# Efficient Verification of a Class of Linear Hybrid Automata Using Linear Programming*

Li Xuandong, Pei Yu, Zhao Jianhua, Li Yong, Zheng Tao, and Zheng Guoliang

State Key Laboratory of Novel Software Technology
Department of Computer Science and Technology
Nanjing University, Nanjing
Jiangsu, P.R. China 210093
lxd@nju.edu.cn

**Abstract.** In this paper, we show that for a class of linear hybrid automata called *zero loop-closed automata*, the satisfaction problem for linear duration properties can be solved efficiently by linear programming. We give an algorithm based on depth-first search method to solve the problem by traversing all the simple paths (with no repeated node occurrence) in an automaton and checking their corresponding sequences of locations for a given linear duration property.

## 1 Introduction

The model checking problem for real-time 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 [1-4]. So an important question for the analysis and design of hybrid systems is identification of subclasses of such systems and corresponding restricted classes of analysis problems that can be settled algorithmically [2].

In this paper, we consider the problem of checking linear hybrid automata for linear duration properties. We show that for a class of linear hybrid automata called *zero loop-closed automata*, the satisfaction problem of linear duration properties can be solved efficiently by linear programming. We give an algorithm based on depth-first search method to solve the problem by traversing all the simple paths (with no repeated node occurrence) in an automaton and checking their corresponding sequences of locations for a given linear duration property.

The paper is organized as follows. In next section, we recall the notion of linear hybrid automata, and introduce linear duration properties. In Sect. 3, we define *zero loop-closed automata*. Section 4 gives an efficient algorithm to check *zero loop-closed automata* for linear duration properties. The last section discusses the related work and contains some conclusion.

* This work is supported by the National Natural Science Foundation of China under Grant 60073031 and Grant 69703009, and by International Institute for Software Technology, The United Nations University (UNU/IIST).

T. Margaria and T. Melham (Eds.): CHARME 2001, LNCS 2144, pp. 465–479, 2001.

## 2 Linear Hybrid Automata and Linear Duration Properties

### 2.1 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 [1]. 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* $\mathcal{H} = (Z, X, V, E, v_I, \alpha, \beta)$, *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* $(v, \phi, \psi, v')$ *where* $v, v'$ *are in* $V$, $\phi$ *is a set of variable constraints which are of the form* $a \leq x \leq b$, *and* $\psi$ *is a set of reset actions which are of the form* $y := c$ ($x \in X$, $y \in X$, $a, b, c$ *are real numbers,* $a$ *and* $b$ *may be* $\infty$; *if* $a$ ($b$) *is* $-\infty$ ($\infty$), *then* $a \leq x \leq b$ *is taken to be* $x \leq b$ ($a \leq x$); *if* $a = b$, *then* $a \leq x \leq b$ *is taken to be* $x = a$).
- $v_I$ *is an initial location.*
- $\alpha$ *is a labeling function which maps each location in* $V$ *to a state in* $Z$.
- $\beta$ *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* $\dot{x} = a \in \beta(v)$.

□

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 two inches per second; when the pump is on, the water level rises by one inch 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 inches. A design of the monitor can be modelled by the hybrid automaton depicted in Fig. 1. The initial location of the automaton is $v_0$. The other four locations $v_1, v_2, v_3, v_4$ 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.

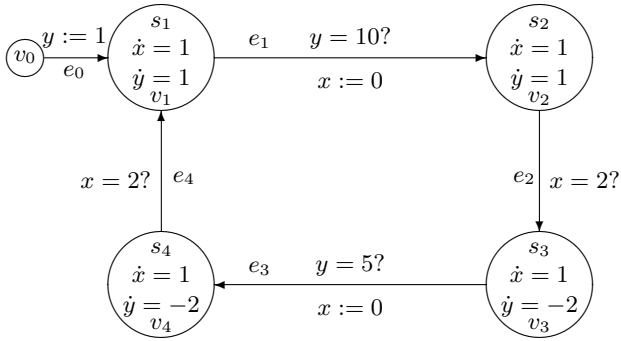


**Fig. 1.** A hybrid automaton modelling a water-level monitor

We use *sequences of locations* to represent the untimed behaviour of linear hybrid automata. A sequence of locations is of the form

$$v_0 \overset{(\phi_0,\psi_0)}{\longrightarrow} v_1 \overset{(\phi_1,\psi_1)}{\longrightarrow} \dots \overset{(\phi_m,\psi_m)}{\longrightarrow} v_{m+1}\,,$$

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 $\psi_i$ when the variable constraints in set $\phi_i$ are satisfied. For a linear hybrid automaton $\mathcal{H} = (Z, X, V, E, v_I, \alpha, \beta)$, if $v_0$ is $v_I$ and $(v_i, \phi_i, \psi_i, v_{i+1}) \in E$ for each $i$ $(0 \le i \le m)$, then the sequence of locations represents a untimed behaviour of $\mathcal{H}$.

The behaviour of linear hybrid automata can be represented by *timed sequences*. Any timed sequence is of the form $(s_1, t_1)\hat{}(s_2, t_2)\hat{}\dots\hat{}(s_m, t_m)$ where $s_i$ $(1 \le i \le m)$ is a state and $t_i$ $(1 \le i \le m)$ 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 $\mathcal{H} = (Z, X, V, E, v_I, \alpha, \beta)$, a timed sequence $(s_1, t_1)\hat{}(s_2, t_2)\hat{}\dots\hat{}(s_m, t_m)$ $(m \ge 1)$ represents a behaviour of $\mathcal{H}$ if and only if there is a untimed behaviour of the automaton*

$$v_0 \overset{(\phi_0,\psi_0)}{\longrightarrow} v_1 \overset{(\phi_1,\psi_1)}{\longrightarrow} \dots \overset{(\phi_m,\psi_m)}{\longrightarrow} v_{m+1}$$

*such that*

- *for each $i$ $(1 \le i \le m)$, $\alpha(v_i) = s_i$; and*
- *$t_1, t_2, \dots, t_m$ satisfy all the variable constraints in $\phi_i$ $(1 \le i \le m)$, i.e. for each variable constraint $a \le x \le b$ in $\phi_i$, if there is a reset action $x := c$ in $\psi_j$ $(0 \le j < i)$ and $x := d$ is not in $\psi_k$ for any $k$ $(j < k < i)$, then*

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

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

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

For a linear hybrid automaton $\mathcal{H}$, for a transition $e = (v, \phi, \psi, v')$ in $\mathcal{H}$, if $e$ is labeled with a variable constraint $a \leq x \leq b$, i.e. $a \leq x \leq b \in \phi$, 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.

### 2.2 Linear Duration Properties

Linear duration properties are linear inequalities on integrated durations of system states. Here we use Duration Calculus (DC) [5] to describe this kind of properties. 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 $\int S[a, b] \hat{=} \int_a^b S(t)dt$, where $[a, b]$ $(b \geq a)$ is a bounded interval of time. A linear duration property in DC is of the form $\sum_{i=1}^{n} c_i \int S_i \leq M$, where $S_i$s are system states and $M$, $c_i$s are real numbers.

The requirement of the water-level monitor is that the monitor must keep the water level in between 1 and 12 inches, which 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 inch per second, and when the control is in locations $v_3$ or $v_4$, the water level falls by 2 inch 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 one inch, 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

$$1 + \int s_1 + \int s_2 - 2(\int s_3 + \int s_4) \leq 12\,;$$
$$1 + \int s_1 + \int s_2 - 2(\int s_3 + \int s_4) \geq 1\,.$$

## 3 Zero Loop-Closed Automata

*Zero loop-closed automata* form a subclass of linear hybrid automata. In the following, we define this class of linear hybrid automata.

For a linear hybrid automaton $\mathcal{H} = (Z, X, V, E, v_I, \alpha, \beta)$ , a *path segment* is a sequence of locations $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$ which satisfies $(v_i, \phi_i, \psi_i, v_{i+1}) \in E$ for each $i$ $(1 \leq i \leq m-1)$. A *path* is a path segment starting at $v_I$. A path is called *simple* if all locations in the path are distinct. For a simple

path $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$, if there is $v_i$ $(1 < i \leq m)$ 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 $(v_m, \phi, \psi, v_i)$ is the *end* transition in the loop. Notice that a loop may have many different loop-enter paths. For a loop $\rho$, if $\rho_1$ is a loop-enter path of $\rho$, we say that $\rho$ *can be entered through* $\rho_1$. For example, in the automaton depicted in Fig. 2, the sequence of locations

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

is a loop, and the sequence of locations

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

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

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

is a loop, and the sequence of locations

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

is a loop-enter of the loop.

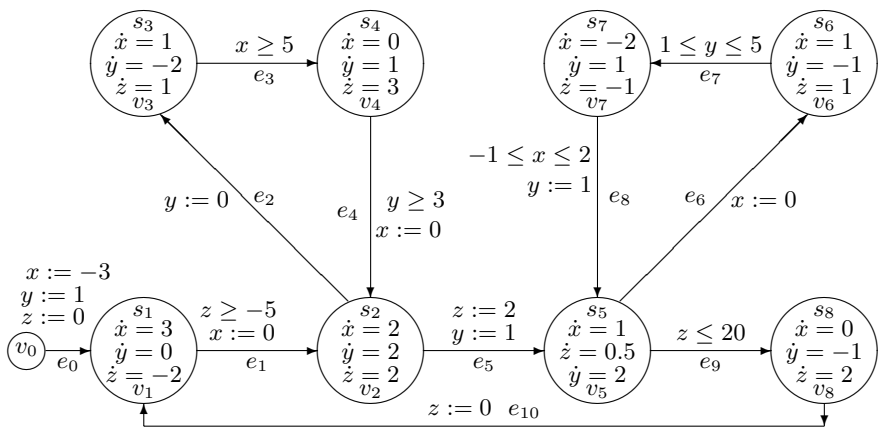


**Fig. 2.** A loop-closed automaton

Let $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \cdots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$ be a path in a linear hybrid automaton $\mathcal{H}$. For a variable constraint $a \leq x \leq b$ labeled on a transition $(v_i, \phi_i, \psi_i, v_{i+1})$ $(1 < i < m)$, its *reference point* is a transition $(v_j, \phi_j, \psi_j, v_{j+1})$ $(1 \leq j < i)$ such that

– $x$ is reset by the transition $(v_j, \phi_j, \psi_j, v_{j+1})$, and
– $x$ is not reset by any transition $(v_k, \phi_k, \psi_k, v_{k+1})$ $(j < k < i)$,

which means that for calculating the value of $x$ when the automaton stay in $v_i$ along the path in order to check if the variable constraint $a \leq x \leq b$ is satisfied, we need to refer the value of $x$ which is reset to by the transition $(v_j, \phi_j, \psi_j, v_{j+1})$. We say that the variable constraint $a \leq x \leq b$ *combines* the transitions $(v_i, \phi_i, \psi_i, v_{i+1})$ and $(v_j, \phi_j, \psi_j, v_{j+1})$.

Let $\mathcal{H}$ be a linear hybrid automaton, and $\rho$ be a loop in $\mathcal{H}$ which is of the form $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$. We say that any transition $(v_i \xrightarrow{(\phi_i,\psi_i)} v_{i+1})$ $(1 \leq i < m-1)$, which is not the end transition of $\rho$, *is inside* $\rho$. We defined that $\rho$ is *closed* if any variable constraint does not combine transition occurrences inside and outside of the loop, i.e. the following condition holds:

– for any variable constraint $a \leq x \leq b$ labeled on a transition $(v_i, \phi_i, \psi_i, v_{i+1})$ $(1 \leq i < m)$ in $\rho$, its reference point is in $\rho$, i.e., for any simple path or loop

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

such that $v_1 = u_n$, if there is no transition $(v_j, \phi_j, \psi_j, v_{j+1})$ $(1 \leq j < i)$ in $\rho$ resetting $x$, then $x$ is reset to $c$ by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ of $\rho$ 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 constraint labeled on a transition outside $\rho$, its reference point is not inside $\rho$, i.e., for any simple path segment

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

such that $v_1 = u_1$, there is no variable constraint $a \leq x \leq b$ labeled on the transition $(u_{n-1}, \phi'_{n-1}, \psi'_{n-1}, u_n)$ such that
  – $x$ is reset by a transition $(v_{i-1}, \phi_{i-1}, \psi_{i-1}, v_i)$ $(1 < i < m-1)$ inside $\rho$, and
  – $x$ is not reset by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ of $\rho$ and by any transition $(u_{k-1}, \phi'_{k-1}, \psi'_{k-1}, u_k)$ $(1 < k < n)$.

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

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

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 variable constraint $y \geq 3$ labeled on $e_4$, its reference point is $e_0$ 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.

**Definition 3.** *A linear hybrid automaton $\mathcal{H}$ is loop-closed if and only if any loop in $\mathcal{H}$ is closed.* □

For example, the automaton depicted in Fig. 2 is a loop-closed automaton. Notice that for some linear hybrid automata which are not loop-closed, we can construct loop-closed automata with the same behaviour from them. For example, the automaton depicted in Fig. 1 is not loop-closed, but we can construct a loop-closed automaton with the same behaviour, which is depicted in Fig. 3.

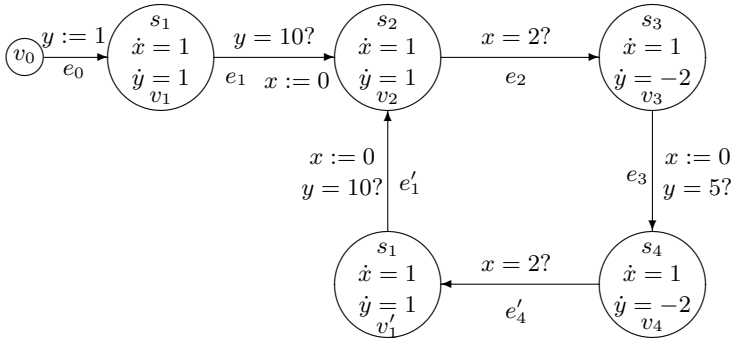


**Fig. 3.** A loop-closed automaton which has the same behaviour as the automaton in Fig. 1

For a loop-closed automaton $\mathcal{H}$, let $\rho$ be a loop in $\mathcal{H}$ 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 $\rho$ is a *zero* loop if for any variable constraint $a \le x \le b$ in $\phi_i$ $(1 \le i < m)$, $a - d \le 0$ and $b - d \ge 0$ where $d$ is the value of $x$ reset by the reference point of the variable constraint $a \le x \le b$, i.e. $d$ satisfies one of the following conditions:

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

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

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

are a zero loop, while $v_2 \xrightarrow{(\emptyset,\{y:=0\})} v_3 \xrightarrow{(\{x\ge 5\},\emptyset)} v_4 \xrightarrow{(\{y\ge 3\},\{x:=0\})} v_2$ is a nonzero loop.

For a loop-closed automaton $\mathcal{H}$, let $\rho$ be a loop in $\mathcal{H}$ 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 $\rho$ is *free* if it is not constrained

by any variable constraint outside $\rho$, i.e. for any simple path segment

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

such that $v_1 = u_1$, for any variable constraint $a \le x \le b$ labeled on the transition $(u_{n-1}, \phi'_{n-1}, \psi'_{n-1}, u_n)$, $x$ is reset by a transition $(u_i, \phi'_i, \psi'_i, u_{i+1})$ $(1 \le i < n-1)$ or by the end transition $(v_{m-1}, \phi_{m-1}, \psi_{m-1}, v_m)$ of $\rho$. For example, in the automaton depicted in Fig. 2, the nonzero loop

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

is free, but the zero loop $v_5 \xrightarrow{(\emptyset,\{x:=0\})} v_6 \xrightarrow{(\{1\le y\le 5\},\emptyset)} v_7 \xrightarrow{(\{-1\le x\le 2\},\{y:=1\})} v_5$ is not free since it is constrained by the variable constraint $z \le 20$ on the transition $e_9$.

**Definition 4.** *A zero loop-closed automaton is a loop-closed automaton in which any nonzero loop is free.* □

For example, the automata depicted in Figs. 2 and 3 are zero loop-closed automata.

Although the definition of zero loop-closed automaton is not simple, we can develop an efficient algorithm to check if a linear hybrid automaton is zero loop-closed, which is described in the appendix. Zero loop-closed automata form a decidable class of linear hybrid automata. We think there are a number of real systems that are of the loop-closed property so that some of them can be modelled by zero loop-closed automata. For example, for control systems, the loop-closed property means that every repetition of a control process starts from the same control conditions. In next section, we will show that the satisfaction problem of zero loop-closed automata for linear duration properties can be solved by linear programming.

## 4 Checking Zero Loop-Closed Automata for Linear Duration Properties

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

The satisfaction problem of linear hybrid automata for linear duration properties are defined as follows. Let $\mathcal{P} = \sum_{i=1}^{n} \int c_i S_i \le M$ be a linear duration property, and $\sigma = (s_1, t_1)^\frown(s_2, t_2)^\frown \dots ^\frown(s_m, t_m)$ be a timed sequence. The integrated duration of $S_i$ over $\sigma$ can be calculated as $\int S_i = \sum_{j\in\alpha_i} \delta_j$ where $\alpha_i = \{j \mid (0 \le j \le m) \wedge (s_j \Rightarrow S_i)\}$. Let $\theta(\sigma, \mathcal{P}) = \sum_{i=1}^{n} c_i(\sum_{j\in\alpha_i} \delta_j)$. A timed sequence $\sigma$ satisfies a linear duration property $\mathcal{P}$ if and only if $\theta(\sigma, \mathcal{P}) \le M$. A linear hybrid automaton satisfies a linear duration property if and only if every timed sequence representing its behaviour satisfies the linear duration property.

Let $\mathcal{H} = (Z, X, V, E, v_I, \alpha, \beta)$ be a linear hybrid automaton, and $\rho$ be a sequence of locations of the form $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \dots \xrightarrow{(\phi_m,\psi_m)} v_{m+1}$. We define

$\tau(\rho)$ to be the set of the timed sequences of the form $(s_1, t_1)^\frown(s_2, t_2)^\frown \ldots ^\frown(s_m, t_m)$ satisfying that

- $s_i = \alpha(v_i)$ for each $i$ $(1 \leq i \leq m)$, and
- $t_1, t_2, \ldots, t_m$ satisfy all the variable constraints in all $\phi_i$ $(1 \leq i \leq m)$, i.e. for each variable constraint $a \leq x \leq b$ in $\phi_i$,
  - if there is a reset action $x := c$ in $\psi_j$ $(1 \leq j < i)$ and $x := d$ is not in $\psi_k$ for any $k$ $(j < k < i)$, then

$$a \leq c + w_{j+1}t_{j+1} + w_{j+2}t_{j+2} + \ldots + w_i t_i \leq b\,;$$

  - if $x := c$ is not in any $\psi_j$ $(1 \leq j < i)$ and $x := d$ is in $\psi_m$, then

$$a \leq d + w_1 t_1 + w_2 t_2 + \ldots + w_i t_i \leq b\,;$$

  - if $x := c$ is not in any $\psi_j$ $(1 \leq j < i)$ and in $\psi_m$, then

$$a \leq w_1 t_1 + w_2 t_2 + \ldots + w_i t_i \leq b\,,$$

  where for each $l$ $(1 \leq l \leq i)$, $\dot{x} = w_l \in \beta(v_l)$.

We define that a sequence $\rho$ of locations satisfies a linear duration property if every timed sequence in $\tau(\rho)$ satisfies the linear duration property. It follows that a linear hybrid automaton satisfies a linear duration property if its every path satisfies the linear duration property.

Now let us consider to solve the problem for a finite path of a linear hybrid automaton. Let $\rho = v_1 \overset{(\phi_1,\psi_1)}{\longrightarrow} v_2 \overset{(\phi_2,\psi_2)}{\longrightarrow} \ldots \overset{(\phi_m,\psi_m)}{\longrightarrow} v_{m+1}$ be a path in a linear hybrid automaton $\mathcal{H}$, and every timed sequence in $\tau(\rho)$ be of the form $(s_1, t_1)^\frown(s_2, t_2)^\frown \ldots ^\frown(s_m, t_m)$. We know that $t_1, t_2, \ldots, t_m$ should satisfy all the variable constraints in all $\phi_i$ $(1 \leq i \leq m)$, which forms a group of linear inequalities denoted by $C$. If the group $C$ of linear inequalities has no solution, then $\tau(\rho) = \emptyset$, which means that there is no timed sequence representing the behaviour of $\mathcal{H}$ corresponding to $\rho$; otherwise the problem of checking if $\rho$ satisfies a linear duration property $\mathcal{P} = \sum_{i=1}^{n} \int c_i S_i \leq M$ is equivalent to the problem of finding the maximum value of the linear function

$$\sum_{i=1}^{n} c_i \Big( \sum_{u \in \alpha_i} t_u \Big) \quad \text{where} \quad \alpha_i = \{u \mid (1 \leq u \leq m) \wedge (s_u \Rightarrow S_i)\}$$

subject to the linear constraint $C$ and checking whether it is not greater than $M$. The latter is a linear programming problem. So we reduce the problem into a linear programming problem for a finite path of a linear hybrid automaton.

We know that for a linear hybrid automaton, there could be infinite paths and the number of paths could be infinite. So we attempt to solve the problem based on a finite set of finite paths. In the following, we show how to solve the problem for zero loop-closed automata based on a finite set of finite sequences of locations.

Let $\mathcal{H}$ be a linear hybrid automaton, and $\rho$ be a path in $\mathcal{H}$ which is of the form $v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$. If $\rho$ is not a simple path, then we can find $v_i$ and $v_j$ $(1 \leq i < j \leq m)$ such that $v_i = v_j$, and then we can get a path $\rho_1$ which is constructed from $\rho$ by removing any $v_k$ $(i < k \leq j)$. By applying the above *elimination* step repeatedly, we can get a simple path $\rho'$. We say that $\rho$ is an *extension* of $\rho'$. We define that any simple path is an extension of itself. It is clear that a linear hybrid automaton $\mathcal{H}$ satisfies a linear duration property $\mathcal{P}$ if and only if for any simple path $\rho$ in $\mathcal{H}$, any extension of $\rho$ satisfies $\mathcal{P}$. In the following, for a linear duration property $\mathcal{P}$, for a simple path $\rho$ in a zero loop-closed automaton, we define a sequence of locations such that any extension of $\rho$ satisfies $\mathcal{P}$ if and only if the sequence of locations satisfies $\mathcal{P}$.

For a sequence $\rho$ of locations in a zero loop-closed automaton, for a linear duration property $\mathcal{P}$, by linear programming we can calculate the supremum of the set $\{\theta(\sigma, \mathcal{P}) \mid \sigma \in \tau(\rho)\}$. If the supremum is larger than zero, we say that $\rho$ is *violable* for $\mathcal{P}$, otherwise we say that $\rho$ is not violable for $\mathcal{P}$. Notice that if a sequence $\rho$ of locations is violable for a linear duration property $\mathcal{P}$, by repeating $\rho$ with finite many times we can construct a sequence of locations which does not satisfy $\mathcal{P}$.

Let $\mathcal{H} = (Z, X, V, E, v_I, \alpha, \beta)$ be a linear hybrid automaton, and $\mathcal{P}$ be a linear duration property which is of the form $\sum_{i=1}^{n} \int c_i S_i \leq M$. For a location $v$ in $\mathcal{H}$, if there is $S_i$ $(1 \leq i \leq n)$ such that $\alpha(v) \Rightarrow S_i$ and $c_i > 0$, then we say that $v$ is *positive* for $\mathcal{P}$. Notice that for a zero loop-closed automaton $\mathcal{H}$, for a sequence $\rho$ of locations in $\mathcal{H}$ which is violable for a linear duration property $\mathcal{P}$, there must be a location in $\rho$ which is positive for $\mathcal{P}$.

For two sequences of locations

$$\rho_1 = v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m \text{ and}$$
$$\rho_2 = u_1 \xrightarrow{(\phi'_1,\psi'_1)} u_2 \xrightarrow{(\phi'_2,\psi'_2)} \ldots \xrightarrow{(\phi'_{n-1},\psi'_{n-1})} u_n$$

such that $v_m = u_1$, let $\rho_1 \diamond \rho_2$ be the sequence of locations

$$v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m \xrightarrow{(\phi'_1,\psi'_1)} u_2 \xrightarrow{(\phi'_2,\psi'_2)} \ldots \xrightarrow{(\phi'_{n-1},\psi'_{n-1})} u_n\,.$$

For a linear hybrid automaton, we can find all loops using depth-first search method whose algorithm is described in the appendix. According to the order that the loops are found out, for any loop $\rho$ we let $o(\rho)$ be a integer which represents the position of $\rho$ in the order.

Let $\rho = v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{m-1},\psi_{m-1})} v_m$ be a loop in a zero loop-closed automaton, $\rho_1$ be a loop-enter path of $\rho$, and for each $i$ $(1 < i < m)$, $\rho_i = \rho_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{i-1},\psi_{i-1})} v_i$. For any linear duration property $\mathcal{P} = \sum_{i=1}^{n} \int c_i S_i \leq M$, let $\mu(\rho_1, \rho, \mathcal{P})$ be a sequence of locations, which is defined recursively as follows:

– if $\rho$ is a nonzero loop, then

$$\mu(\rho_1, \rho, \mathcal{P}) = \begin{cases} v_1 & \text{if } \rho' \text{ is not violable for } \mathcal{P} \\ v_1 \xrightarrow{(\emptyset,\emptyset)} v \xrightarrow{(\emptyset,\psi_{m-1})} v_m & \text{if } \rho' \text{ is violable for } \mathcal{P} \end{cases},$$

where $v$ is a location in $\rho'$ which is positive for $\mathcal{P}$ and

$$\rho' = v_1 \overset{(\phi_1,\psi_1)}{\longrightarrow} w_2 \overset{(\phi_2,\psi_2)}{\longrightarrow} w_3 \overset{(\phi_3,\psi_3)}{\longrightarrow} \ldots \overset{(\phi_{m-2},\psi_{m-2})}{\longrightarrow} w_{m-1} \overset{(\phi_{m-1},\psi_{m-1})}{\longrightarrow} v_m$$

where for each $i$ $(1 < i < m)$, if there is not any loop which can be entered through $\rho_i$, then $w_i = v_i$, otherwise

$$w_i = \mu(\rho_i, \rho_{i1}, \mathcal{P}) \diamond \mu(\rho_i, \rho_{i2}, \mathcal{P}) \diamond \ldots \diamond \mu(\rho_i, \rho_{in_i}, \mathcal{P})$$

where $\rho_{i1}, \rho_{i2}, \ldots, \rho_{in_i}$ are all the loops which can be entered through $\rho_i$ and which are such that $o(\rho_{ij}) < o(\rho_{ij+1})$ for any $j$ $(1 \leq j < n_i)$;

– if $\rho$ is a zero loop, then

$$\mu(\rho_1, \rho, \mathcal{P}) = v_1 \overset{(\phi'_1,\psi_1)}{\longrightarrow} w_2 \overset{(\phi'_2,\psi_2)}{\longrightarrow} w_3 \overset{(\phi'_3,\psi_3)}{\longrightarrow} \ldots \overset{(\phi'_{m-2},\psi_{m-2})}{\longrightarrow} w_{m-1} \overset{(\phi'_{m-1},\psi_{m-1})}{\longrightarrow} v_m$$

where

– for each $i$ $(1 < i < m)$, if there is not any loop which can be entered through $\rho_i$, then $w_i = v_i$, otherwise

$$w_i = \mu(\rho_i, \rho_{i1}, \mathcal{P}) \diamond \mu(\rho_i, \rho_{i2}, \mathcal{P}) \diamond \ldots \diamond \mu(\rho_i, \rho_{in_i}, \mathcal{P})$$

where $\rho_{i1}, \rho_{i2}, \ldots, \rho_{in_i}$ are all the loops which can be entered through $\rho_i$ and which are such that $o(\rho_{ij}) < o(\rho_{ij+1})$ for any $j$ $(1 \leq j < n_i)$, and

– for each $i$ $(1 < i < m)$, $\phi'_i = \phi_{i1} \cup \phi_{i2} \cup \phi_{i3}$,

$$\begin{aligned}
\phi_{i1} &= \{d \leq x \mid a \leq x \leq b \in \phi_i \text{ and } a = d\}, \\
\phi_{i2} &= \{x \leq d \mid a \leq x \leq b \in \phi_i \text{ and } b = d\}, \\
\phi_{i3} &= \{x = d \mid a \leq x \leq b \in \phi_i \text{ and } a = b = d\},
\end{aligned}$$

where $d$ is such that $x := d \in \psi_j$ $(1 \leq j < i)$ and $x := c \notin \psi_k$ for any $k$ $(j < k < i)$ or that $x := d \in \psi_{m-1}$ $(1 \leq j < i)$ and $x := c \notin \psi_k$ for any $k$ $(1 \leq k < i)$.

Let $\rho$ be a simple path of a zero loop-closed automaton which is of the form $v_1 \overset{(\phi_1,\psi_1)}{\longrightarrow} v_2 \overset{(\phi_2,\psi_2)}{\longrightarrow} \ldots \overset{(\phi_{m-1},\psi_{m-1})}{\longrightarrow} v_m$, and for each $i$ $(1 < i < m)$, $\rho_i = v_1 \overset{(\phi_1,\psi_1)}{\longrightarrow} v_2 \overset{(\phi_2,\psi_2)}{\longrightarrow} \ldots \overset{(\phi_{i-1},\psi_{i-1})}{\longrightarrow} v_i$. For any linear duration property $\mathcal{P}$, let $\omega(\rho, \mathcal{P})$ be a sequence of locations, which is defined as follows:

$$\omega(\rho, \mathcal{P}) = v_1 \overset{(\phi_1,\psi_1)}{\longrightarrow} w_2 \overset{(\phi_2,\psi_2)}{\longrightarrow} w_3 \overset{(\phi_3,\psi_3)}{\longrightarrow} \ldots \overset{(\phi_{m-2},\psi_{m-2})}{\longrightarrow} w_{m-1} \overset{(\phi_{m-1},\psi_{m-1})}{\longrightarrow} v_m$$

where for each $i$ $(1 < i < m)$, if there is not any loop which can be entered through $\rho_i$, then $w_i = v_i$, otherwise

$$w_i = \mu(\rho_i, \rho_{i1}, \mathcal{P}) \diamond \mu(\rho_i, \rho_{i2}, \mathcal{P}) \diamond \ldots \diamond \mu(\rho_i, \rho_{in_i}, \mathcal{P})$$

where $\rho_{i1}, \rho_{i2}, \ldots, \rho_{in_i}$ are all the loops which can be entered through $\rho_i$ and which are such that $o(\rho_{ij}) < o(\rho_{ij+1})$ for any $j$ $(1 \leq j < n_i)$.

**Lemma 1.** *Let $\mathcal{H}$ be a zero loop-closed automaton, and $\mathcal{P}$ be a linear duration property. For any simple path $\rho$ in $\mathcal{H}$, any extension of $\rho$ satisfies $\mathcal{P}$ if and only if $\omega(\rho, \mathcal{P})$ satisfies $\mathcal{P}$.* □

The detailed proof of Lemma 1 is omitted here because of space consideration. By Lemma 1, we can get the following theorem.

**Theorem 1.** *A zero loop-closed automaton $\mathcal{H}$ satisfies a linear duration property $\mathcal{P}$ if and only if for each simple path $\rho$ of $\mathcal{H}$, $\omega(\rho, \mathcal{P})$ satisfies $\mathcal{P}$.* □

```
currentpath := ⟨v_I⟩;
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;
           construct ω(ρ, P) for the current path ρ;
           check if ω(ρ, P) satisfies P;
           if no, then return false;
           if node is not in currentpath
           then append node to currentpath;
        end
until currentpath = ⟨⟩;
return true.
```

**Fig. 4.** Algorithm checking zero loop-closed automata for linear duration properties

Based on Theorem 1, we can develop an efficient algorithm to check if a zero loop-closed automaton $(X, V, E, v_I, \alpha, \beta)$ satisfies a linear duration property. The algorithm is based on depth-first search method and described in Fig. 4. The main data structure in the algorithm is a list *currentpath* of locations which is used to record the current paths. We traverse all simple paths to check if there is a simple path $\rho$ such that $\omega(\rho, \mathcal{P})$ does not satisfy $\mathcal{P}$. In the algorithm, we need to solve linear programs. Linear programming is well studied, and can be solved with a polynomial-time algorithm in general. The number of the linear programs we need to solve equals the number of all nonzero loops and simple paths in the automaton. The number of variables in each linear program is not larger than the numbers of the locations in the longest simple paths and in all the loops.

## 5 Conclusion

In this paper, we have shown that for a class of linear hybrid automata called zero loop-closed automata, the satisfaction problem for linear duration properties

can be solved efficiently by linear programming. 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 zero loop-closed automata is not contained by the decidable classes of hybrid systems we have found in the literature so far. In [2], the decidability of a class of linear hybrid systems called *integration graphs* is reduced to the verification problem for timed automata. In integration graphs, it is not allowed to test a variable in a loop which has different change rate in different locations. So the class of integration graphs does not contain that of zero loop-closed automata. In [4], 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. So the class of zero loop-closed automata is not contained by that of initialized rectangular automata. 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.

The idea to check linear duration properties by linear programming comes from [6] in which the problem for real-time automata is solved by linear programming technique, which is well established. By developing the techniques in [6], we show that by linear programming the problem can be solved totally for a class of linear hybrid automata in [9,10]. In [9,10], 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, and the presented algorithm is of high complexity in some case because we need to unfold loops so that the number and size of the linear programs we need to solve become very large. Compared with the work in [9,10], this paper makes two new contributions: First the class of zero loop-closed automata presented this paper is not included by the decidable class of linear hybrid automata defined in [9,10], and secondly the approach in this paper is based on automata directly and leads itself to an efficient implementation.

The algorithm presented in this paper has been implemented. We think there are a number of real systems that are of the loop-closed property (for example, for control systems, the loop-closed property means that every repetition of a control process starts from the same control conditions). An important topic for future work is to do case studies in practical use.

## References

1. Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, pp. 278-292.
2. Y. Kesten, A. Pnueli, J. Sifakis, S. Yovine. Integration Graphs: A Class of Decidable Hybrid Systems. In *Hybrid System, LNCS 736*, pp.179-208.

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. In *Theoretical Computer Science*, 138(1995), pp.3-34.
4. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's Decidable About Hybrid Automata? In *Journal of Computer and System Sciences*, 57:94-124, 1998.
5. Zhou Chaochen, C.A.R. Hoare, A.P. Ravn. A Calculus of Durations. In *Information Processing Letter*, 40, 5, 1991, pp.269-276.
6. C. Zhou, J. Zhang, L. Yang, and X. Li. Linear Duration Invariants. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863*, pp.88-109.
7. Thomas 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, 2000.
8. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: a model checker for hybrid systems. In *Software Tools for Technology Transfer*, 1:110-122, 1997.
9. Li Xuandong, Dang Van Hung, and Zheng Tao. Checking Hybrid Automata for Linear Duration Invariants. In *Advances in Computing Science - ASIAN'97, LNCS 1345*, Springer-Verlag, 1997, pp.166-1180.
10. Li Xuandong, Zheng Tao, Hou Jianmin, Zhao Jianhua, and Zheng Guoliang. Hybrid Regular Expressions. In *Hybrid Systems: Computation and Control, LNCS 1386*, Springer-Verlag, 1998, pp.384-399.

## A Algorithm to Check if a Linear Hybrid Automaton is Zero Loop-Closed

An efficient algorithm is described in Fig. 5, which is to check if a linear hybrid automaton $(X, V, E, v_I, \alpha, \beta)$ is zero 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 for any variable constraint in the loop, its reference point is in the loop. Then we check if any loop is such that any other loop with the same loop-start node satisfies that for any variable constraint in the loop, its reference point is in 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, its reference point is not inside the loop; and if any simple path segment is such that any nonzero loop is free. 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.

```
currentpath := ⟨v_I⟩; loopset := ∅;
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 the loop satisfies
                   that for any variable constraint in the loop,
                   its reference point is in the loop;
                   if no (the loop is not closed), then return false;
                end
           else append node to currentpath;
        end
until currentpath = ⟨⟩;
```

```
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 for any variable constraint
           in the loop, its reference point is in the loop;
           if no, then return false;
   end;
```

```
for any loop in loopset with a loop-start node v do
   begin currentpath := ⟨v⟩;
    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;
               check if the current path satisfies that
               for any variable constraint outside a loop,
               its reference point is not inside the loop;
               if no, then return false;
               check if the current path is such that any nonzero loop is free;
               if no, then return false;
               if node is not in currentpath
               then append node to currentpath;
            end
     until currentpath = ⟨⟩;
   end;
return true.
```

**Fig. 5.** Algorithm for checking if a linear hybrid automaton is zero loop-closed