Modeling and analysis of hybrid systems

What's decidable about hybrid automata?

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen

SS 2010

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: Given to sets of states R and R', is a state in R' reachable from a state in R? (safety)
- Language inclusion: Is the set of traces doable from states from R contained in a given trace set? (lifeness)

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

What is decidable about hybrid automata?

A particularly interesting class:

What is decidable about hybrid automata?

A particularly interesting class:

- all conditions, effects, and flows are described by rectagular sets.

What is decidable about hybrid automata?

A particularly interesting class:

- all conditions, effects, and flows are described by rectagular sets.

Definition

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

Rectangular automaton

Definition

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

- finite set of locations Loc,
- finite set of real-valued variables $Var = \{x_1, \ldots, x_n\}$,
- a function $Con : Loc \to 2^{Var}$ assigning controlled variables to locations,
- 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 $Init : Loc \to \mathcal{R}^n$.

Rectangular automaton with ϵ-moves: Lab contains ϵ (also denoted by τ).

State space

- States: $\sigma = (l, \vec{x}) \in (Loc \times \mathbb{R}^n)$ with $\vec{x} \in Inv(l)$

State space

- States: $\sigma = (l, \vec{x}) \in (Loc \times \mathbb{R}^n)$ with $\vec{x} \in Inv(l)$
- State space: $\Sigma \subseteq Loc \times \mathbb{R}^n$ is the set of all states

State space

- States: $\sigma = (l, \vec{x}) \in (Loc \times \mathbb{R}^n)$ with $\vec{x} \in Inv(l)$
- State space: $\Sigma \subseteq Loc \times \mathbb{R}^n$ is the set of all states
- Is the state space rectangular?

State space

- States: $\sigma = (l, \vec{x}) \in (Loc \times \mathbb{R}^n)$ with $\vec{x} \in Inv(l)$
- State space: $\Sigma \subseteq Loc \times \mathbb{R}^n$ is the set of all states
- Is the state space rectangular?
- Do the initial states build a rectangular set?

State space

- States: $\sigma = (l, \vec{x}) \in (Loc \times \mathbb{R}^n)$ with $\vec{x} \in Inv(l)$
- State space: $\Sigma \subseteq Loc \times \mathbb{R}^n$ is the set of all states
- Is the state space rectangular?
- Do the initial states build a rectangular set?
- May we use conjunctions to specify the invariants?

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 to a nondeterministically chosen value from $post_i$ (the projection of $post$ to the ith dimension), such that the values of the variables $x_i \notin jump$ are unchanged.

Operational semantics

Operational semantics

$$\frac{\begin{array}{c}(l, a, \mathit{pre}, \mathit{post}, \mathit{jump}, l') \in Edge \\ \vec{x} \in \mathit{pre} \quad \vec{x}' \in \mathit{post} \quad \forall i \notin \mathit{jump}. x'_i = x_i \quad \vec{x}' \in Inv(l')\end{array}}{(l, \vec{x}) \xrightarrow{a} (l', \vec{x}')} \quad \texttt{Rule}_{\texttt{Discrete}}$$

Operational semantics

$$\frac{\begin{array}{c}(l, a, \textit{pre}, \textit{post}, \textit{jump}, l') \in Edge \\ \vec{x} \in \textit{pre} \quad \vec{x}' \in \textit{post} \quad \forall i \notin \textit{jump}. x'_i = x_i \quad \vec{x}' \in Inv(l')\end{array}}{(l, \vec{x}) \xrightarrow{a} (l', \vec{x}')} \quad \texttt{Rule}_{\texttt{Discrete}}$$

$$\frac{(t = 0 \wedge \vec{x} = \vec{x}') \vee (t > 0 \wedge (\vec{x}' - \vec{x})/t \in Act(l)) \quad \vec{x}' \in Inv(l)}{(l, \vec{x}) \xrightarrow{t} (l, \vec{x}')} \quad \texttt{Rule}_{\texttt{Time}}$$

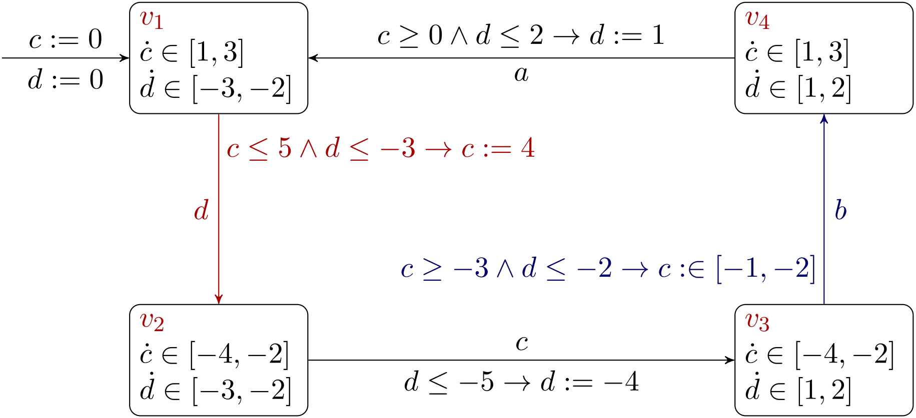
Operational semantics

$$\frac{\begin{array}{c}(l, a, \mathit{pre}, \mathit{post}, \mathit{jump}, l') \in Edge \\ \vec{x} \in \mathit{pre} \quad \vec{x}' \in \mathit{post} \quad \forall i \notin \mathit{jump}. x'_i = x_i \quad \vec{x}' \in Inv(l')\end{array}}{(l, \vec{x}) \stackrel{a}{\rightarrow} (l', \vec{x}')} \quad \texttt{Rule}_{\texttt{ Discrete}}$$

$$\frac{(t = 0 \wedge \vec{x} = \vec{x}') \vee (t > 0 \wedge (\vec{x}' - \vec{x})/t \in Act(l)) \quad \vec{x}' \in Inv(l)}{(l, \vec{x}) \stackrel{t}{\rightarrow} (l, \vec{x}')} \quad \texttt{Rule}_{\texttt{ Time}}$$

- execution step: $\rightarrow = \stackrel{a}{\rightarrow} \cup \stackrel{t}{\rightarrow}$
- path: $\sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \ldots$
- initial path: path $\sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \ldots$ with $\sigma_0 = (l_0, \vec{x}_0)$, $\vec{x}_0 \in Init(l_0) \cap Inv(l_0)$
- reachability of a state: exists a run leading to the state

Initialized rectangular automaton



Definition?
Trajectories?

Reverse automata

Rectangular automata are reversible.

Remarks

- If we replace rectangular sets with linear sets, 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.

Remarks

- If we replace rectangular sets with linear sets, 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.

What do the restrictions actually mean?

Remarks

- If we replace rectangular sets with linear sets, 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.

What do the restrictions actually mean? (Rectangularity is preserved)

Remarks

- If we replace rectangular sets with linear sets, 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.

What do the restrictions actually mean? (Rectangularity is preserved)
This class lies at the boundary of decidability.

Decidability

The reachability problem is decidable for initialized rectangular automata:

Decidability

The reachability problem is decidable for initialized rectangular automata:

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 $i \in jump$.

Decidability

The reachability problem is decidable for initialized rectangular automata:

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 $i \in jump$.

The language inclusion problem is decidable for initialized rectangular automata with bounded nondeterminism:

Decidability

The reachability problem is decidable for initialized rectangular automata:

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 $i \in jump$.

The language inclusion problem is decidable for initialized rectangular automata with bounded nondeterminism:

Definition

A rectangular automaton A has bounded nondeterminism, if

- all initial and flow sets are bounded, and
- for every edge e and every index i in the *jump* set of e, the interval $post_i$ of e is bounded.

Decidability

The reachability problem is decidable for initialized rectangular automata:

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 $i \in jump$.

The language inclusion problem is decidable for initialized rectangular automata with bounded nondeterminism:

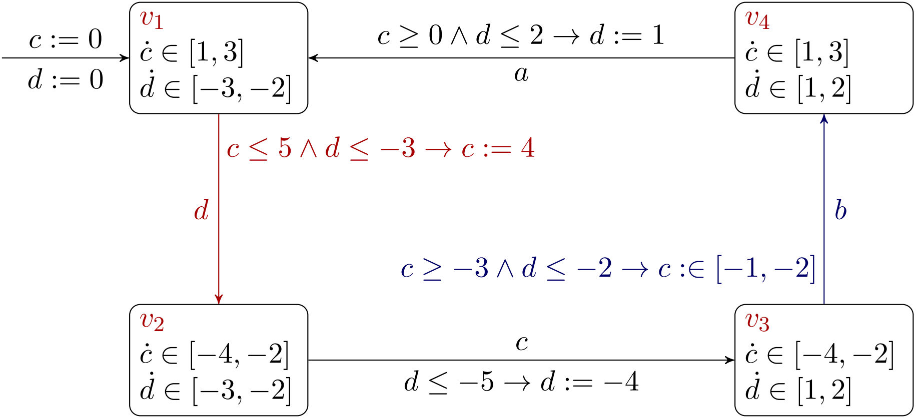
Definition

A rectangular automaton A has bounded nondeterminism, if

- all initial and flow sets are bounded, and
- for every edge e and every index i in the *jump* set of e, the interval $post_i$ of e is bounded.

Both problems becomes undecidable if one of the restrictions is relaxed.

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 language inclusion problem for initialized rectangular automata with bounded nondeterminism is complete for PSPACE.

Decidability results

Lemma

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

Lemma

The language inclusion problem 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, i.e.,

A timed automaton is a rectangular automaton with deterministic jumps, i.e.,

- $Init(l)$ is empty or a singleton for each $l \in Loc$,
- for each edge, $post_i$ is a single value for each $i \in jump$,

A timed automaton is a rectangular automaton with deterministic jumps, i.e.,

- $Init(l)$ is empty or a singleton for each $l \in Loc$,
- for each edge, $post_i$ is a single value for each $i \in jump$,

and every variable is a clock, i.e.,

A timed automaton is a rectangular automaton with deterministic jumps, i.e.,

- $Init(l)$ is empty or a singleton for each $l \in Loc$,
- for each edge, $post_i$ is a single value for each $i \in jump$,

and every variable is a clock, i.e.,

- $Act(l)(x) = [1, 1]$ for all locations l and variables x.

A timed automaton is a rectangular automaton with deterministic jumps, i.e.,

- $Init(l)$ is empty or a singleton for each $l \in Loc$,
- for each edge, $post_i$ is a single value for each $i \in jump$,

and every variable is a clock, i.e.,

- $Act(l)(x) = [1, 1]$ for all locations l and variables x.

Lemma

The reachability and the language inclusion problems for timed automata are complete for PSPACE.

Decidability results

Timed automaton

$\uparrow$

Initialized stopwatch automaton

- 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 language inclusion 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 $f : \{1, \ldots, n\} \to \kappa_-$. 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

$\uparrow$

Initialized stopwatch automaton

$\uparrow$

Initialized singular automaton

- A variable x_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 language inclusion 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

↑

Initialized stopwatch automaton

↑

Initialized singular automaton

↑

Initialized rectangular automaton

Lemma

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

Lemma

The language inclusion problem 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.