the transition guards of the extending model M'= App(M), since the absolute value of the difference between the values of a variable y in ρ in the moment where the transition e is taken and its digitization $[\rho]_e$ is strictly less than $\beta(x, e)$ (Proposition 1). This ensures that $[\rho]_e$ corresponds actually to a computation in M'.

Lemma 1: Let k<n such that the transition e_{k} is of the form $\xrightarrow{d}$, i.e., a discrete move between locations. Then, $|E_e^k(y)|<\beta(y, e)$ implies that for every $l \in \mathbb{N}$ and $u \in \mathbb{N}$, $l \prec \nu^k(y) \prec u$ with $\prec \in \{<,\leq\}$ implies that $\nu_e^k(y) \in \{l-\beta(y, e),\dots, u+\beta(y, e)\}$

Lemma 1 ensures that under the assumption that ρ corresponds to a computation of M, which means in particular that the guard of d is satisfied in ρ, if $|E_e^k(y)|<\beta(y, e)$, then the guard of d in the approximate system is also satisfied in $[\rho]_e$. This is due to the fact that the guard of d in M'= App(M) is *relaxed* by $\beta(y, e)$.

Proof: We have by hypothesis

$$|E_e^k(y)|=|n_e^k(y)- n^k(y)|<b(y, e) \qquad (2)$$

We consider that $l \prec \nu_e^k(y) \prec u+\beta(y, e)$. Since (2) holds by hypothesis, and since $\nu_e^k(y)$ is an integer, we have necessarily $l-\beta(y, e) \leq \nu_e^k(y) \leq u+\beta(y, e)$ then $\nu_e^k(y) \in \{l-\beta(y, e),\dots, u+\beta(y, e)\}$. □

Proposition 2: Let ρ be a computation of a FP-DVTG M, then for each $\epsilon \in [0, 1[$, $[\rho]_\epsilon$ is a computation of M'= App(M).

Proof: Proposition 1 ensures that for a given FP-DVTG M), for each transition e, for each variable y, for each e and for each computation ρ of M we have $|E_e^k(y)|<\beta(y, e)$. The lemma ensures that if $|E_e^k(y)|<\beta(y, e)$ and a guard of e is satisfied in ρ then the relaxed guard by $\beta(y, e)$ is also satisfied in $[\rho]_e$ so $[\rho]_e$ a computation of M'= App(M). □

6.2. Solving the Reachability Problem

In this subsection we show how we can solve the verification problem of reachability formulas for FP-DVTG's. Let M'= (Σ , L, d, Π, X, ?, a, θ) be an extending FP-DVTG obtained from a FP-DVTG model M, the verification problem we consider is $s_0 \models \phi$ where s_0= (ℓ_0, ν_0) is an integer state and is a closed reachability formula given by

$$f = \exists\Diamond (p \wedge \bigwedge_{j=1}^{m} x_j)$$

The consideration of disjunctive reachability formulas of the form (1) (see section 3) is then straightforward.

Our result is based on a reduction of the problem $s_0 \models \phi$ to the reachability problem in a finite state graph. This reduction is made in two steps: The first step remains consistent in considering a *relaxed* FP-DVTG M'_ϕ obtained by adding to M' a target location # which is reachable from s_0 if and only if $s_0 \models \phi$ holds. Then, using the digitization result given in the previous subsection, we show that this reachability problem can be solved by reasoning on a finite graph deduced from $(N(S_{M'_\phi}), \xrightarrow{?} \cup \xrightarrow{1})$.

So, let us define the model M'_ϕ. As we said, we construct M'_ϕ as an extension of M' such that the set of locations of M'_ϕ is the set L augmented by a new location # satisfying a special new atomic proposition at_#, and every location in L that satisfies the state formula π has a transition to # which is guarded by $\bigwedge_{j=1}^{m} \xi_j$.

Formally, $M'_f = (S \cup \{at\_\#\}, L \cup \{\#\}, d_1, P_1, X, ?_1, a_1, q_1)$ where

- $d_1 = d \cup \{(\ell,\#) \mid \ell \in L \text{ and } P(\ell) \models p\}$
- $P_1(\ell) = P(\ell)$ for every $\ell \in L$ and $P_1(\#) = \{at\_\#\}$
- $?_1(d) = ?(d)$ for every $d \in d$ and $?_1(d) = \bigwedge_{j=1}^{m} x_j$ for every $d \in d_1 \setminus d$
- $a_1(d) = a(d)$ for every $d \in d$ and $a_1(d) = \emptyset$ for every $d \in d_1 \setminus d$,
- $q_1(\ell, x) = q(\ell, x)$ for every $\ell \in L$ and $x \in X$, and $q_1(\#, x) = 1$ for every $x \in X$.

Then, it is clear that $s_0 \models \phi$ holds if and only if some state of the form (#,ν) is reachable from s_0 in M_ϕ; i.e.

$s_0 \models \phi$ holds in M' if and only if

$s_0 \models \exists\Diamond$at_# holds in M' (3)

Now, let us consider the problem s_0 ² ∃◊at_# in M'$_\phi$. We recall that s_0 ² ∃◊at_# holds if and only if there exists in M'$_\phi$ some computation sequence starting from s_0

$$s = (\ell_0, n_0, t_0);\ ¼\ ;\ (\ell_n, n_n, t_n)$$

such that $\ell_n = \#$. Let

$$r_s = (\ell_0, 0);\ ¼\ ;\ (\ell_n, n_n)$$

be the trail corresponding to s. Then, by Proposition 2, for each value e in [0, 1[, we have $[\rho_s]_e$ corresponds to a computation of M'$_\phi$. Hence, there exists some integer computation sequence

$$s' = (\ell_0, n_0, 0);\ (\ell_1, n'_1, t'_1);\ ¼\ ;\ (\ell_n, n'_n, t'_n)$$

in the model M'$_\phi$ such that $\ell_n = \#$. So, we can solve the problem s_0 ² ∃◊at_# by considering only integer computations of M'$_\phi$ which means that we can restrict our attention to integer states of M'$_\phi$ and consider only the countable state graph G= (N($S_{M'_\phi}$),? $^1\cup$.). However, this state graph is infinite. We show that nevertheless, we can solve the problem s_0 ² ∃◊at_# by considering a finite graph G' derived from G.

Let us first introduce some notations. For every variable x, we define two constants c_x and n_x. Let c_x be the maximal constant which is compared with x in M'$_\phi$, i.e., c_x= max({c | x≤ c or x≥ c is a guard of M'$_\phi$}); and let n_x be the lcm (lowest common multiple) of the set {n | ∃c, x≡$_n$c or x?$_n$c is a guard of M'$_\phi$. We also introduce two functions ϑ and λ defined as follows: ϑ(a, n)= c iff a≡$_n$c holds and λ(a, n)= if ϑ(a, n)= 0 then n else ϑ(a, n). Let ν be an integer valuation, we denote by λ(ν) the integer valuation defined by

$$l(n)(x) = \text{if } n(x) \pounds c_x + n_x \text{ then } n(x) \text{ else } c_x + l(n(x) - c_x, n_x).$$

In order to construct the graph G', we observe first that all the variables of M'$_\phi$ are monotonic (never decrease). Thus, for every variable x, it is clear that beyond c_x, the exact value of x is not relevant for the guards of the form x≤ c or x≥ c. Moreover, for the guards of the form ≡$_n$ or ?$_n$, we can find a finite amount of values of x beyond c_x that represent all the possible behaviors of valuations of x w. r. t the guards of M'$_\phi$. Indeed, for every guard x≡ $_{ni}c_i$ of M'$_\phi$, it is easy to see that if ν (x) ≥ c_x+n_x+1, then ν(x)≡ $_{ni}c_i$ holds iff c_x+λ(ν(x)-c_x, n_x)≡ $_{ni}c_i$ holds too. Thus, we have the following fact.

Lemma 2: For every integer valuation ν ∈ [X? N], and for every guard g in M'$_\phi$, ν ? g iff λ (ν)? g.

By Lemma 2, we can consider a finite graph G'= (Q,? $^1_\equiv \cup$.) induced by G and defined by:

- $Q = \{(\ell, n)\ Î\ N(S_{M'_f}):\ "\ x Î\ X,\ 0 £ n(x) £ c_x + n_x\}$,
- (ℓ, n)? $^1_\circ$ $(\ell,\ l([n+1]_\ell))$

Then, we have the following result.

Lemma 3: (ℓ_0, ν_0) ² ∃◊at_# holds in M'$_\phi$ if and only if there exists some state (#,ν) which is reachable from the state (ℓ_0, λ(ν_0)) in the graph G'.

Finally, from (3) and Lemma 3, we get the following result.

Theorem: Let M be a relaxed FP-DVTG, M' its extension, s_0 an integer state in N(S_M), and ϕ a closed reachability formula. Then, if s_0 ² ϕ is true in M then s_0 ² ϕ is true in the finite graph (N ($S_{M'_\phi}$), ? $^1\cup$.).

7. Conclusion

We proposed an approximation method for verifying reachability properties over duration systems, while other existing formal verification approaches in general fail. We believe that our approximation allows a good analysis of duration systems. Besides, we can combine this result and the results described in [15] about the verification of MTL logic over timed graphs, in order to verify a large fragment of MTL logic over duration systems.

In a future work, we plan to combine our results with those described in [13] showing that every Linear Hybrid Automata can be transformed into a stopwatch automata recognizing the same language, and the results in [15] to develop a tool dedicated for the verification of a large fragment of MTL over linear hybrid automata.

References

[1] Alur R. and Dill D., "A Theory of Timed Automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183-235, 1994.

[2] Alur R. and Dill D., "Automata for Modeling Real Time Systems," *in Proceedings of International Colloquium on Automata and Programming* (*ICALP'90)*, Warwick, 1990.

[3] Alur R., Courcoubetis C., and Dill D., "Model Checking for Real-Time Systems," *in Proceedings of 5th Symposium on Logic in Computer Science (LICS'90)*, Philadelphia, USA, 1990.

[4] Alur R., Courcoubetis C., Henzinger T., and Ho P. H., "Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems," *in Proceedings of Hybrid Systems*, Springer-Verlag, 1993.

[5] Alur R., Courcoubetis C., Halbwatchs N., Henzinger T. A., Ho P. H., Nicollin X., Olivero A., Sifakis J., and Yovine S., "The Algorithmic Analysis of Hybrid Systems," *Theoretical Computer Science*, vol. 138, pp. 3-34, 1995.

[6] Bouajjani A. and Robbana R., "Verifying ω-Regular Properties for Subclasses of Linear Hybrid Systems," *in Proceedings of Computer-Aided Verification (CAV'95)*, Belgium, 1995.

[7] Bouajjani A., Echahed R., and Robbana R., "On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures," *in Proceedings of Hybrid Systems and Autonomous Control*, New York, USA, 1995.

[8] Bouajjani A., Enchahed R., and Robbana R., "Verifying Invariance Properties of Timed Systems with Duration Variables," *in Proceedings of Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFTS'94)*, 1994.

[9] Bouajjani A., Echahed R., and Robbana R., "Verification of Context-Free Timed Systems Using Linear Hybrid Observers," *in Proceedings of Computer-Aided Verification (CAV'94)*, USA, 1994.

[10] Bouajjani A., Lakhnech Y., and Robbana R., "From Duration Calculus to Linear Hybrid Systems," *in Proceedings of Computer-Aided Verification (CAV'95)*, Belgium, 1995.

[11] Cerans K., "Decidability of Bisimulation Equivalence for Parallel Timer Processes," *in Proceedings of Computer-Aided Verification (CAV'92)*, Montreal, Canada, 1992.

[12] Chaochen Z., Hoare R., and Ravn P., "A Calculus of Durations," *Information Processing Letters*, vol. 40, pp. 269-276, 1991.

[13] Frank C. and Kim L., "The Impressive Power of Stopwatches," *in Proceedings of Conference on Concurrency Theory (CONCUR'00)*, Pennsylvania, USA, 2000.

[14] Henzinger A, Kopke W., Puri A., and Varaiya P., "What's Decidable about Hybrid Automata?," *Journal of Computer and System Science*, vol. 57, pp. 94-124, 1998.

[15] Henzinger A., Manna Z., and Pnuelli A., "What Good are Digital Clocks?," *in Proceedings of International Colloquium on Automata Languages and Programming*, 1992.

[16] Henzinger A., Nicollin X, Sifakis J., and Yovine S., "Symbolic Model-Checking for Real Time Systems," *in Proceedings of 7th Symposium on Logic in Computer Science (LICS'92)*, Santa Cruz, USA, 1992.

[17] Kesten Y., Pnueli A., Sifakis J., and Yovine S., "Decidable Integration Graphs," *Information and Computation*, vol. 150, no. 2, pp. 209-243, 1999.

[18] Kesten Y., Pnueli A., Sifakis J., and Yovine S., "Integration Graphs: A Class of Durable Hybrid Systems," *in Proceedings of Hybrid Systems*, Springer Verlag, 1993.

[19] Nicollin X, Olivero A., Sifakis J., and Yovine S., "An Approach to the Description and Analysis of Hybrid Systems," *in Proceedings of Hybrid Systems*, Springer-Verlag, 1993.

[20] Pnueli A. and Shahar E., "Liveness and Acceleration in Parameterized Verification," *in Proceedings of Computer-Aided Verification (CAV'00)*, Chicago, USA, 2000.

[21] Robbana R., "Réduction et Vérification de Systèmes Temps-Réel Distribués," *in Colloque Francophone de l'Ingénierie des Protocoles (CFIP'99)*, Nancy, France, 1999.

[22] Robbana R, "Verification of Duration Systems Using an Approximation Approach," *Journal of Computer Science and Technology*, vol. 18, no. 2, pp.153-162, 2003

[23] Robbana R., "Verification of Integrated Timed Systems," *in Proceedings of Maghrebian Conference on Software Engineering (MCSEA'98) and Artificial Intelligence*, Tunis, 1998.

Narjes Berregeb received an engineering degree in computer science from the Faculty of Sciences of Tunis in 1992, and a PhD degree in computer science on automatic proofs by induction in associative commutative and observational theories from Henri Poincaré University at Nancy. Currently, she is a lecturer at the Institut National des Sciences Appliquées et de Technologie at Tunis. Her research interests include theorem proving techniques and formal methods for specification and verification.

Riadh Robbana received an engineering degree in computer science from the Faculty of Sciences of Tunis in 1991, and a PhD degree in computer science on hybrid systems verification from Joseph Fourier University at Grenoble. Currently, he is a lecturer and head of the Department of Applied Mathematics and Computer Science at Ecole Polytechnique of Tunisia. His research interests include formal verification of real-time systems.