ELSEVIER

The Journal of Logic and
Algebraic Programming 52–53 (2002) 79–108

THE JOURNAL OF
LOGIC AND
ALGEBRAIC
PROGRAMMING

www.elsevier.com/locate/jlap

Positive loop-closed automata: a decidable class of hybrid systems

Xuandong Li [*], Jianhua Zhao, Yu Pei, Yong Li, Tao Zheng, Guoliang Zheng

Department of Computer Science and Technology, State Key Laboratory of Novel Software Technology, Nanjing University, Jiangsu, Nanjing 210093, People's Republic of China

Abstract

The model-checking problem for real-time and hybrid systems is very difficult, even for a well-formed class of hybrid systems—the class of linear hybrid automata—the problem is still undecidable in general. So an important question for the analysis and design of real-time and hybrid systems is the identification of subclasses of such systems and corresponding restricted classes of analysis problems that can be settled algorithmically. In this paper, we show that for a class of linear hybrid automata called *positive loop-closed automata*, the satisfaction problem for *linear duration properties* can be solved by linear programming. We extend the traditional regular expressions with duration constraints and use them as a language to describe the behaviour of this class of linear hybrid automata. The extended notation is called *duration-constrained regular expressions*. Based on this formalism, we show that the model-checking problem can be reduced formally to linear programs.
© 2002 Elsevier Science Inc. All rights reserved.

Keywords: Real-time and hybrid systems; Model checking; Hybrid automata; Linear duration properties

1. Introduction

Hybrid systems are real-time systems that allow continuous state changes, over time periods of positive duration, as well as discrete state changes, in zero time. The formalism of hybrid automata [2] has become a standard model for real-time and hybrid systems.

A class of hybrid systems can be modelled by linear hybrid automata. Informally, a linear hybrid automaton is a conventional automaton extended with a set of variables, which are used to model the continuous state changes of hybrid systems and are assumed to be piecewise-linear functions of time. The states of the automaton called *locations* are

[*] Corresponding author.
E-mail addresses: lxd@nju.edu.cn (X. Li), zhaojh@nju.edu.cn (J. Zhao), peiyu@seg.nju.edu.cn (Y. Pei), liyong@seg.nju.edu.cn (Y. Li), zt@nju.edu.cn (T. Zheng), zhenggl@nju.edu.cn (G. Zheng).

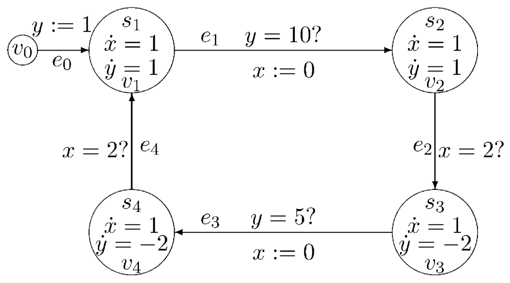

$y := 1$
e_0
v_0
s_1 $\dot{x} = 1$ $\dot{y} = 1$ v_1
e_1 $y = 10?$ $x := 0$
s_2 $\dot{x} = 1$ $\dot{y} = 1$ v_2
e_2 $x = 2?$
s_3 $\dot{x} = 1$ $\dot{y} = -2$ v_3
e_3 $y = 5?$ $x := 0$
s_4 $\dot{x} = 1$ $\dot{y} = -2$ v_4
$x = 2?$ e_4

Fig. 1. A hybrid automaton modelling a water-level monitor.

assigned with a system state and with a change rate for each variable, such as $\dot{x} = w$ (x is a variable, w is a real number), and the transitions of the automaton are labeled with constraints on the variables such as $a \leqslant x \leqslant b$ and /or with reset actions such as $x := c$ (x is a variable, a, b, and c are real numbers). The automaton starts at one of the initial locations with all variables initialised to their initial values. As time progresses, the values of all variables change continuously according to the rate associated with the current location. At any time, the system can change its current location from v to v' provided that there is a transition e from v to v' whose labeling conditions are satisfied by the current value of the variables. With a location change by a transition e, all the variables are reset to the new value accordingly by the reset actions labeled on e. Transitions are assumed to be instantaneous.

Let us consider an example of a water-level monitor in [3]. The water level in a tank is controlled through a monitor, which continuously senses the water level and turns a pump on and off. The water level changes as a piecewise-linear function of time. When the pump is off, the water level falls by 2 in. per second; when the pump is on, the water level rises by 1 in. per second. Suppose that initially the water level is one inch and the pump is on. There is a delay of two seconds from the time that the monitor signals to change the status of the pump to the time that the change becomes effective. The requirement of the water-level monitor is that the monitor must keep the water level in between 1 and 12 in. A design of the monitor is modelled by the hybrid automaton depicted in Fig. 1. The automaton has four locations v_1, v_2, v_3, v_4 which are assigned with the system states s_1, s_2, s_3, s_4 respectively. In the locations v_1 and v_2, the pump is on; in the locations v_3 and v_4, the pump is off. The variable y is used to model the water-level, and x is used to specify the delays: whenever the control is in location v_2 or v_3, the value of x indicates how long the signal to switch the pump off or on has been sent.

The model-checking problem for real-time and hybrid systems is very difficult, even for a well-formed class of hybrid systems—the class of linear hybrid automata—the problem is still undecidable in general [2,5,6]. So an important question for the analysis and design of real-time and hybrid systems is identification of subclasses of such systems and corresponding restricted classes of analysis problems that can be settled algorithmically [5]. In recent years there have been some works on searching for decidable analysis problems for a subclass of linear hybrid automata [3,5–7,11–13]. In this paper, for *linear duration properties* we give a new decidable subclass of linear hybrid automata called *positive loop-closed automata*. Based on linear programming, we solve the satisfaction problem of *positive loop-closed automata* for *linear duration properties*.

Linear duration properties are linear inequalities on integrated durations of system states. Here we use duration calculus (DC) [1] to describe this kind of properties. DC is a logic to specify and reason about requirements for real-time systems. It is an extension of Interval temporal logic which can be used to reason about integrated constraints over time-dependent and Boolean value states without explicit mention of absolute time. In DC, states are modelled as Boolean functions from reals (representing continuous time) to {0, 1}, where 1 denotes state presence, and 0 denotes state absence. For a state S, the integral variable $\int S$ of DC is a function from bounded and closed intervals to reals which stands for the accumulated presence time (duration) of state S over the intervals, and is defined formally by $\int S[a,b] \hat{=} \int_a^b S(t)\,\mathrm{d}t$, where $[a,b](b \geqslant a)$ is a bounded interval of time. A *linear duration property* in DC is of the form

$$\sum_{i=1}^{m} c_i \int S_i \leqslant M,$$

where S_is are system states, and M and c_is are real numbers. For example, the requirement of the water-level monitor, which is that the monitor must keep the water level in between 1 and 12 in., can be expressed by linear duration properties as well. We know that when the control is in locations v_1 or v_2, the water level rises 1 in. per second, and when the control is in locations v_3 or v_4, the water level falls by 2 in. per second. Furthermore, for an interval $[0, t]$, the accumulated time that the system stays in s_1 or s_2 is $\int s_1 + \int s_2$, and the accumulated time that the system stays in s_3 or s_4 is $\int s_3 + \int s_4$. Therefore, the water level at time t, given that at the beginning the water level is 1 in., is $1 + \int s_1 + \int s_2 - 2(\int s_3 + \int s_4)$. Hence, the requirement for the water-level monitor can be described by the following linear duration properties:

$$1 + \int s_1 + \int s_2 - 2\left(\int s_3 + \int s_4\right) \leqslant 12,$$

$$1 + \int s_1 + \int s_2 - 2\left(\int s_3 + \int s_4\right) \geqslant 1.$$

In this paper, we consider the problem of checking linear hybrid automata for linear duration properties. We extend the traditional regular expressions with duration constraints and use them as a language to describe the behaviour of linear hybrid automata. The extended notation is called *duration-constrained regular expressions*. Based on this formalism, we show that the model-checking problem can be reduced formally to linear programs for *positive loop-closed automata*. The paper is organised as follows. In Section 2, we recall the notion of linear hybrid automata. Section 3 defines *positive loop-closed automata*. Section 4 introduces *duration-constrained regular expressions* to describe the behaviour of linear hybrid automata. Section 5 shows that based on *duration-constrained regular expressions*, the model-checking problem for *positive loop-closed automata* can be reduced formally to linear programs. Section 6 discusses the related work and contains some conclusions.

2. Linear hybrid automata

A linear hybrid automaton is a conventional automaton extended with a finite set of real-valued variables. We use a simplified version of linear hybrid automata defined in [2]. The simplification is that any linear hybrid automaton considered in this paper has just one initial location, no initial condition, and no transition to the initial location (we suppose that each variable with an initial value is reset to the initial value by the transitions from the initial location).

Definition 1. A linear hybrid automaton is a tuple $H = (Z, X, V, E, v_I, \alpha, \beta)$, where

- Z is a finite set of system states.
- X is a finite set of real-numbered variables.
- V is a finite set of *locations*.
- E is *transition* relation whose elements are of the form (v, ϕ, ψ, v') where v, v' are in V, ϕ is a set of *variable constraints* of the form $a \leqslant x \leqslant b$, and ψ is a set of *reset actions* of the form $y := c$ ($x \in X$, $y \in X$, a, b, c are real numbers, a and b may be ∞; if a (b) is $-\infty$ (∞), then $a \leqslant x \leqslant b$ is taken to be $x \leqslant b$ ($a \leqslant x$); if $a = b$, then $a \leqslant x \leqslant b$ is taken to be $x = a$).
- v_I is an *initial* location.
- α is a labeling function which maps each location in V to a state in Z.
- β is a labeling function which maps each location in V to a set of *change rates* which are of the form $\dot{x} = a$ ($x \in X$ and a is a real number). For any location v, for any $x \in X$, there is one and only one $\dot{x} = a \in \beta(v)$.

We use *sequences of locations* to represent the evolution of a linear hybrid automaton from state to state. A sequence of locations is of the form

$$v_0 \xrightarrow{(\phi_0,\psi_0)} v_1 \xrightarrow{(\phi_1,\psi_1)} \cdots \xrightarrow{(\phi_m,\psi_m)} v_{m+1},$$

which indicates that the automaton start from location v_0, move to v_{i+1} from v_i with executing the reset actions in set ψ_i when the variable constraints in set ϕ_i are satisfied. For a linear hybrid automaton $H = (Z, X, V, E, v_I, \alpha, \beta)$, a *path segment* is a sequence of locations $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$ which satisfies $(v_i, \phi_i, \psi_i, v_{i+1}) \in E$ for each i ($1 \leqslant i \leqslant m-1$). A *path* in H is a path segment starting at v_I. A path (path segment) is called *simple path* (*path segment*) if all locations in the path (path segment) are distinct. A simple path (path segment) of the form $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$ is *bounded* if it cannot be extended into a longer simple path (path segment), i.e. there is no location v in H such that $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m \xrightarrow{(\phi,\psi)} v$ is a simple path (path segment).

The behaviour of linear hybrid automata can be represented by *timed sequences*. Any timed sequence is of the form $(s_1, t_1)^\wedge(s_2, t_2)^\wedge \cdots {}^\wedge(s_m, t_m)$, where s_i ($1 \leqslant i \leqslant m$) is a state and t_i ($1 \leqslant i \leqslant m$) is a nonnegative real number, which represents a behaviour of an automaton that the system starts at the state s_1, stays there for t_1 time units, then changes to s_2 and stays in s_2 for t_2 time units, and so on.

Definition 2. For a linear hybrid automaton $H = (Z, X, V, E, v_I, \alpha, \beta)$, a timed sequence $(s_1, t_1)^\wedge(s_2, t_2)^\wedge \cdots {}^\wedge(s_m, t_m)$ ($m \geqslant 1$) represents a behaviour of H if there is a path of the automaton

$$v_0 \xrightarrow{(\phi_0,\psi_0)} v_1 \xrightarrow{(\phi_1,\psi_1)} \cdots \xrightarrow{(\phi_m,\psi_m)} v_{m+1}$$

satisfying that

- for each i ($1 \leqslant i \leqslant m$), $\alpha(v_i) = s_i$; and

- $t_1, t_2, \ldots, t_m$ satisfy all the variable constraints in ϕ_i $(1 \leqslant i \leqslant m)$, i.e. for each variable constraint $a \leqslant x \leqslant b$ in ϕ_i, if there is a reset action $x := c$ in ψ_j $(0 \leqslant j < i)$ and $x := d$ is not in ψ_k for any k $(j < k < i)$, then

  $$a \leqslant c + w_{j+1}t_{j+1} + w_{j+2}t_{j+2} + \cdots + w_i t_i \leqslant b,$$

  where for each l $(j < l \leqslant i)$, $\dot{x} = w_l \in \beta(v_l)$.

For example, for the linear hybrid automaton depicted in Fig. 1, the timed sequence $(s_1, 9)^\wedge(s_2, 2)^\wedge(s_3, 3.5)^\wedge(s_4, 2)$ is a behaviour.

For a linear hybrid automaton H, for a transition $e = (v, \phi, \psi, v')$ in H, if e is labeled with a variable constraint $a \leqslant x \leqslant b$, i.e. $a \leqslant x \leqslant b \in \phi$, then we say that x is *tested* by e; if e is labeled with a reset action $x := c$, i.e. $x := c \in \psi$, then we say that x is *reset* by e. Notice that if a transition is labeled with a variable constraint $x = c$, we can take it as the transition resets the variable x to c. For example, for the automaton depicted in Fig. 1, we can say that the transitions e_1 and e_3 reset the variable y to 10 and 5 respectively, and the transitions e_2 and e_4 reset the variable x to 2.

3. Positive loop-closed automata

Positive loop-closed automata form a subclass of linear hybrid automata. In the following, we define this class of linear hybrid automata, and give an algorithm for identifying them.

3.1. Definition of positive loop-closed automata

A *positive loop-closed automaton* is a restricted linear hybrid automaton with its loops satisfying two restriction conditions. So we first define loops in linear hybrid automata. For a simple path in a linear hybrid automaton of the form $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$, if there is v_i $(1 < i \leqslant m)$ such that $(v_m, \phi, \psi, v_i) \in E$, then the sequence

$$v_i \xrightarrow{(\phi_i,\psi_i)} v_{i+1} \xrightarrow{(\phi_{i+1},\psi_{i+1})} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m \xrightarrow{(\phi,\psi)} v_i$$

is a *loop*, v_i is the *loop-start node* of the loop, $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{i-1},\psi_{i-1})} v_i$ is a *loop-enter path* of the loop, and (v_m, ϕ, ψ, v_i) is the *end* transition in the loop. Notice that a loop may have many different loop-enter paths. For a loop ρ, if ρ_1 is a loop-enter path of ρ, we say that ρ *can be entered through* ρ_1. For example, in the automaton depicted in Fig. 2, the sequence of locations

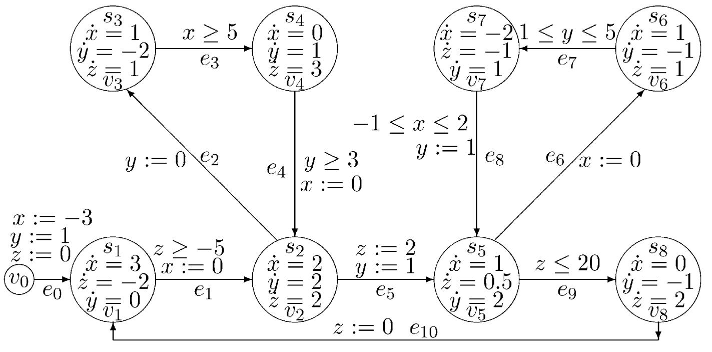
$$v_2 \xrightarrow{(\emptyset,\{y:=0\})} v_3 \xrightarrow{(\{x\geqslant 5\},\emptyset)} v_4 \xrightarrow{(\{y\geqslant 3\},\{x:=0\})} v_2$$

is a loop, and the sequence of locations

$$v_0 \xrightarrow{(\emptyset,\{x:=-3,y:=1,z:=0\})} v_1 \xrightarrow{(\{z\geqslant -5\},\{x:=0\})} v_2$$

is a loop-enter of the loop; and the sequence of locations

$$v_5 \xrightarrow{(\emptyset,\{x:=0\})} v_6 \xrightarrow{(\{1\leqslant y\leqslant 5\},\emptyset)} v_7 \xrightarrow{(\{-1\leqslant x\leqslant 2\},\{y:=1\})} v_5$$



s_3 $\dot{x}=1$ $\dot{y}=-2$ $\dot{z}=1$ v_3
$x \geq 5$
e_3
s_4 $\dot{x}=0$ $\dot{y}=1$ $\dot{z}=3$ v_4
s_7 $\dot{x}=-2$ $\dot{z}=-1$ $\dot{y}=1$ v_7
$1 \leq y \leq 5$
e_7
s_6 $\dot{x}=1$ $\dot{y}=-1$ $\dot{z}=1$ v_6
$-1 \leq x \leq 2$
$y := 1$ e_8
$y := 0$ e_2
e_4 $y \geq 3$ $x := 0$
e_6 $x := 0$
$x := -3$ $y := 1$ $z := 0$
v_0 e_0
s_1 $\dot{x}=3$ $\dot{z}=-2$ $\dot{y}=0$ v_1
$z \geq -5$ $x := 0$ e_1
s_2 $\dot{x}=2$ $\dot{y}=2$ $\dot{z}=2$ v_2
$z := 2$ $y := 1$ e_5
s_5 $\dot{x}=1$ $\dot{z}=0.5$ $\dot{y}=2$ v_5
$z \leq 20$ e_9
s_8 $\dot{x}=0$ $\dot{y}=-1$ $\dot{z}=2$ v_8
$z := 0$ e_{10}

Fig. 2. A loop-closed automaton.

is a loop, and the sequence of locations

$$v_0 \xrightarrow{(\emptyset,\{x:=-3,y:=1,z:=0\})} v_1 \xrightarrow{(\{z\geqslant -5\},\{x:=0\})} v_2 \xrightarrow{(\emptyset,\{y:=1,z:=2\})} v_5$$

is a loop-enter of the loop.

Then we introduce *loop-closedness* which is one condition satisfied by any loop in a *positive loop-closed automaton*. Intuitively, *loop-closedness* means that any variable constraint inside (outside) a loop is not related to any transition outside (inside) the loop, which is defined formally as follows. In a linear hybrid automaton H, let ρ be a loop of the form $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$, and ρ_1 be a path of the form

$$u_1 \xrightarrow{(\phi'_1,\psi'_1)} u_2 \xrightarrow{(\phi'_2,\psi'_2)} \cdots \xrightarrow{(\phi'_{i-1},\psi'_{i-1})} u_i \xrightarrow{(\phi'_i,\psi'_i)} \cdots \xrightarrow{(\phi'_{m-1},\psi'_{m-1})} u_m.$$

Suppose that $v_1 = u_i$, i.e. the loop ρ starts from u_i. It follows that by replacing u_i in ρ_1 with ρ we can get another path ρ_2 of the form

$$u_1 \xrightarrow{(\phi'_1,\psi'_1)} \cdots \xrightarrow{(\phi'_{i-2},\psi'_{i-2})} u_{i-1} \xrightarrow{(\phi'_{i-1},\psi'_{i-1})} \rho \xrightarrow{(\phi'_i,\psi'_i)} u_{i+1} \xrightarrow{(\phi'_{i+1},\psi'_{i+1})} \cdots \xrightarrow{(\phi'_{m-1},\psi'_{m-1})} u_m.$$

We say that in the path ρ_2, a variable constraint inside ρ is *related to* a transition outside ρ if the following condition holds:

- there is a variable constraint $a \leqslant x \leqslant b$ which is labeled on a transition $(v_j, \phi_j, \psi_j, v_{j+1})$ $(1 \leqslant j < m)$ in ρ,
- any transition $(v_k, \phi_k, \psi_k, v_{k+1})$ $(1 \leqslant k < j)$ in ρ does not reset the variable x, and
- there is a transition $(u_l, \phi'_l, \psi'_l, u_{l+1})$ $(1 \leqslant l < i-1)$ in ρ_2 resetting the variable x, but any transition $(u_k, \phi'_k, \psi'_k, u_{k+1})$ $(l < k < i)$ in ρ_2 does not reset the variable x,

which means that when the automaton stays in v_j along the path ρ_2, in order to check if the variable constraint $a \leqslant x \leqslant b$ is satisfied, we need to refer the value of x which is reset to by the transition $(u_l, \phi'_l, \psi'_l, u_{l+1})$. We say that in the path ρ_2, a variable constraint outside ρ is *related to* a transition inside ρ if the following condition holds:

- there is a variable constraint $a \leqslant x \leqslant b$ which is labeled on a transition $(u_j, \phi'_j, \psi'_j, u_{j+1})$ $(i \leqslant j < n)$ in ρ_2,
- any transition $(u_k, \phi'_k, \psi'_k, u_{k+1})$ $(i \leqslant k < j)$ in ρ_2 does not reset the variable x, and
- there is a transition $(v_l, \phi_l, \psi_l, v_{l+1})$ $(1 \leqslant l < m-1)$ in ρ resetting the variable x, but the end transition $(v_{m-1}, \phi_{m-i}, \psi_{m-1}, v_m)$ of ρ does not reset x,

which means that when the automaton stays in u_j along the path ρ_2, in order to check if the variable constraint $a \leqslant x \leqslant b$ is satisfied, we need to refer the value of x which is reset to by the transition $(v_l, \phi_l, \psi_l, v_{l+1})$.

Let ρ be a loop in a linear hybrid automaton H of the form

$$v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m.$$

We defined that ρ is *closed* if for any path in H containing ρ, any variable constraint inside ρ is not related to any transition outside ρ, and any variable constraint outside ρ is not related to any transition inside ρ, i.e. the following condition holds:

- any variable constraint inside ρ is not related to any transition outside ρ, i.e., for any simple path or loop

  $$u_1 \xrightarrow{(\phi'_1,\psi'_1)} u_2 \xrightarrow{(\phi'_2,\psi'_2)} \cdots \xrightarrow{(\phi'_{n-1},\psi'_{n-1})} u_n \quad (n > 1)$$

  satisfying that $v_1 = u_n$,
  - for any variable constraint $a \leqslant x \leqslant b$ labeled on a transition $(v_i, \phi_i, \psi_i, v_{i+1})$ $(1 \leqslant i \leqslant m)$ in ρ, if there is no transition $(v_j, \phi_j, \psi_j, v_{j+1})$ $(1 \leqslant j < i)$ in ρ resetting x, then x is reset to c by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ of ρ and by the transition $(u_{n-1}, \phi'_{n-1}, \psi'_{n-1}, u_n)$, i.e. $x := c \in \psi_{m-1}$ and $x := c \in \psi'_{n-1}$; and
  - for any variable y reset to d by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ of ρ, y is reset to d by the transition $(u_{n-1}, \phi'_{n-1}, \psi'_{n-1}, u_n)$, i.e.

  $$y := d \in \psi_{m-1} \;\Rightarrow\; y := d \in \psi'_{n-1};$$

- any variable constraint outside ρ is not related to any transition inside ρ, i.e., for any simple path segment

  $$u_1 \xrightarrow{(\phi'_1,\psi'_1)} u_2 \xrightarrow{(\phi'_2,\psi'_2)} \cdots \xrightarrow{(\phi'_{n-1},\psi'_{n-1})} u_n$$

  satisfying that $v_1 = u_1$, there is no variable constraint $a \leqslant x \leqslant b$ labeled on the transition $(u_{n-1}, \phi'_{n-1}, \psi'_{n-1}, u_n)$ satisfying that
  - x is reset by a transition $(v_i, \phi_i, \psi_i, v_{i+1})$ $(1 \leqslant i < m - 1)$ in ρ, and
  - x is not reset by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ of ρ and by any transition $(u_k, \phi'_k, \psi'_k, u_{k+1})$ $(1 \leqslant k < n - 1)$.

For example, in the automaton depicted in Fig. 2, the loop

$$v_2 \xrightarrow{(\emptyset,\{y:=0\})} v_3 \xrightarrow{(\{x\geqslant 5\},\emptyset)} v_4 \xrightarrow{(\{y\geqslant 3\},\{x:=0\})} v_2$$

is a closed loop. But it is not closed if we remove the reset action $y := 0$ from the transition e_2 since now for the path

$$v_0 \xrightarrow{(\emptyset,\{x:=-3,y:=1,z:=0\})} v_1 \xrightarrow{(\{z\geqslant -5\},\{x:=0\})}$$
$$v_2 \xrightarrow{(\emptyset,\{y:=0\})} v_3 \xrightarrow{(\{x\geqslant 5\},\emptyset)} v_4 \xrightarrow{(\{y\geqslant 3\},\{x:=0\})} v_2,$$

the variable constraint $y \geqslant 3$ labeled on e_4 is related to the transition e_0 which is outside the loop. That a loop is closed implies that the variable values inside the loop do not depend on their values outside the loop, and that the variable values outside the loop do not depend on their values inside the loop.

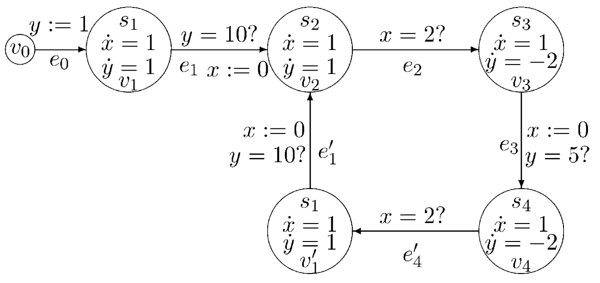


Fig. 3. A loop-closed automaton which has the same behaviour as the automaton in Fig. 1.

Definition 3. A linear hybrid automaton H is *loop-closed* if each loop in H is closed.

For example, the automaton depicted in Fig. 2 is a loop-closed automaton. We have discovered that for some loops which are not closed, if we change their loop-start nodes then they become closed. For example, the only loop in the automaton depicted in Fig. 1 is not closed, but if the location v_2 is the loop-start node instead of the location v_1 then the loop is closed. So we introduce a new location v_1' such that the location v_2 becomes the loop-start node, and get a loop-closed automaton with the same behaviour which is depicted in Fig. 3.

The other restriction condition satisfied by any loop in a *positive loop-closed automaton* is *reducibleness*. It follows that if any loop in a loop-closed automaton satisfies the *reducibleness* condition then the automaton is a *positive loop-closed automaton*. For defining the *reducibleness* condition, we first need to introduce *zero loops* and *nonzero loops*.

For a loop-closed automaton H, let ρ be a loop in H which is of the form $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$. We say that ρ is a *zero loop* if for any variable constraint $a \leqslant x \leqslant b$ in ϕ_i $(1 \leqslant i < m)$, $a - d \leqslant 0$ and $b - d \geqslant 0$ where d satisfies one of the following two conditions:

- x is reset to d by a transition $(v_j, \phi_j, \psi_j, v_{j+1})$ $(1 \leqslant j < i)$, but not reset by any transition $(v_k, \phi_k, \psi_k, v_{k+1})$ $(j < k < i)$; or
- x is reset to d by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$, but not reset by any transition $(v_k, \phi_k, \psi_k, v_{k+1})$ $(1 \leqslant k < i)$.

A loop is called *nonzero loop* if it is not a zero loop. According to the variable constraints on the transitions of a loop, if a loop is a zero loop, then a repetition of the loop may take no time; if a loop is a nonzero loop, then a repetition of the loop must take time. For example, in the automaton depicted in Fig. 2,

$$v_5 \xrightarrow{(\emptyset,\{x:=0\})} v_6 \xrightarrow{(\{1\leqslant y\leqslant 5\},\emptyset)} v_7 \xrightarrow{(\{-1\leqslant x\leqslant 2\},\{y:=1\})} v_5$$

and

$$v_1 \xrightarrow{(\{z\geqslant -5\},\{x:=0\})} v_2 \xrightarrow{(\emptyset,\{z:=2,y:=1\})} v_5 \xrightarrow{(\{z\leqslant 20\},\emptyset)} v_8 \xrightarrow{(\emptyset,\{z:=0\})} v_1$$

are zero loops, while

$$v_2 \xrightarrow{(\emptyset,\{y:=0\})} v_3 \xrightarrow{(\{x\geqslant 5\},\emptyset)} v_4 \xrightarrow{(\{y\geqslant 3\},\{x:=0\})} v_2$$

is a nonzero loop.

Then we define the *reducibleness* condition. For a loop-closed automaton H, a variable x is *positive* if $\dot{x} > 0$ for any location; a variable constraint $a \leqslant x \leqslant b$ is called *positive constraint* if x is positive and $b \neq \infty$. Let ρ be a loop in H which is of the form

$$v_1 \xrightarrow{(\phi_1, \psi_1)} v_2 \xrightarrow{(\phi_2, \psi_2)} \cdots \xrightarrow{(\phi_{m-1}, \psi_{m-1})} v_m.$$

We say that ρ is *reducible* if for any bounded simple path segment

$$u_1 \xrightarrow{(\phi'_1, \psi'_1)} u_2 \xrightarrow{(\phi'_2, \psi'_2)} \cdots \xrightarrow{(\phi'_{n-1}, \psi'_{n-1})} u_n$$

satisfying that $v_1 = u_1$,

- either ρ is not constrained by any variable constraint in the path segment, i.e. for any variable constraint $a \leqslant x \leqslant b$ labeled on a transition $(u_i, \psi'_i, \phi'_i, u_{i+1})$ $(1 \leqslant i < n)$, x is reset by a transition $(u_j, \phi'_j, \psi'_j, u_{j+1})$ $(1 \leqslant j < i)$ or by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ of ρ; or
- ρ is constrained by a positive constraint in the path segment, i.e. there is a positive constraint $a \leqslant x \leqslant b$ labeled on a transition $(u_i, \psi'_i, \phi'_i, u_{i+1})$ $(1 \leqslant i < n)$ satisfying that x is not reset by any transition $(u_j, \phi'_j, \psi'_j, u_{j+1})$ $(1 \leqslant j < i)$ and by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ of ρ.

For example, in the automaton depicted in Fig. 2, the nonzero loop

$$v_2 \xrightarrow{(\emptyset, \{y := 0\})} v_3 \xrightarrow{(\{x \geqslant 5\}, \emptyset)} v_4 \xrightarrow{(\{y \geqslant 3\}, \{x := 0\})} v_2$$

is reducible, but the zero loop

$$v_5 \xrightarrow{(\emptyset, \{x := 0\})} v_6 \xrightarrow{(\{1 \leqslant y \leqslant 5\}, \emptyset)} v_7 \xrightarrow{(\{-1 \leqslant x \leqslant 2\}, \{y := 1\})} v_5$$

is not reducible since along the path segment

$$v_5 \xrightarrow{(\{z \leqslant 20\}, \emptyset)} v_8 \xrightarrow{(\emptyset, \{z := 0\})} v_1 \xrightarrow{(\{z \geqslant -5\} \leqslant, \{x := 0\})} v_2,$$

it is constrained only by the variable constraint $z \leqslant 20$ on the transition e_9 which is not positive. For a nonzero loop, if it is constrained by a positive constraint in a path segment, then the positive constraint will be violated by unfolding the loop finite many times since every repetition of the nonzero loop must take time. So that a nonzero loop is reducible means that either it is not constrained by any variable constraint (in this case, we just need to unfold it one time for checking a given linear duration property), or it is constrained by a positive constraint so that any path of the automaton satisfying enforced time constraints only contains finite many times repetition of the loop (in this case, the model checking problem is decidable).

Definition 4. A *positive loop-closed automaton* is a loop-closed automaton in which each nonzero loop is reducible.

For example, the automata depicted in Figs. 2 and 3 are positive loop-closed automata. Although the definition of positive loop-closed automaton is not simple, we can give an efficient algorithm to check if a linear hybrid automaton is positive loop-closed, which is described in Appendix A.

Positive loop-closed automata form a decidable class of linear hybrid automata for linear duration properties. For real systems, the condition of positive loop-closedness is rational. For a real system (e.g., a control system), the loop-closedness means that every repetition of a control process starts from the same control conditions. Furthermore, for a control system, in most case, a repetition of a control process takes time, and a task containing the repetitions of the control process need to be finished in a given time (as a nonzero loop is constrained by a positive constraint). So we think that there are a number of real systems that satisfy the condition of positive loop-closedness and that can thus be modeled by positive loop-closed automata. In next section, we will introduce *duration-constrained regular expressions* to represent the behaviour of this class of linear hybrid automata. Based on this formalism, in Section 5 we show that the satisfaction problem of positive loop-closed automata for linear duration properties can be solved by linear programming.

4. Duration-constrained regular expressions

We know that the number of timed sequences to express the behaviour of a linear hybrid automaton may be infinite. In this section, we introduce *duration-constrained regular expressions* as a finite representation of behaviour of loop-closed automata, which is an extension of regular expressions with *duration constraints*.

4.1. Definition of duration-constrained regular expressions

While a regular expression over a set of states/transitions (alphabet) is a finite representation of a (infinite) set of sequences of states/transitions, a *duration-constrained regular expression* will be a finite representation of a set of timed sequences of states. By incorporating *duration constraints* into regular expressions, we get *duration-constrained regular expressions*.

A *duration constraint* in Duration Calculus is of the form $a \leqslant \sum_{i=1}^{m} c_i \int S_i \leqslant b$, where S_is are states, and a, b, and c_is are real numbers (a and b may be ∞). We use $S_1, S_2, \ldots, S_n$ to range over the states occurring in duration constraints and linear duration properties, and $s_1, s_2, \ldots, s_m$ to range over the states occurring in timed sequences. Let $\sigma = (s_1, t_1)^\wedge (s_2, t_2)^\wedge \cdots {}^\wedge(s_m, t_m)$ be a timed sequence. For a duration constraint $a \leqslant \sum_{i=1}^{n} c_i \int S_i \leqslant b$, for each i $(1 \leqslant i \leqslant n)$, the integrated duration of state S_i over σ can be calculated as $\int S_i = \sum_{u \in \alpha_i} t_u$ where $\alpha_i = \{u | (1 \leqslant u \leqslant m) \wedge (s_u = S_i)\}$. Consequently, σ satisfies the duration constraint if and only if $a \leqslant \sum_{i=1}^{n} c_i (\sum_{u \in \alpha_i} t_u) \leqslant b$.

Definition 5. For a *duration-constrained regular expression* (*DRE*) R, its language over a finite set Z of states is denoted by $L(R)$. Let $\mathbf{R}^+$ be the set of nonnegative real numbers. DREs and their languages are defined recursively as follows:
(1) ε is a DRE, and $L(\varepsilon) = \{\varepsilon\}$.
(2) If $s \in Z$, then s is a DRE, and $L(s) = \{(s, t) \mid t \in \mathbf{R}^+\}$.
(3) If R_1 and R_2 are DREs, then $R_1{}^\wedge R_2$ is a DRE, and

$$L(R_1{}^\wedge R_2) = \{\sigma_1{}^\wedge \sigma_2 \mid \sigma_1 \in L(R_1),\ \sigma_2 \in L(R_2)\}.$$

(4) If R_1 and R_2 are DREs, then $R_1 \oplus R_2$ is a DRE, and

$$L(R_1 \oplus R_2) = L(R_1) \cup L(R_2).$$

(5) If R is a DRE, then R^* is a DRE, and

$$L(R^*) = \left\{ \sigma_1{}^\wedge \cdots {}^\wedge\sigma_m \,\middle|\, m \geqslant 0 \text{ and } \bigwedge_{i=1}^{m} (\sigma_i \in L(R)) \right\},$$

where $\sigma_1{}^\wedge \cdots {}^\wedge\sigma_m \hat{=} \varepsilon$ when $m = 0$.

(6) If R is a DRE, and Δ is a set of duration constraints, then (R, Δ) is a DRE, and

$$L((R, \Delta)) = \{\sigma \mid \sigma \in L(R) \text{ satisfies all duration constraints in } \Delta\}.$$

Duration-constrained regular expressions form a simple formalism to model real-time and hybrid systems. Although the traditional regular expressions are powerful enough to describe the behaviour of finite automata, it is not the case for DREs to describe the behaviour of all linear hybrid automata. The reason is that the behaviour of linear hybrid automata is more complicated than the one of traditional automata since time is introduced. Nevertheless, DREs are powerful enough for describe the behaviour of loop-closed automata. In the following subsection, we show that any loop-closed automata can have its behaviour represented by a DRE.

4.2. From loop-closed automata to duration-constrained regular expressions

Now we show that for a loop-closed automaton, we can construct a DRE to represent its behaviour. The constructing process consists of two steps: first, for a given loop-closed automaton H, we construct a regular expression K over the set of its locations which represents the set of all paths in H; then, we construct a DRE to represent its behaviour by incorporating duration constraints into K and by replacing locations with states in K. Here for simplicity and being consistent with regular expressions, we denote a sequence of locations in a linear hybrid automaton by the form $v_1{}^\wedge v_2{}^\wedge \cdots {}^\wedge v_m$.

Let H be a linear hybrid automaton, and ρ be a path segment in H which is of the form $v_1{}^\wedge v_2{}^\wedge \cdots {}^\wedge v_m$. If ρ is not a simple path segment, then we can find v_i and v_j $(1 \leqslant i < j \leqslant m)$ such that $v_i = v_j$, and then we can get a path segment ρ_1 which is constructed from ρ by removing any v_k $(i < k \leqslant j)$. By applying the above *elimination* step repeatedly, we can get a simple path segment ρ'. We say that ρ is an *extension* of ρ'. We define that any simple path segment is an extension of itself. In the following, we first construct a regular expression for a loop which represents the set of the extensions of the loop, then construct a regular expression for a simple path which represents the set of the extensions of the simple path.

Let H be a loop-closed automaton, $\rho = v_1{}^\wedge v_2{}^\wedge \cdots {}^\wedge v_m$ be a loop in H, ρ_1 be a loop-enter path of ρ, and $\rho_i = \rho_1{}^\wedge v_2{}^\wedge \cdots {}^\wedge v_i$ $(1 < i < m)$. Let $E(\rho_1, \rho)$ be a regular expression over the locations in H, which is defined recursively as follows:

$$E(\rho_1, \rho) = v_1{}^\wedge G_2{}^\wedge v_2{}^\wedge G_3{}^\wedge v_3{}^\wedge \cdots {}^\wedge G_{m-1}{}^\wedge v_{m-1},$$

where if there is not any loop which can be entered through ρ_i $(1 < i < m)$, then $G_i = \varepsilon$, otherwise $G_i = (E(\rho_i, \rho_{i1}) \oplus E(\rho_i, \rho_{i2}) \oplus \cdots \oplus E(\rho_i, \rho_{in_i}))^*$ where $\rho_{i1}, \rho_{i2}, \ldots, \rho_{in_i}$ are all the loops which can be entered through ρ_i. Notice that the above recursive definition is convergent since the number of loops in H are finite and here each loop is combined with a loop-enter path.

Let H be a loop-closed automaton, $\rho = v_1{}^\wedge v_2{}^\wedge \cdots {}^\wedge v_m$ be a simple path in H, and $\rho_i = v_1{}^\wedge v_2{}^\wedge \cdots {}^\wedge v_i$ for any i $(1 < i \leqslant m)$. Let $F(\rho)$ be a regular expression over the locations in H, which is defined as follows:

$$F(\rho) = v_1{}^\wedge G_2{}^\wedge v_2{}^\wedge G_3{}^\wedge v_3{}^\wedge \cdots {}^\wedge G_{m-1}{}^\wedge v_{m-1}{}^\wedge G_m{}^\wedge v_m,$$

where if there is not any loop which can be entered through $\rho_i (1 < i \leqslant m)$, then $G_i = \varepsilon$, otherwise $G_i = (E(\rho_i, \rho_{i1}) \oplus E(\rho_i, \rho_{i2}) \oplus \cdots \oplus E(\rho_i, \rho_{in_i}))^*$, where $\rho_{i1}, \rho_{i2}, \ldots, \rho_{in_i}$ are all the loops which can be entered through ρ_i.

From the definition of F, it follows directly that for a loop-closed automaton, for a simple path ρ in the automaton, $F(\rho)$ represents the set of the paths which contains all extensions of ρ. It implies that for a loop-closed automaton, the set of its paths can be represented by $F(\rho_1) \oplus F(\rho_2) \oplus \cdots \oplus F(\rho_n)$, where $\rho_1, \rho_2, \ldots, \rho_n$ are all simple paths in the automaton.

Then, for a loop-closed automaton $H = (Z, X, V, E, v_I, \alpha, \beta)$ whose set of paths is represented by a regular expression K, we construct a DRE to represent its behaviour by incorporating duration constraints into K and by replacing locations with states in K. Each duration constraint incorporated into K is corresponding to a variable constraint in H. For a simple path or loop in H, let $K_1 = F(\rho)$ or $K_1 = E(\rho_1, \rho)$ which is of the form

$$K_1 = v_1{}^\wedge G_2{}^\wedge v_2{}^\wedge G_3{}^\wedge v_3{}^\wedge \cdots {}^\wedge G_{m-1}{}^\wedge v_{m-1}{}^\wedge G_m{}^\wedge G_{m+1},$$

where for each i $(1 < i \leqslant m)$, either $G_i = \varepsilon$ or $G_i = (G_{i1} \oplus G_{i2} \oplus \cdots \oplus G_{in_i})^*$ (if $K_1 = E(\rho_1, \rho)$, then $G_m = \varepsilon$ and $G_{m+1} = \varepsilon$, otherwise $G_{m+1} = v_m$). Since we can give a state assigned to a location in H with many names, we let each location occurrence in K_1 correspond to a different state name for avoiding name collision in duration constraints. Let $D(K_1)$ be a DRE which is defined recursively as follows:

$$D(K_1) = (s_1{}^\wedge R_2{}^\wedge s_2{}^\wedge R_3{}^\wedge s_3{}^\wedge \cdots {}^\wedge R_{m-1}{}^\wedge s_{m-1}{}^\wedge R_m, \Delta),$$

where

- each s_i $(1 \leqslant i < m)$ is the state name corresponding to v_i,
- if $G_i = \varepsilon$ $(1 < i \leqslant m)$, then $R_i = \varepsilon$, otherwise

$$R_i = (D(G_{i1}) \oplus D(G_{i2}) \oplus \cdots \oplus D(G_{in_i}))^*,$$

- Δ is the set of duration constraints whose elements are of the form

$$a - d \leqslant \sum_{i=1}^{n} c_i \int S_i \leqslant b - d,$$

where

- $a \leqslant x \leqslant b$ is a variable constraint labeled on a transition $(v_i, \phi_i, \psi_i, v_{i+1})$ $(1 \leqslant i < m)$;
- either x is reset to d by a transition $(v_j, \phi_j, \psi_j, v_{j+1})$ $(1 \leqslant j < i)$ and is not reset by any transition $(v_k, \phi_k, \psi_k, v_{k+1})$ $(j < k < i)$, or x is reset to d by the transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ and is not reset by any transition $(v_k, \phi_k, \psi_k, v_{k+1})$ $(1 \leqslant k < i)$ (in this case let $j = 0$);
- $\{S_1, S_2, \ldots, S_n\}$ is the set of the state names which are s_k $(j < k \leqslant i)$ or occur in $D(G_k)$ $(j < k \leqslant i$, and any transition occurring in G_k does not reset $x)$; and
- c_k $(1 \leqslant k \leqslant n)$ is the change rate of x in the location corresponding to S_k.

From Definitions 2 and 5, it follows that for a simple path ρ in H, for any timed sequence in $L(D(F(\rho)))$, if we replace all state names which correspond to the same location v with $\alpha(v)$, then we get a timed sequence which represents a behaviour of H. It implies that if $\rho_1, \rho_2, \ldots, \rho_n$ are all the simple paths in H, then the behaviour of H can be represented modulo renaming by

$$D(F(\rho_1)) \oplus D(F(\rho_2)) \oplus \cdots \oplus D(F(\rho_n)).$$

For example, for the hybrid automaton depicted in Fig. 3, its behaviour can be expressed by the following DRE R:

$$\begin{aligned}R = {} & \varepsilon \oplus s_0 \oplus \left(s_0{}^\wedge s_1{}^\wedge R_1, \left\{\int s_1 = 9\right\}\right) \\ & \oplus \left(s_0{}^\wedge s_1{}^\wedge R_1{}^\wedge s_2, \left\{\int s_1 = 9, \int s_2 = 2\right\}\right) \\ & \oplus \left(s_0{}^\wedge s_1{}^\wedge R_1{}^\wedge s_2{}^\wedge s_3, \left\{\int s_1 = 9, \int s_2 = 2, 2s_3 - s_2 = 5\right\}\right) \\ & \oplus \left(s_0{}^\wedge s_1{}^\wedge R_1{}^\wedge s_2{}^\wedge s_3{}^\wedge s_4, \left\{\int s_1 = 9, \int s_2 = 2, 2s_3 - s_2 = 5, \int s_4 = 2\right\}\right)\end{aligned}$$

where

$$\begin{aligned}R_1 = {} & \left(s_2'{}^\wedge s_3'{}^\wedge s_4'{}^\wedge s_1', \left\{\int s_2' = 2, 2\int s_3' - \int s_2' = 5,\right.\right. \\ & \qquad\qquad \left.\left.\int s_4' = 2, \int s_1' - 2\int s_4' = 5\right\}\right)^*,\end{aligned}$$

and s_1', s_2', s_3', s_4' are respectively the another state names corresponding to the locations v_1', v_2, v_3, v_4.

We have given a transition procedure from loop-closed automata to DREs. Whether there is an inverse transition is an interesting question. It seems that both formalisms have the same expressive power. Since we introduce DREs just for checking positive loop-closed automata, we leave this open problem here.

5. Checking positive loop-closed automata for linear duration properties

In this section, we solve the problem of checking positive loop-closed automata for linear duration properties.

A timed sequence $\sigma = (s_1, t_1)^\wedge(s_2, t_2)^\wedge \cdots {}^\wedge(s_m, t_m)$ satisfies a linear duration property P of the form $\sum_{i=1}^n c_i \int S_i \leqslant M$ if and only if $\sum_{i=1}^n c_i (\sum_{u \in \alpha_i} t_u) \leqslant M$, where $\alpha_i = \{u \mid (1 \leqslant u \leqslant m) \wedge (s_u \Rightarrow S_i)\}$. A linear hybrid automaton satisfies a linear duration property if and only if every timed sequence representing a behaviour of the automaton satisfies the linear duration property. A DRE R satisfies a linear duration property P, denoted by $R \models P$, if and only if any timed sequences $\sigma \in L(R)$ satisfies P. So, for a loop-closed automaton whose behaviour can be represented by a DRE, the satisfaction problem for a linear duration property can be solved by checking if the DRE satisfies the linear duration property. In the following, we consider the problem of checking if a DRE satisfies a linear duration property.

5.1. Some concepts concerning duration-constrained regular expressions

First we need to introduce more concepts about DREs which will be used in solving the problem.

For a DRE R, if $L(R) = \emptyset$, then R is said to be *empty*.

A *simple* DRE is a DRE in which there is no occurrence of the combinators $*$ (repetition) and $\oplus$ (union). From Definition 5, it follows that by renaming states, any simple DRE R

can be rewritten as a simple DRE R' of the form $(s_1{}^\wedge s_2{}^\wedge \cdots {}^\wedge s_n, \Delta)$ such that $L(R) = L(R')$, where $s_1, s_2, \ldots, s_n$ are states. Therefore, from now on, we assume that any simple DRE is of the form $(s_1{}^\wedge s_2{}^\wedge \cdots {}^\wedge s_n, \Delta)$. Intuitively, a simple path is corresponding to a simple path segment in an automaton.

For any simple DRE $R = (s_1{}^\wedge s_2{}^\wedge \cdots {}^\wedge s_n, \Delta)$ such that $L(R) \neq \emptyset$, if each duration constraint $a \leqslant \sum_{i=1}^{m} c_i \int S_i \leqslant b \in \Delta$ satisfies that $a \leqslant 0$ and $b \geqslant 0$, then R is said to be a *zero-simple* DRE, otherwise R is said to be a *nonzero-simple* DRE. The intuitive meanings of zero-simple DREs and nonzero-simple DREs are respectively corresponding to zero loops and nonzero loops in an automaton.

By a *normal form* we mean a DRE of the form $R_1 \oplus R_2 \oplus \cdots \oplus R_m$, where each R_i $(1 \leqslant i \leqslant m)$ is a simple DRE.

For a DRE R, its *sub-expressions* are defined recursively by

(1) R is a sub-expression of R.
(2) If $R = R_1{}^\wedge R_2$ or $R = R_1 \oplus R_2$, where R_1 and R_2 are DREs, then all the sub-expressions of R_1 or of R_2 are sub-expressions of R.
(3) If $R = R_1^*$ or $R = (R_1, \Delta)$ where R_1 is a DRE, then all the sub-expressions of R_1 are sub-expressions of R.

Let R be a DRE, and R_1 be a sub-expression of R. Replacing an occurrence of R_1 in R with a letter X, we obtain a *context* of X, denoted by $C(X)$. Any context of X, $C(X)$, is associated with a set of duration constraints which are enforced on the variable X by the context, which is denoted by $\Gamma(C(X))$. If a context $C(X)$ does not enforce any duration constraint on X, then $\Gamma(C(X)) = \emptyset$.

Definition 6. A context $C(X)$ of X and $\Gamma(C(X))$ are defined recursively as

(1) X is a context of X, and $\Gamma(X) = \emptyset$.
(2) If $C_1(X)$ is a context of X and R is a DRE, then $C(X) = C_1(X){}^\wedge R$ and $C(X) = R{}^\wedge C_1(X)$ are contexts of X, and $\Gamma(C(X)) = \Gamma(C_1(X))$.
(3) If $C_1(X)$ is a context of X and R is a DRE, then $C(X) = C_1(X) \oplus R$ and $C(X) = R \oplus C_1(X)$ are contexts of X, and $\Gamma(C(X)) = \Gamma(C_1(X))$.
(4) If $C_1(X)$ is a context of X, then $C(X) = C_1(X)^*$ is a context of X, and $\Gamma(C(X)) = \Gamma(C_1(X))$.
(5) If $C_1(X)$ is a context of X and Δ is a set of duration constraints, then $C(X) = (C_1(X), \Delta)$ is a context of X, and $\Gamma(C(X)) = \Gamma(C_1(X)) \cup \Delta$.

For any context $C(X)$, replacing X in $C(X)$ with a DRE, say R, we obtain a DRE, denoted by $C(R)$.

5.2. Basic idea for solving the satisfaction problem

Let $R = (s_1{}^\wedge s_2{}^\wedge \cdots {}^\wedge s_m, \Delta)$ be a simple DRE, and $P = \sum_{i=1}^{n} c_i \int S_i \leqslant M$ be a linear duration property. From the definition of DREs, it follows that every $\sigma \in L(R)$ is of the form $(s_1, t_1){}^\wedge(s_2, t_2){}^\wedge \cdots {}^\wedge(s_m, t_m)$, and that for any duration constraint $a \leqslant \sum_{i=1}^{n} c_i' S_i' \leqslant b \in \Delta$, $t_1, t_2, \ldots, t_m$ must satisfy

$$a \leqslant \sum_{i=1}^{n} c_i' \left(\sum_{u \in \beta_i} t_u \right) \leqslant b,$$

where $\beta_i = \{u \mid (1 \leqslant u \leqslant m) \wedge (s_u \Rightarrow S_i')\}$, which form a group of linear inequalities on $t_1, t_2, \ldots, t_m$ denoted by C. If the group C of linear inequalities has no solutions, then $L(R) = \emptyset$; otherwise the problem of checking $R \models P$ is equivalent to the problem of finding the maximal value of the linear function $\sum_{i=1}^{n} c_i (\sum_{u \in \alpha_i} t_u)$ where $\alpha_i = \{u \mid (1 \leqslant u \leqslant m) \wedge (s_u \Rightarrow S_i)\}$ subject to the linear constraint C and checking whether it is not greater than M. The latter is a linear programming problem.

Let $N = R_1 \oplus R_2 \oplus \cdots \oplus R_m$ be a normal form. It follows that each R_i $(1 \leqslant i \leqslant m)$ is a simple DRE. For a linear duration property P, since

$$N \models P \quad \Leftrightarrow \quad \bigwedge_{i=1}^{m} R_i \models P,$$

the problem of checking N for P can be solved by solving m problems of checking $R_i \models P$ for $i = 1, 2, \ldots, m$, which can be solved by linear programming.

Therefore, for a general DRE R, for a linear duration property P, if we can effectively find a normal form N satisfying that $R \models P$ if and only if $N \models P$, then we can check $R \models P$ effectively.

5.3. *Foundation of algorithm*

For a timed sequence $\sigma = (s_1, t_1)^\wedge(s_2, t_2)^\wedge \cdots {}^\wedge(s_m, t_m)$, for an integrated duration $\sum_{i=0}^{n} c_i \int S_i$, let $\theta(\sigma, \sum_{i=0}^{n} c_i \int S_i)$ be the value of $\sum_{i=1}^{n} c_i \int S_i$ evaluated over σ, i.e.

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right) = \sum_{i=1}^{n} c_i \left(\sum_{u \in \alpha_i} t_u\right),$$

where $\alpha_i = \{u \mid (1 \leqslant u \leqslant m) \wedge (s_u \Rightarrow S_i)\}$.

For a state s, for an integrated duration $\sum_{i=0}^{n} c_i \int S_i$, let

$$\zeta\left(s, \sum_{i=0}^{n} c_i \int S_i\right) = \sum_{u \in \beta} c_u,$$

where $\beta = \{u \mid (1 \leqslant u \leqslant n) \wedge (s \Rightarrow S_u)\}$.

For any simple DRE R such that $L(R) \neq \emptyset$, for any integrated duration $\sum_{i=1}^{n} c_i \int S_i$, let $m_\theta(R, \sum_{i=1}^{n} c_i \int S_i)$ and $M_\theta(R, \sum_{i=1}^{n} c_i \int S_i)$ denote respectively the infimum and supremum of the set

$$\left\{\theta\left(\sigma, \sum_{i=1}^{n} c_i \int S_i\right) \middle| \ \sigma \in L(R)\right\}.$$

Notice that $m_\theta(R, \sum_{i=1}^{n} c_i \int S_i)$ and $M_\theta(R, \sum_{i=1}^{n} c_i \int S_i)$ can be calculated by linear programming, and that for a simple DRE R, for a linear duration property $P = \sum_{i=1}^{n} c_i \int S_i \leqslant M$, if $M_\theta(R, \sum_{i=1}^{n} c_i \int S_i) > 0$, then there is a concatenation of finint many Rs which does not satisfy P, that is, $R^* \not\models P$.

For a nonzero-simple DRE R, we say that a duration constraint

$$a \leqslant \sum_{i=1}^{n} c_i \int S_i \leqslant b$$

is *positive* for R if each state occurring in R is in $\{S_1, S_2, \ldots, S_n\}$, $b \neq \infty$, and $c_i > 0$ for any i $(1 \leqslant i \leqslant n)$. We say that a context $C(X)$ is *bounded* for R if there is a positive duration constraint for R in $\Gamma(C(X))$. Intuitively, a positive duration constraint is corresponding to a positive constraint in an automaton. The intuitive meaning of $C(R)$, which results from putting a nonzero-simple DRE R into a bounded context $C(X)$ for R, is corresponding to that a nonzero loop in an automaton is constrained by a positive constraint.

We say that a context $C(X)$ is *free* if $\Gamma(C(X)) = \emptyset$. The intuitive meaning of $C(R)$, which results from putting a nonzero-simple DRE R into a free context $C(X)$, is corresponding to that a nonzero loop in an automaton is not constrained by any variable constraint.

Let R be a DRE representing the behaviour of a positive loop-closed automaton, and R_1 be a sub-expression of R which is an nonzero-simple DRE. Replacing an occurrence of R_1^* in R with X, we get a context $C(X)$. Notice that since any positive loop-closed automaton satisfies that any nonzero loop is reducible, either $C(X)$ is bounded for R or $C(X)$ is free.

Let R be a nonzero-simple DRE, and $C(X)$ be a bounded context for R. Notice that for any duration constraint $a \leqslant \sum_{i=1}^{n} c_i \int S_i \leqslant b$ which is positive for R, $m_\theta(R, \sum_{i=1}^{n} c_i \int S_i) > 0$. Let $\omega(C(X), R)$ denote the minimal value of the set

$$\left\{ b/m_\theta\left(R, \sum_{i=1}^{n} c_i \int S_i\right) \middle| \, a \leqslant \sum_{i=1}^{n} c_i \int S_i \leqslant b \in \Gamma(C(X)) \text{ is positive for } R \right\}.$$

The intuitive meaning of $\omega(C(X), R)$ is that in any timed sequence $\in L(C(R))$ there is no occurrence of any concatenation of more than $\omega(C(X), R)$ timed sequences in $L(R)$.

For a DRE R and a linear duration property P, we attempt to find a normal form N such that $L(R) \models P$ if and only if $L(N) \models P$ by the following procedure:

- *Step* 0. Let $R' := R$.
- *Step* 1. For R', distributing $^\wedge$ over $\oplus$, and Δ over $\oplus$, we obtain Q. If Q is a normal form, then we have done.
- *Step* 2. For a sub-expression Q_S of Q which is of the form $Q_S = Q_1{}^*$, replacing an occurrence of Q_S in Q with X, we obtain a context $C_Q(X)$ such that $Q = C_Q(Q_S)$.
- *Step* 3. Finding a DRE Q'_S in which there is no occurrence of combinator $*$ satisfying that $C_Q(Q_S) \models P$ if and only if $C_Q(Q'_S) \models P$. Let $R' := C_Q(Q'_S)$ and goto Step 1.

Obviously the procedure is correct. The problem is how to find Q'_S in Step 3. The following lemmas and theorems will help to solve that problem.

Lemma 1. *Let R and R' be DREs, $C(X)$ be a context, and P be a linear duration property. If for any $\sigma \in L(R)$, there is $\sigma' \in L(R')$ satisfying that*

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma', \sum_{i=0}^{n} c_i \int S_i\right)$$

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$, then $C(R') \models P$ implies $C(R) \models P$.

Lemma 2. *Let R and R' be nonzero-simple DREs, $P = \sum_{i=0}^{n} c_i \int S_i \leqslant M$ be a linear duration property, and $C(X)$ be a free context. If for any $\sigma \in L(R)$, there is $\sigma' \in L(R')$ satisfying that $\theta(\sigma, \sum_{i=0}^{n} c_i \int S_i) \leqslant \theta(\sigma', \sum_{i=0}^{n} c_i \int S_i)$, then $C(R') \models P$ implies $C(R) \models P$.*

Lemma 3. *Let $P = \sum_{i=1}^{n} c_i \int S_i \leqslant M$ be a linear duration property, R be a nonzero-simple DRE such that $M_\theta(R, \sum_{i=0}^{n} c_i \int S_i) \leqslant 0$, and $C(X)$ be a free context. Then for any $\sigma \in L(C(R^*))$, there is $\sigma' \in L(C(\varepsilon))$ satisfying that $\theta(\sigma, \sum_{i=1}^{n} c_i \int S_i) \leqslant \theta(\sigma', \sum_{i=1}^{n} c_i \int S_i)$.*

Lemma 4. *Let R be a nonzero-simple DRE, and $C(X)$ be a bounded context for R. Then $L(C(\bigoplus_{j=0}^{p} R^j)) \supseteq L(C(R^*))$, where $p = \lfloor \omega(C(X), R) \rfloor + 1$.*

These lemmas can be proved by induction on the structure of context, and their detailed proofs are presented in Appendix B. From these lemmas, we can prove the following theorems.

Theorem 1. *Let R_1 and R_2 be DREs, P be a linear duration property, and $C(X)$ be a context. Then $C((R_1 \oplus R_2)^*) \models P$ iff $C(R_1^{*\wedge} R_2^*) \models P$.*

Proof. By Definition 5, $L((R_1{}^*)^\wedge(R_2{}^*)) \subseteq L((R_1 \oplus R_2)^*)$. From Lemma 1, the half of the claim follows, i.e.

$$C((R_1 \oplus R_2)^*) \models P \text{ implies } C((R_1{}^*)^\wedge(R_2{}^*)) \models P.$$

The other half can be proved as follows. For any integrated duration $\sum_{i=1}^{n} c_i \int S_i$, for any $\sigma_1 \in L(R_1)$ and $\sigma_2 \in L(R_2)$, since

$$\theta\left(\sigma_1{}^\wedge\sigma_2, \sum_{i=1}^{n} c_i \int S_i\right) = \theta\left(\sigma_1, \sum_{i=1}^{n} c_i \int S_i\right) + \theta\left(\sigma_2, \sum_{i=1}^{n} c_i \int S_i\right),$$

we have $\theta(\sigma_1{}^\wedge\sigma_2, \sum_{i=1}^{n} c_i \int S_i) = \theta(\sigma_2{}^\wedge\sigma_1, \sum_{i=1}^{n} c_i \int S_i)$. Furthermore, for any $\sigma \in L(C((R_1 \oplus R_2)^*)$, it can be permuted into a $\sigma' \in L(C((R_1{}^*)^\wedge(R_2{}^*)))$. Hence, from Lemma 1, the result follows. □

Theorem 2. *Let $R = (s_1{}^\wedge s_2{}^\wedge \cdots {}^\wedge s_m, \Delta)$ be a zero-simple DRE, P be a linear duration property, and $C(X)$ be a context. Let $R' = (s_1{}^\wedge s_2{}^\wedge \ldots {}^\wedge s_m, \Delta')$, where $\Delta' = \Delta_1 \cup \Delta_2 \cup \Delta_3$,*

$$\Delta_1 = \left\{ 0 \leqslant \sum_{i=1}^{n} c_i S_i \,\middle|\, \begin{array}{l} 0 \leqslant \sum_{i=1}^{n} c_i S_i \leqslant b \in \Delta, b \neq 0, \\ \textit{and}\ \exists j \cdot (1 \leqslant j \leqslant n \wedge c'_j < 0) \end{array} \right\},$$

$$\Delta_2 = \left\{ \sum_{i=1}^{n} c_i S_i \leqslant 0 \,\middle|\, \begin{array}{l} a \leqslant \sum_{i=1}^{n} c_i S_i \leqslant 0 \in \Delta, a \neq 0, \\ \textit{and}\ \exists j \cdot (1 \leqslant j \leqslant n \wedge c'_j > 0) \end{array} \right\},$$

$$\Delta_3 = \left\{ 0 \leqslant \sum_{i=1}^{n} c_i S_i \leqslant 0 \mid 0 \leqslant \sum_{i=1}^{n} c_i S_i \leqslant 0 \in \Delta \right\}.$$

Then $C(R^*) \models P$ iff $C(R') \models P$.

Proof. The half of the claim that $C(R') \models P$ implies $C(R^*) \models P$ is explained as follows. By Definition 5, any $\sigma \in L(R^*)$ is of the form $\sigma_1{}^\wedge\sigma_2{}^\wedge \cdots {}^\wedge\sigma_n$, where

$$\sigma_i = (s_1, t_{i1})^\wedge(s_2, t_{i2})^\wedge \cdots {}^\wedge(s_m, t_{im}) \in L(R) \quad (i = 1, 2, \ldots, n).$$

For any j $(1 \leqslant j \leqslant m)$, let $t'_j = t_{1j} + t_{2j} + \cdots + t_{nj}$, and let

$$\sigma' = (s_1, t'_1)^\wedge(s_2, t'_2)^\wedge \cdots {}^\wedge(s_m, t'_m).$$

Since for any i $(1 \leqslant i \leqslant n)$, $t_{i1}, t_{i2}, \ldots, t_{im}$ satisfy Δ, $t'_1, t'_2, \ldots, t'_m$ satisfy Δ' as well. It follows that $\sigma' \in L(R')$. Since $\theta(\sigma, \sum_{i=1}^n c_i \int S_i) = \theta(\sigma', \sum_{i=1}^n c_i \int S_i)$ for any integrated duration $\sum_{i=1}^n c_i \int S_i$, the first half of the claim follows from Lemma 1.

The other half of the claim, i.e. $C(R^*) \models P$ implies $C(R') \models P$, can be proved as follows. For any $\sigma' = (s_1, t_1)^\wedge(s_2, t_2)^\wedge \cdots {}^\wedge(s_m, t_m) \in L(R')$, since $t_1, t_2, \ldots, t_m$ satisfy Δ',

- for any $0 \leqslant \sum_{i=1}^m c_i \int S_i \leqslant b \in \Delta$, we have $\theta(\sigma', \sum_{i=1}^m c_i \int S_i) \geqslant 0$;
- for any $a \leqslant \sum_{i=1}^m c_i \int S_i \leqslant 0 \in \Delta$, we have $\theta(\sigma', \sum_{i=1}^m c_i \int S_i) \leqslant 0$; and
- for any $0 \leqslant \sum_{i=1}^m c_i \int S_i \leqslant 0 \in \Delta$, we have $\theta(\sigma', \sum_{i=1}^m c_i \int S_i) = 0$.

Because for any $a \leqslant \sum_{i=1}^m c_i \int S_i \leqslant b$ in Δ, $a \leqslant 0$ and $b \geqslant 0$, and because Δ is a finite set, we can choose a natural number p satisfying that

$$\text{for any } a \leqslant \sum_{i=1}^m c_i \int S_i \leqslant b \in \Delta,\ a \leqslant \theta(\sigma', \sum_{i=1}^m c_i \int S_i)/p \leqslant b.$$

For each i $(1 \leqslant i \leqslant m)$, let $b_i = t_i/p$, and let $\sigma_b = (s_1, b_1)^\wedge(s_2, b_2)^\wedge \cdots {}^\wedge(s_m, b_m)$. Obviously, $\sigma_b \in L(R)$. Let

$$\sigma = \underbrace{\sigma_b{}^\wedge\sigma_b{}^\wedge \cdots {}^\wedge\sigma_b}_{p}.$$

It follows that $\sigma \in L(R^*)$. Since $\theta(\sigma, \sum_{i=0}^n c_i \int S_i) = \theta(\sigma', \sum_{i=0}^n c_i \int S_i)$ for any integrated duration $\sum_{i=0}^n c_i \int S_i$, by Lemma 1, $C(R^*) \models P$ implies $C(R') \models P$. □

Theorem 3. *Let $P = \sum_{i=1}^n c_i \int S_i \leqslant M$ be a linear duration property, R be a nonzero-simple DRE satisfying that $M_\theta(R, \sum_{i=0}^n c_i \int S_i) > 0$, and $C(X)$ be a free context. Then there is a state s occurring in R satisfying that $\zeta(s, \sum_{i=0}^n c_i \int S_i) > 0$, and $C(R^*) \models P$ iff $C(s) \models P$.*

Proof. Since R is a nonzero-simple DRE, any $\sigma \in L(R)$ is of the form

$$(s_1, t_1)^\wedge(s_2, t_2)^\wedge \cdots {}^\wedge(s_m, t_m),$$

and $\theta(\sigma, \sum_{i=0}^n c_i \int S_i) = \sum_{i=1}^m \zeta(s_i, \sum_{i=0}^n c_i \int S_i) t_i$. If there is not any state s_i $(1 \leqslant i \leqslant m)$ occurring in R satisfying that $\zeta(s_i, \sum_{i=0}^n c_i \int S_i) > 0$, then

$$M_\theta\left(R, \sum_{i=0}^n c_i \int S_i\right) \leqslant 0,$$

which results in a contradiction. Therefore, there is a state s occurring in R such that $\zeta(s, \sum_{i=0}^n c_i \int S_i) > 0$.

The claim that $C(R^*) \models P$ implies $C(s) \models P$ can be proved as follows. Let $\sigma = (s, t) \in L(s)$. Let $\sigma_R = (s_1, t_1)^\wedge(s_2, t_2)^\wedge \cdots {}^\wedge(s_m, t_m) \in L(R)$ satisfying that $\theta(\sigma_R, \sum_{i=0}^n c_i \int S_i) = M_\theta(R, \sum_{i=0}^n c_i \int S_i)$. Since $M_\theta(R, \sum_{i=0}^n c_i \int S_i) > 0$ and $C(X)$ is free, there is a natural number p satisfying that

$$\theta\left(\sigma_R, \sum_{i=0}^{n} c_i \int S_i\right) p \geqslant \theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right).$$

Let

$$\sigma' = \underbrace{\sigma_R{}^\wedge\sigma_R{}^\wedge \cdots {}^\wedge\sigma_R}_{p}.$$

Then $\sigma' \in L(R^*)$. Since $\theta(\sigma', \sum_{i=0}^{n} c_i \int S_i) \geqslant \theta(\sigma, \sum_{i=0}^{n} c_i \int S_i)$, by Lemma 2, the claim holds.

The other claim that $C(s) \models P$ implies $C(R^*) \models P$ can be proved as follows. For any $\sigma \in L(R^*)$, let $t = \max(\theta(\sigma, \sum_{i=0}^{n} c_i \int S_i)/\zeta(s, \sum_{i=0}^{n} c_i \int S_i) + 1, 1)$. Then $\sigma' = (s, t) \in L(s)$. Since $\theta(\sigma', \sum_{i=0}^{n} c_i \int S_i) \geqslant \theta(\sigma, \sum_{i=0}^{n} c_i \int S_i)$, by Lemma 2, the claim holds. □

Theorem 4. *Let $P = \sum_{i=1}^{n} c_i \int S_i \leqslant M$ be a linear duration property, R be a nonzero-simple DRE satisfying that $M_\theta(R, \sum_{i=0}^{n} c_i \int S_i) \leqslant 0$, and $C(X)$ be a free context. Then $C(R^*) \models P$ if and only if $C(\varepsilon) \models P$.*

Proof. By Definition 5, $L(R^*) \supseteq L(\varepsilon)$ holds, which by Lemma 1 implies a half of the claim, i.e. $C(R^*) \models P$ implies $C(\varepsilon) \models P$. The other half is straightforward from Lemma 3. □

Theorem 5. *Let R be a nonzero-simple DRE, P be a linear duration property, and $C(X)$ be a bounded context for R. Then $C(R^*) \models P$ if and only if $C(\bigoplus_{j=0}^{p} R^j) \models P$, where $p = \lfloor \omega(C(X), R) \rfloor + 1$.*

Proof. One half of the claim, i.e. $C(R^*) \models P$ implies $C(\bigoplus_{j=0}^{p} R^j) \models P$ is exactly the same as the proof of Theorem 4. The other half of the claim is a direct consequence of Lemma 4. □

5.4. Model-checking algorithm

Let R be a DRE representing the behaviour of a positive loop-closed automaton, and R_1 be a sub-expression of R which is an nonzero-simple DRE. Replacing an occurrence of R_1^* in R with X, we get a context $C(X)$. Since any positive loop-closed automaton satisfies that any nonzero loop is reducible, either $C(X)$ is bounded for R or $C(X)$ is free. So based on the above theorems, we can develop an algorithm to check if a positive loop-closed automaton H satisfies a linear duration property $P = \sum_{i=0}^{n} c_i \int S_i \leqslant M$ as follows.

Step 0. Construct a DRE R to represent the behaviour of H, and let $R' := R$.

Step 1. For R', distributing $^\wedge$ over $\oplus$, and Δ over $\oplus$, we obtain Q.

Step 2. Finding a sub-expression Q_S of Q which has one of the following four forms:
(1) $Q_S = (R_1 \oplus R_2 \oplus \cdots \oplus R_k)^*$ $(k \geqslant 2)$, where every R_i $(1 \leqslant i \leqslant m)$ is a simple DRE.
(2) $Q_S = R_1^*$, where R_1 is a nonzero-simple DRE.

(3) $Q_S = R_1^*$, where R_1 is a zero-simple DRE.
(4) $Q_S = R_1^*$, where R_1 is a simple DRE satisfying that $L(R_1) = \emptyset$.

If such Q_S cannot be found, goto Step 7 (note that it is not difficult to prove that if we cannot find out such a Q_S, then Q is a normal form); otherwise replacing the occurrence of Q_S in Q with X, we get a context $C_Q(X)$ such that $Q = C_Q(Q_S)$. Then, if Q_S has the first form, goto Step 3; if Q_S has second form, goto Step 4; if Q_S has the third form, goto Step 5; if Q_S has the fourth form, goto Step 6.

Step 3. By Theorem 1, we transform Q into $Q' = C_Q((R_1)^{*\wedge}(R_2)^{*\wedge} \cdots {}^{\wedge}(R_m)^*)$. Thus, let $R' := Q'$, and goto Step 1.

Step 4. If $C_Q(X)$ is bounded for R_1, then by Theorem 5 we transform Q into $Q' = C_Q(\bigoplus_{j=0}^{p} R_1^j)$, where p is defined in Theorem 5. Therefore, let $R' := Q'$, and goto Step 1. Otherwise, $C_Q(X)$ is free. If $M_\theta(R_1, \sum_{i=1}^{n} c_i \int S_i) > 0$, then by Theorem 3, we transform Q into $Q' = C_Q(s)$ where s is a state defined in Theorem 3. Let $R' := Q'$, and goto Step 1. Otherwise, by Theorem 4, we transform Q into $Q' = C_Q(\varepsilon)$. Let $R' := Q'$, and goto Step 1.

Step 5. By Theorem 2, we transform Q into $Q' = C_Q(R_1')$, where R_1' is the simple DRE defined in Theorem 2. Let $R' := Q'$, and goto Step 1.

Step 6. Since $L(R_1) = \emptyset$, let $R' := C_Q(\varepsilon)$ and goto Step 1.

Step 7. Since Q is a normal form now, we check $Q \models P$ by linear programming. If $Q \models P$, then $R \models P$, i.e. H satisfies P; otherwise $R \not\models P$, i.e. H does not satisfy P.

5.5. Algorithm complexity

The above algorithm is based on linear programming. The linear programming problem has been well-studied, and can be solved with a polynomial-time algorithm in general. Indeed many software packages have been developed to efficiently find solutions for linear programs. In the algorithm, sometimes we need to unfold the combinator $*$ (loop) a finite number of times (shown in Theorem 5). Each iteration will make the linear programming problem larger and hence this is the main source of complexity of the algorithm.

We have discovered that for a subclass of positive loop-closed automata, the satisfaction problem for linear duration properties can be solved efficiently without unfolding loops. We call this class of linear hybrid automata by *zero loop-closed automata*, which is described in [18]. A *zero loop-closed automaton* is a positive loop-closed automaton in which any nonzero loop is not constrained by any variable constraint outside the loop. In this case, we do not need to use Theorem 5 so that we can use another approach to solving problem efficiently, which is to traverse all the simple paths in an automaton and checking their corresponding sequences of locations for a given linear duration property.

6. Related work and conclusion

In this paper, we have shown that for a class of linear hybrid automata called positive loop-closed automata, the satisfaction problem for linear duration properties can be solved by linear programming. We extend the traditional regular expressions with duration constraints and use them as a language to describe the behaviour of this class of linear hybrid automata. The extended notation is called duration-constrained regular expressions. Based on this formalism, we show that the model-checking problem can be reduced formally to linear programs.

In general the model checking problem is undecidable for the class of linear hybrid automata. This paper gives a new result for the decidability of the model checking problem because the class of positive loop-closed automata is not contained by the decidable classes of hybrid systems we have found in the literature so far. In [5], the decidability of a class of linear hybrid systems called *integration graphs* is reduced to the verification problem for timed automata [10]. In integration graphs, it is not allowed to test a variable in a loop which has different change rate in different locations. In [6], a class of hybrid automata, *initialized rectangular automata*, are proved to be decidable for linear temporal logic (LTL) requirements. A symbolic method is presented in [7] such that the tool HYTECH [8] which runs a symbolic procedure can terminate on initialized rectangular automata. Any initialized rectangular automaton requires that any variable must be reset when its change rate is changed. In [3], an automatic approach, which attempt to construct the reachable region by symbolic execution, has been presented. But the procedures often do not terminate. In [11–13], several approaches to verifying hybrid systems are presented, but they do not result in any decidable class of hybrid systems.

The idea to check linear duration properties by linear programming comes from [4] in which the problem for real-time automata is solved by linear programming technique, which is well established. By developing the techniques in [4], we show in [19,20] that by linear programming technique the problem can be solved totally for a class of linear hybrid automata which is included by the class of positive loop-closed automata. In [19,20], we describe the decidable class of linear hybrid automata by using an extension of regular expressions with time constraints, but do not give any direct definition of the decidable hybrid automata. In [18], we show that for a subclass of positive loop-closed automata, the problem can be solved efficiently based on depth-first search method. In [5] the problem for timed automata has been solved by mixed integer/linear programming techniques. In [16,17], a integer time verification technique is developed for solving the problem for timed automata. In [9], an algorithm has been developed for checking duration-bounded reachability which asks whether there is a behaviour of an automaton from a start state to a target state, such that the accumulated duration along the behaviour satisfies a constraint. In that paper, the coefficients corresponding to the state durations are restricted to nonnegative integers.

We have developed a model checker based on the result presented in this paper, which accepts a linear hybrid automaton, expresses its behaviour with a duration-constrained regular expression if it is a positive loop-closed automaton, and check it for a given linear duration property. The tool is implemented in Java, and the linear programming software package which is integrated in the tool is from OR_Objects of DRA Systems which is a free collection of Java classes for developing operations research, scientific and engineering applications (http://OpsResearch.com/OR-Objects/index.html).

For real systems, the condition of positive loop-closedness is rational. For example, for a control system, the loop-closedness means that every repetition of a control process starts from the same control conditions. Furthermore, in most case, a repetition of a control process takes time, and a task containing the repetitions of the control process need to be finished in a given time (as a nonzero loop is constrained by a positive constraint). So we think that there are a number of real systems that satisfy the condition of positive loop-closedness and that can thus be modeled by positive loop-closed automata. An important topic for future work is to do case studies in practical use.

Appendix A

A.1. Algorithm to check if a linear hybrid automaton is positive loop-closed

An efficient algorithm is described in Fig. 4, which is to check if a linear hybrid automaton $(X, V, E, v_I, \alpha, \beta)$ is positive loop-closed. The algorithm is based on depth-first search method. The main data structure in the algorithm includes a list *currentpath* of locations which is used to record the current paths, and a set *loopset* of loops which records all the loops in the automaton. The algorithm consists of three steps. First, we find out all loops, and check if any simple path is such that any loop satisfies that any variable constraint inside the loop is not related to any transition outside the loop. Then we check if any loop is such that any other loop with the same loop-start node satisfies that any variable constraint inside the loop is not related to any transition outside the loop. Last, for any loop, from the loop-start node we traverse all simple path segment to check if any simple path segment satisfies that for any variable constraint outside a loop is not related to any transition inside the loop; and if any bounded simple path segment is such that any nonzero loop is reducible. The complexity of the algorithm is proportional to the number of the simple paths and the size of the longest simple path in an automaton.

Appendix B

B.1. Proof of lemmas

In this section, we present the proof of Lemmas 1–4. These lemmas will be proved by induction on the structure of context. Since the proof for the structures $R \oplus C_1(X)$ and $R^\wedge C_1(X)$ is similar to the one of the structures $C_1(X) \oplus R$ and $C_1(X)^\wedge R$, it is left for the reader.

Lemma 1. *Let R and R' be DREs, $C(X)$ be a context, and P be a linear duration property. If for any $\sigma \in L(R)$, there is $\sigma' \in L(R')$ satisfying that*

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma', \sum_{i=0}^{n} c_i \int S_i\right)$$

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$, then $C(R') \models P$ implies $C(R) \models P$.

Proof. The lemma follows immediately from the following claim: if for any $\sigma_1 \in L(R)$, there is $\sigma_1' \in L(R')$ such that

$$\theta\left(\sigma_1, \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma_1', \sum_{i=0}^{n} c_i \int S_i\right)$$

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$, then for any $\sigma \in L(C(R))$, there is $\sigma' \in L(C(R'))$ such that

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma', \sum_{i=0}^{n} c_i \int S_i\right)$$

```
$currentpath := \langle v_I \rangle$; $loopset := \emptyset$;
repeat
  $node$ := the last node of $currentpath$;
  if $node$ has no new successive node
  then delete the last node of $currentpath$
  else begin
      $node$ := a new successive node of $node$;
      if $node$ is in $currentpath$ (we discover a loop)
      then begin put the loop into $loopset$;
                 check if the current path is such that any variable
                 constraint inside the loop is not related to
                 any transition outside the loop;
                 if no (the loop is not closed), then return false;
           end
      else append $node$ to $currentpath$; end
until $currentpath = \langle\rangle$;
```

```
for any loop in $loopset$ do
  begin check if the loop is such that any other loop with the same
      loop-start node satisfies that any variable constraint inside
      the loop is not related to any transition outside the loop;
      if no, then return false;
  end;
```

```
for any loop in $loopset$ with a loop-start node $v$ do
  begin $currentpath := \langle v \rangle$;
   repeat
     $node$ := the last node of $currentpath$;
     if $node$ has no new successive node
     then begin check if the current path is such that any nonzero
                loop is reducible; if no, then return false;
                delete the last node of $currentpath$; end
     else begin $node$ := a new successive node of $node$;
               check if the current path satisfies that any variable
               constraint outside a loop is not related to any
               transition inside the loop;
               if no, then return false;
               if $node$ is not in $currentpath$
               then append $node$ to $currentpath$; end
   until $currentpath = \langle\rangle$;
  end;
return true.
```

Fig. 4. Algorithm for checking if a linear hybrid automaton is zero loop-closed.

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$. We prove the claim by induction on the structure of context.

- *Basic case*: Let $C(X) = X$. Then $C(R) = R$ and $C(R') = R'$. By assumption, the basic case holds.
- *Induction step*: Assume that the claim holds for a context $C_1(X)$, and let $C(X)$ be defined from $C_1(X)$.

(1) Let $C(X) = C_1(X) \oplus R_1$. Then

$$C(R) = C_1(R) \oplus R_1 \quad \text{and} \quad C(R') = C_1(R') \oplus R_1.$$

Since, by Definition 5,

$$L(C(R)) = L(C_1(R)) \cup L(R_1) \quad \text{and} \quad L(C(R')) = L(C_1(R')) \cup L(R_1),$$

by the inductive hypothesis, the claim holds.

(2) Let $C(X) = C_1(X)^\wedge R_1$. Then

$$C(R) = C_1(R)^\wedge R_1 \quad \text{and} \quad C(R') = C_1(R')^\wedge R_1.$$

For any $\sigma = \sigma_1{}^\wedge\sigma_{R_1} \in L(C(R))$, where $\sigma_1 \in L(C_1(R))$ and $\sigma_{R1} \in L(R_1)$, by the inductive hypothesis, there is $\sigma_1' \in L(C_1(R'))$ such that

$$\theta\left(\sigma_1, \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma_1', \sum_{i=0}^{n} c_i \int S_i\right)$$

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$. Let $\sigma' = \sigma_1'{}^\wedge\sigma_{R_1}$. It follows that $\sigma' \in L(C(R'))$. Since

$$\theta\left(\sigma', \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma_1', \sum_{i=0}^{n} c_i \int S_i\right) + \theta\left(\sigma_{R_1}, \sum_{i=0}^{n} c_i \int S_i\right)$$

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$, we have

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma', \sum_{i=0}^{n} c_i \int S_i\right)$$

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$, i.e. the claim holds.

(3) Let $C(X) = C_1(X)^*$. Then $C(R) = C_1(R)^*$ and $C(R') = C_1(R')^*$. For any

$$\sigma = \sigma_1{}^\wedge\sigma_2{}^\wedge \cdots {}^\wedge\sigma_m \in L(C(R)),$$

where each $\sigma_i \in L(C_1(R))$ $(1 \leqslant i \leqslant m)$, by the inductive hypothesis, for each i $(1 \leqslant i \leqslant m)$, there is $\sigma_i' \in L(C_1(R'))$ such that

$$\theta\left(\sigma_i, \sum_{j=0}^{n} c_j \int S_j\right) = \theta\left(\sigma_i', \sum_{j=0}^{n} c_j \int S_j\right)$$

for any integrated duration $\sum_{j=0}^{n} c_j \int S_j$. Let
$\sigma' = \sigma_1'{}^\wedge\sigma_2'{}^\wedge \cdots {}^\wedge\sigma_m'$. It follows that $\sigma' \in L(C(R'))$. Similarly the previous case, we have

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma', \sum_{i=0}^{n} c_i \int S_i\right)$$

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$, i.e. the claim holds.

(4) Let $C(X) = (C_1(X), \Delta)$. Then

$C(R) = (C_1(R), \Delta)$ and $C(R') = (C_1(R'), \Delta)$.

By Definition 5, any $\sigma \in L(C(R))$ is in $L(C_1(R))$ and satisfies any linear duration constraint in Δ, any $\sigma' \in L(C(R'))$ is in $L(C_1(R'))$ and satisfies any linear duration constraint in Δ. Let σ is in $L(C_1(R))$. By the inductive hypothesis, there is $\sigma' \in L(C_1(R'))$ such that

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right) = \theta\left(\sigma', \sum_{i=0}^{n} c_i \int S_i\right)$$

for any integrated duration $\sum_{i=0}^{n} c_i \int S_i$. It follows that if σ is in $L(C(R))$, σ' is in $L(C(R'))$, which implies the claim holds. □

Lemma 2. *Let R and R' be nonzero-simple DREs, $P = \sum_{i=0}^{n} c_i \int S_i \leqslant M$ be a linear duration property, and $C(X)$ be a free context. If for any $\sigma \in L(R)$, there is $\sigma' \in L(R')$ satisfying that $\theta(\sigma, \sum_{i=0}^{n} c_i \int S_i) \leqslant \theta(\sigma', \sum_{i=0}^{n} c_i \int S_i)$, then $C(R') \models P$ implies $C(R) \models P$.*

Proof. The lemma follows immediately from the following claim: If for any $\sigma_1 \in L(R)$, there is $\sigma_1' \in L(R')$ satisfying that $\theta(\sigma_1, \sum_{i=0}^{n} c_i \int S_i) \leqslant \theta(\sigma_1', \sum_{i=0}^{n} c_i \int S_i)$, then for any $\sigma \in L(C(R))$, there is $\sigma' \in L(C(R))$ satisfying that

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i \int S_i\right) \leqslant \theta\left(\sigma', \sum_{i=0}^{n} c_i \int S_i\right).$$

We prove this claim by induction on the structure of context.

- *Basic case*: Let $C(X) = X$. Then $C(R) = R$ and $C(R') = R'$. Bu assumption, the basic case holds.
- *Induction step*: Assume that the claim holds for a context $C_1(X)$, and let $C(X)$ be defined from $C_1(X)$.

(1) Let $C(X) = C_1(X) \oplus R_1$. Then

$$C(R) = C_1(R) \oplus R_1 \quad \text{and} \quad C(R') = C_1(R') \oplus R_1.$$

Since, by Definition 5,

$$L(C(R)) = L(C_1(R)) \cup L(R_1) \quad \text{and} \quad L(C(R')) = L(C_1(R')) \cup L(R_1),$$

by the inductive hypothesis, the claim holds.

(2) Let $C(X) = C_1(X)^\wedge R_1$. Then

$$C(R) = C_1(R)^\wedge R_1 \quad \text{and} \quad C(R') = C_1(R')^\wedge R_1.$$

From Definition 5, it follows that any $\sigma \in L(C(R))$ is of the form $\sigma_1{}^\wedge\sigma_{R_1}$, where $\sigma_1 \in L(C_1(R))$ and $\sigma_{R1} \in L(R_1)$. By the inductive hypothesis, there is $\sigma_1' \in L(C_1(R'))$ satisfying that

$$\theta\left(\sigma_1, \sum_{i=0}^{n} c_i \int S_i\right) \leqslant \theta\left(\sigma_1', \sum_{i=0}^{n} c_i \int S_i\right).$$

Let $\sigma' = \sigma_1'{}^\wedge\sigma_{R_1}$. It follows that $\sigma' \in L(C(R'))$. Since for any linear duration constraint $a \leqslant \sum_{i=0}^{n} c_i' \int S_i' \leqslant b$,

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i' \int S_i'\right) = \theta\left(\sigma_1, \sum_{i=0}^{n} c_i' \int S_i'\right) + \theta\left(\sigma_R, \sum_{i=0}^{n} c_i' \int S_i'\right)$$

and

$$\theta\left(\sigma', \sum_{i=0}^{n} c_i' \int S_i'\right) = \theta\left(\sigma_1', \sum_{i=0}^{n} c_i' \int S_i'\right) + \theta\left(\sigma_R, \sum_{i=0}^{n} c_i' \int S_i'\right),$$

we have $\theta(\sigma, \sum_{i=0}^{n} c_i \int S_i) \leqslant \theta(\sigma', \sum_{i=0}^{n} c_i \int S_i)$, i.e. the claim holds.

(3) Let $C(X) = C_1(X)^*$. Then

$$C(R) = C_1(R)^* \quad \text{and} \quad rC(R') = C_1(R')^*.$$

By Definition 5, any $\sigma \in L(C(R))$ is of the form $\sigma_1{}^\wedge\sigma_2{}^\wedge \cdots {}^\wedge\sigma_m$, where $\sigma_i \in L(C_1(R))$ for each i $(1 \leqslant i \leqslant m)$. By the inductive hypothesis, for each i $(1 \leqslant i \leqslant m)$, there is $\sigma_i' \in L(C_1(R'))$ such that

$$\theta\left(\sigma_i, \sum_{i=0}^{n} c_i \int S_i\right) \leqslant \theta\left(\sigma_i', \sum_{i=0}^{n} c_i \int S_i\right)$$

Let $\sigma' = \sigma_1'{}^\wedge\sigma_2'{}^\wedge \cdots {}^\wedge\sigma_m'$. It follows that $\sigma' \in L(C(R'))$. Similarly the previous case, we have $\theta(\sigma, \sum_{i=0}^{n} c_i \int S_i) \leqslant \theta(\sigma', \sum_{i=0}^{n} c_i \int S_i)$, i.e. the claim holds.

(4) Let $C(X) = (C_1(X), \Delta)$. Then

$$C(R) = (C_1(R), \Delta) \quad \text{and} \quad C(R') = (C_1(R), \Delta).$$

Since $C(X)$ is free, $\Delta = \emptyset$. It follows that $L(C(R)) = L(C_1(R))$ and $L(C(R')) = L(C_1(R'))$. By the inductive hypothesis, the claim holds. □

Lemma 3. *Let $P = \sum_{i=1}^{n} c_i \int S_i \leqslant M$ be a linear duration property, R be a nonzero-simple DRE such that $M_\theta(R, \sum_{i=0}^{n} c_i \int S_i) \leqslant 0$, and $C(X)$ be a free context. Then for any $\sigma \in L(C(R^*))$, there is $\sigma' \in L(C(\varepsilon))$ satisfying that $\theta(\sigma, \sum_{i=1}^{n} c_i \int S_i) \leqslant \theta(\sigma', \sum_{i=1}^{n} c_i \int S_i)$.*

Proof. We prove the claim by induction on the structure of context.

- *Basic case*: Let $C(X) = X$. Then $C(R^*) = R^*$ and $C(\varepsilon) = \varepsilon$. Since $M_\theta(R, \sum_{i=0}^{n} c_i \int S_i) \leqslant 0$, the basic case holds.
- *Induction step*: Assume that the claim holds for a context $C_1(X)$, and let $C(X)$ be defined from $C_1(X)$.

(1) Let $C(X) = C_1(X) \oplus R_1$. Then

$$C(R^*) = C_1(R^*) \oplus R_1 \text{ and } C(\varepsilon) = C_1(\varepsilon) \oplus R_1.$$

Since, by Definition 5, $L(C(R^*)) = L(C_1(R^*)) \cup L(R_1)$, by the inductive hypothesis, the claim holds.

(2) Let $C(X) = C_1(X)^\wedge R_1$. Then $C(R^*)) = C_1(R^*)^\wedge R_1$, and $C(\varepsilon) = C_1(\varepsilon)^\wedge R_1$. From Definition 5, it follows that any $\sigma \in L(C(R^*))$ is of the form $\sigma_1{}^\wedge\sigma_{R_1}$, where $\sigma_1 \in$

$L(C_1(R^*))$ and $\sigma_{R1} \in L(R_1)$. Let By the inductive hypothesis, there is $\sigma_1' \in L(C_1(\varepsilon))$ satisfying that

$$\theta\left(\sigma_1, \sum_{i=0}^{n} c_i \int S_i\right) \leqslant \theta\left(\sigma_1', \sum_{i=0}^{n} c_i \int S_i\right).$$

Let $\sigma' = \sigma_1'{}^\wedge\sigma_{R_1}$. it follows that $\sigma' \in L(C(\varepsilon))$. Since for any linear duration constraint $a \leqslant \sum_{i=0}^{n} c_i' \int S_i' \leqslant b$,

$$\theta\left(\sigma, \sum_{i=0}^{n} c_i' \int S_i'\right) = \theta\left(\sigma_1, \sum_{i=0}^{n} c_i' \int S_i'\right) + \theta\left(\sigma_R, \sum_{i=0}^{n} c_i' \int S_i'\right)$$

and

$$\theta\left(\sigma', \sum_{i=0}^{n} c_i' \int S_i'\right) = \theta\left(\sigma_1', \sum_{i=0}^{n} c_i' \int S_i'\right) + \theta\left(\sigma_R, \sum_{i=0}^{n} c_i' \int S_i'\right),$$

we have $\theta(\sigma, \sum_{i=0}^{n} c_i \int S_i) \leqslant \theta(\sigma', \sum_{i=0}^{n} c_i \int S_i)$, i.e. the claim holds.

(3) Let $C(X) = C_1(X)^*$. Then

$$C(R^*)) = C_1(R^*)^* \text{ and } C(\varepsilon) = C_1(\varepsilon)^*.$$

By Definition 5, any $\sigma \in L(C(R^*))$ is of the form $\sigma_1{}^\wedge\sigma_2{}^\wedge \cdots {}^\wedge\sigma_m$, where $\sigma_i \in L(C_1(R^*))$ for each i $(1 \leqslant i \leqslant m)$. By the inductive hypothesis, for each i $(1 \leqslant i \leqslant m)$, there is $\sigma_i' \in L(C_1(\varepsilon))$ satisfying that

$$\theta\left(\sigma_i, \sum_{i=0}^{n} c_i \int S_i\right) \leqslant \theta\left(\sigma_i', \sum_{i=0}^{n} c_i \int S_i\right).$$

Let $\sigma' = \sigma_1'{}^\wedge\sigma_2'{}^\wedge \cdots {}^\wedge\sigma_m'$. It follows that $\sigma' \in L(C(\varepsilon))$. Similarly the previous case, we have $\theta(\sigma, \sum_{i=0}^{n} c_i \int S_i) \leqslant \theta(\sigma', \sum_{i=0}^{n} c_i \int S_i)$, i.e. the claim holds.

(4) Let $C(X) = (C_1(X), \Delta)$. Then

$$C(R^*) = (C_1(R^*), \Delta) \text{ and } C(\varepsilon) = (C_1(\varepsilon), \Delta).$$

Since $C(X)$ is free, $\Delta = \emptyset$. It follows that $L(C(R^*)) = L(C_1(R^*))$ and $L(C(\varepsilon)) = L(C_1(\varepsilon))$. By the inductive hypothesis, the claim holds. □

Lemma 4. *Let R be a nonzero-simple DRE, and $C(X)$ be a bounded context for R. Then $L(C(\bigoplus_{j=0}^{p} R^j)) \supseteq L(C(R^*))$, where $p = \lfloor\omega(C(X), R)\rfloor + 1$.*

Proof. We will prove the following more general claim: Let R be a nonzero-simple DRE, and $(C(X), \Delta)$ be a bounded context for R. Then, for any

$$\sigma \in L((C(R^*), \Delta)), \quad \sigma \in L\left(\left(C\left(\bigoplus_{j=0}^{p} R^j\right), \Delta\right)\right),$$

where $p = \lfloor\omega((C(X), \Delta), R)\rfloor + 1$. It is clear that if we let $\Delta = \emptyset$, we get the lemma. In the following, we prove this claim by induction on the structure of context.

- *Basic case*: Let $C(X) = X$. Then $(C(R^*), \Delta) = (R^*, \Delta)$ and

  $$\left(C\left(\bigoplus_{j=0}^{p} R^j\right), \Delta\right) = \left(\bigoplus_{j=0}^{p} R^j, \Delta\right).$$

  By Definition 5, any $\sigma \in L((R^*, \Delta))$ is of the form $\sigma_1{}^\wedge\sigma_2{}^\wedge \cdots {}^\wedge\sigma_m$ where $\sigma_i \in L(R)$ for any i $(1 \leqslant i \leqslant m)$. Since $\omega((C(X), \Delta), R)$ is the minimal value of the set

  $$\left\{ b/m_\theta\left(R, \sum_{i=1}^{n} c_i \int S_i\right) \middle|\, a \leqslant \sum_{i=1}^{n} c_i \int S_i \leqslant b \in \Delta \text{ is positive for } R \right\}$$

  if $m > p$, then there must be a linear duration constraint in Δ which is not satisfied by σ. Thus, $m \leqslant p$. It follows that $\sigma \in L((\bigoplus_{j=0}^{p} R^j, \Delta))$, which implies the basic case holds.
- *Induction step*: Assume that the claim holds for a context $C_1(X)$, and let $C(X)$ be defined from $C_1(X)$.

(1) Let $C(X) = C_1(X) \oplus R_1$. Then $(C(R^*), \Delta) = (C_1(R^*) \oplus R_1, \Delta)$, and

$$\left(C\left(\bigoplus_{j=0}^{p} R^j\right), \Delta\right) = \left(C_1\left(\bigoplus_{j=0}^{p} R^j\right) \oplus R_1, \Delta\right).$$

By Definition 6, $\Gamma((C(X), \Delta)) = \Gamma((C_1(X), \Delta))$. It follows that

$$\omega((C(X), \Delta), R) = \omega((C_1(X), \Delta), R).$$

Since $(C(X), \Delta)$ be a bounded context for R, $(C_1(X), \Delta)$ be a bounded context for R. Since, by Definition 5,

$$L((C(R^*), \Delta)) = L((C_1(R^*), \Delta)) \cup L((R_1, \Delta)),$$

by the inductive hypothesis, the claim holds.

(2) Let $C(X) = C_1(X)^\wedge R_1$. Then $(C(R^*), \Delta) = (C_1(R^*)^\wedge R_1, \Delta)$, and

$$\left(C\left(\bigoplus_{j=0}^{p} R^j\right), \Delta\right) = \left(C_1\left(\bigoplus_{j=0}^{p} R^j\right)^\wedge R_1, \Delta\right).$$

By Definition 6, $\Gamma((C(X), \Delta)) = \Gamma((C_1(X), \Delta))$. It follows that

$$\omega((C(X), \Delta), R) = \omega((C_1(X), \Delta), R).$$

By Definition 5, any $\sigma \in L((C(R^*), \Delta))$ is of the form $\sigma_1{}^\wedge\sigma_{R_1}$, where $\sigma_1 \in L(C_1 (R^*))$ and $\sigma_{R1} \in L(R_1)$. Let

$$\Delta' = \left\{ \min\left(a, \theta\left(\sigma_1, \sum_{i=1}^{n} c_i \int S_i\right)\right) \leqslant \sum_{i=1}^{n} c_i \int S_i \leqslant b \,\middle|\, a \leqslant \sum_{i=1}^{n} c_i \int S_i \leqslant b \in \Delta \right\}.$$

It follows that $\sigma_1 \in L((C_1(R^*), \Delta'))$. Since $(C(X), \Delta)$ be a bounded context for R, $(C_1(X), \Delta')$ be a bounded context for R. By the inductive hypothesis, $\sigma_1 \in L((C_1(\bigoplus_{j=0}^{p_1} R^j), \Delta'))$, where $p_1 = \lfloor \omega((C_1(X), \Delta'), R) \rfloor + 1$. Since $p_1 \leqslant p$, we have $\sigma_1 \in L((C_1(\bigoplus_{j=0}^{p} R^j), \Delta'))$. It follows that

$$\sigma \in L\left(\left(C\left(\bigoplus_{j=0}^{p} R^j\right), \Delta\right)\right),$$

i.e. the claim holds.

(3) Let $C(X) = C_1(X)^*$. Then

$$(C(R^*), \Delta) = (C_1(R^*)^*, \Delta) \quad \text{and}$$
$$\left(C\left(\bigoplus_{j=0}^{p} R^j\right), \Delta\right) = \left(C_1\left(\bigoplus_{j=0}^{p} R^j\right)^*, \Delta\right).$$

By Definition 6, $\Gamma((C(X), \Delta)) = \Gamma((C_1(X), \Delta))$. It follows that

$$\omega((C(X), \Delta), R) = \omega((C_1(X), \Delta), R).$$

By Definition 5, any $\sigma \in L((C(R^*), \Delta))$ is of the form $\sigma_1{}^\frown\sigma_2{}^\frown \cdots {}^\frown\sigma_m$, where $\sigma_i \in L(C_1(R^*))$ for each i $(1 \leqslant i \leqslant m)$. For each i $(1 \leqslant i \leqslant m)$, let $\Delta_i = \{\min(a, \theta(\sigma_i, \sum_{i=1}^n c_i \int S_i)) \leqslant \sum_{i=1}^n c_i \int S_i \leqslant b \mid a \leqslant \sum_{i=1}^n c_i \int S_i \leqslant b \in \Delta\}$. It follows that $\sigma_i \in L((C_1(R^*), \Delta_i)$ for any i $(1 \leqslant i \leqslant m)$. Since $(C(X), \Delta)$ is a bounded context for R, $(C_1(X), \Delta_i)$ is a bounded context for R. By the inductive hypothesis, for each i $(1 \leqslant i \leqslant m)$, $\sigma_i \in L((C_1(\bigoplus_{j=0}^{p_i} R^j), \Delta_i))$, where $p_i = \lfloor \omega((C_1(X), \Delta_i'), R) \rfloor + 1$. Since $p_i \leqslant p$ for any i $(1 \leqslant i \leqslant m)$, $\sigma_i \in L((C_1(\bigoplus_{j=0}^{p} R^j), \Delta_i'))$. It follows that $\sigma \in L((C(\bigoplus_{j=0}^{p} R^j), \Delta))$ i.e. the claim holds.

(4) Let $C(X) = (C_1(X), \Delta_1)$. Then

$$(C(X), \Delta) = ((C_1(X), \Delta_1), \Delta) = (C_1(X), \Delta_1 \cup \Delta).$$

It follows that $\omega((C(X), \Delta), R) = \omega((C_1(X), \Delta_1 \cup \Delta), R)$. Since $(C(X), \Delta)$ is a bounded context for R, $(C_1(X), \Delta_1 \cup \Delta)$ is a bounded context for R. By the inductive hypothesis, the claim holds. □

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China (Nos. 60073031 and 69703009), Jiangsu Province Research Project (No. BK2001033), and by International Institute for Software Technology, United Nations University (UNU/IIST).

References

[1] Zhou Chaochen, C.A.R. Hoare, A.P. Ravn, A calculus of durations, Information Processing Letter 40 (5) (1991) 269–276.

[2] T.A. Henzinger, The theory of hybrid automata, in: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996), 1996, pp. 278–292.

[3] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, The algorithmic analysis of hybrid systems, Theoretical Computer Science 138 (1995) 3–34.

[4] Zhou Chaochen, Zhang Jinzhong, Yang Lu, Li Xiaoshan, Linear duration invariants, in: Formal Techniques in Real-Time and Fault-Tolerant Systems, Lecture Notes in Computer Science, vol. 863, Springer, Berlin, 1994, pp. 88–109.

[5] Y. Kesten, A. Pnueli, J. Sifakis, S. Yovine, Decidable integration graphs, Information and Computation 150 (2) (1999) 209–243.

[6] T.A. Henzinger, P.W. Kopke, Anuj Puri, Pravin Varaiya, What's decidable about hybrid automata? Journal of Computer and System Sciences 57 (1998) 94–124.

[7] T.A. Henzinger, Rupak Majumdar, Symbolic model checking for rectangular hybrid systems, in: Proceedings of the Sixth Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 00), Lecture Notes in Computer Science, Springer, Berlin, 2000.

[8] T.A. Henzinger, P.-H. Ho, H. Wong-Toi, HYTECH: a model checker for hybrid systems, Software Tools for Technology Transfer 1 (1997) 110–122.

[9] R. Alur, C. Courcoubetis, T.A. Henzinger, Computing accumulated delays in real-time systmes, in: Proceedings of the Fifth Conference on Computer-Aided Verification, Lecture Notes in Computer Science, vol. 818, Springer, Berlin, 1993, pp. 181–193.

[10] R. Alur, D. David, A theory of timed automata, Theoretical Computer Science 126 (1994) 183–235.

[11] T.A. Henzinger, P.-H. Ho, H. Wong-Toi, Algorithmic analysis of nonlinear hybrid systems, IEEE Transactions on Automatic Control 43 (1998) 540–554.

[12] T.A. Henzinger, Vlad Rusu, Reachability verification for hybrid automata, in: Proceedings of the First International Workshop on Hybrid Systems: Computation and Control (HSCC 1998), Lecture Notes in Computer Science, vol. 1386, Springer, Berlin, 1998, pp. 190–204.

[13] N. Halbwachs, Y.-E. Proy, P. Raymond, Verification of linear hybrid systems by means of convex approximations, in: Proceedings of the International Symposium on Static Analysis, Lecture Notes in Computer Science, Springer, Berlin, 1994.

[14] G.S. Avrunin, J.C. Corbett, L.K. Dillon, Analyzing partially-implemented real-time systems, IEEE Transactions on Software Engineering 24 (8) (1998) 602–614.

[15] E. Asarin, P. Caspi, O. Maler, A Kleene theorem for timed automata, in: Proceedings of Logic in Computer Science, IEEE Computer Society, 1997, pp. 160–171.

[16] Z. Jianhua, D.V. Hung, On checking parallel real-time systems for linear duration properties, in: Formal Techniques in Real-Time and Fault-Tolerant Systems, Lecture Notes in Computer Science, vol. 1486, Springer, Berlin, 1998, pp. 241–250.

[17] Z. Jianhua, D. Van Hung, Checking timed automata for some discretisable duration properties, Research Report 145, UNU/IIST, Macau, 1998.

[18] L. Xuandong, P. Yu, Z. Jianhua, L. Yong, Z. Tao, Z. Guoliang, Efficient verification of a class of linear hybrid automata using linear programming, in: Correct Hardware Design and Verification Methods, Lecture Notes in Computer Science, vol. 2144, Springer, Berlin, 2001, pp. 465–479.

[19] L. Xuandong, D. Van Hung, Z. Tao, Checking hybrid automata for linear duration invariants, in: Advances in Computing Science—ASIAN'97, Lecture Notes in Computer Science, vol. 1345, Springer, Berlin, 1997, pp. 166–180.

[20] L. Xuandong, Z. Tao, H. Jianmin, Z. Jianhua, Z. Guoliang, Hybrid regular expressions, in: Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 1386, Springer, Berlin, 1998, pp. 384–399.