Hybrid Systems

decidable, undecidable, and in between

Eugene Asarin

LIAFA - Université Paris 7 and CNRS
Two goals

• A general introduction to Hybrid Systems from computer science standpoint
Two goals

• A general introduction to Hybrid Systems from computer science standpoint
• Decidability issues or Dynamics and computation (for HS, but this is not important)
• Hybrid Systems = Discrete + Continuous
**Introductory equations**

- **Hybrid Systems** = Discrete + Continuous
- **Hybrid Automata** = A class of models of Hybrid systems
Introductory equations

- **Hybrid Systems** = Discrete + Continuous
- **Hybrid Automata** = A class of models of Hybrid systems
- **Original motivation (1990)** = physical plant + digital controller
Introductory equations

• **Hybrid Systems** = Discrete+Continuous
• **Hybrid Automata** = A class of models of Hybrid systems
• **Original motivation (1990)** = physical plant + digital controller
• **New applications** = also scheduling, biology, economy, numerics, and more
• Hybrid Systems = Discrete+Continuous
• Hybrid Automata = A class of models of Hybrid systems
• Original motivation (1990) = physical plant + digital controller
• New applications = also scheduling, biology, economy, numerics, and more
• Hybrid community = Control scientists’ + Applied mathematicians + Some computer scientists’
1. Hybrid automata - the model
2. Reachability analysis of Hybrid systems
   - Verification and reachability problems
   - Exact methods
     - The curse of undecidability
     - Decidable classes
     - Between decidable and undecidable
     - Can realism help?
   - Approximate methods
   - Beyond reachability, beyond verification
   - Verification tools
3. Conclusions and perspectives
1. The Model
The first example

I’m sorry, a thermostat.
The first example

I’m sorry, a thermostat.

- When the heater is OFF, the room cools down:
  \[ \dot{x} = -x \]

- When it is ON, the room heats:
  \[ \dot{x} = H - x \]
I’m sorry, a thermostat.

• When the heater is OFF, the room cools down:

\[ \dot{x} = -x \]

• When it is ON, the room heats:

\[ \dot{x} = H - x \]

• When \( t > M \) it switches OFF
• When \( t < m \) it switches ON
The first example

I’m sorry, a thermostat.

- When the heater is OFF, the room cools down:
  \[ \dot{x} = -x \]

- When it is ON, the room heats:
  \[ \dot{x} = H - x \]

- When t>M it switches OFF
- When t<m it switches ON

A strange creature...
Some mathematicians prefer to write

\[ \dot{x} = f(x, q) \]

where

\[
\begin{align*}
    f(x, \text{Off}) &= -x \\
    f(x, \text{On}) &= H - x
\end{align*}
\]

with some switching rules on \( q \).
Some mathematicians prefer to write

\[ \dot{x} = f(x, q) \]

where

\[ f(x, \text{Off}) = -x \]
\[ f(x, \text{On}) = H - x \]

with some switching rules on \( q \).
But we will draw an automaton!
Hybrid automaton

On
\[ \dot{x} = H - x \]
\[ x \leq M \]
\[ x = M \]

Off
\[ \dot{x} = -x \]
\[ x \geq m \]
\[ x = m / \gamma \]

Guard

Reset

Invariant

Dynamic

Label
A formal definition: It is a tuple ...
Hybrid automaton

A formal definition: It is a tuple . . .
For those who know timed automata

\[ \dot{x} = H - x \]
\[ x \leq M \]
\[ x = M \]
\[ x = m / \gamma \]
\[ x = m \]
\[ x \geq m \]
\[ \dot{x} = -x \]

Guard
Reset
Invariant
Label
Dynamic

For those who know timed automata

<table>
<thead>
<tr>
<th>Element</th>
<th>Timed Aut.</th>
<th>Hybrid Aut.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete locations</td>
<td>$q \in Q$ (finite)</td>
<td>$q \in Q$ (finite)</td>
</tr>
<tr>
<td>Continuous variables</td>
<td>$\vec{x} \in \mathbb{R}^n$</td>
<td>$\vec{x} \in \mathbb{R}^n$</td>
</tr>
<tr>
<td>$x$ dynamics</td>
<td>$\dot{x} = 1$</td>
<td>$\dot{x} = f(x)$ (and more)</td>
</tr>
<tr>
<td>Guards</td>
<td>bool. comb. of $x_i \leq c_i$</td>
<td>$\vec{x} \in G$</td>
</tr>
<tr>
<td>Updates</td>
<td>$x_i := 0$</td>
<td>$\vec{x} := u(\vec{x})$</td>
</tr>
<tr>
<td>Invariants</td>
<td>same as guards</td>
<td>same as guards</td>
</tr>
</tbody>
</table>
For those who know timed automata

<table>
<thead>
<tr>
<th>Element</th>
<th>Timed Aut.</th>
<th>Hybrid Aut.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete locations</td>
<td>$q \in Q$ (finite) $\bar{x} \in \mathbb{R}^n$ $\dot{x} = 1$ bool. comb. of $x_i \leq c_i$ $x_i := 0$ same as guards</td>
<td>$q \in Q$ (finite) $\bar{x} \in \mathbb{R}^n$ $\dot{x} = f(x)$ (and more) $\bar{x} \in G$ $\bar{x} := u(\bar{x})$ same as guards</td>
</tr>
<tr>
<td>Continuous variables</td>
<td>$x$ dynamics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Guards</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Updates</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Invariants</td>
<td></td>
</tr>
</tbody>
</table>

Hybrid automata = a (too much) far going generalization of timed automata.
Semantic issues

• A trajectory (run) is an \( f : \mathbb{R} \rightarrow Q \times \mathbb{R}^n \)

• Some mathematical complications (notion of solution, existence and unicity not so evident).

• Zeno trajectories (infinitely many transitions in a finite period of time).
  • can be forbidden
  • one can consider trajectories up to the first anomaly (Sastry et al., everything OK)
  • one can consider the complete Zeno trajectories (very funny : Asarin-Maler 95)
Variants

- Discrete-time \((x_{n+1} = f(x_n))\) or continuous-time \(\dot{x} = f(x)\)
- Deterministic (e.g. \(\dot{x} = f(x)\)) or non-deterministic (e.g. \(\dot{x} \in F(x)\))
- Eager or lazy.
- With control and/or disturbance (e.g. \(\dot{x} = f(x, u, d)\))
- Various restrictions on dynamics, guards and resets: “Piecewise trivial dynamics”. LHA, RectA, PCD, PAM, SPDI . . . They are still highly non-trivial.
Special classes of Hybrid Automata

• The famous one: *Linear Hybrid Automata*

\[ \dot{x} = \begin{cases} c_1 & \text{if } x \in P_1 \backslash x := A_1 x + b_1 \\ c_2 & \text{if } x \in P_2 \backslash x := A_2 x + b_2 \end{cases} \]
• My favorite: $PCD = $ Piecewise Constant Derivatives

\[ \dot{x} = c_i \text{ for } x \in P_i \]
PCD is a linear hybrid automaton (LHA).
PCD is a linear hybrid automaton (LHA)
PCD is a linear hybrid automaton (LHA)
PCD is a linear hybrid automaton (LHA)

\[ \dot{x} = a_4 \quad \text{Inv}(\ell_4) \]
\[ \dot{x} = a_5 \quad \text{Inv}(\ell_5) \]
\[ \dot{x} = a_6 \quad \text{Inv}(\ell_6) \]
\[ \dot{x} = a_7 \quad \text{Inv}(\ell_7) \]
\[ \dot{x} = a_8 \quad \text{Inv}(\ell_8) \]
PCD is a linear hybrid automaton (LHA)

\[ \begin{align*}
\dot{x} &= a_4 \\
\text{Inv}(\ell_4) \\
\dot{x} &= a_5 \\
\text{Inv}(\ell_5) \\
\dot{x} &= a_6 \\
\text{Inv}(\ell_6) \\
\dot{x} &= a_7 \\
\text{Inv}(\ell_7) \\
\dot{x} &= a_8 \\
\text{Inv}(\ell_8) \\
\end{align*} \]
The most illustrative: *Piecewise Affine Maps*

\[ x := A_i x + b_i \text{ for } x \in P_i \]
How to model?

- a control system
How to model?

- a control system
- a scheduler with preemption
How to model?

- a control system
- a scheduler with preemption
- a genetic network
How to model?

- a control system
- a scheduler with preemption
- a genetic network

A network of interacting Hybrid automata
Hybrid languages

- SHIFT
- Charon
- Hysdel
- IF, Uppaal (Timed + ϵ)
- why not Simulink? or Simulink+CheckMate.
What to do with a hybrid model

- Simulate
  - With Matlab/Simulink
  - With dedicated tools
- Analyze with techniques from control science:
  - Stability analysis
  - Optimal control
  - etc..
- Analyze with your favorite techniques. The most important invention is the model.
2. Reachability
Verification and reachability problems

• Is automatic verification possible for HA?
• Is automatic verification possible for HA?
• Safety: are we sure that HA never enters a bad state?
• It can be seen as reachability: verify that
  \[ \neg \text{Reach}(Init, Bad) \]
Verification and reachability problems

• Is automatic verification possible for HA?
• Safety: are we sure that HA never enters a bad state?
• It can be seen as reachability: verify that
  \[ \neg \text{Reach}(Init, Bad) \]
• It is a natural and challenging mathematical problem.
• Many works on decidability
• Some works on approximated techniques
The reachability problem

Given a hybrid automaton $\mathcal{H}$ and two sets $A, B \subset Q \times \mathbb{R}^n$, find out whether there exists a trajectory of $\mathcal{H}$ starting in $A$ and arriving to $B$. All parameters rational.
Exact methods: The curse of undecidability

- Koiran et al.: \texttt{Reach} is undecidable for 2d PAM.
- AM95: \texttt{Reach} is undecidable for 3d PCD.
- HPKV95 Many results of the type: “3clocks + 2 stopwatches = undecidable”
Anatomy of Undecidability — Preliminaries

Proof method: simulation of 2-counter (Minsky) machine, TM etc...

- A counter: values in $\mathbb{N}$; operations: $C + +$, $C − −$; test $C > 0$?
- A Minsky (2 counter) machine
  
  $q_1 : D + +; \text{ goto } q_2$
  
  $q_2 : C − −; \text{ goto } q_3$
  
  $q_3 : \text{ if } C > 0 \text{ then goto } q_2 \text{ else } q_1$

- Reachability is undecidable (and $\Sigma^0_1$-complete) for Minsky machines.
Simulating a counter

### Counter vs. PAM

<table>
<thead>
<tr>
<th>Counter</th>
<th>PAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>State space $\mathbb{N}$</td>
<td>State space $[0; 1]$</td>
</tr>
<tr>
<td>State $C = n$</td>
<td>$x = 2^{-n}$</td>
</tr>
<tr>
<td>$C++$</td>
<td>$x := x/2$</td>
</tr>
<tr>
<td>$C--$</td>
<td>$x := 2x$</td>
</tr>
<tr>
<td>$C &gt; 0?$</td>
<td>$x &lt; 0.75?$</td>
</tr>
</tbody>
</table>
Encoding a state of a Minsky Machine

**Minsky Machine**

State space \( \{q_1, \ldots, q_k\} \times \mathbb{N} \times \mathbb{N} \)

State \((q_i, C = m, D = n)\)

<table>
<thead>
<tr>
<th>Minsky Machine</th>
<th>PAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>State space ( {q_1, \ldots, q_k} \times \mathbb{N} \times \mathbb{N} )</td>
<td>State space ([1; k + 1])</td>
</tr>
<tr>
<td>State ((q_i, C = m, D = n))</td>
<td>(x = i + 2^{-m}, \ y = 2^{-n})</td>
</tr>
</tbody>
</table>
Simulating a Minsky Machine

<table>
<thead>
<tr>
<th>Minsky Machine</th>
<th>PAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>State space ( {q_1, \ldots, q_k} \times \mathbb{N} \times \mathbb{N} )</td>
<td>State space ([1; k + 1] \times [0; 1])</td>
</tr>
<tr>
<td>State ((q_i, C = m, D = n))</td>
<td>(x = i + 2^{-m}, y = 2^{-n})</td>
</tr>
</tbody>
</table>

\[\begin{array}{l}
q_1 : D++; \text{goto } q_2 \\
q_2 : C--; \text{goto } q_3 \\
q_3 : \text{if } C > 0 \text{ then goto } q_2 \text{ else } q_1
\end{array}\]

\[
\begin{array}{ll}
x := x + 1 & \text{if } 1 < x \leq 2 \\
y := y/2 & \\
\end{array}
\]

\[
\begin{array}{ll}
x := 2(x - 2) + 3 & \text{if } 2 < x \leq 3 \\
y := y & \\
\end{array}
\]

\[
\begin{array}{ll}
x := x - 1 & \text{if } 3 < x < 4 \\
y := y & \\
\end{array}
\]

\[
\begin{array}{ll}
x := x - 2 & \text{if } x = 4 \\
y := y & \\
\end{array}
\]
we have proved that Reach is undecidable for 2d PAMs.

Undecidability proofs for other classes of HA are similar.
PCD on the blackboard
Exact methods: Decidable classes

\[ \text{Reach}(x, y) \iff \exists \text{ a trajectory from } x \text{ to } y \]

Reach is decidable for

- AD: timed automata
- HKPV95: initialized rectangular automata, extensions of timed automata
- LPY01: special linear equations + full resets.

**Method**: finite bisimulation

*(stringent restrictions on the dynamics)*

KPSY: Integration graphs???

EJCOMI - Nancy - 2007 – p. 28/6
Reach is decidable for
- MP94: 2d PCD + Key idea
- CV96: 2d multi-polynomial systems.
- ASY01: 2d “non-deterministic PCD” (wait a minute)
Simple Polygonal Differential Inclusion = the non-deterministic version of PCD =

- A partition of the plane into polygonal regions
- A constant differential inclusion for each region

\[ \dot{x} \in \angle_{a}^{b} \text{ if } x \in R_i \]
Simple Polygonal Differential Inclusion =
Difficulties

Too many trajectories (even locally)
Difficulties

Too many signatures
Difficulties

Self-crossing trajectories
Plan of solution

- Simplify trajectories
- Enumerate types of signatures
- Test reachability for each type using accelerations
Simplification 1: Straightening
Simplification 2: Removing self-crossings

Bottom line: \( \text{Reach}(x, y) \iff \exists \text{ a simple piecewise straight trajectory from } x \text{ to } y \)
Key topological remark

Simple curves on the plane are very simple (Jordan, Poincaré-Benedixson, applied by Maler-Pnueli)
Signatures of simplified trajectories

• **Representation Theorem:** Any edge signature can be represented as

\[ \sigma = r_1(s_1)^{k_1} r_2(s_2)^{k_2} \ldots r_n(s_n)^{k_n} r_{n+1} \]

• **Properties**
  - \( r_i \) is a seq. of pairwise different edges;
  - \( s_i \) is a simple cycle;
  - \( r_i \) and \( r_j \) are disjoint
  - \( s_i \) and \( s_j \) are different

Proof based on Jordan’s theorem (MP94)
Classification of signatures

Any edge signature belongs to a type

\[ r_1(s_1)^* r_2(s_2)^* \ldots r_n(s_n)^* r_{n+1} \]

There are finitely many types!
How to explore one type?

Recipe: compute successors and accelerate cycles.
Successors (by $\sigma$)

One step ($\sigma = e_1 e_2$)

$I' = \text{Succ}_{e_1 e_2}(x) = [f_b(x), f_a(x)] = F(x)$
Successors (by $\sigma$)

Several steps ($\sigma = e_1e_2e_3$)

$I'_3 = \text{Succ}_{\sigma}(x) = [f'_b(x), f'_a(x)] = F'(x)$
Successors (by $\sigma$)

Several steps ($\sigma = e_1 e_2 e_3 e_4 e_5$)

$I' = \text{Succ}_{\sigma}(x) = [f''_{b}(x), f''_{a}(x)] = F''(x)$
Successors (by $\sigma$)

One cycle ($\sigma = s = e_1 e_2 \cdots e_8 e_1$)

$I' = \text{Succ}_{\sigma}(x) = \left[ f''_b(x), f''_a(x) \right] = F''(x)$
Successors (by $\sigma$)

One cycle iterated: \(\approx\) solution of fixpoint equation (acceleration) \(\left(\text{Succ}_\sigma(I) = I\right)\)
The calculus of TAMF

- **Fact:** All successors are TAMF
- **Affine function (AF):**
  \[ f(x) = ax + b \text{ with } a > 0 \]
- **Affine multi-valued function (AMF):**
  \[ \tilde{F}(x) = [f_1(x), f_2(x)] \]
- **Truncated affine multi-valued function (TAMF):**
  \[ F(x) = \tilde{F}(x) \cap J \text{ if } x \in S \]

**Lemma:** AF, AMF and TAMF are closed under composition.

**Lemma:** Fixpoint equations \( F(I) = I \) can be explicitly solved (without iterating)
for each type of signature $\tau$ do
  test whether $x \xrightarrow{\tau} y$

To test $x \xrightarrow{\tau} y$ for $\tau = r_1(s_1)^* r_2(s_2)^* \ldots r_n(s_n)^* r_{n+1}$
compute $\text{Succ}_r$ and accelerate $(\text{Succ}_s)^*$
Main result for SPDI

Reachability is decidable for SPDI
SPeeDI the tool
Between Decidable and Undecidable
More complex 2d systems

What happens if . . .

• . . . we allow jumps?
• . . . the PCD is on a 2d surface?
• . . . ?

The answer is: we know that we do not know.

More precisely: it is equivalent to a well known open problem.
Reminder: the Reference Model

- 1d piecewise affine maps (PAMs): $f : \mathbb{R} \to \mathbb{R}$
  
  $f(x) = a_i x + b_i$ for $x \in I_i$

Old Open Problem. Is reachability decidable for 1d PAM?
Theorem. 2d LHA can simulate 1d PAM and vice versa.

Corollary. Reachability is decidable for 2d LHA iff it is decidable for 1d PAM.
LHA \equiv PAM - proof

- LHA simulates PAM

\[ \gamma(e', x, y) = (e, a_i x + b_i, 0) \]

- PAM simulates LHA
$PCD$ on surfaces $\equiv$ $iPAM$
PCD on surfaces $\equiv$ iPAM

Reachability?
PCD on surfaces \equiv iPAM

Reachability?
PCD on surfaces $\equiv$ iPAM

Reachability?
PCD on surfaces $\equiv$ iPAM

Reachability?
PCD on surfaces $\equiv$ iPAM

**Theorem.** PCDs on 2d surfaces can simulate 1d injective PAM and vice versa

**Corollary.** Reachability is decidable for PCDs on 2d surfaces iff it is decidable for 1d injective PAMs.
Local Summary

- Reachability undecidable for $\dim \geq 2$ in discrete time and $\dim \geq 3$ in continuous time
Local Summary

• Reachability undecidable for $\dim \geq 2$ in discrete time and $\dim \geq 3$ in continuous time

• Reachability decidable on the plane in continuous time without jumps.
Local Summary

- Reachability undecidable for $\dim \geq 2$ in discrete time and $\dim \geq 3$ in continuous time.
- Reachability decidable on the plane in continuous time without jumps.
- Difficult question for jumps on the plane or for 2d manifolds.
• Reachability undecidable for $\text{dim} \geq 2$ in discrete time and $\text{dim} \geq 3$ in continuous time.
• Reachability decidable on the plane in continuous time without jumps.
• Difficult question for jumps on the plane or for 2d manifolds.
• General remark: it seems that undecidability is related to chaotic dynamics.
Can realism help?

Maybe even undecidability is an artefact? Maybe it never occurs in real systems?
Proof method – Abstract View

- Proof by simulation of an infinite state machine by a DS
- State of machine $\leftrightarrow$ state of the DS
- Dynamics of DS simulates transitions of the machine
Consequences for bounded DS witnessing undecidability

- Important states (sets) of the DS are very dense (have accumulation points)
- Dynamics should be very precise (at least around accumulation points)
- It is difficult (impossible) to realize such systems physically
- ...and also: dynamics should be chaotic...

infinite state
Reachability is decidable for realistic, unprecise, noisy, “fuzzy”, “robust” systems

Arguments:

• The only known proof method uses unbounded precision (or unbounded state space)
• Noise could regularize...
• This world is nice and bad things never happen...
• Engineers design systems and never deal with undecidability.
- All the arguments are weak
- The problem is interesting
- I know 4 natural formalizations of “realism”
  - Non-zero noise: undecidable ($\Sigma_1$-hard)
  - uniform noise: open problem
  - Infinitesimal noise: undecidable and co-r.e. ($\Pi_1^0$-complete)
  - Stochastic noise: $\Delta_2^0$-complete for TM
• Both positive or negative solution would be interesting for the second one
• Most of these effects are not specific for a class of systems, they can be ported to any reasonable class.
• All this is very intriguing.
Approximate methods for reachability

- In practice approximate methods should be used for safety verification.
- Several tools, many methods.
- General principles are easy, implementation difficult.
Abstract algorithm

For example consider forward breadth-first search.

\[ F = \text{Init} \]
\[ \text{repeat} \]
\[ \quad F = F \cup \text{SuccFlow}(F) \cup \text{SuccJump}(F) \]
\[ \text{until} \quad \text{fixpoint } |(F \cap \text{Bad} \neq \emptyset) | \text{ tired} \]

A standard verification (semi-)algorithm.
How to implement it

Needed data structure for (over-)approximate representation of subsets of $\mathbb{R}^n$, and algorithms for efficient computing of

- unions, intersections;
- inclusion tests;
- SuccFlow;
- SuccJump.
Known implementations

- Polyhedra (HyTech - exact. Checkmate)
- “Griddy polyhedra” (d/dt)
- Ellipsoids (Kurzhanski, Bochkarev)
- Level sets of functions (Tomlin)
- Zonotopes (Girard)
Does it work?

Up to 10 dimensions. Sometimes.
Using advanced verification techniques

- Searching for better data-structures (SOS, *DD)
- Abstraction and refinement
- Combining model-checking and theorem proving
- Acceleration
- Bounded model-checking
Beyond verification

Generic verification algorithms + hybrid data structures allow:

• Model-checking
• Controller synthesis
• Phase portrait generation
3. Final Remarks
Conclusions for a pragmatical user

- A useful and proper model: HA. Modeling languages available.
- Simulation possible with old and new tools
- No hope for exact analysis
- In simple cases approximated analysis (and synthesis) with guarantee is possible using verification paradigm. Tools available
- (Not discussed) Some control-theoretical techniques available (stability, optimal control etc).
Perspectives for a researcher

• Obtain new decidability results (nobody cares for undecidability).
• Explore noise-fuzziness-realism issues
• Apply modern model-checking techniques to approximate verification of HS
• Create hybrid theory of formal languages
• etc.