What's decidable about hybrid automata?

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen

SS09

Literature

Henzinger et al.: What's decidable about hybrid automata?

Journal of Computer and System Sciences, 57:94–124, 1998

Motivation

- The special class of *timed automata* with TCTL is *decidable*, thus model checking is possible.
- What about other classes of hybrid systems?

What is decidable about hybrid automata?

Two central problems for the analysis of hybrid automata:

- *Reachability*: Check if the trajectories of the automaton meet a given *safety* requirement
- *ω-language emptiness problem*: Check if the trajectories of the automaton meet a given *lifeness* requirement

Both problems are decidable in certain special cases, and undecidable in certain general cases.

What is decidable about hybrid automata?

A particularly interesting class:

- the set of trajectories are *piecewise-linear envelopes*
- e.g. flow condition $\dot{x} \in [1, 3]$ called *rectangular flow constraints*
- automata with rectangular constraints are called *rectangular-flow atomata*
- this class lies at the boundary of decidability

What is decidable about hybrid automata?

The reachability problem for rectangular-flow automata is decidable under the following restrictions:

- values of variables with different flow constraints are not compared, and
- whenever the flow constraint of a variable changes, the value of the variable is re-initialized.

The ω-language emptiness problem is decidable for rectangular-flow automata under the restriction of bounded nondeterminism:

- the successor of a bounded region is bounded.

The reachability problem becomes undecidable if one of the restrictions is relaxed.

Rectangular regions

Definition

- Given a positive integer n, a subset of $\mathbb{R}^n$ is called a *region*.
- A region $\mathcal{R} \subset \mathbb{R}^n$ is *rectangular* if it is a cartesian product of (possibly unbounded) intervals, all of whose endpoints are rational.
- The set of rectangular regions in $\mathbb{R}^n$ is denoted $\mathcal{R}^n$.

Reminder: Hybrid automaton

Definition

A *hybrid automaton* $\mathcal{H}$ is a tuple $\mathcal{H} = (Loc, Var, Lab, Edge, Act, Inv, Init)$ with

- finite set of locations Loc,
- finite set of real-valued variables Var,
- finite set of synchronization labels Lab, $\tau \in Lab$ (stutter label)
- finite set of edges $Edge \subseteq Loc \times Lab \times 2^{V^2} \times Loc$ (including stutter transitions $(l, \tau, \mathtt{Id}, l)$ for each location $l \in Loc$),
- Act is a function assigning a set of activities $f : \mathbb{R}^+ \to V$ to each location; the activity sets are time-invariant, i.e., $f \in Act(l)$ implies $(f + t) \in Act(l)$, where $(f + t)(t') = f(t + t')$ f.a. $t' \in \mathbb{R}^+$,
- a function Inv assigning an invariant $Inv(l) \subseteq V$ to each location $l \in Loc$,
- initial states $Init \subseteq \Sigma$.

with

- valuations $\nu : Var \to \mathbb{R}$, V is the set of valuations
- state: $(l, \nu) \in Loc \times V$, Σ is the set of states
- transitions: discrete and time

Rectangular automaton

Definition

An *n-dimensional rectangular automaton A* is a tuple
$\mathcal{H} = (Loc, Var, Lab, Edge, Act, Inv, Init)$ with

- finite set of locations Loc,
- finite set of real-valued variables $Var = \{v_1, \ldots, v_n\}$,
- finite set of synchronization labels Lab,
- finite set of edges $Edge \subseteq Loc \times Lab \times \mathcal{R}^n \times \mathcal{R}^n \times 2^{\{1,\ldots,n\}} \times Loc$,
- a flow function $Act : Loc \to \mathcal{R}^n$,
- an invariant function $Inv : Loc \to \mathcal{R}^n$,
- initial states $Init : Loc \to \mathcal{R}^n$.

n-dimensional rectangular automaton with ϵ-moves: Lab contains ϵ (also denoted by τ).

Rectangular automaton

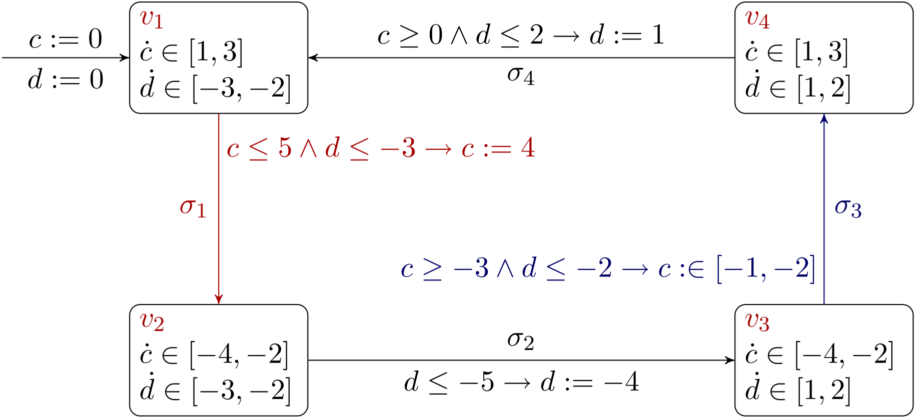
- Flows: first time derivatives of the flow trajectories in location $l \in Loc$ are within $Act(l)$
- Jumps: $e = (l, a, pre, post, jump, l') \in Edge$ may move control from location l to location l' starting from a valuation in pre, changing the value of each variable $v_i \in jump$ to a nondeterministically chosen value from $post_i$ (the projection of $post$ to the ith dimension), and leaving the values of the other variables unchanged.

Notes:

- If we replace rectangular regions with linear regions, we obtain *linear hybrid automata*, a super-class of rectangular automata.
- A timed automaton is a rectangular automaton with deterministic jumps (defined later) such that every variable is a clock.

- A *timed word* is a word from $(Lab \cup \mathbb{R}^{\geq 0})^*$.
- A timed word is *divergent*, if the sum of the time letters of the word is ∞.
- A *run* of a rectangular automaton A is a (finite or infinite) sequence $q_0 \xrightarrow{\tau_0} q_1 \xrightarrow{\tau_1} \ldots$ with $q_0 \in Init$.
- The run $q_0 \xrightarrow{\tau_0} q_1 \xrightarrow{\tau_1} \ldots$ *accepts* the word $\tau_0 \tau_1 \ldots$.
- The ω-language $Lang(A)$ of A is the set of all divergent time words that are accepted by runs of A.
- The *reachable zone* $Reach(A)$ of A is the set $Post^*(Init)$, where $Post^\pi(Z) = \{(l, \nu) \in \Sigma \mid \exists (l', \nu') \in Z.\ (l', \nu') \xrightarrow{\pi} (l, \nu)\}$.

Initialized rectangular automaton



Definition?

Trajectories?

A subset of the state space $Loc \times \mathbb{R}^n$ is called a *zone*. Each zone Z is decomposable into a collection $\bigcup_{l\in Loc}\{l\} \times Z_l$ of zones. The zone Z is *rectangular* iff each Z_l is rectangular. A zone is *multirectangular*, if it is a finite union of rectangular zones.

Lemma

For every multirectangular zone Z of a rectangular automaton A, and every label $\pi \in Lab \cup \mathbb{R}^{\geq 0}$, the zones
$Post^{\pi}(Z) = \{(l,\nu) \in \Sigma \mid \exists (l',\nu') \in Z.\ (l',\nu') \xrightarrow{\pi} (l,\nu)\}$ *and*
$Pre^{\pi}(Z) = \{(l,\nu) \in \Sigma \mid \exists (l',\nu') \in Z.\ (l,\nu) \xrightarrow{\pi} (l',\nu')\}$ *are multirectangular.*

Consequence: we could also allow disjunctions and conjunctions on edges.
Note: The reachable zone of rectangular automaton A is an infinite union of rectangular zones, and may not be multirectangular.

Proof:

- It suffices to prove it for a single transition π, then the lemma follows for runs.
- It suffices to prove it for elementary regions of the form $Z = (\{l\}, \mathcal{R})$ with $\mathcal{R}$ rectangular.
- π can represent a jump or a flow
- $\pi = (l, a, pre, post, jump, l')$: $Post^{\pi}(Z) = \{l'\} \times S$ with

$$S_i = \begin{cases} \mathcal{R}_i \cap pre_i \cap post_i \cap Inv(l')_i & \text{if } i \notin jump_i \\ post_i \cap Inv(l')_i & \text{if } i \in jump_i \text{ and } \mathcal{R}_i \cap pre_i \neq 0 \\ \emptyset & \text{if } i \in jump_i \text{ and } \mathcal{R}_i \cap pre_i = 0 \end{cases}$$

- $\pi = 0$: $Post^{\pi}(Z) = Z$
- $\pi = t$, $t \in \mathbb{R}^{\geq 0}$: $Post^{\pi}(Z) = \{l\} \times S$ with

$$S_i = Inv(l)_i \quad \cap \quad \overset{(}{[}\, inf(\mathcal{R}_i) + \pi \cdot inf(Act(l)_i) \quad , \quad \infty)$$
$$\cap \quad (-\infty \quad , \quad sup(\mathcal{R}_i) + \pi \cdot sup(Act(l)_i)\, \overset{)}{]}$$

Initialization condition

Definition

A rectangular automaton A is *initialized*, if for every edge $(l, a, pre, post, jump, l')$ in the edge set of A, and every variable index $i \in \{1, \ldots, n\}$ with $Act(l)_i \neq Act(l')_i$, we have that $i \in jump$.

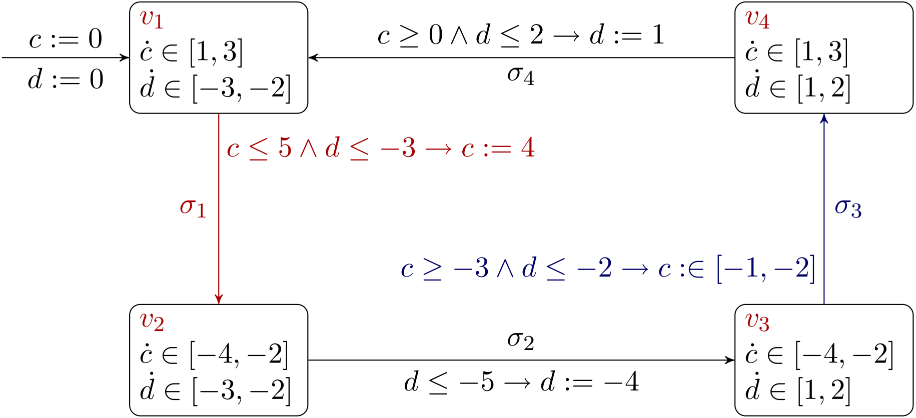
I.e., whenever a variable changes its dynamics, its value gets nondeterministically re-initialized.

Definition

A rectangular automaton A has *bounded nondeterminism*, if

- all initial and flow regions are bounded, and
- for every edge e of A and every coordinate i in the jump set of e, the interval $post_i$ of e is bounded.

Initialized rectangular automaton



This rectangular automaton is initialized and has bounded nondeterminism.

Decidability results

Lemma

The reachability problem for initialized rectangular automata is complete for PSPACE.

Lemma

The ω-language emptiness problems for initialized rectangular automata with bounded nondeterminism is complete for PSPACE.

Timed automaton

$\uparrow$

Initialized stopwatch automaton

$\uparrow$

Initialized singular automaton

$\uparrow$

Initialized rectangular automaton

A timed automaton is a rectangular automaton with deterministic jumps such that every variable is a clock, i.e., $Act(l)(x) = [1,1]$ for all locations l and variables x.

Lemma

The reachability and the ω-language emptiness problems for timed automata are complete for PSPACE.

Decidability results

Timed automaton

↑

Initialized stopwatch automaton

- A has deterministic jumps, if (1) $Init(l)$ is empty or a sigleton for all l, and (2) the post-interval for each variable from the jump-set of each edge is a singleton.
- A *stopwatch* is a variable with derivatives 0 or 1 only.
- A *stopwatch automaton* is a rectangular automaton with deterministic jumps and stopwatch variables only.
- Initialized stopwatch automata can be polynomially encoded by timed automata.

Lemma

The reachability and the ω-language emptiness problems for initialized stopwatch automata are complete for PSPACE.

However, the reachability problem for non-initialized stopwatch automata is undecidable.

Proof idea:

Notice, that a timed automaton is a stopwatch automaton such that every variable is a clock.

Assume that C is an n-dimensional initialized stopwatch automaton with ϵ-moves. Let κ_C be the set of rational constants used in the definition of C, and let $\kappa_- = \kappa_C \cup \{-\}$.

We define an n-dimensional timed automaton D_C with locations $Loc_{D_C} = Loc_c \times \kappa_-^{1,\ldots,n}$. Each location (l, f) of D_C consists of a location l of C and a function $f : \{1, \ldots, n\} \rightarrow \kappa_-$. Each state $q = ((l, f), \vec{x})$ of D_C represents the state $\alpha(q) = (l, \vec{y})$ of C, where $y_i = x_i$ if $f(i) = -$, and $y_i = f(i)$ if $f(i) \neq -$.

Intuitively, if the ith stopwatch of C is running (slope 1), then its value is tracked by the value of the ith clock of D_C; if the ith stopwatch is halted (slope 0) at value $k \in \kappa_C$, then this value is remembered by the current location of D_C.

Decidability results

Timed automaton

↑

Initialized stopwatch automaton

↑

Initialized singular automaton

- A variable v_i is a *finite-slope variable* if $flow(l)_i$ is a singleton in all locations l.
- A *singular automaton* is a rectangular automaton with deterministic jumps such that every variable of the automaton is a finite-slope variable.
- Initialized singular automata can be rescaled to initialized stopwatch automata.

Lemma

The reachability and the ω-language emptiness problems for initialized singular automata are complete for PSPACE.

Proof idea: Let B be an n-dimensional initialized singular automaton with ϵ-moves. We define an n-dimensional initialized stopwatch automaton C_B with the same location set, edge set, and label set as B.
Each state $q = (l, \vec{x})$ of C_B corresponds to the state $\beta(q) = (l, \beta(\vec{x}))$ of B with $\beta : \mathbb{R}^n \to \mathbb{R}^n$ defined as follows:
For each location l of B, if $Act_B(l) = \Pi_{i=1}^n [k_i, k_i]$, then $\beta(x_1, \ldots, x_n) = (l_1 \cdot x_1, \ldots, l_n \cdot x_n)$ with $l_i = k_i$ if $k_i \neq 0$, and $l_i = 1$ if $k_i = 0$;
β can be viewed as a rescaling of the state space. All conditions in the automaton B occur accordingly rescaled in C_B.
We have:

- The reachable set of $Reach(B)$ of B is $\beta(Reach(C_B))$.
- $Lang(B) = Lang(C_B)$

Decidability results

Timed automaton

$\uparrow$

Initialized stopwatch automaton

$\uparrow$

Initialized singular automaton

$\uparrow$

Initialized rectangular automaton

Lemma

The reachability problem for initialized rectangular automata is complete for PSPACE.

Lemma

The ω-language emptiness problems for initialized rectangular automata with bounded nondeterminism is complete for PSPACE.

Proof idea: An n-dimensional initialized rectangular automaton A can be translated into a $(2n+1)$-dimensional initialized singular automaton B with ϵ-moves, such that B contains all reachability information about A. The translation is similar to the subset construction for determinizing finite automata.

The idea is to replace each variable c of A by two finite-slope variables c_l and c_u: c_l tracks the least possible value of c, and c_u tracks the greatest possible value of c.