Decidable Integration Graphs *

Y. Kesten†, A. Pnueli**, J. Sifakis***, S. Yovine***

Abstract. *Integration Graphs* are a computational model developed in the attempt to identify simple Hybrid Systems with decidable analysis problems. We start with the class of *constant slope hybrid systems* (CSHS), in which the right hand side of all differential equations is an integer constant. We refer to continuous variables whose right hand side constants are always 1 as *timers*. All other continuous variables are called *integrators*. The first result shown in the paper is that simple questions such as reachability of a given state are undecidable for even this simple class of systems.
To restrict the model even further, we impose the requirement that no test that refers to integrators may appear within a loop in the graph. This restricted class of CSHS is called *integration graphs*. The main results of the paper are that the reachability problem of integration graphs is decidable for two special cases: The case of a single timer and the case of a single test involving integrators.
The expressive power of the integration-graphs formalism is demonstrated by showing that some typical problems studied within the context of the Calculus of Durations and Timed Statecharts can be formulated as reachability problems for restricted integration graphs, and a high fraction of these fall into the subclasses of a single timer or a single test involving integrators.

1 Introduction

Hybrid systems are systems that consist of a mixture of discrete and continuous components. Typically, the continuous components may represent a physical environment which obeys continuous laws of change, while the discrete components may represent a digital controller that senses and manipulates the environment. Characteristic examples are a computer system controlling a robot, a manufacturing plant, or a transport system. Approaches to the specification, description, and analysis of hybrid systems were proposed in [?], [?], and [?].

* This research was supported in part by the France-Israel project for cooperation in Computer Science and by the European Community ESPRIT Basic Research Action Project 6021 (REACT).

† Department of Communication Systems Engineering, Ben-Gurion University, Beer-Sheva, Israel. E-mail: yonit@bgumail.bgu.ac.il

** Department of Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: amir@wisdom.weimann.ac.il

*** VERIMAG, Centre Equation, 2 Ave de Vignate, 38610 Gières, France, E-mail:(Joseph.Sifakis, Sergio.Yovine)@imag.fr

An important question for the analysis and design of hybrid systems is identification of subclasses of such systems and corresponding restricted classes of analysis problems that can be settled algorithmically. In view of the success of model checking of finite-state systems and similar algorithmic approaches to the algorithmic analysis of reactive systems ([?], [?]) and timed systems ([?], [?]), [?], [?]), it is only natural to search for similar decidable analysis problems for hybrid systems. This is the general aim of this paper.

The main results of this search is the identification of *integration graphs*, a class of hybrid systems that seem to avoid the main obstacles to decidability. The main source for undecidability (as will be shown in Section 2) is the presence of *integrators* which are variables whose constant rate of change is not uniformly 1 throughout the system. Integration graphs restrict the potential complexity of constant-slope hybrid systems by disallowing tests involving integrators within loops in the system. Within the class of integration graphs, we give algorithmic solutions to the reachability problem of three important cases:

- Integer computations of an arbitrary integration graph.
- Integration graph with a single timer, and
- Integration graph with a single integrator test along each computation.

Section 2 introduces the notion of *constant slope hybrid systems* (CSHS), which are hybrid systems all of whose differential equations have the form

$$\dot{x} = c,$$

for some integer constant c. Another restriction is that all guard (enabling) conditions of transitions are boolean combinations of linear inequalities with integer coefficients.

We give an example of a CSHS representing the Gas Burner Problem [?], and explain the need for restrictions on tests applied to integrators. In subsection 2.1 we prove that, without these restrictions, the reachability problem becomes undecidable.

Section 3 introduces *integration graphs* which are CSHS's in which integrators are not tested within loops. The section also introduces *finitary timed automata* (FTA), which are a slightly restricted class of timed automata [?], [?], and the notion of *duration formulas* over such automata. It is shown that the reachability problem for integration graphs can be reduced to checking whether a duration formula is satisfied by a computation of an FTA.

Section 4 shows how to solve the duration satisfiability problem for the integer computations of an FTA. The solution is based on constructing a set of equations that characterizes the length of time a computation spends in each automaton state.

Section 5 considers satisfiability of a duration formula by real computations of an FTA with a single timer. It provides an algorithm for solving this problem based on a similar set of characterizing equations.

Section 6 considers satisfiability of a disjunctive duration formula by real computations of an unrestricted FTA.

2 Constant Slope Hybrid Systems

In this section we introduce the class of constant slope hybrid systems. Many hybrid systems analyzed in the literature fall into this class, which is a restricted version of the *linear hybrid automata* discussed in [**?**]. One of the advantages of this class is that the differential equations appearing in states can be trivially solved in closed form and yield solutions that are linear functions of time.

Let $\mathcal{P}$ be a finite set of propositions. Let N denote the natural numbers, R^+ – the non negative reals, and Z – the integers.

A *constant slope hybrid system* (CSHS) consists of the following components:

- S – A finite set of *locations*. In a graphical representation of the system, these are drawn as nodes of the graph.
- Λ – A proposition labeling function $\Lambda : S \mapsto 2^{\mathcal{P}}$, mapping each location $s \in S$ to the set of propositions that are true in s. For a state s and a boolean formula p over Λ, we write $s \models p$ to denote that p evaluates to *true* over $\Lambda(s)$.
- $V = \{x_1, \ldots, x_n\}$ – A finite set of (*data*) *variables*. These are the variables that change continuously within states and discretely via transitions.
- $\mathcal{R}$ – A rate labeling function $\mathcal{R} : S \times V \mapsto \mathsf{Z}$, identifying for each location $s \in S$ and each variable $x \in V$ an integer $c = \mathcal{R}[s][x] \in \mathsf{Z}$ which specifies the (constant) rate at which x changes continuously while being in location s. Thus, the differential equation for x within s is

  $$\dot{x} = c,$$

  where $c = \mathcal{R}[s][x]$.
- $S_I \subseteq S$ – A set of *initial locations*.
- $S_f \subseteq S$ – A set of *final locations*.
- V_0 – Initial values. This is a tuple of values, representing the values of the variables V at the beginning of a computation. By default, a variable that is not assigned an explicit initial value has 0 as an initial value.
- E – A set of *edges*. Each edge $e \in E$ is associated with the following components:
  - A *source* location. This is the location from which the edge departs.
  - A *target* location. This is the location to which the edge connects.
  - An edge *guard* $\star$. This is the condition under which the edge may be traversed. An edge guard is a conjunction of linear equalities/inequalities of the form

    $$\sum_{i=1}^{n} a_i \cdot x_i \sim c,$$

    where $a_i, c \in \mathsf{Z}$, $x_i \in V$ and $\sim$ is one of the comparison relations $\{<, >, =, \neq, \leq, \geq\}$.
  - A multiple *reset* of the form

    $$(y_1, \ldots, y_m) := (0, \ldots, 0), \qquad y_i \in V$$

represented generically as $Y := 0$. When edge e is taken, the variables $y_1, \ldots, y_m$ are reset to 0.

In the graphical representation of CSHS, we represent edge e by drawing an edge from the source location to the target location and label it by the *edge label*

$$\star \ / \ Y.$$

In the case that Y, the set of assigned variables is empty, we use the simpler labeling $\star$?

It is required that no edge departs from a final location.

In Fig. 1 we present a CSHS for the Cat and Mouse system [**?**], representing the situation of a cat chasing a mouse, where the cat and the mouse run at constant velocities, v_c and v_m respectively, and the cat starts running Δ time units later than the mouse. Variables x_c and x_m measure the respective distances of the cat and the mouse from the wall. Variable y is a timer, used to measure the delay Δ in the start time of the cat. By convention, whenever the rate $\mathcal{R}[s][x]$ assigned to variable $x \in V$ within location s equals 0, we omit the differential equation $\dot{x} = 0$ from the label of location s. Thus, the equation $\dot{x}_c = 0$ is implicitly specified for locations *CwMr* and *CwMs* in Fig 1.

Behaviors and Computations

A behavior of a CSHS starts at the initial location $s \in S_I$ with all variables initialized to their initial values. As time progresses, the values of all variables increase according to the rates associated with the current location. At any point in time, the system can change location through an edge connecting location s to s' and labeled by $\star / Y$, provided the current values of the variables satisfy the guard $\star$. With the change of location, all variables in Y are reset to 0, while variables not in V remain unchanged.

A *valuation* ν for V assigns a real value to each variable in V. Let $\mathcal{R}$ be a V-vector of rates (slopes), assigning to each $x \in V$ a real value $\mathcal{R}[x] \in \mathsf{Z}$, denoting the rate of growth of x. For a valuation ν, a rate vector $\mathcal{R}$, and $t \in \mathsf{R}^+$, $\nu + \mathcal{R} \cdot t$ denotes a new valuation ν' such that, for every variable $x \in X$, $\nu'[x] = \nu[x] + \mathcal{R}[x] \cdot t$. For $Y \subseteq X$, we denote by $\nu[Y \leftarrow 0]$ the valuation which resets every $y \in Y$ to 0, and agrees with ν over the rest of the variables.

A triple of the form $\langle s, \nu, t \rangle$, where s is a location, ν is a valuation, and $t \in \mathsf{R}^+$ is called a *situation*.

A *computation segment* of a CSHS is a sequence of situations

$$\langle s_0, \nu_0, t_0 \rangle, \langle s_1, \nu_1, t_1 \rangle, \ldots, \langle s_n, \nu_n, t_n \rangle$$

satisfying:

- [*Consecution*] For every j, $0 \leq j < n$,
  - Either, $s_{j+1} = s_j$ and $\nu_{j+1} = \nu_j + \mathcal{R}[s_j](t_{j+1} - t_j)$.

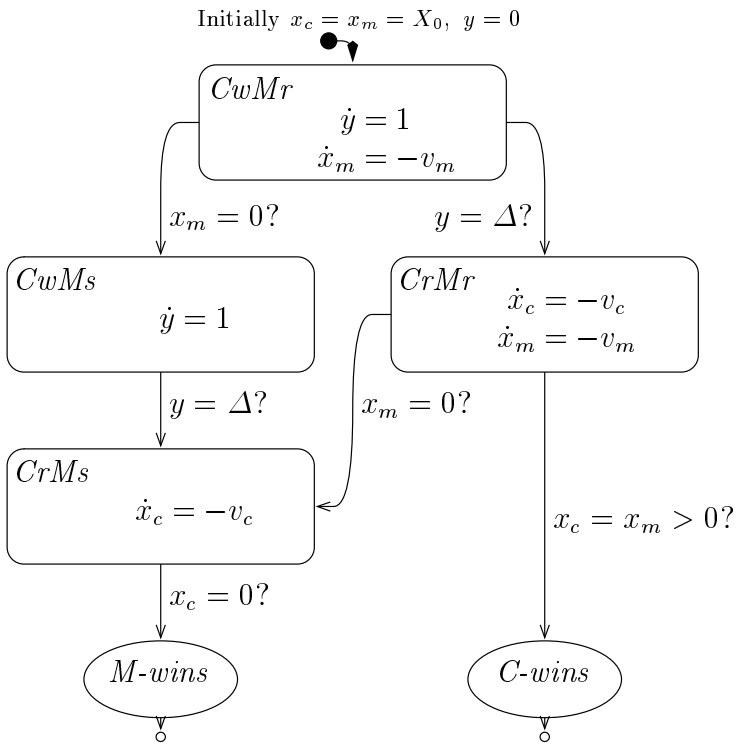


Fig. 1. Cat and Mouse System.

- Or, there is an edge $e \in E$ connecting s_j to s_{j+1} and labeled with $\star\,/Y$ such that $\nu_j + \mathcal{R}[s_j](t_{j+1} - t_j)$ satisfies $\star$ and $\nu_{j+1} = (\nu_j + \mathcal{R}[s_j](t_{j+1} - t_j))[Y \leftarrow 0]$.

- [*Time progress*] For all j, $0 \leq j < n$, $0 \leq t_j \leq t_{j+1}$.

A *computation* of a CSHS is a computation segment satisfying:

- [*Initiation*] $s_0 \in S_I$, $\nu_0 = V_0$, and $t_0 = 0$.
- [*Termination*] $s_n \in S_f$

A computation $\langle s_0, \nu_0, t_0 \rangle, \ldots, \langle s_n, \nu_n, t_n \rangle$ is called an *integer computation* if, for every $i \geq 0$, $t_i \in \mathsf{Z}$ (in fact $t_i \in \mathsf{N}$). For a set of computations $\mathcal{C}$, we denote by $Z(\mathcal{C}) \subseteq \mathcal{C}$ the integer computations in $\mathcal{C}$.

The Reachability Problem

In this paper, we are mainly interested in the reachability problem for constant slope hybrid systems. The reachability problem is:

> Given a final location $s \in S_f$, is there a computation terminating at location s.

In many cases, we are interested in the *non-reachability* or *avoidanve* problem, wishing to ensure that a certain set of states is never visited. An instance of an avoidance problem can be specified on the Cat and Mouse system of Fig. 1 as follows:

Under the assumption

$$\frac{X_0}{v_m} < \Delta + \frac{X_0}{v_c},$$

show that there is no computation that reaches location *C-wins*

As another reachability problem, we consider the Gas Burner system [?]. Consider the CSHS presented in Fig. 2.

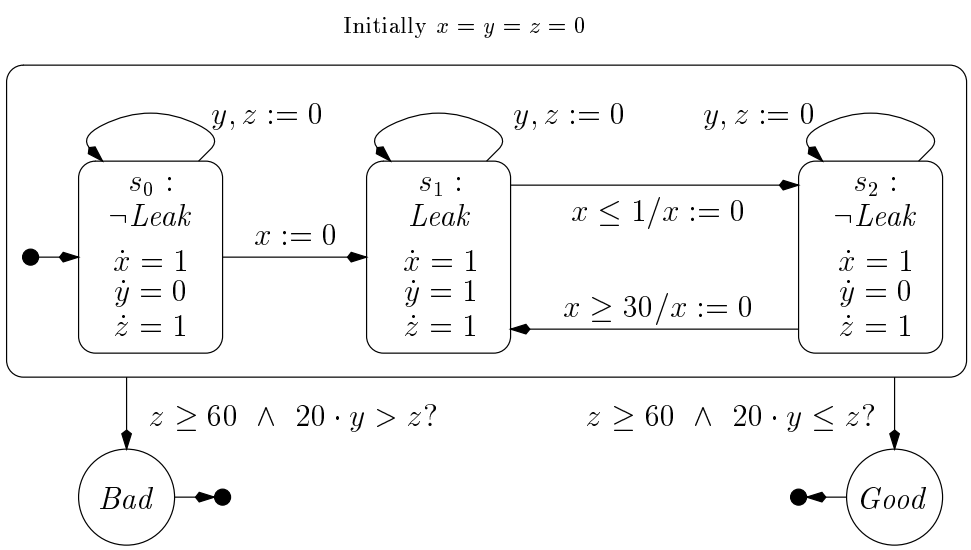


Fig. 2. H-GAS: The gas burner as a hybrid system.

Locations s_0, s_1, and s_2 represent a Gas Burner system that has these three control states. There is a proposition *Leak* which is true only at location s_1, representing a situation at which the system is leaking.

The verification problem posed in [?] can be formulated as follows.
Assuming

1. A continuous leaking period cannot extend beyond 1 time unit.
2. Two disjoint leaking periods are separated by a non-leaking period extending for at least 30 time units.

Prove:

- *Safety-Critical Requirement:* In any interval longer than 60, the *accumulated* leaking time is at most 5% of the interval length.

The CSHS of Fig 2 employs three variables as follows:

- Variable x measures the duration of time in each of the locations s_0, s_1, and s_2. It is reset to 0 on entry to each of these locations.
- Variable y measures the accumulated leaking time. It grows linearly in location s_1, and stays constant in any of the other locations.
- Variable z measures the total elapsed time.

Obviously, system H-GAS ensures assumptions 1 and 2. The only leaking location is s_1 and it is clear that no computation of the system can stay continuously in s_1 for more than 1 time unit and that, between two consecutive (but disjoint) visits to s_1, the system stays at the non-leaking location s_2 for at least 30 time units.

We can view locations s_0, s_1, and s_2, as the *operational* part of the representation. The other locations serve for testing the required property. The system can exit the operational part any time after an interval whose length is at least 60 time units has elapsed. There are two possible exits. If $20 \cdot y > z$, then the accumulated leaking time exceeds 5% of the overall period spent in the operational part. In that case, the system exits to the final location *Bad*, implying a violation of the safety-critical requirement. Otherwise it exits to *Good*, implying that the current run was not found to violate the requirement.

Obviously the safety-critical requirement is valid for this system if and only if location *Bad* is unreachable. This provides another example of an interesting reachability problem: namely, show that no computation of the system of Fig. 2 ever reaches location *Bad*.

2.1 Reachability is Undecidable for CSHS's

In this subsection we show that the reachability problem for CSHS's is undecidable.

The result is based on a reduction of an n-counter machine to a CSHS. The system emulating the n-counter machine only uses guards of the form:

$$u = c \quad \text{or} \quad u \neq c$$

where c is an integer constant .

The Construction

An n-counter machine can be described as a linear labeled program allowing the following basic commands:

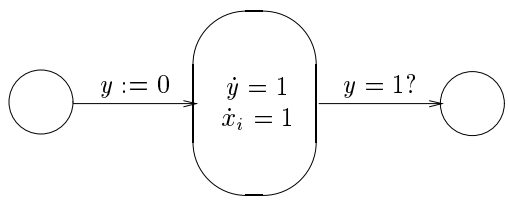
- **go to** ℓ,

- **if** $x_i = 0$ **then go to** ℓ_i **else go to** ℓ_j.
- $x_i := x_i + 1$,
- $x_i := x_i - 1$, (this operation is undefined if x_i=0)
- **stop**

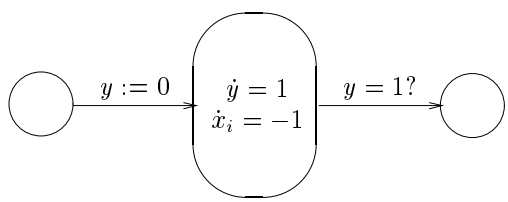
Let P be a program for an n-counter machine with counters $x_1, \ldots, x_n$. Without loss of generality, assume that the first label of P is ℓ_0 and the last command (with label ℓ_t) is a *stop* command . We construct a CSHS S_P which emulates P, i.e., terminates precisely when P does. System S_P uses variables $x_1, \ldots, x_n$ and an additional variable y.

We represent S_P as a graph which has a location (node) for each label of program P. It may have additional locations.

It is not difficult to see how the *go–to* and conditional *go–to* commands can be implemented by edges connecting the corresponding nodes that may be labeled by $x_i = 0$ and $x_i \neq 0$ for the conditional transfer. The commands for incrementing and decrementing a counter x_i can be implemented by the following two subgraphs:



and



According to our conventions, variables not mentioned in the label of a location stay unchanged while executing the location. This is the case with the two extreme locations of the presented subgraphs implementing incrementations and decrementation.

We claim that program P reaches the *stop* command at ℓ_t iff location ℓ_t is reachable in the CSHS S_P.

Conclusion

Since the halting problem for an n-counter machine is undecidable for $n \geq 2$, we conclude that the reachability problem is undecidable for CSHS's of the form considered here. In fact, since our construction uses an additional variable y, it is undecidable for systems having at least 3 variables.

Note that reachability is undecidable even if we restrict ourselves to integer computations.

3 Integration Graphs

Having realized that reachability is undecidable for CSHS's, we attempt to narrow the class of considered systems in the hope that reachability will be decidable for a more restricted class.

3.1 Definition of Integration Graphs

A variable of a CSHS that has the slope +1 at all locations, except perhaps at the final locations, is called a *timer*. All other variables are called *integrators*.

A lower bound of how much we have to restrict the class before reachability becomes decidable is provided by *timed automata* [?]. The differences between a timed automaton and a CSHS can be summarized as:

1. Timed automata do not allow integrators but only timers.
2. The guards allowed by timed automata are conditions of the forms

   $$x \simeq c \quad \text{and} \quad x - y \simeq c$$

   where $\simeq \in \{=, <, >, \leq, \geq\}$, x, y are variables (called *clocks* in [?]), and c is an integer constant.

Motivated by this comparison, consider a test

$$a_1 \cdot x_1 + \cdots + a_n \cdot x_n \sim b, \tag{1}$$

where $x_1, \ldots, x_n$ are variables, $a_1, \ldots, a_n$ and b are integer constants, and $\sim$ is one of $\{=, \neq, <, >, \leq, \geq\}$. Such a test is called *dangerous* if it refers to an integrator or does not have one of the forms listed in item 2 above. Variables that appear in dangerous tests are called *critical variables*.

The implication is that unbridled use of dangerous tests may lead to undecidability. The fact that tests that refer to more than two variables, or contain multiplicative factors with absolute value different from 1, lead to undecidability is proven in [?].

The fact that tests that refer to integrators are dangerous has been established in the undecidability result proven in Section 2. The construction used for the undecidability proof employs integrators to represent the registers x_i and tests them for being zero on edges representing conditional *go-to* commands.

Eliminating dangerous tests altogether is too harsh, since this will exclude systems such as the Cat and Mouse or the Gas Burner from the class we intend to study. For example, the test $20 \cdot y > z$ is dangerous for two reasons. It is not of one of the allowed forms, and it refers to the integrator y.

Instead, we strongly restrict the places where dangerous tests can appear in the graph of a CSHS. An edge in the graph representing a CSHS is called *cyclic* if it is part of a cycle in the graph. A CSHS is called an *integration graph* (IG) if

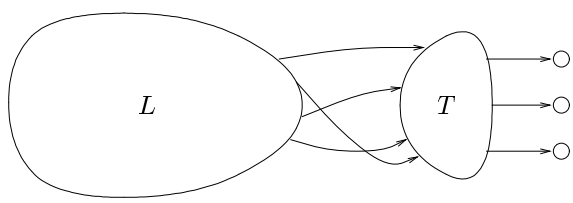
1. Dangerous tests do not appear on cyclic edges.
2. For every critical variable $x \in V$, $\mathcal{R}[s](x) = 0$ for every location s reachable *after* traversing a dangerous test referring to x.

3. The initial values of all timers are 0.

The first restriction ensures that there exists a bound K such that the number of times any computation encounters a dangerous test is bounded by K. In all proofs of undecidability, the constructed counter-examples rely on checking dangerous tests an unbounded number of times. Consequently, there is hope that reachability will be decidable for integration graphs. The second restriction requires that the values of critical variables be frozen immediately after testing them in a dangerous way. This allows referring to their tested values at the end of the computation.

It is not difficult to ascertain that both the Cat and Mouse system (that has no cycles at all) and the Gas Burner system are integration graphs. For the Cat and Mouse system, y can be considered a timer (if we add the equation $\dot{y} = 1$ to states *CrMr* and *CrMs*) while x_c and x_m are integrators. For the Gas Burner system, x and z are timers, while y is an integrator.

It is straightforward to show that any integration graph is equivalent to a system, whose graph can be decomposed into a cyclic (the *looping*) part L with exits into an acyclic (the *testing*) graph T. This decomposition is such that L contains no dangerous tests, both T and transitions from L to T may contain some dangerous tests.



The Cat and Mouse system contains no cycles. Therefore, its decomposition into a looping and a testing part is such that all the system belongs to T while the looping part L is empty. For the Gas Burner system, $T = S_f = \{Bad, Good\}$, while L comprises all other locations.

Without loss of generality, we assume that $S_f \subseteq T$, that is, all final locations belong to T, the testing part of the integration graph.

Partitioned Integration Graphs

An integration graph is said to be *partitioned* if, for every critical variable $x \in V$, the set of non-final locations can be partitioned into the set of potentially *resetting* locations R_x and the set of *non-resetting* locations N_x, such that

- $T \subseteq N_x$
- *No* edge departing from an N_x-location resets variable x.
- *Every* edge connecting an R_x-location to an N_x-location resets variable x.
- *No* edge connects an N_x-location to an R_x-location.

Every integration graph can be transformed into an equivalent partitioned IG at the price of doubling the number of locations for each variable that originally violates the assumed requirements. The two graphs are equivalent in the sense that every final location $s \in S_f$ is reachable in one of them iff it is reachable in the other.

For example, the gas burner system of Fig. 2 violates the requirement of partitionability with respect to the critical variables y and z. To remedy this situation, we present in Fig. 3 a partitioned system which is equivalent to the system of Fig. 2. In this new version, $R_{y,z} = \{s_0, s_1, s_2\}$ while $N_{y,z} = \{s_3, s_4, s_5\}$

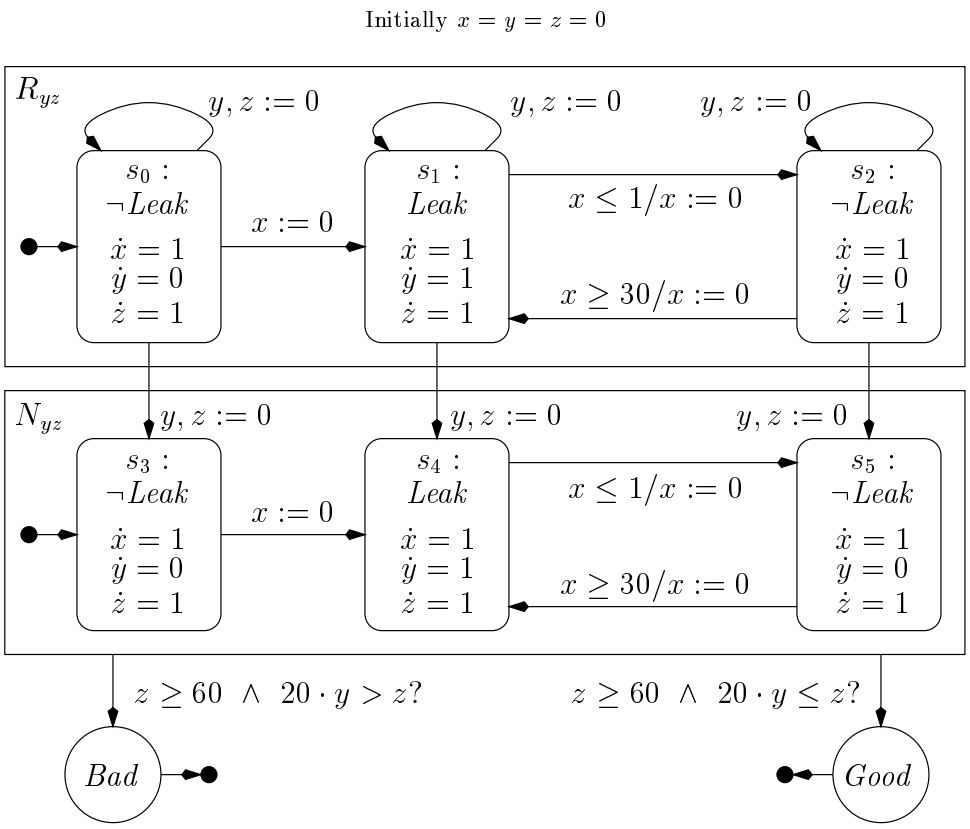


Fig. 3. H-GAS: a partitioned version of the gas burner.

and $S_f = T = \{Bad, Good\}$.

From now on, we assume that the considered integration graph is partitioned.

Single Integrator Tests

In the study of CSHS's, it is possible to restrict our attention to dangerous tests of the form

$$u \sim b$$

where u is an integrator and b is an integer constant. Observe that, for any dangerous test of the more general form

$$a_1 \cdot x_1 + \cdots + a_m \cdot x_m \sim b,$$

we can define a new integrator u whose slope at each location $s \in S$ is given by

$$\mathcal{R}[s][u] = a_1 \cdot \mathcal{R}[s][x_1] + \cdots + a_n \cdot \mathcal{R}[s][x_m]$$

For example, for the gas burner system, we can define a new integrator u whose value is intended to be $20 \cdot y - z$. The slopes of u at locations s_0, s_1 and s_2 are given by -1, 19 and -1, respectively. Then, instead of testing whether $20 \cdot y - z$ is positive, it is sufficient to check for $u > 0$.

3.2 Duration Properties of Finitary Timed Automata

In this subsection we consider the simpler model of timed automata, but ask more complicated questions than just reachability of some final location.

Finitary Timed Automata

We use a simplified version of timed automata ([?], [?]), to which we refer as *finitary timed automata* (FTA). The simplification is that we are interested only in finite computations that reach some final location.

In our framework, an FTA can be presented as a CSHS with the following restrictions:

- All variables have the slope 1 in all states. Consequently, they are all timers, and we can eliminate the rate labeling function $\mathcal{R}$ from the description of a timed automaton.
- The initial values of all variables are 0. Consequently, we need not specify the component V_0 in the description of an FTA.
- All guards are of the forms $l \leq x \leq u$, for a timer $x \in V$ and natural constants $l, u \in \mathsf{N}$.

Following these simplifications, an FTA can be described by the tuple

$$M : \langle S, \Lambda, V, S_I, S_f, E \rangle.$$

We often refer to edges as a tuple $(s, s', \star, Y)$, where s and s' are the source and destination locations, $\star$ is the guard, and Y is the set of variables reset by the edge.

Let $e = \langle s, s', \star, Y \rangle \in E$. We say that e is a *resetting edge* if $Y = V$. For convenience we assume that every initial state $s \in S_I$ is entered by a virtual resetting edge e^-. We denote the set of all virtual resetting edges by E^-.

An FTA may be nondeterministic. It may contain two edges $e_1 = \langle s, s_1, \star_1, Y_1 \rangle$ and $e_2 = \langle s, s_2, \star_2, Y_2 \rangle$, $s_1 \neq s_2$, such that $\star_1$ and $\star_2$ are not mutually exclusive, i.e., $\star_1 \wedge \star_2$ is satisfiable. On the other hand, we require that every two locations, s_1 and s_2, have at most one edge connecting them.

A *trail* is a finite sequence

$$(s_0, t_0), \ldots, (s_n, t_n),$$

such that, for every j, $0 \leq j < n$, $0 \leq t_j \leq t_{j+1}$.

If

$$\gamma = \langle s_0, \nu_0, t_0 \rangle, \langle s_1, \nu_1, t_1 \rangle, \ldots, \langle s_n, \nu_n, t_n \rangle$$

is a computation segment, then

$$\tau = (s_0, t_0), (s_1, t_1), \ldots, (s_n, t_n)$$

is called the trail *corresponding to* γ. A trail that corresponds to some computation segment is called *realizable*. A trail that corresponds to a computation is called *complete*. Obviously, a complete trail is realizable. A trail is called an *integer trail*, if $t_0, t_1, \ldots, t_n$ are all integers. We denote by $\mathcal{T}$ the set of all complete trails of M, and by $Z(\mathcal{T}) \subseteq \mathcal{T}$, the set of complete integer trails of M. We use the shorthand notation $\tau = (\sigma, T)$, where $\sigma = s_0, s_1, \ldots, s_n$ and $T = t_0, t_1, \ldots, t_n$ are the location and time sequence, respectively, associated with the trail τ.

Duration Properties

The questions we intend to pose for finitary timed automata are expressed in a language that includes the propositional calculus augmented with the *duration function* $\int$ and linear inequalities. The version of duration function considered here was inspired by the use of a similar operator in the duration calculus [**?**]. However, the semantics given here to this operator differs from its semantic in the duration calculus.

State formulas are defined in the usual way over the propositions in $\mathcal{P}$ and the boolean operators, and can be evaluated over single locations, using the interpretation assigned to them by the proposition-labeling function Λ.

The duration function $\int$ is a temporal function interpreted over trails. Let φ be a state formula and

$$\tau = (s_0, t_0), \ldots, (s_n, t_n)$$

be a trail. The value of the *duration expression* $\int\varphi$ at position k, $0 \leq k \leq n$ of τ is defined as

$$val(\tau, k, \int\varphi) = \sum_{\substack{0 \leq j < k \\ s_j \models \varphi}} (t_{j+1} - t_j).$$

That is, the value of $\int\varphi$ at position j equals the sum of durations spent in states satisfying φ. *Duration constraints* are inequalities of the form:

$$\sum_{i=1}^{m} a_i \cdot \int\varphi_i \sim c$$

where $\sim \in \{<, >, =, \neq, \leq, \geq\}$, $a_i, c \in \mathsf{Z}$ and φ_i are state formulas.
Duration formulas are boolean combinations of duration constraints.

Let $\tau = (s_0, t_0), \ldots, (s_n, t_n)$ be a trail, and ψ be a duration formula. We say that τ satisfies ψ, denoted $\tau \models \psi$, if ψ evaluates to true when all the duration expressions are evaluated at position n of τ. Let Θ be a set of trails. We say that ψ is *valid* over Θ, if for all $\tau \in \Theta$, $\tau \models \psi$. We say that ψ is *satisfiable* over Θ, if there exists a trail $\tau \in \Theta$ satisfying ψ. Let M be an FTA. We say that ψ is satisfiable (valid) over M if ψ is satisfiable (valid) over $\mathcal{T}$, the set of complete trails of M. Obviously, ψ is valid over M iff $\neg\psi$ is not satisfiable over M.

A duration property is called *conjunctive* if it is a conjunction of duration constraints. Similarly, a duration property is *disjunctive* if it is a disjunction of duration constraints. We use the notations

$$\bigvee\bigwedge\Big(\sum_{i=1}^{m} a_i \cdot \int\varphi_i \sim c\Big), \quad \bigwedge\Big(\sum_{i=1}^{m} a_i \cdot \int\varphi_i \sim c\Big), \quad \text{and} \quad \bigvee\Big(\sum_{i=1}^{m} a_i \cdot \int\varphi_i \sim c\Big)$$

to denote a (general) duration property, a conjunctive duration property, and a disjunctive duration property, respectively.

Decision Problems

Given an FTA M and a duration property ψ, we may ask the following questions:

Problem 1 Validity. Is ψ valid over M?

and

Problem 2 Satisfiability. Is ψ satisfied by some computation of M?

As indicated above, an algorithm for solving one of these problems can be used to solve the other. We will therefore concentrate on finding solutions to the satisfiability problem.

3.3 Reduction of IG-Reachability to FTA-Satisfiability

The reachability problem for integration graphs can be reduced to the satisfiability problem of duration formulas over FTA's.

Let $\mathcal{I}$ be a given integration graph and $\hat{s} \in S_f$ one of its final locations. We are interested in the question whether $\hat{s}$ is reachable by some computation of $\mathcal{I}$.

According to the definition of integration graphs, $\mathcal{I}$ can be decomposed into the looping part L and the acyclic testing part T. Without loss of generality, we can assume that $\hat{s} \in T$.

As a first step, we construct an FTA $M_{\mathcal{I}}$ that represents the behavior of $\mathcal{I}$ ignoring all integrators and dangerous tests. Automaton $M_{\mathcal{I}}$ is obtained from $\mathcal{I}$ by the following transformation:

1. Delete from the integration graph $\mathcal{I}$ all locations and edges that cannot participate in a path from some $s \in S_I$ to $\hat{s}$.
2. Replace all dangerous guards on the remaining edges by the trivial guard T (true).
3. Retain $\hat{s}$ as the only final location.

It is not difficult to see that $M_{\mathcal{I}}$ is an FTA with a single final location $\hat{s}$.

Next, we construct a duration formula ψ that expresses the condition for IG $\mathcal{I}$ to be able to reach location $\hat{s}$. Our first task is to express the values of the critical variables at the end of a computation in terms of duration expressions.

Let $x \in V$ be a critical variable of $\mathcal{I}$. Let x_0 be the initial value specified for x in V_0 and $s_1, \ldots, s_m \in N_x$ be all the non-resetting locations of $\mathcal{I}$. Let $\mathcal{R}_1, \ldots, \mathcal{R}_m$ be the rates of growth of x in $s_1, \ldots, s_m$, respectively. We assume that each s_j has a proposition (as part of $\mathcal{P}$) that uniquely characterize it, i.e., is true at s_j and at no other location. We denote this proposition by at_s_j.

The value of x at the end of a computation can be expressed by the duration expression

$$x_s = x_0 + \mathcal{R}_1 \cdot \int at_s_1 + \cdots + \mathcal{R}_m \cdot \int at_s_m$$

This is based on the observation that any unit of time spent at location s_i contributes $\mathcal{R}_i$ to the final value of x.

Since T is acyclic, there are only finitely many paths $\pi_1, \ldots, \pi_k$ that a computation can follow within T until it reaches location $\hat{s}$. For each $i = 1, \ldots, k$, let the formula ψ_i be the conjunction of all the dangerous tests that appear in $\mathcal{I}$ on edges of π_i, replacing any occurrence of a critical variable x by the expression x_f as defined above.

Finally, we let ψ be the disjunction $\psi_1 \vee \cdots \vee \psi_k$.

Claim 1 (Reduction) *Location $\hat{s}$ is reachable by a computation of $\mathcal{I}$ iff there exists a computation of $M_{\mathcal{I}}$ satisfying ψ.*

Examples of Reduction: Cat and Mouse

Consider applying the described reduction to the Cat and Mouse system. The decomposition of this system into L and T identifies the entire system (being acyclic) as T.

First, consider reachability of location *C-wins*. The diagram contains only one path leading to this location while traversing the two dangerous tests $x_c = x_m$ and $x_m > 0$ (represented in the diagram by the abbreviation $x_c = x_m > 0$). In Fig. 4, we present the FTA obtained by the reduction. The duration formula whose satisfiability should be checked is

$$\psi_C : \left(X_0 - v_c \cdot \int \overline{CrMr} = X_0 - v_m \cdot (\int \overline{CwMr} + \int \overline{CrMr})\right) \wedge \left(X_0 - v_c \cdot \int \overline{CrMr} > 0\right),$$

where we use the shorthand notation $\overline{CrMr}$ and $\overline{CwMr}$ for at_CrMr and at_CwMr respectively.

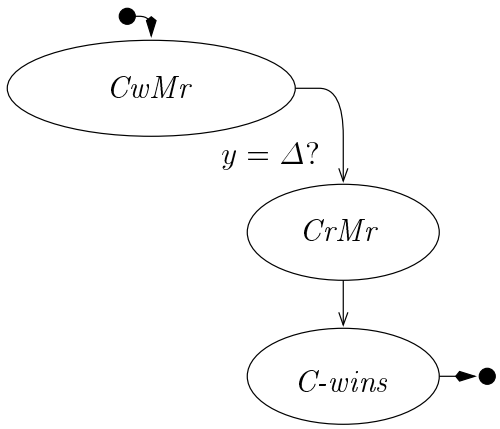


Fig. 4. Reduction of Cat and Mouse: Reachability of *C-wins*.

Next, we consider reachability of location *M-wins*. In Fig. 5, we present the automaton obtained by reduction of the Cat and Mouse system where *M-wins* is taken to be the final state.

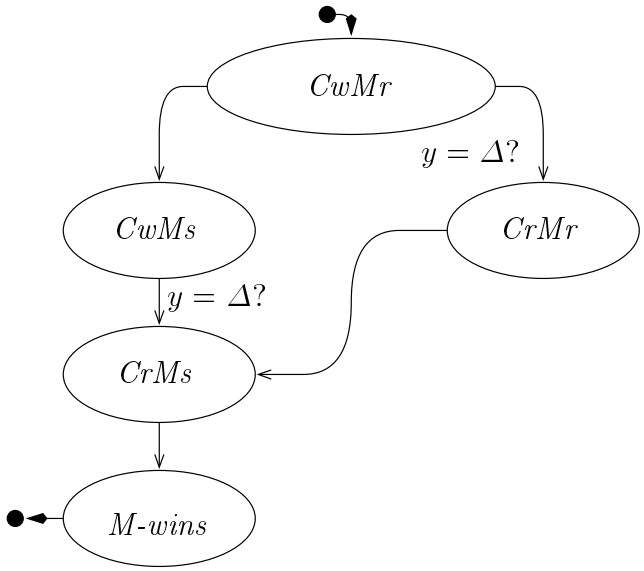


Fig. 5. Reduction of Cat and Mouse: Reachability of *M-wins*.

There are two paths leading to location *M-wins*. While both traverse the same dangerous tests, $x_m = 0$ and $x_c = 0$, they differ in the

However, the conjunction of dangerous tests along these paths yields the same formula $x_m = 0 \wedge x_c = 0$. Consequently, we take

$$\psi_M : \Big(X_0 - v_c \cdot (\int \overline{CrMr} + \int \overline{CrMs}) = 0\Big) \wedge \Big(X_0 - v_m \cdot (\int \overline{CwMr} + \int \overline{CrMr}) = 0\Big)$$

Examples of Reduction: Gas Burner

Next, we consider the example of the Gas Burner. In Fig. 6, we present the FTA obtained by applying the FTA-reduction to the partitionable version of the Gas Burner, as presented in Fig. 3. The corresponding accessibility formula for

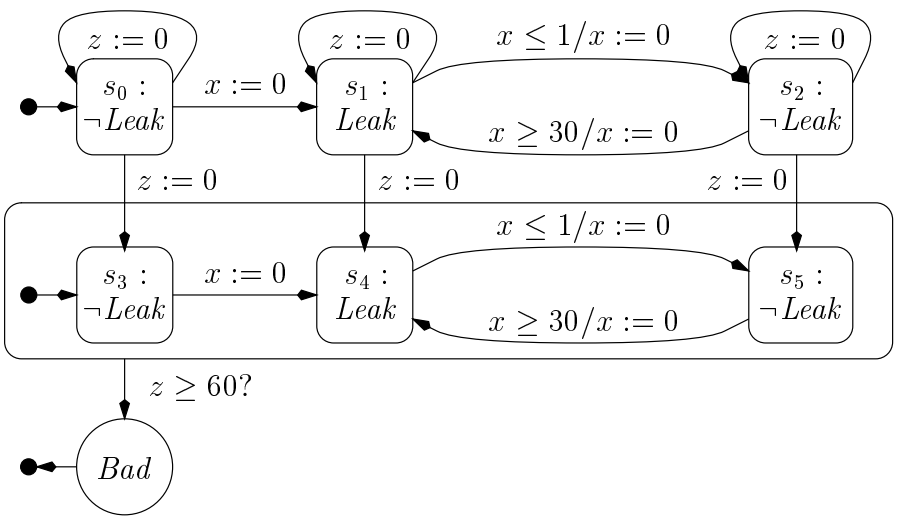


Fig. 6. Reduction of the Gas Burner.

reaching location *Bad* is given by:

$$\psi_{Bad} : 20 \cdot \int at_s_4 \ > \ \int at_s_3 + \int at_s_4 + \int at_s_5.$$

Note that the accumulated durations of the critical variables y and z are taken only over the non-resetting locations s_3, s_4 and s_5 for these variables.

Conclusions

According to Claim 1, in order to find whether location $\hat{s}$ is reachable in the integration graph $\mathcal{I}$, it is sufficient to check that the duration formula ψ is satisfiable over the FTA $M_{\mathcal{I}}$. Consequently, we will concentrate on methods for solving the satisfiability problem of duration formulas over FTA's.

4 Duration Properties over Integer Computations

The set of computations of an FTA consists of both integer and real computations. In the following, we present an algorithm for checking the satisfiability of duration properties over the integer subset of an FTA's computations.

Given an FTA $\hat{M}$, we first discuss the construction of a *0/1-unwinding* FTA M, whose set of computations is exactly the set of integer computations of $\hat{M}$. The *0/1-unwinding* M is an FTA with restricted behaviors such that its set of computations can be analyzed using simple network flow techniques.

Next, we present an algorithm for the satisfiability of a (general) duration property over M.

Note that this solves the problem of reachability of integration graphs by integer computations.

4.1 The Integer Computations of an FTA

We say that the 3-segment

$$\langle s_j, \nu_j, t_j \rangle, \langle s_{j+1}, \nu_{j+1}, t_{j+1} \rangle, \langle s_{j+2}, \nu_{j+2}, t_{j+2} \rangle$$

is *stutteringly equivalent* to the 2-segment

$$\langle s_j, \nu_j, t_j \rangle, \langle s_{j+2}, \nu_{j+2}, t_{j+2} \rangle,$$

if $s_j = s_{j+1}$, $t_j \leq t_{j+1} \leq t_{j+2}$, and $\nu_{j+1} = \nu_j + (t_{j+1} - t_j)$.

A computation segment γ' is said to be a *stuttering variant* of the segment γ if γ' can be obtained from γ by a finite sequence of steps, each replacing a 3-segment [2-segment] by a stuttering equivalent 2-segment [3-segment]. For a set of computation segments G, we denote by $stutt(G)$ the set of all stuttering variants of computation segments in G.

It is not difficult to see that the set of computations of an FTA is closed under stuttering equivalence.

Let $M = \langle S, \Lambda, V, S_I, S_f, E \rangle$ and $\hat{M} = \langle \hat{S}, \hat{\Lambda}, \hat{V}, \hat{S}_I, \hat{S}_f, \hat{E} \rangle$ be FTA's and $\Phi : S \to \hat{S}$ be a function mapping locations of M to locations of $\hat{M}$. We say that Φ is a *homomorphism*, if

- $\forall s \in S, \Lambda(s) = \hat{\Lambda}(\Phi(s))$
- $\Phi : S_I \to \hat{S}_I$ and $\Phi : S_f \to \hat{S}_f$.
- $V \supseteq \hat{V}$
- For every $e = \langle s_1, s_2, \star, Y \rangle \in E$,
  - Either, there exists an edge $\hat{e} = \langle \Phi(s_1), \Phi(s_2), \hat{\star}, \hat{Y} \rangle \in \hat{E}$ such that
    $\hat{Y} = Y \cap \hat{V}$
    $\star \to \hat{\star}$, i.e., $\star$ implies $\hat{\star}$
    In that case we write $\hat{e} = \Phi(e)$.
  - Or, $\Phi(s_1) = \Phi(s_2)$, $\star|_{\hat{V}} = \mathrm{T}$ and $\hat{V} = Y \cap \hat{V} = \phi$, where $\star|_{\hat{V}}$ is the restriction of $\star$ to constraints over $\hat{V} \subseteq V$. We can view this case as though e is mapped by Φ to a self-edge connecting $\Phi(s_1)$ to itself.

For a valuation $\nu : V \to \mathsf{R}$, we denote by $\nu|_{\hat{V}}$ its restriction to $\hat{V} \subseteq V$.

For an M-computation

$$\gamma = \langle s_0, \nu_0, t_0\rangle, \langle s_1, \nu_1, t_1\rangle, \ldots, \langle s_n, \nu_n, t_n\rangle,$$

we denote by

$$\Phi(\gamma) = \langle \Phi(s_0), \nu_0|_{\hat{V}}, t_0\rangle, \langle \Phi(s_1), \nu_1|_{\hat{V}}, t_1\rangle, \ldots, \langle \Phi(s_n), \nu_n|_{\hat{V}}, t_n\rangle$$

the homomorphic image of γ under the homomorphism Φ.

Similarly, we write $\Phi(\mathcal{C}_M)$ to denote the set of all homomorphic images of the computations of M.

The following is an immediate result of the above definition:

Claim 2 *Let $M = \langle S, \Lambda, V, S_I, S_f, E\rangle$ and $\hat{M} = \langle \hat{S}, \hat{\Lambda}, \hat{V}, \hat{S}_I, \hat{S}_f, \hat{E}\rangle$ be* FTA *'s, and $\Phi : S \to \hat{S}$ be a homomorphism. Let $\mathcal{C}$ and $\hat{\mathcal{C}}$ denote the sets of computations of M and $\hat{M}$, respectively. Then*

$$\Phi(\mathcal{C}) \subseteq \hat{\mathcal{C}}.$$

Let $\pi : e_0, s_0, \ldots, e_n, s_n$ be a path in M. We say that π is a *complete* path if $e_0 \in E^-$ and $s_n \in S_f$.

4.2 0/1-Unwinding

Given an FTA $\hat{M} = \langle \hat{S}, \hat{\Lambda}, \hat{V}, \hat{S}_I, \hat{S}_f, \hat{E}\rangle$, we say that an FTA $M = \langle S, \Lambda, V, S_I, S_f, E\rangle$ is a *0/1-unwinding* of $\hat{M}$ if

- There exists a surjective homomorphism $\Phi : S \to \hat{S}$.
- Every complete path π in M has a realizable trail tracing π.
- Every location $s \in S$ is associated with a fixed visit length $v_s \in \{0, 1\}$, such that each visit to s in a realizable trail lasts precisely v_s time units. That is, if

  $$\ldots, (s, t_s), (s', t_{s'}), \ldots \in \mathcal{T}$$

  then $t_{s'} = t_s + v_s$.
- $stutt(\Phi(\mathcal{C})) = Z(\hat{\mathcal{C}})$

Claim 3 *For every* FTA*, there exists a 0/1-unwinding.*

Proof: Let $\hat{M} = \langle \hat{S}, \hat{\Lambda}, \hat{V}, \hat{S}_I, \hat{S}_f, \hat{E}\rangle$ be an FTA. We have to prove that there exists an FTA $M = \langle S, \Lambda, V, S_I, S_f, E\rangle$ such that M is a 0/1-unwinding of $\hat{M}$. We will do so by definig M.

For every variable $x \in \hat{V}$, let c_x be the smallest integer such that $c \leq x$ or $x \leq c$ is a subformula on a guard in $\hat{\star}$. Let $y \notin \hat{V}$ be a new variable. We define $V = \hat{V} \cup \{y\}$, with $c_y = 1$

We define a set of *location annotations* as follows. For every variable $x \in V$, let i be an integer in the range $\{0..c_x\}$.

- *Equality annotations* - For every variable $x \in V$, add either $x = i$ or $x > c_x$ to the annotation. We denote the set of equality annotations by $\Omega_=$.
  For $\alpha \in \Omega_=$, we denote by $\alpha[Y \leftarrow 0]$ the annotation which replaces $(x = i) \in \alpha$ by $x = 0$ for every variable $x \in Y$, and agrees with α on all other variables.
- *Inequality annotations* - For every variable $x \in V$, add either $x \in [i..i+1]$ or $x > c_x$ to the annotation. We denote the set of inequality annotations by $\Omega_{\leq\geq}$.

We denote by $\Omega = \Omega_= \cup \Omega_{\leq\geq}$ the set of all annotations. We can now define M as follows:

- $S = \{\langle \hat{s}, \alpha \rangle \mid \hat{s} \in \hat{S}, \alpha \in \Omega\}$.
  We say that $s = \langle \hat{s}, \alpha \rangle$ is an *equality (inequality) location* if $\alpha \in \Omega_=$ ($\alpha \in \Omega_{\leq\geq}$).
  For $s = \langle \hat{s}, \alpha \rangle \in S$, we say that $s' = \langle \hat{s}', \alpha' \rangle \in S$ is the *successor* of s ($s' = successor(s)$) if
  - Either, s is an equality location, and the following conditons hold:
    - For every $x \in V$ s.t. $(x = i) \in \alpha$, then if $i \leq c_x$ then $(x \in [i..i+1]) \in \alpha'$, else $x > c_x \in \alpha'$.
    - For every $x \in V$ s.t. $x < c_x \in \alpha$ then $x < c_x \in \alpha'$.
  - Or, s is an inequality location, and the following conditons hold:
    - For every $x \in V$ s.t. $(x \in [i..i+1]) \in \alpha$ then $(x = i+1) \in \alpha'$.
    - For every $x \in V$ s.t. $x < c_x \in \alpha$ then $x < c_x \in \alpha'$.
- Λ - For every $s = \langle \hat{s}, \alpha \rangle \in S$, $\Lambda(s) = \Lambda(\hat{s})$.
- $S_I = \{\langle \hat{s}, \alpha_0 \rangle \mid \hat{s} \in \hat{S}_I\}$, where α_0 is the equality annotation with $x = 0$ for every $x \in V$.
- $S_f = \{\langle \hat{s}, \alpha_0 \rangle \mid \hat{s} \in \hat{S}_f\}$.
- E -
  - For every edge $\hat{e} = \langle \hat{s}, \hat{s}', \hat{\star}, \hat{Y} \rangle \in \hat{E}$ and equality location $\langle \hat{s}, \alpha \rangle \in S$ such that α satisfies $\hat{\star}$, $e = \langle \langle \hat{s}, \alpha \rangle, \langle \hat{s}', \alpha[Y \leftarrow 0] \rangle, \hat{\star} \wedge (y = 0)?, \hat{Y} \rangle \in E$.
  - For every equality location $s \in S$, $e = \langle s, successor(s), y = 0?, \phi \rangle \in E$.
  - For every inequality location $s \in S$, $e = \langle s, successor(s), y = 1?, y \rangle \in E$.

  Note that the guard on every edge in E constrains the variable y, to either 0 or 1. These constraints enforce a 0/1-visit duration in every equality/inequality location of M.

It is straightforward to show that M as defined above, is a 0/1-unwinding of $\hat{M}$.

In figure 7 we present an FTA $\hat{M}$ with a single variable x, and two transitions. The variable is reset at the first transition and tested on the second transition. In figure 8 we present an FTA M, the 0/1-unwinding of $\hat{M}$, where

$$\Phi(s_i) = \ldots = \Phi(s_o) = s_1, \ \Phi(s_I) = s_I, \text{ and } \Phi(s_f) = s_f$$

The construction of the 0/1-unwinding of an FTA resembles the construction of the region graph for an FTA ([?], [?]), and has a similar complexity. Namely, the 0/1-unwinding of an FTA with k locations and clocks $x_1, \ldots, x_n$ with respective ranges $[0..c_1], \ldots, [0..c_n]$, can contain up to $2k \cdot (c_1 + 2) \cdots (c_n + 2)$ locations.

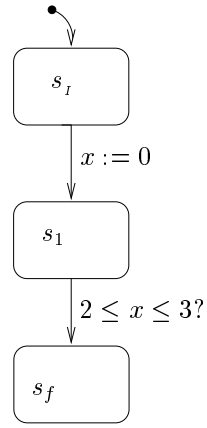


Fig. 7. $\hat{M}$: An FTA

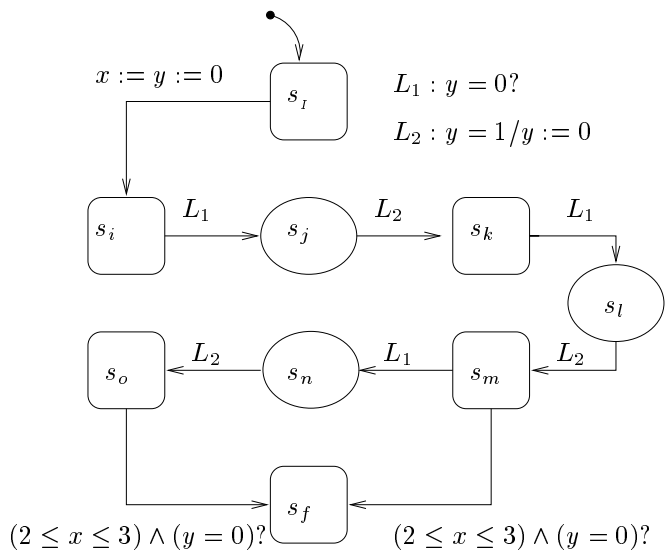


Fig. 8. M : 0/1 - unwinding of $\hat{M}$

4.3 The Verification Algorithm

Let $M = \langle S, \Lambda, V, S_I, S_f, E \rangle$ be an FTA. For $e : \langle s, s', \star, Y \rangle \in E$, we say that e *departs* from s and *arrives* at s'. For $s \in S$ and $e \in E$ departing from s, we define

- $succ(s)$ — the set of edges departing from s.
- $pred(s)$ — the set of edges arriving at s.
- $pred(e) = pred(s)$

Let $\tau = (s_0, t_0), \ldots, (s_n, t_n)$ be a trail. For every $s \in S$ and $e \in E$, we define

- n_s – the number of occurrences of location s in τ.
- m_e – the number of times e was taken in τ.
- r_e – the sequence number of e in the list of edges visited by τ, sorted according to the order of their first visit. That is, $r_e = k$ if e is the k'th edge visited by τ.

Let $\hat{M} = \langle \hat{S}, \hat{\Lambda}, \hat{V}, \hat{S}_I, \hat{S}_f, \hat{E} \rangle$ be an FTA and $M = \langle S, \Lambda, V, S_I, S_f, E \rangle$ be a 0/1-unwinding of $\hat{M}$. Let

$$\psi = \bigvee \bigwedge \Big(\sum_{i=1}^{m} a_i \cdot \int \varphi_i \sim c \Big)$$

be a duration formula. The set of constraints $C(M, \psi)$ characterizing the attributes of a complete trail satisfying ψ, is summarized in figure 9.

$$C(M,\psi): \begin{cases}
\bullet\ \textbf{Flow:}\ \text{For every } s \in S - S_f \\
\qquad n_s = \sum_{e_j \in pred(s)} m_j = \sum_{e_i \in succ(s)} m_i . \\
\bullet\ \textbf{Initiation and Termination:} \\
\qquad \sum_{e \in E^-} m_e = 1 \\
\qquad \sum_{s \in S_f} n_s = \sum_{s \in S_f} \sum_{e_j \in pred(s)} m_j = 1 \\
\bullet\ \textbf{Accessibility:} \\
\quad \text{For every edge } e \in E^- \\
\qquad (m_e = r_e = 1) \vee (m_e = r_e = 0) \\
\quad \text{For every edge } e \in E - E^- \\
\qquad \Big(m_e = r_e = 0 \Big) \vee \Big(m_e > 0 \wedge \bigvee_{e' \in pred(e)} (0 < r_{e'} < r_e) \Big) \\
\bullet\ \textbf{Visit Durations:}\ \text{For every } s \in S \\
\qquad \Delta(s) = n_s \cdot v_s \\
\bullet\ \textbf{The Duration Property:} \\
\qquad \bigvee \bigwedge \Big(\sum_{i=1}^{m} a_i \cdot \sum_{\substack{s \in S \\ s \models \varphi_i}} \Delta(s) \sim c \Big)
\end{cases}$$

Fig. 9. Attributes of Complete Trails Satisfying ψ

Proposition 3. *The set of constraints $C(M,\psi)$ has a solution iff ψ is satisfiable over M iff ψ is satisfiable by an integer computation of $\hat{M}$.*

Proof: It is straightforward to show that the values of n_s, m_e and r_e corresponding to a computation of M which satisfies ψ, satisfy the constraints $C(M,\psi)$.

In the other direction, let $\mathcal{E} = \{n_s, m_e, r_e \mid e \in E, s \in S\}$ be a set of values satisfying the constraints $C(M,\psi)$. We have to show that there exists a complete path π in M satisfying ψ. We will do so by constructing π such that for every $e \in E$ the number of occurrences of e in π is exactly m_e. By the *flow* constraint and the *initiation and termination* constraint this ensures that for every $s \in S$, the number of occurrences of $s \in \pi$ is exactly n_s. From the *duration property* and *visit duration* constraints, any such path satisfies ψ. The *accessibility* constraints, together with the edge ordering r_e, ensures that cyclic paths in M which are unreachable from an initial location, do not contribute to the characterizing attributes.

We construct π incrementally, proceeding through a sequence of paths

$$\Pi : \pi_1, \pi_2, \ldots,$$

where, for each $j = 1, \ldots,$ π_{j+1} is a suffix of π_j. Let

$m_e(\pi_j)$: number of occurences of e in π_j
$\widetilde{m}_e(\pi_j) : m_e - m_e(\pi_j)$

for every $j = 1, \ldots$ and $e \in E$.

The sequence of paths Π is constructed inductively as follows. We start with $\pi_1 : e_f$, where e_f is the only predecessor edge to a final state, with $m_e \neq 0$. For simplicity, we only specify the list of edges followed by each path. The visited states can be inferred. For every $i = 1, \ldots,$ such that $\pi_i : e_1^i, e_2^i, \ldots$ we construct $\pi^{i+1} : e, e_1^i, e_2^i, \ldots$ where $e \in E$ is an edge with maximal r_e satisfying

- e is a predecessor of e_1^i in M.
- $\widetilde{m}_e(\pi_i) > 0$

Let $\pi_j = e_1^j, e_2^j, \ldots \in \Pi$. For $e_1^j \notin E^-$, let $s^j \in S$ denote the location from which e_1^j departs. The following *induction hypothesis* can be stated over the sequence Π:

For every $s \in S - (S_f \cup \{s^j\})$,

$$\sum_{e_i \in pred(s)} \widetilde{m}_i(\pi_j) = \sum_{e_i \in succ(s)} \widetilde{m}_i(\pi_j)$$

and

$$\sum_{e_i \in pred(s^j)} \widetilde{m}_i(\pi_j) = 1 + \sum_{e_i \in succ(s^j)} \widetilde{m}_i(\pi_j)$$

From the *induction hypothesis* and the *flow* constraint, for every $\pi_k : e_1^k, e_2^k, \ldots \in \Pi$, such that $e_1^k \notin E^-$, there exists and edge $e' \in E$, such that e' is a predecessor of e_1^k in M and $\widetilde{m}_{e'}(\pi_k) > 0$. Namely, for every $\pi_k : e_1^k, e_2^k, \ldots \in \Pi$, such that $e_1^k \notin E^-$, we can construct $\pi_{k+1} \in \Pi$. Obviously, Π is a finite sequence, therefore, for some $l \geq 1$ and $e^- \in E^-$, $\pi_l : e_1^l = e^-, e_2^l, \ldots, e_f \in \Pi$. Namely, for some $l \geq 1$, π_l is a complete path in M.

It remains to show that for every $e \in E$, $\widetilde{m}_e(\pi_l) = 0$. Assume the contrary, i.e., for some $e \in E$, $\widetilde{m}_e(\pi_l) > 0$. Let $\mathcal{E} = \{e \in E \mid \widetilde{m}_e(\pi_l) > 0\}$. Let e be an edge in $\mathcal{E}$ such that for every $\tilde{e} \in \mathcal{E}$, $r_e \leq r_{\tilde{e}}$, i.e. e has the minimal value of r_e among all edges in $\mathcal{E}$. From the *accessibility* constraint, there exists an edge $e' \in E$ a predecessor of e such that $m_{e'} > 0$ and $r_{e'} < r_e$. Obviously, $e' \notin \mathcal{E}$, thus $e' \in \pi_l$. Let $s \in S$ be the location common to both e and e'. From the *induction hypothesis* and the *flow* constraint, there exists an edge $e'' \in E$ such that e'' is a predecessor of s and $e'' \in \mathcal{E}$. Minimality of r_e within $\mathcal{E}$ implies that $r_{e''} \geq r_e$. The constraints governing the construction of the sequence of paths Π imply that $r_{e'} \geq r_{e''}$. However, the last two inequalities contradict our assumption that $r_{e'} < r_e$. Therefore, our initial assumption that for some $e \in E$, $\widetilde{m}_e(\pi_l) > 0$ must be wrong. ∎

Complexity Considerations

The problem of finding an integer solution to $C(M, \psi)$ for the unknowns

$$n_s \geq 0, \quad m_e \geq 0, \quad r_e \geq 0$$

is a generalization of the classical *integer linear programming* (ILP) problem. The generalization is due to the fact that the components relating to *accessibility* and the *duration property* contain each disjunctions, while the classical ILP problem only allows conjunction of linear constraints.

The ILP problem is shown in [**?**] to be NP complete. The key element in the proof that ILP is in NP is showing that if an ILP has a solution it also has a solution whose bit complexity (i.e., number of bits required for its presentation) is polynomial in the size of the problem (also measured in the number of presentation bits). It is straightforward to generalize this analysis in order to show that the problem of finding an integer solution to $C(M, \psi)$ is also in NP. Note that the size of $C(M, \psi)$ is linear in the sizes of M and ψ. This leads trivially to an upper bound which is exponential in the size of M and may become doubly exponential in the size of $\hat{M}$, the original IG before unwinding.

5 Real Computations: Single Timer

In the previous section, we presented an algorithm for satisfiability of duration properties over the integer computations of an FTA. In this and the following section, we deal with satisfiability of duration properties over the entire set of an FTA's computations, including real computations.

First, we restrict the FTA to a single timer, checking satisfiability of general duration formulas. Note that this solves the problem of reachability of integration graphs with a single timer (and any number of integrators).

Next, in section 6 we give an algorithm for satisfiability over an unrestricted FTA, restricting the verified property to disjunctive durations.

5.1 Characterization of Complete Trails

Let $M = \langle S, \Lambda, V, S_I, S_f, E \rangle$ be an FTA satisfying:

- $V = \{x\}$.
- Every loop in M contains at least one resetting edge.

We refer to M as a *single-timer* FTA.

Let $\pi = e_0, s_0, \ldots, e_n, s_n$ be a path in M. We call π an *rr-path* if e_0 and e_n are resetting edges. An rr-path is called *basic* if for every $i = 1, 2, \ldots n-1$, e_i is not a resetting edge. We denote by Π the (finite) set of all basic rr-paths in M. Since there is only one timer, the guard of each edge e_i has the form $l_i \leq x \leq u_i$.

Let $\pi = e_0, s_0, \ldots, e_n, s_n$ be a path in M. We say that

$$\tau = (s_0, t_0), \ldots, (s_n, t_n)$$

is a *trail tracing* π.
Let s be a location appearing one or more times in τ. We define the *visit duration* of the trail τ at location s to be

$$\Delta(s, \tau) \quad = \sum_{0 \leq i < n,\ s_i = s} (t_{i+1} - t_i)$$

For a location s not visited by τ, we take $\Delta(s, \tau) = 0$. Thus, $\Delta(s, \tau)$ is the time spent by τ at location s accumulated over all visits of τ at s.

When the trail τ is obvious from the context, we write $\Delta(s, \tau)$ simply as $\Delta(s)$. Note that if π is a basic path, τ can visit each $s \in S$ at most once, with the possible exception of $s_0 = s_n$. In this special case, $\Delta(s_0, \tau) = t_1 - t_0$, implying that the duration of the last visit of τ at $s_n = s_0$ is taken as 0.

Claim 4 *Let $\pi = e_0, s_0, \ldots, e_n, s_n \in \Pi$ be a basic rr-path, and $\tau = (s_0, t_0), \ldots, (s_n, t_n)$ be a trail tracing π. Then τ is realizable iff*

$$\bigwedge_{i=1}^{n} \left(l_i \ \leq \ \sum_{j=0}^{i-1} \Delta(s_j) \ \leq \ u_i \right)$$

where, for every $i = 1, 2, \ldots, n$, $l_i \leq x \leq u_i$ is the guard associated with e_i.

Proof: First, assume that τ is realizable. This means that there exists some computation segment

$$\gamma = \langle s_0, \nu_0, t_0 \rangle, \langle s_1, \nu_1, t_1 \rangle, \ldots, \langle s_n, \nu_n, t_n \rangle$$

such that:

- $\nu_0[x] = \nu_n[x] = 0$, and
- for all i, $0 \le i < n-1$, $\nu_{i+1}[x] = \nu_i[x] + t_{i+1} - t_i = t_{i+1} - t_0$.

Clearly, for all i, $1 \le i \le n$, $l_i \le \nu_{i-1}[x] + t_i - t_{i-1} \le u_i$.

Since π is basic, $\Delta(s_j) = t_{j+1} - t_j$ for all j, $0 \le j \le n-1$ and it is easy to show by induction that $\nu_{i-1}[x] + t_i - t_{i-1} = \sum_{j=0}^{i-1} \Delta(s_j)$ for all i, $1 \le i \le n$.

Hence,

$$\bigwedge_{i=1}^{n} \left(l_i \ \le \ \sum_{j=0}^{i-1} \Delta(s_j) \ \le \ u_i \right)$$

In the other direction, let

$$\gamma = \langle s_0, \nu_0, t_0 \rangle, \langle s_1, \nu_1, t_1 \rangle, \ldots, \langle s_n, \nu_n, t_n \rangle$$

such that

- $\nu_0[x] = \nu_n[x] = 0$, and
- for all i, $0 \le i < n-1$, $\nu_{i+1}[x] = \nu_i[x] + \Delta(s_i) = \nu_i[x] + t_{i+1} - t_i$.

Clearly, for all i, $1 \le i \le n$, $\nu_{i-1}[x] + t_i - t_{i-1} = \sum_{j=0}^{i-1} \Delta(s_j)$ and $l_i \le \nu_{i-1}[x] + t_i - t_{i-1} \le u_i$.

Hence, γ is a computation segment and τ is realizable.

∎

Let $\pi = e_0, s_0, \ldots, e_n, s_n$ and $\pi' = e'_0, s'_0, \ldots, e'_k, s'_k$ be two rr-paths such that $e_n = e'_0$. We define the *fusion* of π and π' to be the path

$$\pi \circ \pi' = e_0, s_0, \ldots, e_n, s'_0, \ldots, e'_k, s'_k$$

Let $\tau = (s_0, t_0), \ldots, (s_n, t_n)$ be a realizable trail tracing the rr-path $\lambda = e_0, s_0, \ldots, e_n, s_n$. Obviously, λ can be uniquely presented as a fusion $\pi_1 \circ \cdots \circ \pi_k$ of $k \ge 1$ basic rr-paths.

- For each $\pi \in \Pi$, let $n(\pi, \lambda)$ denote the number of times the basic rr-path π appears in the fusion λ. If π does not appear in λ then $n(\pi, \lambda) = 0$.
- For each location s, let $\Delta^\pi(s, \tau)$ denote the accumulated visit duration of τ in location s when we restrict our attention to visits at s while tracing the basic rr-path π.

If both π_1 and π_2 with $\pi_1 \ne \pi_2$ visit location s and $\lambda = \pi_1 \circ \pi_2$, then $\Delta^{\pi_1}(s, \tau)$ refers to the first visit to s, while $\Delta^{\pi_2}(s, \tau)$ refers to the second visit. If π does not appear in λ then $\Delta^\pi(s, \tau) = 0$ for every $s \in S$.

Thus, the accumulated visit duration of τ in location s is

$$\Delta(s, \tau) = \sum_{\pi \in \Pi} \Delta^\pi(s, \tau)$$

When τ and λ are understood from the context, we write $n(\pi, \lambda)$ and $\Delta^\pi(s, \tau)$ simply as $n(\pi)$ and $\Delta^\pi(s)$. We refer to the sets $\{n(\pi) \mid \pi \in \Pi\}$ and $\{\Delta^\pi(s) \mid \pi \in \Pi, s \in S\}$ as the *attributes* of τ and π. Note that $n(\pi, \lambda)$ is determined by the path λ and is common to all trails tracing λ. On the other hand, $\Delta^\pi(s, \tau)$ may vary among different trails tracing λ.

Claim 5 *Let λ be an rr-path. Then, there exists a realizable trail τ tracing λ with attributes $n(\pi)$ and $\Delta^{\pi}(s_j)$ for each $\pi \in \Pi$ and $s_j \in \pi$, iff*

$$\bigwedge_{\pi \in \Pi} \bigwedge_{i=1}^{|\pi|} \left(n(\pi) \cdot l_i \;\leq\; \sum_{j=0}^{i-1} \Delta^{\pi}(s_j) \;\leq\; n(\pi) \cdot u_i \right) \tag{2}$$

where, for every $i = 1, 2, \ldots, |\pi|$, $l_i \leq x \leq u_i$ is the guard of edge e_i in π.

Proof: Let λ be an rr-path. For each $\pi \in \Pi$, we denote by $\pi(k)$ the $k'th$ occurrence of π in λ, and by $\tau_{\pi(k)}$ any trail tracing $\pi(k)$. We denote by $\Delta^{\pi(k)}(s_j)$ the visit duration of $\tau_{\pi(k)}$ in state s_j. Obviously, a trail tracing λ is realizable iff for every $\pi \in \Pi$ and every occurrence $k \geq 1$ of π in λ, $\tau_{\pi(k)}$ is realizable.

First, let τ be a realizable trail tracing λ. Let $\pi \in \Pi$ be a basic rr-path appearing once or more in λ. Then from Claim 4, for each occurrence $k \geq 1$ of π in λ,

$$\bigwedge_{i=1}^{|\pi|} \left(l_i \;\leq\; \sum_{j=0}^{i-1} \Delta^{\pi(k)}(s_j) \;\leq\; u_i \right)$$

Summing over all occurrences of π in λ and observing that $\Delta^{\pi}(s) = \sum_{k=1}^{n(\pi)} \Delta^{\pi(k)}(s)$, we obtain

$$\bigwedge_{i=1}^{|\pi|} \left(n(\pi) \cdot l_i \;\leq\; \sum_{j=0}^{i-1} \Delta^{\pi}(s_j) \;\leq\; n(\pi) \cdot u_i \right)$$

Finally, repeating the above for every $\pi \in \Pi$, we get

$$\bigwedge_{\pi \in \Pi} \bigwedge_{i=1}^{|\pi|} \left(n(\pi) \cdot l_i \;\leq\; \sum_{j=0}^{i-1} \Delta^{\pi}(s_j) \;\leq\; n(\pi) \cdot u_i \right)$$

In the other direction, let $n(\pi)$ and $\Delta^{\pi}(s_j)$ for every $\pi \in \Pi$, $s_j \in \pi$ be a set of attributes satisfying (2). Let $\pi \in \Pi$ be a basic rr-path appearing once or more in λ. Then with every occurrence $k \geq 1$ of π in λ we associate a trail $\tau_{\pi(k)}$ with the following attributes

$$\Delta^{\pi(k)}(s_j) = \frac{\Delta^{\pi}(s_j)}{n(\pi)}$$

for every $s_j \in \pi$. From Claim 4, $\tau_{\pi(k)}$ is realizable. Repeating the above for every $\pi \in \Pi$ we get a realizable trail tracing λ.

∎

Let us denote by $R \subseteq E$ the set of resetting edges. Let $e \in R$ be a resetting edge. We define the following sets of basic rr-paths associated with e

$$pred(e) = \{\pi \mid \pi = e_0, s_0, \ldots, e, s_n \in \Pi\}$$

$$succ(e) = \{\pi \mid \pi = e, s_0, \ldots, e_n, s_n \in \Pi\}$$

Let $\pi = e_0, s_0, \ldots, e_n, s_n \in \Pi$. Then

$$pred(\pi) = pred(e_0)$$

$$last(\pi) = s_n$$

$$first(\pi) = e_0$$

Let $\tau = \pi_0 \circ \cdots \circ \pi_n$ be a complete trail. For every $\pi \in \Pi$, we define

- $r(\pi)$ – the sequence number of π in the list of basic rr-paths visited by τ, sorted according to the order of their first visit.

The set of constraints (C) characterizing the attributes of a complete trail, is summarized in figure 10.

$$(C): \begin{cases} \bullet \ \textbf{Flow:} \\ \text{For every resetting edge } e \in R \text{ such that } e \notin E^- \text{ and } e \text{ does not arrive at } s \in S_f, \\ \qquad \displaystyle\sum_{\pi \in pred(e)} n(\pi) = \sum_{\pi \in succ(e)} n(\pi) \\ \bullet \ \textbf{Initiation and Termination:} \\ \qquad \displaystyle\sum_{first(\pi) \in E^-} n(\pi) = \sum_{last(\pi) \in S_f} n(\pi) = 1 \\ \bullet \ \textbf{Accessibility:} \\ \text{For every path } \pi = e_0, \ldots, s_n \in \Pi \text{ such that } e_0 \in E^- \\ \qquad (r(\pi) = n(\pi) = 0) \vee (r(\pi) = n(\pi) = 1) \\ \text{For every other path } \pi \in \Pi \\ \qquad \Big(r(\pi) = n(\pi) = 0\Big) \vee \Big(\displaystyle\bigvee_{\pi' \in pred(\pi)} (0 < r(\pi') < r(\pi))\Big) \\ \bullet \ \textbf{Time Constraints:} \text{ For every } \pi \in \Pi, \\ \qquad \displaystyle\bigwedge_{i=1}^{|\pi|} \left(n(\pi) \cdot l_i \ \leq \ \sum_{j=0}^{i-1} \Delta^{\pi}(s_j) \ \leq \ n(\pi) \cdot u_i \right) \\ \qquad \text{For every } s \notin \pi, \quad \Delta^{\pi}(s) = 0 \end{cases}$$

Fig. 10. Single Timer: The Attributes of Complete Trails

The following proposition states that the constraint set (C) is a precise characterization of a complete trail.

Proposition 4. *A set of values for the variables $n(\pi)$, $\Delta^{\pi}(s)$ is an attribute set for a complete trail iff it is a solution for (C).*

Proof: The set $\{n(\pi), r(\pi) \mid \pi \in \Pi\}$ satisfies the first three constraints iff there exists a complete path λ, such that for every $\pi \in \Pi$, $n(\pi)$ is the number of occurrences of π in λ. The proof is based on a constuction similar to the construction presented in the proof of Proposition 5 in section 4.

Claim 5 guarantees that the set $\Omega = \{n(\pi), \Delta^{\pi}(s) \mid$ for every $\pi \in \Pi$ and $s \in S\}$ is an attribute set of a realizable trail tracing λ iff Ω satisfies the last constraint.

∎

Corollary 5. *The duration formula*

$$\psi = \bigvee \bigwedge \Big(\sum_{i=1}^{n} a_i \cdot \int \varphi_i \sim c \Big)$$

is satisfiable over a single timer FTA *M iff the set of equalities/inequalities consisting of (C) plus the constraint*

$$\bigvee \bigwedge \Big(\sum_{i=1}^{n} a_i \cdot \Big(\sum_{\substack{\pi \in \Pi \\ s \models \varphi_i}} \Delta^{\pi}(s) \Big) \sim c \Big)$$

has a solution.

Complexity Considerations

The system (C) is a generalization of a mixed (integer/real) linear programming problem. The generalization stems from the fact that the components of the *accessibility* requirement and the *property* constraint contain disjunction. It is not difficult to see that, if (C) has a solution, then it also has a solution in which the real-valued variables, which are the various $\Delta^{\pi}(s_j)$, assume rational values. By an analysis similar to the one carried out in Section 4 and inspired by [**?**], we can show that checking whether $(C) + \psi$ has a solution is in NP, where the size parameter is the combined size of $(C) + \psi$.

Unfortunately, unlike the situation with system $C(M, \psi)$, the size of system (C) is no longer polynomial in the size of M (the original FTA), due to the presence of the set Π which, in the worst case can be of size exponential in that of M. For the cases that there exist a fixed upper bound (independent of $|M|$) on the length of all possible rr-paths, the size of $(C) + \psi$ is still polynomial in the size of M. However, when the size of rr-paths is only bounded by $|M|$, the only upper bound we can provide is doubly exponential in the size of M.

6 Real Computations: Disjunctive Durations

In section 4 we presented an algorithm for the satisfiability of general duration properties over the integer subset of an FTA's computation. In the following, we

show that the same algorithm can be used to check satisfiability over the entire set of an FTA's computations, providing we restrict the property to disjunctive durations.

Note that this solves the problem of reachability of integration graphs with at most one dangerous test along each path.

6.1 Digitization of FTA Computations

A time sequence $T = t_0, \ldots, t_n$, $t_i \in \mathsf{R}^+$ is called an *integer* time sequence if $t_i \in \mathsf{Z}$, for every $i = 0, \ldots, n$. We denote by $[0,1)$ the set of real numbers ϵ, satisfying $0 \leq \epsilon < 1$. Let $T = t_1, \ldots, t_n$ be a time sequence. For every $\epsilon \in [0,1)$, we define the integer time sequence $[T]_\epsilon = [t_1]_\epsilon, \ldots, [t_n]_\epsilon$ as follows:

$$[t_i]_\epsilon = \lfloor t_i \rfloor \quad \text{if } t_i \leq (\lfloor t_i \rfloor + \epsilon), \qquad \text{otherwise } [t_i]_\epsilon = \lceil t_i \rceil$$

Claim 6 *Let M be an* FTA*, and $\tau = (\sigma, T) \in \mathcal{T}_M$ be a complete trail of M. Then, for every $\epsilon \in [0,1)$, $[\tau]_\epsilon = (\sigma, [T]_\epsilon) \in \mathcal{T}_M$.*

Proof: Similar to the proof given for timed transition systems in [**?**].

The proof relies on the assumption that all enabling conditions use the non-strict inequality $\leq$. Allowing strict inequality invalidates the claim.

6.2 Disjunctive Durations over Digitizable Computations

Lemma 6. *Let*

$$0 < a_1 \leq a_2 \leq \ldots \leq a_m \leq 1 \quad \textit{and} \quad 0 < b_1 \leq b_2 \leq \ldots \leq b_n \leq 1$$

be two non-decreasing sequences of real numbers. If

$$\sum_{i=1}^{m} a_i \succ \sum_{j=1}^{n} b_j,$$

where $\succ \in \{\geq, >\}$, then there exists an $\epsilon \in [0,1)$, such that

$$\sum_{i=1}^{m} [a_i]_\epsilon \succ \sum_{j=1}^{n} [b_j]_\epsilon.$$

The proof of this lemma is presented in the appendix.

Proposition 7. *Let M be an* FTA *and ψ be a disjunctive duration property. The formula ψ is satisfiable over M iff ψ is satisfiable over $Z(\mathcal{T}_M)$.*

Proof outline: Let $\tau = (s_0, t_0), \ldots, (s_n, t_n) \in \mathcal{T}_M$, and ψ be a disjunctive duration property. Let

$$\psi' = \sum_{i=1}^{m} a_i \cdot \int \varphi_i \sim c$$

be a single disjunct of ψ. Interpreting ψ' over τ, we get

$$\psi' = \Big(\sum_{i=1}^{m} a_i \cdot \sum_{\substack{j=0 \\ s_j \models \varphi_i}}^{n-1} \Big(t_{j+1} - t_j\Big)\Big) \sim c$$

which can be rewritten as

$$\sum_{j=0}^{n} c_j \cdot t_j \sim \sum_{j=0}^{n} d_j \cdot t_j + c \tag{3}$$

where $c_j, d_j \in \mathsf{N}$. For every expression $(\alpha_j \cdot t_j)$ in equation 3, where $\alpha \in \{c, d\}$ and

$$t_j = \lfloor t_j \rfloor + \delta_j, \qquad 0 \leq \delta_j < 1$$

we rewrite the expression as follows

$$\overbrace{1 + \ldots + 1}^{\alpha_j \cdot \lfloor t_j \rfloor - times} + \overbrace{\delta_j + \ldots + \delta_j}^{\alpha_j - times}$$

turning equation 3 into the form used in Lemma 8. Thus, if ψ' is satisfiable over τ, there exists an $\epsilon \in [0, 1)$ such that ψ' is satisfiable over $[\tau]_\epsilon$, for every disjunct ψ' of ψ. Claim 6 completes the proof.

∎

The idea of analyzing dense-time computations by digitization is discussed in [?]. Using their terminology, Claim 6 states that the set of complete trails of an FTA is *closed under digitization*. Similarly, a direct result of Lemma 8 is that *conjuctive* duration properties are *closed under inverse digitization*.

Conclusion

Let M be an FTA and

$$\psi = \bigvee \Big(\sum_{i=1}^{m} a_i \cdot \textstyle\int \varphi_i \sim c\Big)$$

be a disjunctive duration formula. The satisfiability of ψ over M can be checked using the algorithm described in section 4.

7 Discussion and Related Work

In this paper we explore constant slope hybrid systems, searching for decidable subsets which can still be used for modeling realistic problems. We show that the reachability problem for even this restricted subset of hybrid systems is

undecidable. We start by considering integration graphs which allow integrators but query them only in a restricted way at the end of a computation.

We then look for decidable subsets of integration graphs. We first reduce their reachability problem to the satisfiability problem of duration properties over the simpler model of finite timed automata (FTA). For the latter, we establish decidability for three interesting subclasses. First, we present a decision procedure for the satisfiability of (general) duration properties over the integer computations of an FTA. This solves the reachability problem of integration graphs by integer computations. Then we show that the same decision procedure can be used to solve the problem of disjunctive duration properties, over the entire set of an FTA's computations. This solves the reachability problem of integration graphs with a single dangerous test on each path to a finite location. This decision procedure can be used to solve the Gas Burner problem presented in section 3. The Cat and Mouse problem, however, is specified with conjunctive duration properties, thus can not be solved with this procedure. Finally, we present a second (similar) decision procedure solving the satisfiability of general duration properties over the entire set of an FTA's computations, but restrict the FTA to a single timer. This solves the reachability problem of integration graphs with a single timer. The second procedure can be used to solve the Cat and Mouse problem (see Fig. 4 and Fig. 5). Similarly, it can be used to solve the Gas Burner problem, provided we consider the variable z an integrator.

Duration properties of finitary timed automata have been studied independently in [**?**]. Their method involves the construction of a region graph annotated with duration bounds. With the resulting graph, they can verify satisfiability of duration expressions over the entire set of computations of an FTA. Their duration expressions, constrained by both an upper bound u and a lower bound l, are of the form

$$l \leq (\sum_{i=1}^{m} a_i \cdot \int \varphi_i) \leq u$$

At a first glance, this seems to solve the problem of satisfiability of conjunctive duration properties with two conjuncts, over the entire set of an FTA's computations. However, the method presented in [**?**] is restricted to duration expressions with *positive* coefficients ($a_i \in N$). This case can easily be answered over the subset of integer computations of an FTA, replacing Lemma 8 with the following (simpler) proposition:

Proposition 8. *Let*

$$0 < a_1 \leq a_2 \leq \ldots \leq a_m \leq 1$$

be an increasing sequence. If

$$l \leq \sum_{i=1}^{m} a_i \leq u$$

where $l, u \in \mathsf{N}$*, then there exists an* $\epsilon \in [0,1)$*, such that*

$$l \leq \sum_{i=1}^{m} [a_i]_\epsilon \leq u$$

The following choice of ϵ proves the proposition:

$$a_{m-u} < \epsilon \leq a_{m-l+1}$$

Hybrid systems, and their decidable subset of Integration Graphs, are studied in the general context of *reactive systems*, which are systems that do not terminate. In this general context, we are interested in *relative* rather than *absolute* durations. For relative durations, we need duration expressions with negative coefficients, as can be seen in both the Cat and Mouse and the Gas Burner examples. The restriction to positive coefficients means a restriction to absolute durations.

An earlier version of the work reported in this paper was published in [?]. Since then, several publications which both extend and reuse the techniques described here, have been reported.

In [?] it is shown that more general duration properties like *response* (always eventually) and *persistence*, can be verified over CSHS with a single timer, if we assume a timer reset at every transition.

Decidability of the reachability problem for CSHS has been studied in several works. In [?] it is shown that reachability in CSHS is decidable under the constraints that only non-strict inequalities are used in edge guards, and timers are reset only at integer values. This result is based on a digitization technique which is a generalization of the digitization presented in [?]. In [?] it is shown that reachability in CSHS with two variables is decidable, under the restriction of a single integrator whose rate-labellings are constrained to -1, 0 or 1. Finally, in [?] it is shown that reachability in CSHS is decidable under the constraint that every integrator x is reset on each transition which causes a change in the rate-labelling associated with x.

In [?] another class of CSHS has been considered. These systems allow guards to be arbitrary polyhedra and rate-labellings to be rational values. However, they do not allow variables to be reset on transitions. In [?] a decision procedure for the reachability problem is given for systems with 2 continuous variables, while undecidability is proven for systems with 3 or more.

Using linear programming for the verification of duration properties has also been considered in [?]. In this work, the algorithm is defined for a subset of timed automata, specified by a special class of extended regular expressions. In contrast to our result, requiring *integer* linear programming, their systems can be solved by (rational) linear programming. This result is extended in [?] to a larger class of systems. The same approach has been used in [?] to develop a semi-decision procedure for the verification of duration invariants over timed regular expressions ([?]).

References

[ACH93] R. Alur, C. Courcoubetis, and T. Henzinger. Computing accumulated delays in real-time systems. In C. Courcoubetis, editor, *Proc. 5th Workshop on Computer Aided Verification*, volume 697 of *Lect. Notes in Comp. Sci.*, pages 181–193. Springer-Verlag, 1993.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comp. Sci.*, 138:3–34, 1995.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126:183–235, 1994.

[AH93] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[AMC97] E. Asarin, O. Maler, and P. Caspi. A kleene theorem for timed automata. In *Proc. 12th IEEE Symp. Logic in Comp. Sci.*, 1997.

[AMP95] A. Asarin, O. Maler, and A. Pnueli. On the analysis of dynamical systems having piecewise-constant derivatives. *Theor. Comp. Sci.*, 138:35–65, 1995.

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[BDM+98] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: a model-checking tool for real-time systems. In *Proc. 10th Conference Computer-Aided Verification, CAV'98*, volume 1427 of *Lect. Notes in Comp. Sci.*, Vancouver, Canada, June 1998. Springer-Verlag.

[BER94] A. Bouajjani, R. Echahed, and R. Robbana. Verifying Invariance Properties of Timed Systems with Duration Variables. In *FTRTFT'94*, volume 863 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 1994.

[BES95] A. Bouajjani, R. Echahed, and J. Sifakis. On Model Checking for Real Time Properties with Durations. In *Proc. 8th IEEE Symp. Logic in Comp. Sci.*, 1995.

[BH98] V. Braberman and D. Van Hung. On checking timed automata for linear duration invariants. In *RTSS'98*, 1998.

[BR95] A. Bouajjani and R. Robbana. Verifying ω-Regular Properties for Subclass of Linear Hybrid Systems. In *Proc. 7th Intl. Conference on Computer Aided Verification (CAV'95)*, volume 939 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 1995.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. *ACM Trans. Prog. Lang. Sys.*, 8:244–263, 1986.

[CHR92] Z. Chaochen, C.A.R Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1992.

[CJLX94] Z. Chaochen, Z. Jingzhong, Y. Lu, and L. Xiaoshan. Linear duration invariants. In *FTRT'94*, volume 863. Lect. Notes in Comp. Sci., 1994.

[Dil89] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lect. Notes in Comp. Sci.* Springer Verlag, New York, 1989.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: the next generation. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 56–65. IEEE Computer Society Press, 1995.

[HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, To appear. Special issue for STOC 95.

[HMP92] T. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *Proc. 19th Int. Colloq. Aut. Lang. Prog.*, volume 623 of *Lect. Notes in Comp. Sci.*, pages 545–558. Springer-Verlag, 1992.

[KPSY93] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration Graphs: A Class of Decidable Hybrid Systems. In A. Ravn and H. Rischel, editors, *Workshop on Hybrid Systems*, volume 736 of *Lect. Notes in Comp. Sci.*, pages 179–208. Springer-Verlag, 1993.

[LHT97] X. Li, D. Van Hung, and Z. Tao. Checking hybrid automata for linear duration invariants. In *Advances in Computing Science*, volume 1345. Lect. Notes in Comp. Sci., 1997.

[MMP92] O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lect. Notes in Comp. Sci.*, pages 447–484. Springer-Verlag, 1992.

[NSY93] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. *Acta Informatica*, 30:181–202, 1993.

[Pap81] C.H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.

Appendix

In the following, we present the proof of Lemma 8 (subsection 6.2).

Lemma 8. *Let*

$$0 < a_1 \leq a_2 \leq \ldots \leq a_m \leq 1 \quad \textit{and} \quad 0 < b_1 \leq b_2 \leq \ldots \leq b_n \leq 1$$

be two non-decreasing sequences of real numbers. If

$$\sum_{i=1}^{m} a_i \;\succ\; \sum_{j=1}^{n} b_j,$$

where $\succ \in \{\geq, >\}$*, then there exists an* $\epsilon \in [0,1)$*, such that*

$$\sum_{i=1}^{m} [a_i]_\epsilon \;\succ\; \sum_{j=1}^{n} [b_j]_\epsilon.$$

Proof: The proof is by induction on $n \geq 0$ for all m.

For $n = 0$, it is enough to take $\epsilon = 0$. Observing that $[c]_0 \geq c$, we write

$$\sum_{i=1}^{m} [a_i]_0 \;\geq\; \sum_{i=1}^{m} a_i \;\succ\; 0 \;=\; \sum_{j=1}^{0} [b_j]_0$$

Assume the lemma is true for n and show for $n+1$. We assume that

$$\sum_{i=1}^{m} a_i \;\succ\; \sum_{j=1}^{n} b_j + b_{n+1} \tag{4}$$

Since $b_{n+1} > 0$, m must be positive. We consider several cases:

Case: $a_m > b_{n+1}$
In this case, we take any ϵ satisfying $b_{n+1} < \epsilon < a_m$. This yields

$$\sum_{i=1}^{m} [a_i]_\epsilon \;\geq\; 1 \;\succ\; 0 \;=\; \sum_{j=1}^{n+1} [b_j]_\epsilon.$$

Consequently, we assume from now on that $a_m \leq b_{n+1}$.

Case: $a_m = b_{n+1}$
Subtracting $a_m = b_{n+1}$ from both sides of inequality (4), we obtain

$$\sum_{i=1}^{m-1} a_i \;\succ\; \sum_{j=1}^{n} b_j.$$

Applying the induction hypothesis to this inequality, we obtain an ϵ such that

$$\sum_{i=1}^{m-1} [a_i]_\epsilon \;\succ\; \sum_{j=1}^{n} [b_j]_\epsilon.$$

Adding $[a_m]_\epsilon = [b_{n+1}]_\epsilon = 1$ to both sides yields

$$\sum_{i=1}^{m} [a_i]_\epsilon \;\succ\; \sum_{j=1}^{n+1} [b_j]_\epsilon.$$

Case: $a_m < b_{n+1}$, $\succ$ is $>$
Subtracting a_m from the left-hand side and the bigger b_{n+1} from the right-hand side of inequality (4), we obtain

$$\sum_{i=1}^{m-1} a_i \;>\; \sum_{j=1}^{n} b_j.$$

Applying the induction hypothesis to this equation, we obtain an ϵ such that

$$\sum_{i=1}^{m-1} [a_i]_\epsilon \;>\; \sum_{j=1}^{n} [b_j]_\epsilon. \tag{5}$$

This ϵ must be smaller than a_{m-1} because, otherwise, the left hand side of (5) evaluates to 0, and we get the contradictory inequality

$$0 \;>\; \sum_{j=1}^{n} [b_j]_\epsilon \;\geq\; 0.$$

Consequently, $\epsilon < a_{m-1} \leq a_m < b_{n+1}$ and we may add $[a_m]_\epsilon = [b_{n+1}]_\epsilon = 1$ to both sides of (5), obtaining

$$\sum_{i=1}^{m} [a_i]_\epsilon \;\succ\; \sum_{j=1}^{n+1} [b_j]_\epsilon.$$

The remaining cases will deal with $\geq$.

Case: $a_m < b_{n+1} < 1$
Taking ϵ that satisfies $b_{n+1} < \epsilon < 1$, we obtain

$$\sum_{i=1}^{m} [a_i]_\epsilon \;=\; 0 \;\geq\; 0 \;=\; \sum_{j=1}^{n+1} [b_j]_\epsilon.$$

From now on, we assume that $b_{n+1} = 1$.

Case: $b_n < a_m < b_{n+1} = 1$
Taking ϵ that satisfies $b_n < \epsilon < a_m$, we obtain

$$\sum_{i=1}^{m} [a_i]_\epsilon \;\geq\; 1 \;=\; \sum_{j=1}^{n+1} [b_j]_\epsilon.$$

Case: $a_m \leq b_n \leq b_{n+1} = 1$
Subtracting a_m from the left-hand side of (4) and the not smaller b_n from its right-hand side, we obtain (substituting 1 for b_{n+1})

$$\sum_{i=1}^{m-1} a_i \;\geq\; \sum_{j=1}^{n-1} b_j + 1.$$

Applying the induction hypothesis to this case (that has n elements on its right-hand side), we obtain an ϵ such that

$$\sum_{i=1}^{m-1} [a_i]_\epsilon \;\geq\; \sum_{j=1}^{n-1} [b_j]_\epsilon + 1. \tag{6}$$

This ϵ must be smaller than a_{m-1} because, otherwise, the left hand side of (6) evaluates to 0, and we get the contradictory inequality

$$0 \;\geq\; \sum_{j=1}^{n-1} [b_j]_\epsilon + 1 \;\geq\; 1.$$

Consequently, $\epsilon < a_{m-1} \leq a_m \leq b_n$ and we may add $[a_m]_\epsilon = [b_n]_\epsilon$ to both sides of (6), obtaining

$$\sum_{i=1}^{m} [a_i]_\epsilon \;\succ\; \sum_{j=1}^{n+1} [b_j]_\epsilon.$$

This concludes the proof. ∎