Decidability for the Reachability Problem in Initialized Almost Rectangular Hybrid Automata

Nima Roohi

***Abstract*—Hybrid automata which model systems with both discrete and continuous behaviors are useful for analysis of the embedded systems. However decidability is a big problem in analyzing these models. In this paper we weaken the flow condition on the class Initialized Rectangular Hybrid Automata and prove the reachability problem is still decidable for this extended class of automata. While preserving all the reachability information, we use and extend current methods to translate a given automaton to a timed automaton with constants that could be non-rational. We then prove that reachability is still decidable for this class of timed automaton. Our approach is simple and enlarge a class of hybrid automata which is somehow known as a border of decidability for the reachability problem.**

I. Introduction

In modeling a digital system (with digital variables) which interacts with a physical environment (with analog variables), hybrid automata [4] can be a very useful model. For this reason hybrid automata is used quite a lot in modeling embedded systems. Modeling a system allows us to analyze it and check if that system satisfy its requirements or not. One of the main problems in the analysis of hybrid automata is the reachability problem. Answer to this problem makes us able to verify if the model satisfy some safety properties or not.

It is well-known that the reachability problem is decidable for some special cases of hybrid automata, and undecidable for many general cases. Henzinger *et. al.* proved in [5] that the reachability problem is decidable for the class of Initialized Rectangular Hybrid Automata (IRHA). In IRHA flow, guard, reset, and invariant of different variables are independents of each other and they are all bounded by two min/inf and max/sup values (unbounded constraints as well as strict constraints are also allowed). In this paper we relax the flow condition on each variable of a hybrid automata in the following way: Each variable in a IRHA has the flow of the form $a \le \dot{x} \le b$ for some constants a and b. But here we let flow of a variable to be in the form $a_l x + b_l \le \dot{x} \le a_u x + b_u$ for some constant a_l, a_u, b_l, and b_u. For the sake of simplicity we only consider bounded flow and non-strict constraints (more general cases can be considered in future works). We call the extended class Initialized *Almost* Rectangular Hybrid Automata (IARHA), and prove that the reachability problem is still decidable for this class of hybrid automata.

II. Preliminaries

$\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$ are respectively the set of *natural*, *rational*, and *real* numbers. $\mathbb{N}_+$ and $\mathbb{Q}_+$ are respectively the set of *positive* natural and rational numbers, and $\mathbb{R}_{\ge 0}$ is the set of non-negative real numbers. $\le$, $\ge$, $>$ and $<$ are the *ordering relations* on real numbers with their ordinary meanings. We assume ∞ is strictly larger than all real numbers and $-\infty$ is strictly smaller than all real numbers. For all $r \in \mathbb{R}$, we define $\mathsf{abs}(r)$ as an absolute value of r. We also define r^- and r^+ to be $\lim_{\epsilon \to 0} r - \mathsf{abs}(\epsilon)$ and $\lim_{\epsilon \to 0} r + \mathsf{abs}(\epsilon)$ respectively. For all numbers $a, b \in \mathbb{R}$, $[a, b]$ is defined to be the set $\{x \in \mathbb{R} | a \le x \le b\}$. $(a, b]$, $[a, b)$, and (a, b) are defined in a similar way. For any set A, we show the power set of A by $\mathcal{P}(A)$ and if A is finite we show number of elements in A by $|A|$. For all sets A and B, sets $A \cup B$, $A \cap B$, $A - B$, $A \bigtriangleup B$, and $A \times B$ are respectively union, intersection, difference, symmetric difference, and Cartesian product of A and B. A^n, for some $n \in \mathbb{N}_+$, is A if $n = 1$ and $A^{n-1} \times A$ otherwise, $A \to B$ is a (total) function from A to B, and $[A \to B]$ is the set of all (total) functions from A to B. For each functions $f : A \to B$ and $g : C \to D$ function $f \lhd g : A \cup C \to B \cup D : x \mapsto$ $\mathsf{if}\ x \in C\ \mathsf{then}\ g(x)\ \mathsf{else}\ f(x)\ \mathsf{endif}$. What is the correct symbol for this operator? For all positive real numbers r, $\ln(r)$ is the natural logarithm of r. For all functions $f : A \to B$ and $C \subseteq A$, $f(C) = \{b \in B | (\exists a \in C) b = f(a)\}$. For all sets A, functions $f : A \to \mathbb{R}$, and real values $t \in \mathbb{R}$ we define function $f + t : A \to \mathbb{R}$ by $(f + t)(x) = f(x) + t$. For all functions f and function argument a we may omit parentheses from $f(a)$ and write it fa when it causes no confusion. We use von Neumann representation for natural numbers. It means $0 = \{\}$, $1 = \{0\}$, $2 = \{0, 1\}$, $\ldots$, $n = \{1, \ldots, n-1\}$ (for two natural numbers n and m, $n - m$ is still the minus arithmetic expression).

A *rectangular* region is specified by two elements $l, u \in \mathbb{R} \cup \{-\infty, \infty\}$ and defined to be $\{x : \mathbb{R} | l \le x \le u\}$. We show the set of all rectangular regions by $\mathcal{K}$. For each rectangle $k \in \mathcal{K}$ we denote elements of k by l_k and u_k. For all $a \in \mathbb{R}$, we display a rectangular region $[a, a]$ by a when it causes no confusion. Rectangular regions are closed under finite intersection. Therefore we may display a rectangular region as the intersection of finite number of rectangular regions. For each $a_l, a_u \in \mathbb{R}$ and $b_l, b_u \in \mathbb{R}$ a *non-linear* region is defined to be $\{(y, x) \in \mathbb{R}^2 | a_l x + b_l \le y \le a_u x + b_u\}$. We show the set of all non-linear regions by $\mathcal{L}$. For each $p \in \mathcal{L}$ we denote elements of p by a_{l_p}, a_{u_p}, b_{l_p}, and b_{u_p}. We may also show p by $[a_{l_p} x + b_{l_p}, a_{u_p} x + b_{u_p}]$. For all $a, b \in \mathbb{R}$ we display $[ax + b, ax + b]$ by $ax + b$ when it causes no confusion.

For every function $x : \mathbb{R}_{\ge 0} \to \mathbb{R} : t \mapsto x(t)$, $\frac{\mathsf{d}x}{\mathsf{d}t}$ is the first derivation of x with respect to t and is displayed by $\dot{x}$ when t is known from the context. We show $x(0)$ by x_0. If $\dot{x} = ax + b$ for some $a, b \in \mathbb{R}$, then x is known to be:

$$x(t) = \begin{cases} x_0 e^{at} + \dfrac{be^{at} - b}{a} & \text{if } a \neq 0 \\ x_0 + bt & \text{otherwise} \end{cases}$$

Definition 1. A transition system T is a tuple $(\mathtt{S}, \Sigma, \to, \mathtt{S}^{\mathsf{init}})$ in which $\mathtt{S}$ is a possibly infinite set of states, Σ is a possibly infinite set of labels, $\to \subseteq \mathtt{S} \times \Sigma \times \mathtt{S}$ is a transition relation, and $\mathtt{S}^{\mathsf{init}} \subseteq \mathtt{S}$ is a set of initial sates. We write $s \xrightarrow{\alpha} s'$ instead of $(s, \alpha, s') \in \to$. Also we show the set of all transition systems

Department of Computer Science, University of Illinois at Urbana-Champaign, USA

by $\mathcal{T}$. For a transition system $T \in \mathcal{T}$, we display elements of T by $\mathtt{S}_T$, Σ_T, $\rightarrow_T$, and $\mathtt{S}_T^{\mathsf{init}}$. In addition, whenever it makes no ambiguity we may omit the subscript T to make notations simpler.

Definition 2. For all transition systems $A, B \in \mathcal{T}$ we say A is simulated by B (displayed by $A \prec B$) if and only if there exists a relation $R \in \mathtt{S}_A \times \mathtt{S}_B$ such that the following conditions are true for R:

- $\Sigma_A \subseteq \Sigma_B$
- $\forall (s_a, s_b) \in R, s'_a \in \mathtt{S}_A, \alpha \in \Sigma$
  $s_a \xrightarrow{\alpha}_A s'_a \Rightarrow (\exists s'_b \in \mathtt{S}_B) s_b \xrightarrow{\alpha}_B s'_b \wedge (s'_a, s'_b) \in R$
- $(\forall s_a \in \mathtt{S}_A^{\mathsf{init}})(\exists s_b \in \mathtt{S}_B^{\mathsf{init}})(s_a, s_b) \in R$

Also we call A and B bisimilar Correct this definition

Definition 3. An almost rectangular hybrid automaton $\mathtt{H}$ is a tuple ($\mathtt{Q}$, $\mathtt{X}$, $\mathtt{I}$, $\mathtt{F}$, Σ, $\mathtt{E}$, $\mathtt{Q}^{\mathsf{init}}$, $\mathtt{X}^{\mathsf{init}}$), where

- $\mathtt{Q}$ is a finite non-empty set of (discrete) *locations*.
- $\mathtt{X} \subset [\mathbb{R}_{\geq 0} \rightarrow \mathbb{R}]$ is a finite set of *variables*. Assuming time $t \in [0, \infty)$ is an implicit variable in each automaton, each element of $\mathtt{X}$ is defined to be a differentiable function $x : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R} : t \mapsto x(t)$.
- $\mathtt{I} \in [\mathtt{Q} \times \mathtt{X} \rightarrow \mathcal{K}]$ maps each location q and variable x to a rectangular region as the *invariant* of x in q.
- $\mathtt{F} \in [\mathtt{Q} \times \mathtt{X} \rightarrow \mathcal{L}]$ maps each location q and variable x to a region as the possible *flows* of x in q. It means $(\frac{\mathsf{d}x}{\mathsf{d}t}, x) \in \mathtt{F}(q, x)$ wherever the location is q. Assuming $\mathtt{F}(q,x) = [a_l x + b_l, a_u x + b_u]$, we define $\mathsf{crs}(\mathtt{F}(q,x))$ to be $\{r \in \mathbb{R} | a_l r + b_l = a_u r + b_u\}$. Also we simply assume that if $a_l = a_u$ then $b_l \leq b_u$. Finally for all $r \in \mathbb{R}$ we define $\mathtt{F}(q,x)(r)$ to be the rectangular region $[a_l r + b_l, a_u r + b_u]$.
- Σ is a finite set of *alphabet*.
- $\mathtt{E}$ is a finite set of *edges*. Each edge $e \in \mathtt{E}$ itself is a tuple of (s, d, b, g, j, r) in which
  - $s, d \in \mathtt{Q}$ are *source* and *destination* locations, respectively.
  - $b \in \Sigma$ is the *label* of e.
  - $g \in [\mathtt{X} \rightarrow \mathcal{K}]$ maps each variable x to a rectangular region as the *guard* condition of e for x.
  - $j \in \mathcal{P}(\mathtt{X})$ is the set of variables that their values will have *jump* after traversing e.
  - $r \in [j \rightarrow \mathcal{K}]$ maps each variable x to a rectangular region as the possible *reset* values of x after traversing e.

  We write $\mathtt{S}e$, $\mathtt{D}e$, $\mathtt{B}e$, $\mathtt{G}e$, $\mathtt{J}e$, and $\mathtt{R}e$ to denote different elements of an edge e, respectively. Also we show $(\mathtt{G}e)(x)$ and $(\mathtt{R}e)(x)$ respectively by $\mathtt{G}(e,x)$ and $\mathtt{R}(e,x)$.
- $\mathtt{Q}^{\mathsf{init}} \subseteq \mathtt{Q}$ is the set of initial locations.
- $\mathtt{X}^{\mathsf{init}} \in [\mathtt{Q}^{\mathsf{init}} \times \mathtt{X} \rightarrow \mathcal{K}]$ maps each location q and variable x to the set of *initial* values for x in q.

We show the set of all almost rectangular hybrid automata (ARHA) by $\mathcal{H}$. We only consider this class of hybrid automata in this paper, therefore whenever we write hybrid automaton (automata) we mean almost rectangular hybrid automaton (automata). For a hybrid automaton A, we display elements of A by $\mathtt{Q}_A$, $\mathtt{X}_A$, $\mathtt{I}_A$, $\mathtt{F}_A$, Σ_A, $\mathtt{E}_A$, $\mathtt{S}_A$, $\mathtt{D}_A$, $\mathtt{B}_A$, $\mathtt{G}_A$, $\mathtt{J}_A$, $\mathtt{R}_A$, $\mathtt{Q}_A^{\mathsf{init}}$, and $\mathtt{X}_A^{\mathsf{init}}$. Also we define a valuation function $\nu_A : \mathtt{X}_A \rightarrow \mathbb{R}$ that assigns a value to each variable of A. We show the set of all possible valuation functions for A by $\boldsymbol{\nu}_A$. Finally, whenever it makes no ambiguity we may omit the subscript A to make notations simpler.

For all hybrid automata A and for all variables $x \in \mathtt{X}_A$ we define $\mathsf{cnst}_A(x)$ to be the set of all constants in regions related to x. Formally $\mathsf{cnst}_A(x) = C_{\mathtt{I}} \cup C_{\mathtt{G}} \cup C_{\mathtt{R}} \cup C_{\mathtt{X}^{\mathsf{init}}}$ in which

- $C_{\mathtt{I}} = \{r \in \mathbb{R} | (\exists q \in \mathtt{Q}) r \in \{l_{\mathtt{I}(q,x)}, u_{\mathtt{I}(q,x)}\}\}$
- $C_{\mathtt{G}} = \{r \in \mathbb{R} | (\exists e \in \mathtt{E}) r \in \{l_{\mathtt{G}(e,x)}, u_{\mathtt{G}(e,x)}\}\}$
- $C_{\mathtt{R}} = \{r \in \mathbb{R} | (\exists e \in \mathtt{E}) r \in \{l_{\mathtt{R}(e,x)}, u_{\mathtt{R}(e,x)}\} \wedge x \in \mathtt{J}e\}$
- $C_{\mathtt{X}^{\mathsf{init}}} = \{r \in \mathbb{R} | (\exists q \in \mathtt{Q}^{\mathsf{init}}) r \in \{l_{\mathtt{X}^{\mathsf{init}}(q,x)}, u_{\mathtt{X}^{\mathsf{init}}(q,x)}\}\}$

Also for all hybrid automata A, function $\mathsf{cnst}(A)$ is defined to be $\bigcup_{x \in \mathtt{X}} \mathsf{cnst}_A(x)$.

We define the semantics of a hybrid automaton as a transition system it represents [7]. The semantics of a hybrid automaton A is defined by transition system $[\![A]\!] = (\mathtt{S}, \Sigma, \rightarrow, \mathtt{S}^{\mathsf{init}})$ in which

- $\mathtt{S}_{[\![A]\!]} = \mathtt{Q}_A \times \boldsymbol{\nu}_A$,
- $\Sigma_{[\![A]\!]} = \mathtt{E}_A \cup \mathbb{R}_{\geq 0}$,
- $\mathtt{S}^{\mathsf{init}}_{[\![A]\!]} = \{(q, \nu) \in \mathtt{S}_{[\![A]\!]} | q \in \mathtt{Q}_A^{\mathsf{init}} \wedge (\forall x \in \mathtt{X}_A) \nu(x) \in \mathtt{X}_A^{\mathsf{init}}(q,x)\}$, and
- $\rightarrow_{[\![A]\!]} = \rightarrow_1 \cup \rightarrow_2$ where
  - $\rightarrow_1$ is the set of time transitions and for all $t \in \mathbb{R}_{\geq 0}$ $(q, \nu) \xrightarrow{t}_1 (q', \nu')$ if and only if $q = q'$ and for all $x \in \mathtt{X}_A$ there exists a function $f_x : [0, t] \rightarrow \mathbb{R}$ such that $f_x(0) = \nu(x)$, $f_x(t) = \nu'(x)$, $(\forall t' \in [0, t])$ $f_x(t') \in \mathtt{I}_A(q,x)$ and $\frac{\mathsf{d}f_x}{\mathsf{d}t}(t') \in \mathtt{F}(q,x)(f_x(t'))$.
  - $\rightarrow_2$ is the set of jump transitions and for all $e \in \mathtt{E}_A$ $(q, \nu) \xrightarrow{\mathtt{B}e}_2 (q', \nu')$ if and only if $q = \mathtt{S}_A e$, $q' = \mathtt{D}_A e$, $(\forall x \in \mathtt{X}_A)$ $\nu(x) \in \mathtt{I}(q,x) \cap \mathtt{G}(e,x)$, $\nu'(x) \in \mathtt{I}(q',x)$. Also $x \in \mathtt{J}_A e \Rightarrow \nu'(x) \in \mathtt{R}(e,x)$ and $x \notin \mathtt{J}_A e \Rightarrow \nu(x) = \nu'(x)$.

For all hybrid automata $A, B \in \mathcal{H}$, A is simulated by B if and only if $[\![A]\!]$ is simulated by $[\![B]\!]$. Similarly A and B are bisimilar if and only if $[\![A]\!]$ and $[\![B]\!]$ are bisimilar.

Definition 4. For a hybrid automaton $A \in \mathcal{H}$, function $\mathsf{norm}(A)$ returns a hybrid automaton B in which all elements except $\mathtt{E}_A$ remain unchanged. $\mathtt{E}_B = \{(\mathtt{S}e, \mathtt{D}e, \mathtt{B}e, g, \mathtt{J}e, r) | \exists e \in \mathtt{E}_A\}$ such that $r : \mathtt{J}e \rightarrow \mathcal{K} : x \mapsto \mathtt{R}(e,x) \cap \mathtt{I}(\mathtt{D}e, x)$ and $g : \mathtt{X} \rightarrow \mathcal{K} : x \mapsto \mathtt{G}(e,x) \cap$ if $x \in \mathtt{J}e$ then $\mathtt{I}(\mathtt{S}e, x)$ else $\mathtt{I}(\mathtt{S}e, x) \cap \mathtt{I}(\mathtt{D}e, x)$ endif.

It is easy to see that for all hybrid automata $A, B \in \mathcal{H}$, $A = \mathsf{norm}(B)$ implies $A \cong B$. Also for all hybrid automata $A \in \mathsf{norm}(\mathcal{H})$ the following conditions are true:

- $(\forall e \in \mathtt{E}, x \in \mathtt{X}) \mathtt{G}(e,x) \subseteq \mathtt{I}(\mathtt{S}e, x)$
- $(\forall e \in \mathtt{E}, x \in \mathtt{J}) \mathtt{R}(e,x) \subseteq \mathtt{I}(\mathtt{D}e, x)$
- $(\forall e \in \mathtt{E}, x \in (\mathtt{X} - \mathtt{J}e)) \mathtt{G}(e,x) \subseteq \mathtt{I}(\mathtt{D}e, x)$

Definition 5. A hybrid automaton $A \in \mathcal{H}$ is

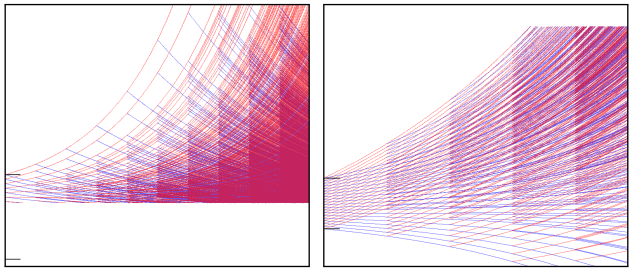
initialized $\Leftrightarrow (\forall e \in \mathtt{E}, x \in \mathtt{X})\mathtt{F}(\mathtt{S}e,x) \neq \mathtt{F}(\mathtt{D}e,x) \Rightarrow x \in \mathtt{J}e$
rectangular $\Leftrightarrow (\forall q \in \mathtt{Q}, x \in \mathtt{X})\mathtt{F}(q,x) \in \mathcal{K}$
singular $\Leftrightarrow (\forall q \in \mathtt{Q}^{\mathsf{init}}, x \in \mathtt{X})|\mathtt{X}^{\mathsf{init}}(q,x)| \leq 1$,
$(\forall e \in \mathtt{E}, x \in \mathtt{J}e)|\mathtt{R}(q,x)| \leq 1$, and
$(\forall q \in \mathtt{Q}, x \in \mathtt{X})\ a_{u_{\mathtt{F}(q,x)}} = a_{l_{\mathtt{F}(q,x)}} \wedge b_{u_{\mathtt{F}(q,x)}} = b_{l_{\mathtt{F}(q,x)}}$
singular linear $\Leftrightarrow A$ is singular, and
$(\forall q \in \mathtt{Q}, x \in \mathtt{X})\mathtt{F}(q,x) \in \mathcal{K}$
timed $\Leftrightarrow (\forall q \in \mathtt{Q}^{\mathsf{init}}, x \in \mathtt{X})\mathtt{X}^{\mathsf{init}}(q,x) \subseteq \{0\}$,
$(\forall e \in \mathtt{E}, x \in \mathtt{J}e)\mathtt{R}(q,x) \subseteq \{0\}$, and
$(\forall q \in \mathtt{Q}, x \in \mathtt{X})\mathtt{F}(q,x) = 1$

We denote to the class of initialized hybrid automata by IARHA, initialized and rectangular hybrid automata by IRHA, initialized and singular hybrid automata by ISARHA, initialized and singular linear hybrid automata by ISRHA, and timed automata by TA.

In this paper we first transform automata from IARHA to ISARHA. We then transform automata from ISARHA to ISRHA, and finally from ISRHA to TA.

A. From IRHA to TA

Henzinger *et. al.* showed in [5] how to convert a hybrid automaton A with n variables to a hybrid automaton B with $2n+1$ variables, while preserving all the reachability information of A. In their work A must be in the class of IRHA and B is always in the class of ISRHA. The general idea is to encode each variable x in $\mathtt{X}_A$ by two variables l_x, u_x in $\mathtt{X}_B$ that represent lower and upper bounds of x, respectively. One additional clock variable is used to model passage of an infinite small amount of time. Also for each new variable there will be two bits encoded in the discrete locations. One represents if the value of that variable is finite or infinite, and the other represents if the bound on that variable is strict or weak ($<$ or $\leq$). Then they define a function $\gamma : \mathtt{Q}_B \times \boldsymbol{\nu}_B \to \mathcal{P}(\mathtt{Q}_A \times \boldsymbol{\nu}_A)$ such that for all $(\forall q \in \mathtt{Q}_B, \nu \in \boldsymbol{\nu}_B)$ state (q,ν) in B is reachable if and only if all states in $\gamma(q,\nu)$ are reachable. Confirm! The approach of replacing a variable x with two variables l_x and u_x works for the class of IRHA because of (at least) three reasons: 1. At time $t = 0$ and for any possible x_0 and location q, either $\mathtt{F}(q,x)(x_0) = \emptyset$ or $\mathtt{F}(q,x)(x_0) \neq \emptyset$. 2. If the lower bound of a variable flow is smaller than or equal to the upper bound of the flow at time 0 then this relation between lower and upper bounds is preserved in any later time. 3. Possible values of x in location q at any time is exactly defined by $\{^1 l_x, u_x{}^1\}$. '$\{^1$' (similarly '$^1\}$') can be '[' or '(' (similarly ']' or ')') based on the value of strictness bit encoded in location q. In general neither of these three properties are satisfied by automata in the class of IARHA, so we need to first transform automata in this class to automata that satisfy all of these properties, and then use them to prove our decision procedure works for the class of IARHA. Figure 3 shows two example non-linear flows. Flow in Figure 1a does not satisfy the first property. Because for some x_0 in initial values of x we have $\mathtt{F}(q,x)(x_0) = \emptyset$ and for some other values we have $\mathtt{F}(q,x)(x_0) \neq \emptyset$. Also flow in Figure 1b does not satisfy the other two properties. Because although the lower bound of $\dot{x}$ is smaller than or equal to the upper bound of $\dot{x}$ at time 0, this relation between lower and upper bounds is not preserved in later times. Moreover it implies that possible values of x in $t > 0$ is not equal to $[l_x(t), u_x(t)]$ in which $l_x(t)$ and $u_x(t)$ are lower bound and upper bound curves respectively.



(a) $x_0 \in [-1,0]$
$-2x-1 \leq \dot{x} \leq 4x+1$

(b) $x_0 \in [-1,1]$
$3x+1 \leq \dot{x} \leq 2x+8$

Fig. 1: Two examples non-linear flows. Horizontal axis is time and vertical axis is value of sample variable x. Initial values bounds are specified by two small line segments at each vertical axes. Invariant of x is assumed to be exactly parts of the vertical axes that are displayed.

B. From Solvable Hybrid Automata to TA

Henzinger *et. al.* showed in [3] that for a hybrid automaton A from a special class of non-linear hybrid automata we can transform A into a timed automaton B by encoding value of each variable $x \in \mathtt{X}_A$ to a value of variable $t_x \in \mathtt{X}_B$. They call this special automaton *solvable*.

Definition 6. For a hybrid automaton A and for a location $q \in \mathtt{Q}$ and a variable $x \in \mathtt{X}$, the variable x is *determined* in q if $\mathtt{F}(q,x) = f(x)$ (for some f) and for all initial values $x_0 \in \mathbb{R}$ the problem $\dot{x} = f(x) \wedge x(0) = x_0$ has an algebraic solution $x_{q,x_0}(t)$ such that for each constant c that appear in constraints, $x_{q,x_0}(t) - c$ has finite number of roots.

For a location $q \in \mathtt{Q}$ and a variable $x \in \mathtt{X}$, q is called *definite* for x iff $|\mathtt{Q}^{\mathsf{init}}(q,x)| \leq 1$. Similarly an edge $e \in \mathtt{E}$ is called *definite* for x iff $x \in \mathtt{J}(e) \wedge |\mathtt{R}(e,x)| = 1$.

In automaton A, a variable x of $\mathtt{X}$ is *solvable* if the following three conditions hold:

1. For all locations $q \in \mathtt{Q}$, variable x is determined in q.
2. For all locations $q \in \mathtt{Q}$, location q is definite for x.
3. For all edge $e \in \mathtt{E}$ if $\mathtt{F}(\mathtt{S}e) \neq \mathtt{F}(\mathtt{D}e)$ then edge e is definite for x.

A non-linear hybrid automaton A is solvable iff all of its variables are solvable.

Lemma 1. All automata in the class of ISARHA are solvable.

Proof. For all $a, b, x_0 \in \mathbb{R}$, if $\dot{x} = ax + b$ (all flows in this class are in this form) then $x(t)$ is $x_0 e^{at} + \frac{be^{at}-b}{a}$ if $a \neq 0$ and $x_0 + bt$ otherwise. This means $x(t)$ has an algebraic solution. Also for all $c \in \mathbb{R}$ we know $x(t) - c$ has at most one root. Therefore first condition is satisfied by the automata in this class.

By definition of the singular automata we know $(\forall q \in \mathtt{Q}^{\mathsf{init}}, x \in \mathtt{X})|\mathtt{X}^{\mathsf{init}}(q,x)| \leq 1$ which imply the second condition. By definition of the singular automata we also know $(\forall e \in \mathtt{E}, x \in \mathtt{J}e)|\mathtt{R}(q,x)| \leq 1$. Moreover by the definition of the initialized automata we know $(\forall e \in \mathtt{E}, x \in \mathtt{X})\mathtt{F}(\mathtt{S}e,x) \neq \mathtt{F}(\mathtt{D}e,x) \Rightarrow x \in \mathtt{J}e$. These two imply the third condition. □

In this paper we use this technique to transform automata in the class of ISARHA to automata in the class of TA. For all hybrid automata A in the class of ISARHA, we display the automata after transformation by $\mathsf{solve}(A)$. It is important to note that if $\mathsf{cnst}(A) \subset \mathbb{Q}$ then $\mathsf{cnst}(\mathsf{solve}(A)) \subset \mathbb{Q} \cup \mathbb{Q} \ln \mathbb{Q}_+$ in which $\mathbb{Q} \ln \mathbb{Q}_+$ is defined to be $\{r \ln v | r \in \mathbb{Q} \wedge v \in \mathbb{Q}_+\}$.

C. Fourier-Motzkin variable elimination method

Having a set of inequalities of the form $I = \bigwedge_{i \in m} \sum_{j \in n} a_{i,j} x_j \sim_i b_i$ in which $m, n \in \mathbb{N}$ and for all $i \in m$ and $j \in n$ we have $a_{i,j} \in \mathbb{R}$ and $b_i \in \mathbb{R}$ are constants, $x_j \in \mathbb{R}$ is variable, and $\sim_i \in \{<, \leq\}$ is an ordering relation, one can use the Fourier-Motzkin variable elimination method [6, Sec 5.4] to decide weather there is any $x_1, \ldots, x_n \in \mathbb{R}$ that satisfy all inequalities in I or not. Variable elimination methods usually take as an input a finite number of of inequalities and produce a new finite number of inequalities as their result. Each variable elimination method should have three characteristics:

1. Number of variables in the output should be smaller than their number in the input.
2. The new system should be satisfiable if and only if the original system is satisfiable.
3. From the answer to the new system it should be possible to find an answer in the original system.

In this paper we only use the Fourier-Motzkin method to decide whether the system I is satisfiable or not. Therefore we are not interested in the third condition. To remove variable x from I using the Fourier-Motzkin method, we should divide constraints in I into three sets: 1. I_+ contains constraints that coefficient of x is positive in them, 2. I_- contains constraints that coefficient of x is negative in them, 3. I_0 contains constraints that coefficient of x is zero in them. If either I_+ or I_- was empty, simply remove all the constraints that contains x (coefficient of x is non-zero for them) and return remaining constraints as result. Otherwise for each pair of constraints in $c_+ \in I_+$ and $c_- \in I_-$, assuming coefficients of x are respectively a_+ and a_-, create a new constraint $a_+ \times c_- + a_- \times c_+$ (obviously coefficient of x in the new constraint is 0). If at least one of c_+ or c_- uses the strict inequality relation (*i.e.* $<$), the new constraint is also strict. Otherwise the new constraint will be non-strict (*i.e.* $\leq$). The new system contains all the new constraints, I_0, and none of the constraints in I_- or I_+. It has one less variable, and one can prove that it satisfies the second condition of the elimination method. We can continue eliminating variables till no more variable exists in the system. All that will remain is a finite number of inequalities such that each inequality compares (strict or non-strict) a real value with 0. Therefore Fourier-Motzkin method reduces the decidability of the original system to the decidability of comparison of real numbers and zero.

III. From IARHA to TA

To translate an automaton $A \in \mathcal{H}$ into a timed automaton, while preserving all the reachability information, the general idea is to use the same method described in section II.B and for each variable x in $\mathtt{X}_A$ define two variables x_l and x_u to track the lower bound and upper bound values of x. In the class of IRHA when a variable x is reset to $[a, b]$ and $\dot{x} \in [c, d]$ we know that after any amount of time $t \in \mathbb{R}_{\geq 0}$ lower value of x is $a + ct$ and upper value of x is $b + dt$. In general this property does not hold for the class of IARHA. If $|\mathsf{crs}(\mathtt{F}(q,x))| = 1$ and $x_c \in \mathsf{crs}(\mathtt{F}(q,x))$ for some $x_c \in \mathbb{R}$ then flow of x is not valid for either x_0^- or x_0^+ which makes no time transition possible (see Figure 3 for two examples). So if the initial values of x contain both x_c^- and x_c^+ then we cannot use the lower bound and upper bound variables idea. Even if the initial values of x contain one of x_c^- and x_c^+ for which no valid x flow exists, there is still some point in the initialization rectangle that can have a time transition and some point has no valid time transition.

Definition 8 solves the first problem by splitting locations of the given automaton into new locations with more restricted invariants that prevent the first problem to occur. Definition 9 solves the second problem by only adding one new variable to the given automaton and restricting values of the new variable in the problematic locations. We also prove that none of these transformations loses the reachability information of the original automaton.

Definition 7. For a hybrid automaton $A \in \mathcal{H}$, function $\mathsf{pivots}_A : \mathtt{Q}_A \to \mathcal{P}(\mathtt{X}_A)$ takes a location q as an argument and returns a subset of variables in A. Variable $x \in \mathtt{X}_A$ is in the returned set if and only if the lower bound and the upper bound of $\mathtt{F}_A(q,x)$ has one and only one common value and that common value belongs to the invariant of x in q. For all $q \in \mathtt{Q}_A$ return value of $\mathsf{pivots}_A(q)$ is formally defined as follows:

$$(\forall x \in \mathtt{X}) x \in \mathsf{pivots}_A(q) \Leftrightarrow |\mathsf{crs}(\mathtt{F}_A(q,x))| = 1 \wedge \mathsf{crs}(\mathtt{F}_A(q,x)) \subseteq \mathtt{I}_A(q,x)$$

Definition 8. Function $\mathsf{split} : \mathcal{H} \to \mathcal{H}$ takes a hybrid automaton A as input, and returns hybrid automaton B which is formally defined as follows:

- $\mathtt{Q}_B = \{(q, Y) \in \mathtt{Q} \times \mathcal{P}\mathtt{X} | Y \subseteq \mathsf{pivots}_A(q)\}$
  Therefore every location q is divided into $2^{|\mathsf{pivots}_A(q)|}$ locations.
- $\mathtt{X}_B = \mathtt{X}_A$
- $\mathtt{I}_B((q,Y),x) =$
  $\mathtt{I}_A(q,x)$ if $x \notin \mathsf{pivots}_A(q)$
  $\mathtt{I}_A(q,x) \cap \{x \in \mathbb{R} | x \leq x_c\}$ if $x \in Y$
  $\mathtt{I}_A(q,x) \cap \{x \in \mathbb{R} | x \geq x_c\}$ otherwise
  In the later two cases we assume $\{x_c\} = \mathsf{crs}(\mathtt{F}_A(q,x))$[1]. Invariants of the variables outside of $\mathsf{pivots}_A(q)$ will not change. All other invariants will be divided on x_c the only point in $\mathsf{crs}(\mathtt{F}_A(q,x))$, so x_c^+ and x_c^- cannot both be in the possible values of x_0.
- $\mathtt{F}_B((q,Y),x) = \mathtt{F}_A(q,x)$
- $\Sigma_B = \Sigma_A$
- $\mathtt{E}_B = \{((\mathtt{S}e, Y), (\mathtt{D}e, Y'), \mathtt{B}e, \mathtt{G}e, \mathtt{J}e, \mathtt{R}e) |$
  $(\exists e \in \mathtt{E}_A, Y, Y' \in \mathcal{P}\mathtt{X})(\mathtt{S}e, Y) \in \mathtt{Q}_B \wedge (\mathtt{D}e, Y') \in \mathtt{Q}_B\}$
  When a location is divided into two (or more) locations, incoming and outgoing edges to and from this location should be duplicated such that we do not lose any possible transition (See Lemma 2).

[1] We know $\mathsf{crs}(\mathtt{F}_A(q,x)$ has only one element. Because $x \in Y$ is only possible when $x \in \mathsf{pivots}_A(q)$ by definition implies $|\mathsf{crs}(\mathtt{F}_A(q,x))| = 1$. So our assumption simply defines x_c in terms of $\mathsf{crs}(\mathtt{F}_A(q,x))$.

- $\mathtt{Q}_B^{\mathsf{init}} = \{(q,Y) \in \mathtt{Q}_B | q \in \mathtt{Q}_A^{\mathsf{init}}\}$
- $\mathtt{X}_B^{\mathsf{init}}((q,Y),x) = \mathtt{X}_A^{\mathsf{init}}(q,x)$

Figure 2 displays an example of applying the split function. Automaton A has two locations 1 and 2 plus one variable x. Automaton B has three locations $(1,\{x\})$, $(1,\{\})$, and $(2,\{\})$ plus the same variable x. Since $x+1$ and $2x+3$ are equal in $x=-2$ (and no other point), location 1 in A is divided into two locations in automaton B. But since $2x$ and $2x+3$ are never equal, location 2 in A is not divided in automaton B. Also since 1 is an initial location in A, both $(1,\{\})$ and $(1,\{x\})$ are initial locations in B (initial values of x is not changed). Also $\mathtt{I}_B((1,\{x\}),x) = \mathtt{I}_B(1,x) \cap (-\infty,-2]$ and $\mathtt{I}_B((1,\{\}),x) = \mathtt{I}_B(1,x) \cap [-2,\infty)$. For each edge in A with 1 as its source and destination locations, we have four new edges in B with source and destination locations as specified in this figure. For each edge in A with 1 as its source or destination location (but not both), we have two new edges in B with source and destination locations as specified in this figure. Other elements of the edges are not changed in automaton B, therefore they are exactly same as edges in A with the same number.

Lemma 2 states that function split does not change behavior of its input automaton.

Lemma 2. For all hybrid automata $A, B \in \mathcal{H}$, A and B are bisimilar if $B = \mathsf{split}(A)$.

$$(\forall A, B \in \mathcal{H}) B = \mathsf{split}(A) \Rightarrow B \cong \mathsf{split}(A)$$

Proof. We just define ${}_aR_b$ to be a function from $\mathtt{Q}_A \times \boldsymbol{\nu}_A$ to $\mathtt{Q}_B \times \boldsymbol{\nu}_B$ and ${}_bR_a$ to be a function from $\mathtt{Q}_B \times \boldsymbol{\nu}_B$ to $\mathtt{Q}_A \times \boldsymbol{\nu}_A$. Based on these two functions it is easy to see that A and B simulate each other and therefore $A \cong B$.

For all $p = ((q,Y),\nu) \in \mathtt{Q}_B \times \boldsymbol{\nu}_B$ we define ${}_bR_a(p)$ to simply be (q,ν). For all $p = (q,\nu) \in \mathtt{Q}_A \times \boldsymbol{\nu}_A$ we define ${}_aR_b(p)$ to be $((q,Y),\nu)$ in which Y is defined as $\{x \in \mathtt{X} | (\exists x_c \in \mathsf{crs}(\mathtt{F}_A(q,x)))P_1 \wedge (P_2 \vee P_3)\}$ such that P_1 is $x \in \mathsf{pivots}_A(q)$, P_2 is $\nu(x) < x_c$, and P_3 is $\nu(x) = x_c \wedge \mathtt{F}_A(q,x)(x^-) \neq \emptyset$. By P_1 we know x_c is unique. When P_2 is true, we must have $x \in Y$ in order to $\nu(x) \in \mathtt{I}_B((q,Y),x)$. When $x = x_c$ we should be careful. In this case, by definition both $x \in Y$ and $x \notin Y$ can be true. But only one of the resulting locations can have time transition. If $\mathtt{F}_A(q,x)(x^-) \neq \emptyset$ is true, it means for values below x_c possible flows of x is not empty, therefore x should belongs to Y in order to have time transitions. □

Automaton $\mathsf{split}(A)$ has a nice property which is for all location $q \in \mathtt{Q}$ and variable $x \in \mathtt{X}$ if $\mathtt{F}(q,x)(x_0)$ has only one element x_c then we know at least one of the followings is true: 1. $x_c \notin \mathtt{I}(q,x)$, 2. $x_c \in \mathtt{I}(q,x) \wedge x_c^+ \notin \mathtt{I}(q,x)$ (it means x_c is the upper bound of the invariant of x in q) 3. $x_c \in \mathtt{I}(q,x) \wedge x_c^- \notin \mathtt{I}(q,x)$ (it means x_c is the lower bound of the invariant of x in q). We define and use $\mathsf{split}(A)$ because we want to use similar approach which is used in [5]. We want to replace every variable x in the given hybrid automaton with two variables that track lower and upper bounds of possible values of x. But there is still one problem in $\mathsf{split}(A)$. As an example assume x is a variable which is reset to $[1,2]$ by edge e and $q = \mathtt{D}e$. Also assume $\mathtt{I}(q,x) = [1,3]$ and $\mathtt{F}(q,x) = [-2x+3, -3x+4]$. If $x_0 \neq 1$ there will be no time transition in q. Because $(\forall x \in (1,2]) -2x+3 > -3x+4$. At $x_0 = 1$ even if we can have time transition (not possible in this example) the same transition is possible in another location q' which is same as q except instead of $x \geq 1$ in the invariant of q, $x \leq 1$ is in the invariant of q'. So for each variable x and location q and based on $\mathtt{F}(q,x)$ and $\mathtt{I}(q,x)$ we may restrict invariants in q such that even if no constraint was forced by $\mathtt{F}(q,x)$ no time transition would be possible in q. Definition 9 formalizes this transformation. After this transformation we can replace each variable x by two variables x_l and x_u that track lower and upper bounds of x respectively.

Definition 9. Function $\mathsf{restr} : \mathcal{H} \to \mathcal{H}$ takes a hybrid automaton A as input, and return hybrid automaton B which is formally defined as follows:

- $\mathtt{Q}_B = \mathtt{Q}_A$
- $\mathtt{X}_B = \mathtt{X}_A \cup \{z\}$ (assuming $z \notin \mathtt{X}_A$)
- $\mathtt{I}_B = \mathtt{I}_A \lhd \{(q,z) \mapsto k\}$ in which k is 0 if and only if the following condition is true and $(-\infty,\infty)$ otherwise. $(\exists x \in \mathtt{X}_A)(\exists x_0 \in \mathtt{I}(q,x)) x_0 \notin \mathtt{F}(q,x)(x_0)$
- $\mathtt{F}_B = \mathtt{F}_A \lhd \{(q,z) \mapsto 1 | q \in \mathtt{Q}_B\}$
- $\Sigma_B = \Sigma_A$
- $\mathtt{E}_B = \{(\mathtt{S}e, \mathtt{D}e, \mathtt{B}e, g, \mathtt{J}e \cup \{z\}, r) | e \in \mathtt{E}_A\}$ where
  $g = \mathtt{G}e \cup \{z \mapsto (-\infty,\infty)\}$
  $r = \mathtt{R}e \cup \{z \mapsto 0\}$
- $\mathtt{Q}_B^{\mathsf{init}} = \mathtt{Q}_A^{\mathsf{init}}$
- $\mathtt{X}_B^{\mathsf{init}} = \mathtt{X}_A^{\mathsf{init}} \lhd \{(q,z) \mapsto 0 | q \in \mathtt{Q}_B^{\mathsf{init}}\}$

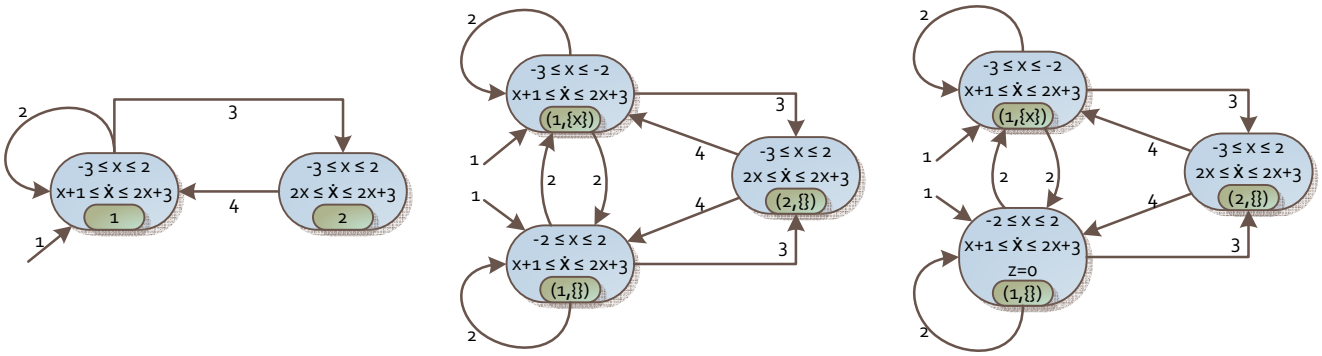
Lemma 3. For all hybrid automata $A \in \mathsf{split}(\mathcal{H})$ and $B \in \mathcal{H}$, A and B are bisimilar if $B = \mathsf{restr}(A)$.

$$(\forall A \in \mathsf{split}(\mathcal{H}), B \in \mathcal{H})$$
$$B = \mathsf{restr}(A) \Rightarrow B \cong \mathsf{restr}(A)$$

Proof. Obviously $B \prec A$, so we only need to show $A \prec B$. The only reason that may prevent B to simulate A is the addition of $((q,Y),z) \mapsto 0$ to some of the invariants in B. Suppose automaton $[\![A]\!]$ can have a time transition t at $((q,Y),x_0)$, for some $q \in \mathtt{Q}$, $Y \subseteq \mathtt{X}$, and $x_0 \in \mathbb{R}$, and automaton $[\![B]\!]$ cannot have that time transition. If we are at the initial state then, by definition of the split function, both $((q,Y \cup \{x\}),x_0)$ and $((q,Y - \{x\}),x_0)$ are initial states too. Moreover, for the one which is different with $((q,Y),x_0)$ we have time transition. If we get to $((q,Y),x_0)$ after traversing an edge e with label $\mathtt{B}e$, again by definition of split, we could went to both $((q,Y \cup \{x\}),x_0)$ and $((q,Y - \{x\}),x_0)$ by traversing another edge with the same label $\mathtt{B}e$. Moreover, for the one which is different with $((q,Y),x_0)$ we have time transition. We know that function restr does not change this property of its input. Having this property in B it is straightforward to create the desired simulation relation. □

Lemma 4. For all automaton $A \in \mathsf{restr}(\mathsf{split}(\mathcal{H}))$, and for all reachable states $(q,\nu) \in (\mathtt{Q}, \boldsymbol{\nu})$, if $\nu(z) > 0$ then $(\forall x \in \mathtt{X})\mathtt{F}(q,x)(\nu(x)) \neq \emptyset$.

Proof. Assume $\mathtt{F}(q,x) = [a_l x + b_l, a_u x + b_u]$. If $a_l = a_u$ we know $b_l \leq b_u$. Therefore $(\forall r \in \mathbb{R})\mathtt{F}(q,x)(r) \neq \emptyset$. If $a_l \neq a_u$ then $(\exists x_c \in \mathbb{R})\mathsf{crs}(\mathtt{F}(q,x)) = \{x_c\}$. If $\mathtt{F}(q,x)(\nu(x)) = \emptyset$ we know either $\nu(x) < x_c$ or $\nu(x) > x_c$. For the case $\nu(x) < x_c$ (similar argument apply to the other case) we know



(a) Hybrid Automaton A (b) Hybrid Automaton $B = \mathsf{split}(A)$ (c) Hybrid Automaton $C = \mathsf{restr}(B)$

Fig. 2: Splitting a sample hybrid automaton in Figure a using function split first and then function restr (results are respectively displayed in Figure b and Figure c). In addition to source and destination locations, each edge has four more elements. Numbers on the edges of automata B and C define these elements based on the edges of automaton A with the same number (the same argument is hold about initialization values of variables). Location 1 in automaton A is divided into two locations $(1,\{x\})$, $(1,\{\})$ in automaton B. Also automaton C has one variable (z) more than automaton B. Dividing location 1 on $x=-2$ guarantees that none of the new locations has both -2^- and -2^+ in its invariant. Also adding $z=0$ to the invariants of location $(1,\{\})$ in automaton C guarantees in no reachable state of automaton C lower bound flow is larger than upper bound flow.

$\mathtt{F}(q,x)(\nu(x)) = \emptyset$ if and only if $\mathtt{F}(q,x)(x_c^-) = \emptyset$. But function restr add $z=0$ to the invariants of locations for which $\mathtt{F}(q,x)(x_c^-) = \emptyset$. □

Theorem 1. For all hybrid automaton $A \in \mathcal{H}$ the following is true:

$$(\forall q \in \mathtt{Q}, x \in \mathtt{X}, k \in \mathcal{K})(l_k \le x_0 \le u_k)$$
$$\Rightarrow$$
$$\max(l_k(t), l_{\mathtt{I}(q,x)}) \le x(t) \le \min(u_k(t), u_{\mathtt{I}(q,x)})$$

in which, assuming $\mathtt{F}(q,x) = [ax+b, cx+d]$, $l_k(t)$ and $u_k(t)$ are defined to be:

$$l_k(t) = \begin{cases} l_k e^{at} + \dfrac{b(e^{at}-1)}{a} & \text{If } a \neq 0 \\ l_k + bt & \text{otherwise} \end{cases}$$

$$u_k(t) = \begin{cases} u_k e^{ct} + \dfrac{d(e^{ct}-1)}{c} & \text{If } c \neq 0 \\ u_k + dt & \text{otherwise} \end{cases}$$

Proof. By definition we know $l_{\mathtt{I}(q,x)} \le x(t) \le u_{\mathtt{I}(q,x)}$, therefore assuming $l_{\mathtt{I}(q,x)} \le l_k(t)$ and $u_k(t) \le u_{\mathtt{I}(q,x)}$ we prove all points between $l_k(t)$ and $u_k(t)$ (inclusive) are reachable and no other point is reachable.

We first prove that no point below $l_k(t)$ is reachable (similar argument proves that no point above $u_k(t)$ is reachable). Because $x(t)$ is differentiable it is continuous, and because $x_0 \ge l_k(0) = l_k$ if $x(t) < l_k(t)$ then $(\exists t_c \in [0,t))(x(t_c) = l_k(t_c) \wedge x(t_c^+) < l_k(t_c^+))$. But this means $\dot{x}(t_c) < l_k(t_c)$ which is a contradiction.

To prove that all the points in between are reachable we need to consider different cases (later cases may use what is proved in the former cases): Confession: I have a serious problem with "THE". "the former cases" or "former cases"?

- $a = c$
  - $a = 0 \wedge c = 0$
    $[l_k + bt, u_k + dt] = [l_k + bt, u_k + bt] \cup [u_k + bt, u_k + dt]$
    If $\dot{x} = b$ then all points in the first set are reachable. All points in the second set are reachable if we let $x_0 = u_k$ and $b \le \dot{x} \le d$.
  - $a \neq 0 \wedge c \neq 0$
    * $b = d$
      $x_0 = \lambda l_k + (1-\lambda)u_k$ for some $\lambda \in [0,1]$.
      $x(t) = x_0 e^{at} + \frac{b(e^{at}-1)}{a}$
      $= \lambda\left(x_0 e^{at} + \frac{b(e^{at}-1)}{a}\right) + (1-\lambda)\left(x_0 e^{at} + \frac{b(e^{at}-1)}{a}\right)$
      $= \lambda l_k(t) + (1-\lambda)u_k(t)$.
    * $l_k = u_k$
      $\dot{x} = \lambda(ax+b) + (1-\lambda)(cx+d)$ for some $\lambda \in [0,1]$.
      $x(t) = x_0 e^{at} + \frac{(\lambda c + (1-\lambda)d)(e^{at}-1)}{a}$
      $= \lambda\left(x_0 e^{at} + \frac{b(e^{at}-1)}{a}\right) + (1-\lambda)\left(x_0 e^{at} + \frac{c(e^{at}-1)}{a}\right)$
      $= \lambda l_k(t) + (1-\lambda)u_k(t)$
    * general case
      $[l_k(t), u_k(t)] = [l_k(t), u_k e^{at} + \frac{b(e^{at}-1)}{a}] \cup [u_k e^{at} + \frac{b(e^{at}-1)}{a}, u_k(t)]$
      All points in the first set are proved to be reachable in the first item and all points in the second set are proved to be reachable in the second item.
- $a \neq c$
  $[l_k(t), u_k(t)] = [l_k(t), u_k e^{at} + \frac{b(e^{at}-1)}{a}] \cup [u_k e^{at} + \frac{b(e^{at}-1)}{a}, u_k(t)]$
  All points in the first set are already proved to be reachable ($a = c$ and $b = d$). To prove all points in the second set ($x_0 = u_k$) are also reachable we consider the following cases:
  - $0 \in [ax_0 + b, cx_0 + d]$
    $[l_k(t), u_k(t)] = [l_k(t), x_0) \cup \{x_0\} \cup (x_0, u_k(t)]$. Obviously x_0 is reachable after any time t. We prove all points in $(x_0, u_k(t)]$ are reachable. Similar argument proves all points in $[l_k(t), x_0)$ are reachable too. Proof

for the case $c = 0$ is trivial, so we only prove the case $c \neq 0$. Let $\dot{x} = \lambda(cx + d)$ for some $\lambda \in (0, 1]$. Therefore $x(t) = x_0 e^{\lambda ct} + \frac{\lambda d(e^{\lambda ct}-1)}{\lambda c}$. We know $\lim_{\lambda \to 0} x(t) = x_0$, which implies all points in $(x_0, u_k(t)]$ are reachable.

- $0 < ax_0 + b$
  $[ax + b, cx + d] = ([0, cx + d] - [0, ax + b]) + \{ax + b\}$. We just proved that for $\dot{x} \in [0, cx + d]$ all the points in $[x_0, x_0 e^{ct} + \frac{d(e^{ct}-1)}{c}]$ and for $\dot{x} \in [0, ax + b]$ all the points in $[x_0, x_0 e^{at} + \frac{b(e^{at}-1)}{a}]$ are reachable. This means if $\dot{x} \in ([0, cx + d] - [0, ax + b])$ then all points in $(x_0 e^{at} + \frac{b(e^{at}-1)}{a}, x_0 e^{ct} + \frac{d(e^{ct}-1)}{c}]$ are reachable. We also know that if $\dot{x} \in ax + b$ then $x_0 e^{at} + \frac{b(e^{at}-1)}{a}$ is reachable.
- $cx_0 + d < 0$ similar to the previous case.

□

From now on we assume that the given automaton satisfy the given conditions in Theorem 1, and proceed to formally define a new automaton in which all the reachability information is preserved. We are given an automaton A and, while all the reachability information is preserved, we create a singular automaton B. The observation alphabet Σ_B is same as Σ_A. For each variable $x \in \mathtt{X}_A$ there are two variables x_l and x_u in $\mathtt{X}_B$. Flow of the new variables are simply lower bound and upper bound flows of the original variable respectively. $\mathtt{F}_B(q, x_l) = a_{1\mathtt{F}_A(q,x)} + b_{1\mathtt{F}_A(q,x)}$ and $\mathtt{F}_B(q, x_u) = a_{2\mathtt{F}_A(q,x)} + b_{2\mathtt{F}_A(q,x)}$. Invariants of the new variables are defined as follows: $\mathtt{I}_B(q, x_l) = (-\infty, u_{\mathtt{I}_A(q,x)}]$ and $\mathtt{I}_B(q, x_u) = [u_{\mathtt{I}_A(q,x)}, \infty)$. For all edge $e \in \mathtt{E}_A$ and $x \in \mathtt{X}_A$, if $x \in \mathtt{J}_A e$ then $\mathtt{R}(e, x) = k_1$ implies $\mathtt{R}(e, x_l) = [l_{k_1}, l_{k_1}]$ and $\mathtt{R}(e, x_u) = [u_{k_1}, u_{k_1}]$, also $\mathtt{G}(e, x) = k_2$ implies $\mathtt{G}(e, x_l) = (-\infty, l_{k_2}]$ and $\mathtt{G}(e, x_u) = [u_{k_2}, \infty)$. On the other hand if $x \notin \mathtt{J}_A e$ then Page 102, second paragraph, the last two lines

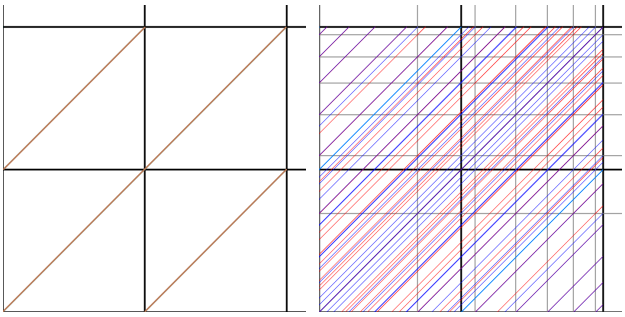
A. From Initialized SARHA to TA

The next step is to use the Clock Translation technique to translate an automaton A in the class of ISARHA to an automaton B in the class of ISRHA. The translation is well explained in [3], we show translated hybrid automaton by $\mathsf{solve}(A)$. The only problem is that if all constants in A are rational, we may end up with some constants in $\mathbb{Q} \ln \mathbb{Q}_+$. But [3] needs that A to be *rationally* solvable, so all such r must belongs to $\mathbb{Q}$. Next we prove that this special case is still decidable.

B. Decidability for Timed Automata

Suppose $A \in \mathcal{H}$ is a hybrid automaton such that $\mathsf{cnst}(A) \subset \mathbb{Q}$. Also suppose $B \in \mathcal{H}$ is another hybrid automaton equal to $\mathsf{solve}(\mathsf{restr}(\mathsf{split}(A)))$. We know $\mathsf{cnst}(B) \subset \mathbb{Q} \cup \mathbb{Q} \ln \mathbb{Q}_+$. In this section we prove that the reachability problem is still decidable for B. Using the same approach used in [2], we first partition $\boldsymbol{\nu}_A$ into a finite set of equivalence classes. We then prove that all valuation functions in each class are bisimilar. And finally, we prove that creating a finite automaton using the defined equivalence relation is decidable.

In [2] it is assumed that all constants are rational, we call a timed automaton with only rational constants a *rational* timed automaton. Similarly we call a timed automaton with constants in $\mathbb{Q} \cup \mathbb{Q} \ln \mathbb{Q}_+$ an *almost rational* timed automaton. Unlike in [2], here we cannot define integer part and fractional part of a value. Because in rational timed automaton, after multiplying every constant by a constant factor, we have a timed automaton in which all constants are integers and 1 is always their common divisors. Therefore if we partition values between minimum 0 and maximum c_x by $c_x + 1$ equally separated lines, it is guaranteed that for all possible set of valid constraints on values of x, values in each partition satisfy the exact same subset of constraints. But in almost rational timed automaton, there is no such common divisor. Furthermore, if we want to consider all the possible values between 0 and c_x, number of partitions will become exponential. Figure 3a shows partitions for a two dimensional rational timed automaton, and Figure 3b shows partitions for a two dimensional almost rational timed automaton if we consider all the possible constants between 0 and 2 (maximum constant in these examples). It is easy to see that number of partitions in Figure 3b are exponentially more than number of partitions in Figure 3a. Furthermore in addition to points, lines, and triangles, bounded regions in Figure 3b can be parallelogram or even trapezoid, which makes comparison of fractional parts difficult. Therefore we use another method to define equivalency between valuations of an almost rational timed automaton.



(a) Rational Timed Automaton (b) Almost Rational Timed Automaton

Fig. 3: Finite partitioning for two two dimensional timed automata. Constants in automaton a are all rational, however constants in automaton b are all either rational or multiplication of a rational and natural logarithm of another rational number. In the rational automaton a all the bounded regions are either points, equal line segments, or equal triangles. However, in the almost rational automaton b all the bounded regions are either points, line segments (not necessarily equal), triangles (not necessarily equal), parallelogram, or even trapezoid.

Definition 10. For a timed automaton A, an equivalence relation $\equiv_A$ is defined on $\boldsymbol{\nu}$ as follows: $(\forall \nu_1, \nu_2 \in \boldsymbol{\nu}) \nu_1 \equiv_A \nu_2 \Leftrightarrow P_1 \wedge P_2$. The following two conditions define P_1 and P_2 respectively.

P_1 : $(\forall x \in \mathtt{X}, c \in \mathsf{cnst}(x), \sim \in \{=, <\})$
$\nu_1(x) \sim c \Leftrightarrow \nu_2(x) \sim c$

P_2 : $(\forall x, y \in \mathtt{X}, c_x \in \mathsf{cnst}(x), c_y \in \mathsf{cnst}(y), \sim \in \{=, <\})$
$\nu_1(x) - c_x \sim \nu_1(y) - c_y \Leftrightarrow \nu_2(x) - c_x \sim \nu_2(y) - c_y$

We may write $\equiv$ instead of $\equiv_A$ whenever A is known from the context. Moreover, we define $\boldsymbol{\nu}/\equiv$ to be $\{[\nu] \in \mathcal{P}\boldsymbol{\nu} | (\forall \nu' \in$

$\boldsymbol{\nu})\nu \equiv \nu' \Leftrightarrow \nu' \in [\nu]\}$.

Lemma 5. For a timed automaton A, $\equiv$ partitions $\boldsymbol{\nu}$ to a finite number of equivalent classes.

Proof. $\mathtt{X}$ is finite, and for all $x \in \mathtt{X}$ we know that $\mathsf{cnst}(x)$ is also finite. Therefore each of P_1 and P_2 in Definition 10 defines a finite number of equivalent classes, which implies $P_1 \wedge P_2$ also defines a finite number of equivalent classes. □

Theorem 2. For a timed automaton A, both of the following propositions are true:

1. $(\forall x \in \mathtt{X}, c \in \mathsf{cnst}(x), \sim \in \{=, <\}, \nu_1, \nu_2 \in \boldsymbol{\nu})$
   $\nu_1 \equiv \nu_2 \Rightarrow (\nu_1(x) \sim c \Leftrightarrow \nu_2(x) \sim c)$
2. $(\forall \nu_1, \nu_2 \in \boldsymbol{\nu}, t_1 \in \mathbb{R}_{\geq 0})$
   $\nu_1 \equiv \nu_2 \Rightarrow (\exists t_2 \in \mathbb{R}_{\geq 0})(\nu_1 + t_1) \equiv (\nu_2 + t_2)$

First proposition guarantees that ν_1 and ν_2 satisfy the exact same set of constraints in A. Second proposition guarantees that for any reachable point from ν_1 there is an equivalent reachable point from ν_2 (therefore $\equiv$ defines a bisimulation).

Proof. First proposition is true by the definition of $\equiv$ (P_1 in Definition 10). For the second proposition, we define function $f_{\nu_1,t_1} : [0,1] \to \mathcal{P}(\boldsymbol{\nu}/\equiv) : \lambda \mapsto \{[\nu_1 + \alpha t_1] | \alpha \in [0, \lambda]\}$ and prove that proposition using the induction on $|f_{\nu_1,t_1}(1)|$.

We know $[\nu_1] \in f_{\nu_1,t_1}(1)$, therefore base of induction, when $|f_{\nu_1,t_1}(1)| = 1$, implies $f_{\nu_1,t_1}(1) = \{[\nu_1]\} = \{[\nu_2]\}$ which means in this case t_2 can be 0.

For the inductive step, we assume $|f_{\nu_1,t_1}(1)| = n+1$ for some $n \in \mathbb{N}_+$. There must be some strictly monotonic function $g : n+1 \to [0,1]$ such that $g(0) = 0$, $g(n) = 1$, and $(\forall i, j \in n+1) i \neq j \Rightarrow [\nu_1 + g(i)t_1] \neq [\nu_1 + g(j)t_1]$. We let $\lambda_n = g(n)$. That is $\lambda_n < 1$ and $|f(\lambda_n)| = n$. Therefore, by the induction hypothesis we know $(\exists t_n \in \mathbb{R}_{\geq 0})(\nu_1 + \lambda_n t_1) \equiv (\nu_2 + t_n)$. Let $\nu_1' = \nu_1 + \lambda_n t_1$, $\nu_2' = \nu_2 + t_n$, and $t_1' = (1 - \lambda_n)t_1$. We know $f_{\nu_1',t_1'}(1) = \{[\nu_1'], [\nu_1' + t_1']\}$, and it sufficient to find t_2' such that $\nu_2' + t_2' \equiv \nu_1' + t_1'$.

By the induction hypothesis we know $[\nu_1'] \neq [\nu_1' + t_1']$, therefore $(\exists z \in \mathtt{X}, c_z \in \mathsf{cnst}(z), \sim \in \{=, <\})\nu_1'(z) \sim c_z \not\Leftrightarrow (\nu_1' + t_1')(z) \sim c_z)$[2]. Fixing that z and c_z, we define t_2' to be

$$t_2' = \begin{cases} c_z - \nu_2'(z) & \text{if } \nu_1'(z) < c_z \\ 0^+ & \text{if } \nu_1'(z) = c_z \end{cases}$$

The case $\nu_1'(z) > c_z$ is not possible, because otherwise by $t_1' > 0$ we know $(\nu_1' + t_1')(z) > c_z$ which is a contradiction. Also it is easy to see $t_2' > 0$. Now we prove $\nu_2' + t_2' \equiv \nu_1' + t_1'$.

We know $(\forall x, y \in \mathtt{X}, \nu \in \boldsymbol{\nu}, t \in \mathbb{R})\ \nu(x) - \nu(y) = \nu(x) + t - \nu(y) - t$. Therefore assuming ν_1' and ν_2' satisfy P_2 in Definition 10, we know $\nu_1' + t_1'$ and $\nu_2' + t_2'$ also satisfy P_2.

For P_1 in Definition 10, it is enough to prove $(\forall x \in \mathtt{X}, c_x \in \mathsf{cnst}(x), \sim \in \{<, =, >\})\ \nu_1'(x) + t_1' \sim c_x \Rightarrow \nu_2'(x) + t_2' \sim c_x$. If $\nu_1'(x) \geq c_x$ we know $\nu_2'(x) \geq c_x$, $t_1' > 0$, and $t_2' > 0$. These imply $\nu_1'(x) + t_1' > c_x$ and $\nu_2'(x) + t_2' > c_x$. Therefore in the following we assume $\nu_1'(x) < c_x$. Furthermore we assume $\nu_1'(x) + t_1' \leq c_x$ because otherwise $|f_{\nu_1',t_1'}(1)| > 2$ which would be a contradiction.

For the case $\nu_1'(z) < c_z$ we have $\nu_1'(z) + t_1' = c_z$ and either $\nu_1'(x) - \nu_1'(z) - c_x + c_z = 0$ or $\nu_1'(x) - \nu_1'(z) - c_x + c_z < 0$ (the other case is not possible since it implies $|f_{\nu_1',t_1'}(1)| > 2$). $\nu_1'(x) - \nu_1'(z) - c_x + c_z = \nu_1'(x) - c_x + t_1'$. Therefore if $\nu_1'(x) - \nu_1'(z) - c_x + c_z = 0$ then $\nu_2'(x) - \nu_2'(z) - c_x + c_z = 0$ and $\nu_1'(x) + t_1' = c_x$. Moreover $\nu_2'(x) + t_2' = \nu_2'(x) + c_z - \nu_2'(z) = c_x = \nu_1'(x) + t_1'$. If $\nu_1'(x) - \nu_1'(z) - c_x + c_z < 0$ then $\nu_2'(x) - \nu_2'(z) - c_x + c_z < 0$ and $\nu_1'(x) + t_1' - \nu_1'(z) - t_1' - c_x + c_z = \nu_1'(x) + t_1' - c_x < 0$. Moreover $\nu_2'(x) + t_2' = \nu_2'(x) + c_z - \nu_2'(z) < c_x$.

For the case $\nu_1'(z) = c_z$ we must have $\nu_1'(x) + t_1' < c_x$ because otherwise $|f_{\nu_1',t_1'}(1)| > 2$. Also $\nu_2'(x) + t_2' < c_x$ is obvious in this case. □

It is remained to prove there is a computational method to decide about each of the following two problems:

1. For each almost rational timed automaton A, every valuation function class $[\nu] \in \boldsymbol{\nu}/\equiv$, and every location $q \in \mathtt{Q}$, whether $(\exists \nu_q \in [\nu])(\forall x \in \mathtt{X})\nu_q(x) \in \mathtt{I}(q, x)$ is true or false.
2. For each almost rational timed automaton A, every two valuation function classes $[\nu_1], [\nu_2] \in \boldsymbol{\nu}/\equiv$, and every edge $e \in \mathtt{E}$, whether we can go from $[\nu_1]$ to $[\nu_2]$ using e or not.

Both of these problems can be easily reduced to a more general problem which is defined and proved to be decidable in Lemma 6.

Lemma 6. For all $n, m \in \mathbb{N}$ and finite set of variables $\mathtt{X}$, there is a decision procedure that tells us whether the following is true or false (assuming $(\forall i \in m, j \in n)\ q_{i,j}, r_i, v_i \in \mathbb{Q} \wedge u_i \in \mathbb{Q}_+ \wedge \sim_i \in \{<, \leq\} \wedge x_j \in \mathtt{X}$):

$$\bigwedge_{i \in m} \left(\left(\sum_{j \in n} q_{i,j} x_j \right) \sim_i r_i + v_i \ln u_i \right)$$

Proof. We can use the Fourier-Motzkin variable elimination method [6] to reduce this problem to

$$\bigwedge_{i \in m'} (0 \sim_i' r_i' + \ln v_i')$$

in which $(\forall i \in m')(r_i' \in \mathbb{Q} \wedge v_i' \in \mathbb{Q}_+ \wedge \sim_i' \in \{<, \leq\})$. This is possible because $(\forall q, r \in \mathbb{Q}, u, v \in \mathbb{Q}_+)(q + r \in \mathbb{Q} \wedge qr \in \mathbb{Q} \wedge q \ln u + r \ln v \in \mathbb{Q} \ln \mathbb{Q}_+ \wedge rq \ln u \in \mathbb{Q} \ln \mathbb{Q}_+)$. Therefore for all $r \in \mathbb{Q}$, $v \in \mathbb{Q}_+$, and $\sim \{<, \leq\}$ we only need to be able to decide whether $r \sim \ln v$ is true or not. Obviously $r \sim \ln v \Leftrightarrow e^r \sim v$. If $r = 0$ then $e^r = 1$ and $1 \sim v$ is known to be decidable. For the case when $r \neq 0$, Aigner *et. al.* proved in [1] that $e^r \notin \mathbb{Q}$. Therefore $e^r \sim v \Leftrightarrow e^r < v$. Now assume $l_i \in \mathbb{Q}_+$ and $u_i \in \mathbb{Q}_+$ are lower and upper bounds of e^r which are equal to it upto $i \in \mathbb{N}_+$ decimal numbers. For all finite $i \in \mathbb{N}_+$ we can compute both l_i and u_i. Shall we put reference for this two? The following two propositions help us to find if $e^r \sim v$ is true or false.

1. $(\exists i \in \mathbb{N}_+) l_i > v \Rightarrow e^r > v$
2. $(\exists i \in \mathbb{N}_+) u_i < v \Rightarrow e^r < v$

There must be some $i \in \mathbb{N}_+$ for which $l_i > v$ or $u_i < v$ becomes true, because otherwise we have $\lim_{i \to \infty} l_i \leq v \leq u_i$ and $\lim_{i \to \infty} l_i = \lim_{i \to \infty} u_i$. This means that $v = e^r$ which is a contradiction because we know $e^r \notin \mathbb{Q}$. Therefore we know there is a decision procedure that always terminates and always decides correctly whether $e^r \sim v$ is true or not. □

[2] We know P_2 in Definition 10 is always satisfied by ν_1' and $\nu_1' + t_1$.

IV. Conclusion

In this paper we relaxed the flow condition on Initialized Rectangular Hybrid Automata from rectangular constraint $a \leq \dot{x} \leq b$ for some rational values a and b to non-linear constraint $a_1 x + b_1 \leq \dot{x} \leq a_2 x + b_2$ for some rational values a_1, a_2, b_1 and b_2 and prove that the reachability problem is still decidable for the extended class of hybrid automata. The first future extension to this work could be considering the case when flows are not necessarily bounded and they can also be strict inequalities. Another future work could be considering another class of hybrid automata, for which reachability is proved to be decidable, and try to relax the precondition on the automata of that class. For example, if $f(x) \leq \dot{x} \leq g(x)$ and $f(x)$ and $g(x)$ intersect only in finite number of points, what are the other sufficient conditions that make it possible to use the same method to achieve a decision procedure.

Acknowledgment

The author would like to thank Prof. Mahesh Viswanathan for his great discussions and comments on the subject of this paper.

References

[1] Martin Aigner and Günter M Ziegler. *Proofs from THE BOOK*. Springer-Verlag, 4 edition, 2010.

[2] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] T.A. Henzinger, Pei-Hsin Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *Automatic Control, IEEE Transactions on*, 43(4):540 –554, apr 1998.

[4] Thomas A. Henzinger. The theory of hybrid automata. pages 278–292. IEEE Computer Society Press, 1996.

[5] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *Journal of Computer and System Sciences*, pages 373–382. ACM Press, 1995.

[6] Daniel Kroening and Ofer Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[7] Pavithra Prabhakar, Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Hybrid automata based cegar for hybrid systems. Under Submission, 2012.