The algorithmic analysis of hybrid system

Authors: R. Alur, C. Courcoubetis etc.
Course teacher: Prof. Ugo Buy
Xin Li, Huiyong Xiao
Nov. 13, 2002

Summary

- What’s a hybrid system?
- Definition of Hybrid Automaton
- Subclasses
- Examples
- Reachability problems of Linear Hybrid Automata
What’s a hybrid system?

• A hybrid system consists of a discrete system with an analog component.
• For example:
  – An automobile engine whose fuel injection (continuous) is regulated by a microprocessor (discrete).
  – A digital controller of an analog plant.
  – Medical equipments, manufacturing controllers, and robots etc.

What’s a hybrid system? (cont’d)

• A run of a hybrid system is a sequence of steps.
• Within each step the system state evolves continuously according to a dynamical law until a transition occurs.
• With time elapsing, when the variable changes to break the invariant condition, state transitions will take place instantaneously.
Hybrid Automaton

- Intuitively – the plant example:
  - The discrete state of the controller → vertices of a graph (locations)
  - The discrete dynamics of the controller → edges of the graph (transitions)
  - The continuous state of the plant → points in $\mathbb{R}^n$
  - The continuous dynamics of the plant → differential equations (activities)
  - Each transition may cause a discrete change in the state of the plant, as determined by a synchronization label.
  - The behavior of the controller depends on the state of the plant: when violating the invariant condition, a transition happens.

Formal definition for Hybrid Automaton

$$\mathcal{H} = (\text{Loc}, \text{Var}, \text{Lab}, \text{Edg}, \text{Act}, \text{Inv})$$

Initial conditions

$$\mathcal{I}_0: \chi(t) = e^{Kt}, \text{ so } \dot{x} = \frac{dx}{dt} = -Kx e^{Kt} = -K\chi$$

$$\mathcal{I}_1: \chi(t) = e^{Kt} + h(1 - e^{Kt}), \text{ so } \dot{x} = K(h - x)$$
Locations

• A unique name identifying each location.
• State invariants:
  – While the control stays in a location, the variables must satisfy the invariant conditions.
  – The state invariants decide how long the automaton can stay in the location.
• Flow relations:
  – How continuous variables evolve.

Arcs

• Each arc represents a state transition from a source location to a target location.
• Synchronization labels:
  – Two hybrid automaton synchronize on the common set of Synchronization labels.
• Guarded assignments:
  – Represent jump conditions using guards and update the state variables by assignments.
  – Assuming two variables $x_i$, $x_i'$ and $x_i'$ refers to the value of $x_i$ after the transition: “$x_i = x_i, x_i' = x_i'$” stands for “$x_i = x_i \land x_i' = 2x_i \land x_i' = x_i$”.
  – “$x = m$” stands for “$x = m \land x' = x$”.


Linear Hybrid Automaton

• Two concepts:
  – A linear term: a linear combination of the variables in \( \mathcal{V} \) with integer coefficients.
  – A linear formula: a boolean combination of inequalities between linear terms over \( \mathcal{V} \).

• Linear Hybrid Automaton: a time-deterministic hybrid system whose activities, invariants, and transition relations can be defined by linear expressions over the set \( \mathcal{V} \) of variables.

Special cases of Linear Hybrid Automaton

• **Discrete system**: All variables are discrete.
  – \( x \) is a discrete variable, if \( \text{Act}(l, x) = 0 \) for each \( l \in \text{Loc} \).

• **Finite-state system**: All variables are propositions.
  – \( x \) is a proposition variable, if \( \mu(e, x) \in \{0,1\} \) for each \( e \in \text{Edg} \).

• **Timed Automaton**:
  – 1) All variables are propositions or clocks,
  – 2) the linear expressions are boolean combinations of inequalities of the form \( x \# c \) or \( x \# y \# c \), where \( c \) is a nonnegative integer and \( \# \in \{<, =, =, >, =\} \).
  – \( x \) is a clock, if \( \text{Act}(l, x) = 1 \) for each \( l \), and \( \mu(e, x) \in [0,x] \) for each \( e \).
Special cases of Linear Hybrid Automaton

- **Multirate timed system**: All variables are propositions or skewed clocks.
  - \( x \) is a skewed clock, if \( \text{Act}(l, x) = k \) for each \( l \), where \( k \in \mathbb{Z} \); and \( \mu(e, x) \in \{0, x\} \) for each \( e \).
  - **N-rate timed system**: a multirate timed system whose skewed clocks proceed at \( n \) different rates.

- **Integrator system**: All variables are propositions or integrators.
  - \( x \) is an integrator, if \( \text{Act}(l, x) = \{0, 1\} \) for each \( l \) and \( \mu(e, x) \in \{0, x\} \) for each \( e \).

- **Parameter**:
  - \( x \) is a parameter, if \( \mu(e, x) = x \) for each \( e \).
  - We obtain parameterized versions of above system by admitting parameters

Example: A mutual-exclusion protocol

- The asynchronous shared-memory system that consists of two processes \( P_1 \) and \( P_2 \) with atomic read and write operations.
- Each process has a critical section and at each time instant, at most one of the two processes is allowed to be in its critical section.
Example: A mutual-exclusion protocol

reach
repeat
repeat
  await k=0
  k := i
  delay b
until k = i
Critical section
k:=0
forever

Reachability problems for Linear Hybrid Automaton

- If there is a run of system $\mathcal{H}$ that starts in state $s$ and ends in state $s'$, then the state $s'$ is reachable from the state $s$, written $\sigma \rightarrow^* \sigma'$.
- reachability question: if $\sigma \rightarrow^* \sigma'$, for two given states $s$ and $s'$ of a hybrid system.
- **Theorem 3.1.** The reachability problem is decidable for simple multirate timed systems.
- **Theorem 3.2.** The reachability problem is undecidable for 2-rate timed systems.
- **Theorem 3.3.** The reachability problem is undecidable for simple integrator systems.
The runs of a hybrid system

- A finite or infinite sequence: ([H] is the set of runs of H)
  \[ \rho: \quad \sigma_0 \mapsto_{f_0} \sigma_1 \mapsto_{f_1} \sigma_2 \mapsto_{f_2} \cdots \]
  where states \( s_i = (l_i, v_i) \in S \), nonnegative reals \( t_i \in \mathbb{R}^0 \), and activities \( f_i \in \text{Act}(l_i) \), such that for all \( i = 0 \):
  - 1. \( f_i(0) = v_i \),
  - 2. for all \( 0 = t = t_i, f_i(t) \in \text{Int}(l_i) \),
  - 3. the state \( s_{i+1} \) is a transition successor of the state \( s_i = (l_i, f_i(t_i)) \).

- For time-deterministic systems, we can omit the subscripts \( f_i \) from the next relation.
- The run \( \rho \) diverges if \( \rho \) is infinite and the infinite sum \( S_{i=0} t_i \) diverges.

The following slides are presented by Xin Li
The algorithmic analysis of hybrid system

- Research motivation
- Background
- Forward analysis
- Backward analysis
- Discussion

Research Motivation

- Purpose of automatic verification: Given a system and a correctness property, does the system satisfy the property?
Research Motivation

• Modeling of hybrid systems:
  The runs of a hybrid system: the state can change in two ways:

<table>
<thead>
<tr>
<th></th>
<th>Nature</th>
<th>Location</th>
<th>Valuation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>Instant &amp; discrete</td>
<td>Change</td>
<td>Transition Relation</td>
</tr>
<tr>
<td>Flow</td>
<td>Continuous</td>
<td>No Change</td>
<td>Activities</td>
</tr>
</tbody>
</table>

Research Motivation

• Reachability issue: Now that a run of a hybrid system is a finite/infinite sequence of “flows” and “jumps”, can we guarantee a system is safe?

“The reachability problem is central to the verification of hybrid systems… a set $R \subseteq \Sigma$ of states is an invariant of the hybrid system $H$ iff no state in $\Sigma-R$ is reachable from an initial state of $H$.”
Research Motivation

- Decidability issue: Are we always able to know if a hybrid system is safe or unsafe?

Reachability analysis is a search over an infinite state space. For linear hybrid system, the termination of this procedure is not guaranteed. Additional techniques (approximation analysis) may help the convergence of this process.

Background

- Sets
  - $\in$ membership  $\subseteq$ subset  $\cap$ set intersection  $\cup$ set union  $- \setminus$ set difference

- Quantifiers
  - Notation: $(\forall x P(x))$ “for all $x$ $P(x)$ is true.”
  - Notation: $(\exists x P(x))$ “there exists an $x$ such that $P(x)$ is true.”

- Proposition Logic:
  - A disjunction $\lor$ is true if either of its parameters are true.
  - A conjunction $\land$ is true only when both parameters (called conjuncts) are true.
Forward Analysis

- General procedure of verification process:
  Start from the initial state, then trace the state change as system runs, finally check if this process converge.

- State change during flow process:
  The forward time closure $<\text{P}>_l$ of P at $l$ is the set of valuations that are reachable from some valuation $v \in P$ $l \in \text{Loc}$, valuation $P \subseteq V, \in P$ by letting time progress.
  
  $v' \in <\text{P}>_l$ iff $\exists v \in V, t \in \mathbb{R}^+, v \in P \land \text{tcp}(v)(t) \land v' = \phi[v](t)$

---

Forward Analysis

What does it mean?

Invariant factor: \( \text{tcp}(v)(t) : time\ can\ progress\): iff $\forall 0 \leq t' \leq t, \phi[v](t) \in \text{Inv}(l)$. \( \phi[v](t) : \) activity at time \( t \).

- State change during jump process:
  
  $v' \in \text{post}_c[P] \text{ iff } \exists v \in V, v \in P \cap \text{Inv}(l) \land (v, v') \in \mu \land v' \in \text{Inv}(l)$

  $\mu$: transition relation. For a linear hybrid system:
  
  $(v, v') \in \mu$ iff $v(\psi) \land \forall x \in \text{Var}. v(\alpha_x) \leq v'(x) \leq v(\beta_x)$

  $\psi \Rightarrow \{ x := [\alpha_x, \beta_x] : x \in \text{Var} \}$
Forward Analysis

• Extension to “region” — a set of state:
  flow: \( <R>’ = l \in \text{loc} \cup (l, <R>_{l})’ \)
  jump: post[R] = \( e = (l, l’) \in \text{edge} \cup (l’, \text{post}_e[R_l]) \)

Combine them together, for the \( i \) step:
\( P_{i+1} = \text{post}_e[<P_i>’_{l_i}] \)

Proposition 4.1: least fixpoint.
Proposition 4.2: linearity of sets.

Forward Analysis

• Example:
Prove \( y \geq 60 \Rightarrow 20z \leq y \).
Forward Analysis

• Analysis:
  Initial state defined by linear formula:
  \[ \psi_1 = (pc = 1 \land x = y = z = 0) \quad pc: \text{control variable} \]
  At location 1: \[ \psi_1 = <x = y = z = 0 \lor \text{post}(2,1)[\psi_2]>_1 ' \]
  At location 2: \[ \psi_2 = <\text{false} \lor \text{post}(1,2)[\psi_1]>_2 ' \]
  For step i: \( \psi_{1,i} = \psi_{1,i-1} \lor <\text{post}(2,1)[\psi_{2,i-1}]>_1 ' \)
  \( \psi_{2,i} = \psi_{2,i-1} \lor <\text{post}(1,2)[\psi_{1,i-1}]>_1 ' \)

Forward Analysis

• Result:
  \[ \psi_R = (pc = 1 \land \psi_1) \cdot (pc = 2 \land \psi_2) \]
  \[ \psi_1 = (x \leq 1 \land x = y = z) \lor (x \leq 1 \land x \leq z \land y + 30x \geq 31z) \]
  \[ \psi_2 = (z \leq 1 \land y = x + z \land x \geq 0) \lor y \geq x + 31z - 30 \]
  Therefore, \( y \geq 60 \Rightarrow 20z \leq y. \)
Backward Analysis

• An “mirror” approach of forward analysis.

The differences:

- The initial state is the “unsafe condition”.
- “Propagation” is done “backward”
- It takes six iterations to converge.
- Converge conditions do not contain that initial state, so the original statement proven.

Discussion

• Other approaches:
  - Approximation analysis.
  - Minimization.
• Questions…