Introduction to Formal Methods for SW and HW Development

11 - Timed and Hybrid Systems: Formal Modeling and Verification

Roberto Sebastiani

Based mostly on the work and slides by Rajeev Alur, with further contributions from:
Andrea Mattioli, Paritosh Pandya, Yusi Ramadian
Trends in Model-Based Design

- **Emerging notations: UML, Stateflow**
  - Visual, Hierarchical, Object oriented
  - Simulation, code generation

- **Steady progress in model checking tools**

- **Control design employs tools** *(Matlab...)*

- **Opportunities to influence design tools**
  - Typically, semantics is not formal
  - Typically, only simulation is supported
  - ....
Advantages
Automated formal verification, Effective debugging tool

Increasing industrial success
In-house groups: Intel, Microsoft, Lucent, Motorola...
Commercial model checkers: FormalCheck by Cadence

Obstacles
Scalability is still a problem (about 500 state vars)
Effective use requires great expertise

A great success story for CS theory impacting practice (see 2007 Turing award), and a vibrant area of research
Hybrid Modeling

State machines + Dynamical systems

(on) $d\dot{x} = kx$
$x < 70$

(off) $d\dot{x} = -k'x$
$x > 60$

$\text{(notation: "$d\dot{x}$" abbreviation of "$dx/dt$")}$
Automotive Applications
Coordination Protocols
Interacting Autonomous Robots
Physics-based Animation
Biomolecular Regulatory Networks
Overview

1. Timed systems: Modeling and Semantics
   - Timed automata

2. Symbolic Reachability Analysis for Timed Systems
   - Making the state space finite
   - Region automata
   - Zone automata

3. Hybrid Systems: Modeling and Semantics
   - Hybrid automata

4. Symbolic Reachability Analysis for Hybrid Systems
   - Linear Hybrid Automata
   - Approximations of reachable sets
Acknowledgements

- Thanks for providing slides & material to:
  - Rajeev Alur & colleagues (Penn University)
  - Paritosh Pandya (IIT Bombay)
  - Andrea Mattioli, Yusi Ramadian (Univ. Trento)

- Disclaimer:
  - Very introductive
  - Only very-partial coverage
  - Mostly computer-science centric
Part 1.
Timed Systems:
Modeling and Semantics
Outline: Part 1

 esposo Timed Automata
Timed Automata
**Simple Light Control**

**WANT:** if press is issued twice quickly then the light will get brighter; otherwise the light is turned off.
Simple Light Control

Solution: Add real-valued clock $x$

Adding continuous variables and constraints to state machines
Modeling: timing constraints

- Finite graph + finite set of (real-valued) clocks

- Vertices $\Rightarrow$ locations
  - Time can elapse
  - Constraint (invariant)

- Edges $\Rightarrow$ switches
  - Reset a clock
  - Constraints

- Meaning of a clock value: time elapsed since the last time it was reset.
Timed Automata

Clocks: \( x, y \)

Guard
Boolean combination of comparisons with integer bounds

Reset
Action performed on clocks

State
( location, x=v, y=u )  where v,u are in R

2 kinds of transitions:

( n, x=2.4, y=3.1415 ) \( \xrightarrow{a} \) ( m, x=0, y=3.1415 )

( n, x=2.4, y=3.1415 ) \( \xrightarrow{\text{wait}(1.1)} \) ( n, x=3.5, y=4.2415 )

Action used for synchronization

\( x <= 5 \& y > 3 \)

\( x := 0 \)
Adding Invariants

Clocks: \( x, y \)

Transitions

\[
( n, x=2.4, y=3.1415 ) \quad \text{wait}(1.1) \quad ( n, x=2.4, y=3.1415 ) \quad \text{wait}(3.2) \quad ( n, x=3.5, y=4.2415 )
\]

Location Invariants

\[
n \quad x \leq 5 \\
( n, x=2.4, y=3.1415 ) \quad a \quad x \leq 5 \land y > 3 \\
x := 0 \\
m \quad y \leq 10 \\
g1 \quad g2 \quad g3 \quad g4
\]

Invariants ensure progress!!
Timed Automata: Formal Syntax

- A timed automaton $A = \langle L, L^0, \Sigma, X, I, E \rangle$
  - $L$ = locations; $L^0 \subseteq L$ = initial locations
  - $\Sigma$ = labels
  - $X$ = clocks
  - $I$ = Invariants
  - $E \subseteq L \times \Sigma \times 2^X \times \Phi(X) \times L$ = set of switches (edges)

A switch = $\langle s, a, \phi, \lambda, s' \rangle$
  - $s, s'$ = locations
  - $a$ = label
  - $\phi$ = clock constraints
  - $\lambda \subseteq X$ = clocks to be reset
Clock constraints and clock interpretations

• Set of clock constraints grammar:

\[ \varphi := x \leq c \mid c \leq x \mid x < c \mid c < x \mid \varphi_1 \land \varphi_2 \]

→ only allow comparison of a clock and a constant

• clock interpretation \( \nu \) :

\( X = \{x, y, z\}, \nu(X) = \{1.0, 1.5, 0\} \)

• \( \nu + \delta \) = clock interpretation after \( \delta \) time

\( \nu(X) + \delta = \{1.2, 1.7, 0.2\} \)

• \( \nu[Y := 0] \) = assigns 0 to each \( x \in Y \)

\( Y = \{y, z\}; \nu[Y := 0] = \{1.0, 0, 0\} \)
Example

- Clocks \{ x, y \}, can be re/set independently
- \( x \) is reset to 0 from \( s_0 \) to \( s_1 \) on \( a \)
- switch \( c \) happens within 1 time-unit from \( a \) because of constraints in \( s_1 \) and \( s_2 \)
- delay between \( b \) and the following \( d \) is >2
- no explicit bounds on time difference between event \( a-b \) or \( c-d \)
Semantics

- **Semantics of $A$ defined in terms of a (infinite) transition system**

$S_A = < Q, Q^0, \rightarrow, \Sigma >$, s.t.:

- $Q = \{ (s, v) \}$
- $Q^0 = \{ (s, v) \}$ where $s \subseteq L^0$ and $v(x)=0$
- $\rightarrow$:
  - State change due to elapse of time
  - State change due to a location switch
- $\Sigma =$ set of labels of $A \cup \mathcal{R}$
State change in transition system

\( q \quad \text{x<2} \quad (q,0) \quad q' \quad a \quad (q,0) \quad \text{Initial state} \)
State change in transition system

- $q(x<2)$
- $(q,0) \xrightarrow{1.2} (q,1.2)$
- $= \text{state change due to elapse of time}$
State change in transition system

- \((q,0) \xrightarrow{1.2} (q,1.2) \xrightarrow{a} (q',1.2)\)
- State change due to a location switch

- \(q \xrightarrow{1.2} q\) and \(q \xrightarrow{a} q' = q \xrightarrow{1.2+a} q'\)
Example: Light Switch

- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.
Example: Light Switch (cont.)

\[(\text{off}, x = y = 0) \xrightarrow{3.5} (\text{off}, x = y = 3.5) \xrightarrow{\text{push}} (\text{off}, x = y = 3.5)
\]

\[(\text{on}, x = y = 0) \xrightarrow{\pi} (\text{on}, x = y = \pi) \xrightarrow{\text{push}} (\text{on}, x = y = \pi)
\]

\[(\text{on}, x = 0, y = \pi) \xrightarrow{3} (\text{on}, x = 3, y = \pi + 3) \xrightarrow{9-(\pi+3)} (\text{off}, x = 0, y = 9)
\]

\[(\text{on}, x = 9-(\pi+3), y = 9) \xrightarrow{\text{click}} (\text{off}, x = 0, y = 9)\ldots\]
Remark: non-zenoneness

1. When the invariant is violated some edge must be enabled

2. Automaton should admit the possibility of time to diverge
Combination of systems

- Complex system = product of interacting transition systems

- $S1 = \langle L_1, L_1^0, \Sigma_1, X_1, I_1, E_1 \rangle$ and $S2 = \langle L_2, L_2^0, \Sigma_2, X_2, I_2, E_2 \rangle$

- Product = $S1 \parallel S2 = \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, I_1 \land I_2, E \rangle$

- Transition iff:
  
  i. Label a belongs to both alphabets $\rightarrow$ synchronized
     (synchronization is blocking: an a-labeled switch cannot be shot singularly)
  
  ii. Label a only in the alphabet of $S1$ $\rightarrow$ asynchronized
  
  iii. Label a only in the alphabet of $S2$ $\rightarrow$ asynchronized
Transition product

- $\Sigma_1 = \{a, b\}$
- $\Sigma_2 = \{a, c\}$
Train - gate controller

- Desired property: $AG(s_2 \rightarrow t_2)$
Product Construction

- \( y := 0, z = 1 \)
- \( x := 0, z := 0 \)
- \( x \leq 5 \land z \leq 1 \)
- \( x > 2 \)

Transition conditions:
- \( y := 0, z = 1 \)
- \( x := 0, z := 0 \)
- \( x \leq 5 \land z \leq 1 \)
- \( x > 2 \)
Part 2. Symbolic Reachability Analysis for Timed Systems
Outline: Part 2

- Reachability analysis
- Making the state space finite
- Region automata
- Zone automata
Reachability Analysis

- Verification of safety requirement: reachability problem
- Input: a time-automaton \( A \) and a set of target locations \( L^F \subseteq L \)
- Determining whether \( L^F \) is reachable in a timed automaton \( A \)
- Location \( s \) of \( A \) is reachable if some state \( q \) with location component \( s \) is a reachable state of the transition system \( S_A \)
Timed/hybrid Systems: problem

The transition system $S_A$ associated to $A$ has infinitely-many states & symbols:

Is finite state analysis possible?
Is reachability problem decidable?

Is finite state analysis possible?
Is reachability problem decidable?
Goal: To partition state-space into finitely many equivalence classes so that equivalent states exhibit similar behaviors
Reachability analysis

**Semantics**

Timed Automaton \( S_A \)

Time-abstract automaton

Region Automaton

Regions:
- Both states and actions are finite sets

Finite set of actions
Infinite set of states.

Infinite set of actions
Infinite set of states.
Outline: Part 2

✓ Reachability Analysis

✂ Making the state space finite

☐ Region automata

☐ Zone automata
Timed Vs Time-Abstract Relations

Transition system associated with a timed/hybrid automaton $A$:

- $S_A$: Labels on continuous steps are delays in $\mathbb{R}$
- Actual delays are suppressed (all continuous steps have same label): $\text{Time-abstract } U_A$
Time-abstract transition system $U_A$

- Only change due to location switch stated explicitly
- Cut system to finitely many labels
- $U_A$ (instead of $S_A$) allows for capturing untimed properties (e.g., reachability, safety)

\[
\begin{align*}
(s_0,0,0) & \xrightarrow{1.2} (s_0,1.2,1.2) \xrightarrow{a} (s_1,0,1.2) \xrightarrow{0.7} (s_1,0.7,1.9) \xrightarrow{b} (s_2,0.7,0) \\
(s_0,0,0) & \xrightarrow{a} (s_1,0,1.2) \xrightarrow{b} (s_2,0.7,0)
\end{align*}
\]
Stable quotients

- Cut to finitely many states
- Collapse equivalent states
- Stable equivalence relation

$\text{Quotient of } U_A = \text{transition system } [U_A]_\sim$
$L^F$-sensitive equivalence relation

- Equivalence relation $\sim = L^F$-sensitive
- All equivalent states in a class belong to either $L^F$ or not $L^F$
Outline: Part 2

- Reachability Analysis
- Making the state space finite
  - Region Automata
- Zone automata
Region Equivalence over clock interpretation

- \( x = 3.7 \)
  - Integral part: \( \lfloor \nu(x) \rfloor = 3.0 \)
  - Fractional part: \( \text{fr}(\nu(x)) = 0.7 \)

- \( \nu \equiv \nu' \) iff

1. For all \( x \), the integral part is the same or both exceed \( c_x \) pict.
   \( (c_x: \text{max constant } x \text{ is compared to}) \)

2. For all \( x, y \) with \( \nu(x) \leq c_x \) and \( \nu(y) \leq c_y \), \( \text{fr}(\nu(x)) \leq \text{fr}(\nu(y)) \) iff \( \text{fr}(\nu'(x)) \leq \text{fr}(\nu'(y)) \) pict.

3. For all \( x, \nu(x) \leq c_x, \text{fr}(\nu(x)) = 0 \) iff \( \text{fr}(\nu'(x)) = 0 \) pict.
Constraint 1

For all $x$, the integral part is the same or both exceed $cx$

$$\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor = 1$$
Constraint 2

For all $x$, $y$ with $v(x) \leq cx$ and $v(y) \leq cy$, $fr(v(x)) \leq fr(v(y))$ iff $fr(v'(x)) \leq fr(v'(y))$
Constraint 3

For all $x$, $v(x) \leq cx$, $fr(v(x)) = 0$ iff $fr(v'(x)) = 0$
Clock regions

- Clock region: equivalence class of clock interpretation $\rightarrow$ finite
- Max number of regions $= k! \cdot 2^k \cdot \prod_{x \in X} (2c_x + 2)$
  - (k is the number of clocks)
- Number of clock regions exponential in the encoding of the clock constraints
Regions, intuitive idea:
Finite partitioning of state space

Definition

An equivalence class (i.e. a region) in fact there is only a finite number of regions!!

\[ w \equiv w' \text{ iff they satisfy the same set of constraints of the form} \]
\[ x_i < c, x_i = c, x_i - x_j < c, x_i - x_j = c \]
for \( c \leq \text{largest const relevant to } x_i \)
An equivalence class (i.e. a region)
Properties of Regions

- The region equivalence relation $\approx$ is a time-abstract bisimulation:
  - Action transitions: If $w \approx v$ and $(l,w) \xrightarrow{a} (l',w')$ for some $w'$, then $\exists v' \approx w'$ s.t. $(l,v) \xrightarrow{a} (l',v')$
  - Delay transitions: If $w \approx v$ then for all real numbers $d$, there exists $d'$ s.t. $w+d \approx v+d'$

- If $w \approx v$ then $(l,w)$ and $(l,v)$ satisfy the same temporal logic formulas
Region automaton

- Equivalent states = identical location + region-equivalent clock
- Classes = finite, stable, $L^F$-sensitive
- Region automaton of $A = R(A)$
  - $Q = \text{equivalence classes of } (s, \nu)$
- Reachability problem $(A, L^F) \rightarrow \text{search } R(A)$
Region graph of a simple timed automata
Complexity of reachability

- Linear with the number of locations
- Exponential with the number of clocks
- Exponential in the encoding of constants

→ PSPACE-complete
Outline: Part 2

☐ Reachability Analysis
☐ Making the state space finite
☐ Region Automata
☒ Zone Automata
Zone automata

- Collapse regions by convex unions of clock regions
- Clock zone $\varphi = \text{set of clock constraints } x-y \leq c, x-y < c, x < c, x \leq c, x = c, x > c, x \geq c$
- $\varphi = \text{convex set in the k-dimensional euclidean space}$
  $\rightarrow \text{Contains all possible relationship for all clock value in a set}$
Zones: symbolic representation

State
(s, x=3.2, y=2.5)

Symbolic state (set of states)
(s, 1≤x≤4,1≤y≤3)

Zone:
conjunction of
x-y≤c, x-y<0,
x<c,x≤c,x=c,x>c,x≥c
Zone automaton

- $Z(A)$ is a transition system $\langle Q, Q^0, \Sigma, \rightarrow \rangle$ s.t.
  - $Q =$ zone of $A$; $\text{Zone} = (s, \phi)$
  - $Q^0 = (s, [X:=0])$, for every initial location $s$ of $A$
  - $\Sigma =$ set of labels or events
  - $\rightarrow = ((s, \phi), a, (s', \text{succ}(\phi, e)))$
    - $\text{succ}(\phi, e) =$ clock interpretation after executing $e$
Symbolic transition

- $\text{succ}(\varphi, e) = (((\varphi \land I(s)) \uparrow) \land I(s) \land \psi)[\lambda := 0]$

1. $\uparrow =$ intersection
2. $\varphi \uparrow =$ interpretation for $\nu + \delta$
3. $\varphi[\lambda := 0] =$ interpretation $\nu[\lambda := 0]$

- closure under the three operations $\Rightarrow$ still a convex set

$\Phi$ fulfill invariant of state $s$
still fulfill invariant of state $s$ after time elapse
fulfill the time constraint of switch $e$
Symbolic Transitions

Thus \((n, 1 \leq x \leq 4, 1 \leq y \leq 3) \implies (m, 3 < x, y = 0)\)
Canonical Data-structures for Zones: Difference-bound Matrices

- Matrix representation of constraints (bounds on a single clock or difference between 2 clocks)
- Reduced form obtained by running all-pairs shortest path algorithm
- Reduced DBM is canonical
- Operations such as reset, time-successor, inclusion, intersection are efficient
- Popular choice in timed-automata-based tools
Difference-bound matrices (DBM)

- \( k \) clocks = \((k+1) \times (k+1)\) matrix \( D \)
- **Example**: \( (0 \leq x_1 < 2) \land (0 < x_2 < 1) \land (x_1 - x_2 \geq 0) \)

### Matrix \( D \)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(\infty)</td>
<td>(0,1)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>1</td>
<td>(2,0)</td>
<td>(\infty)</td>
<td>(\infty)</td>
</tr>
<tr>
<td>2</td>
<td>(1,0)</td>
<td>(0,1)</td>
<td>(\infty)</td>
</tr>
</tbody>
</table>

- \( D_{0i} = \) lower bound
- \( D_{i0} = \) upper bound
- \( D_{ij} = \) upper bound of \( x_i \) and \( x_j \) difference
- \( i,j: (c,1) \rightarrow Xi - Xj \leq c \)
- \( i,j: (c,0) \rightarrow Xi - Xj < c \)
- \( i,j: \infty \rightarrow \) absence of bound
Difference-bound matrices (DBM)

• Upper bound of $x_i - x_j = \text{sum of the upper bounds of } x_i - x_j$ and $x_j - x_i$
• Use all-pairs shortest paths, check DBM
  Satisfiable $\rightarrow$ Canonical
    - Satisfiable = a nonempty clock zone
    - Canonical = Matrices with tightest possible constraints
• Canonical Dbms represent clock zones

<table>
<thead>
<tr>
<th></th>
<th>Matrix $D$</th>
<th></th>
<th>Matrix $D'$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 2</td>
<td></td>
<td>0 1 2</td>
</tr>
<tr>
<td>0</td>
<td>$\infty$ (0,1) (0,0)</td>
<td>(0,1) (0,1) (0,0)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(2,0) $\infty$ $\infty$</td>
<td>(2,0) (0,1) (2,0)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(1,0) (0,1) $\infty$</td>
<td>(1,0) (0,1) (0,1)</td>
<td></td>
</tr>
</tbody>
</table>
Canonical Data-structures for Zones: Difference Bounded Matrices

When are two sets of constraints equivalent?

D1

<table>
<thead>
<tr>
<th>x &lt;= 1</th>
<th>y-x &lt;= 2</th>
<th>z-y &lt;= 2</th>
<th>z &lt;= 9</th>
</tr>
</thead>
</table>

Graph

Shortest Path Closure

D2

<table>
<thead>
<tr>
<th>x &lt;= 1</th>
<th>y-x &lt;= 2</th>
<th>y &lt;= 3</th>
<th>z-y &lt;= 2</th>
<th>z &lt;= 7</th>
</tr>
</thead>
</table>

Graph

Shortest Path Closure
Complexity

• Theoretically:
  - Zone automaton may be exponentially bigger than the region automaton

• Practically:
  - Fewer reachable vertices
    → performances much improved
Implementation

• Verification problem :
  - Input = timed automaton $A_i$
  - Process = searching $R(∥_iA_i)$ or $Z(∥_iA_i)$
    • BDD-based engine (preferably for region construction)
    • On-the-fly enumerative search (preferably for zone construction)
Timed Automata: summary

- Only continuous variables are timers
- Invariants and Guards: \( x < \text{const}, \ x > = \text{const} \)
- Actions: \( x := 0 \)
- Reachability is decidable
- Clustering of regions into zones desirable in practice
- Tools: Uppaal, Kronos, RED ...
- Symbolic representation: matrices
- Techniques to construct timed abstractions of general hybrid systems
Decidable Problems

- Model checking branching-time properties of timed automata
- Reachability in rectangular automata
- Timed bisimilarity: are two given timed automata bisimilar?
- Optimization: Compute shortest paths (e.g. minimum time reachability) in timed automata with costs on locations and edges
- Controller synthesis: Computing winning strategies in timed automata with controllable and uncontrollable transitions
Part 3.
Hybrid Systems: Modeling and Semantics
Outline: Part 3

Hybrid Automata
Hybrid Automata
Hybrid Automata

Locations or modes (discrete states)

Initial condition

Continuous dynamics

Invariant: Hybrid automaton may remain in \( l \) as long as \( X \in \text{Inv}(l) \)

Jump transformation

\( e : g(X) \geq 0 \)

\( J(X, X') \)

\( X \in \text{Inv}(l') \)

\( dX \in \text{Flow}(l') \)
Switched Dynamic Systems

continuous dynamics

flow constraints

F$_1$

F$_2$

F$_3$

x(t)

m(t)

mode select

xdot(t)

integrator

1

S

cont. state

discrete state

discrete event

threshold-driven discrete dynamics

x(t)

e(t)

J$_e$

jump mapping

e(t)

jump dynamics

initial condition

J$_e$

x(t)

continuous dynamics
Hybrid Automata

- Set $L$ of locations, and set $E$ of edges
- Set $X$ of $k$ continuous variables
- State space: $L \times \mathbb{R}^k$, Region: subset of $\mathbb{R}^k$
- For each location $l$,
  - Initial states: region $\text{Init}(l)$
  - Invariant: region $\text{Inv}(l)$
  - Continuous dynamics: $dX$ in $\text{Flow}(l)(X)$
- For each edge $e$ from location $l$ to location $l'$
  - Guard: region $\text{Guard}(e)$
  - Update relation “Jump” over $\mathbb{R}^k \times \mathbb{R}^k$
  - Synchronization labels (communication information)
(Finite) Executions of Hybrid Automata

- State: \((l, x)\) such that \(x\) satisfies \(\text{Inv}(l)\)
- Initialization: \((l, x)\) s.t. \(x\) satisfies \(\text{Init}(l)\)
- Two types of state updates
  - Discrete switches: \((l, x) \rightarrow (l', x')\) if there is an \(a\)-labeled edge \(e\) from \(l\) to \(l'\) s.t. \(x\) satisfies \(\text{Guard}(e)\) and \((x, x')\) satisfies update relation \(\text{Jump}(e)\)
  - Continuous flows: \((l, x) \rightarrow (l, x')\) where \(f\) is a continuous function from \([0, \delta]\) s.t. \(f(0) = x\), \(f(\delta) = x'\), and for all \(t \leq \delta\), \(f(t)\) satisfies \(\text{Inv}(l)\) and \(df(t)\) satisfies \(\text{Flow}(l)(f(t))\)
Example of (linear) Hybrid Automaton Gate for a railroad controller

```
Example of (linear) Hybrid Automaton Gate for a railroad controller

h = 90
Open
h = 90
dh = 0
?lower
lowering
h >= 0
-10 < dh < -9
raising
h <= 90
8 < dh < 10
?lower
?raise
h = 0
closed
h = 0
dh = 0
```
Part 4.
Symbolic Reachability Analysis for Hybrid Systems
Outline: Part 4

- Symbolic Reachability Analysis
- Linear Hybrid Automata (HyTech)
- Polyhedral Flow-pipe Approximations (CheckMate)
Standard Reachability Problem

Model variables $X = \{x_1, \ldots, x_n\}$

- Each var is of finite type, say, boolean

Initialization: $I(X)$ condition over $X$

Update: $T(X,X')$

- How new vars $X'$ are related to old vars $X$ as a result of executing one step of the program

Target set: $F(X)$

Computational problem:

- Can $F$ be satisfied starting with $I$ by repeatedly applying $T$?

Graph Search problem
General Symbolic Solution

Data type: region to represent state-sets

R := I(X)

Repeat

If R intersects F report “yes”
Else if R contains Image(R) report “no”
Else R := R union Image(R)

Image(R): Set of successors of states in R
Termination may or may not be guaranteed
Symbolic Representations

- **Necessary operations on Regions**
  - Union
  - Intersection
  - Negation
  - Projection
  - Renaming
  - Equality/containment test
  - Emptiness test

- **Different choices for different classes**
  - BDDs for boolean variables in hardware verification
  - Size of representation as opposed to number of states
Reachability for Hybrid Systems

- Same algorithm works in principle
- What’s a suitable representation of regions?
  - Region: subset of $\mathbb{R}^k$
  - Main problem: handling continuous dynamics
- Precise solutions available for restricted continuous dynamics
  - Timed automata
  - Linear hybrid automata
- Even for linear systems, over-approximations of reachable set needed
Reachability Analysis for Dynamical Systems

- Goal: Given an initial region, compute whether a bad state can be reached
- Key step is to compute Reach(X) for a given set X under $\frac{dx}{dt} = f(x)$ (hereafter $dx = f(x)$ for short)
Outline: Part 4

☑ Symbolic Reachability Analysis
_format
Linear Hybrid Automata (HyTech)
☑ Polyhedral Flow-pipe Approximations (CheckMate)
Multi-rate Automata

- Modest extension of timed automata
  - Dynamics of the form $dx = \text{const}$ (rate of a clock is same in all locations)
  - Guards and invariants: $x < \text{const}, x > \text{const}$
  - Resets: $x := \text{const}$

- Simple translation to timed automata that gives time-abstract bisimilar system by scaling

\[
\begin{align*}
\frac{dx}{dt} &= 2 \\
\frac{dy}{dt} &= 3 \\
x > 5 \text{ and } y < 1 \\
\frac{du}{dt} &= 1 \\
\frac{dv}{dt} &= 1 \\
u > 5/2 \text{ and } v < 1/3
\end{align*}
\]
Rectangular Automata

- Interesting extension of timed automata
  - Dynamics of the form $dx$ in const interval (rate-bounds of a clock same in all locations)
  - Guards/invariants/resets as before

- Translation to multi-rate automata that gives time-abstract language-equiv system

\[\begin{align*}
\text{dx in [2,3]} & : x < 2 \\
x > 5 & \\
\text{dx} & = 2 \\
\text{dx}_M & = 3 \\
x_M > 5, x_m := 5 & \\
x_m < 2, x_M := 2
\end{align*}\]
Linear Hybrid Automata

- Invariants and guards: linear (Ax <= b)
- Actions: linear transforms (x' := Ax)
- Dynamics: time-invariant, state-independent specified by a convex polytope constraining rates
  E.g. 2 < x <= 3, \dot{x} = \dot{y}

- Tools: HyTech
- Symbolic representation: Polyhedra
- Methodology: abstract dynamics by differential inclusions bounding rates
Example LHA
Gate for a railroad controller

Open
h = 90
dh = 0

lowering
h >= 0
-10 < dh < -9

raising
h <= 90
8 < dh < 10

closed
h = 0
dh = 0

h = 90

?lower

?lower

?lower

?raise

?raise

h = 0
Reachability Computation

Basic element: \((\text{location } l, \text{polyhedron } p)\)
Set of visited states: a list of \((l,p)\) pairs

Key steps:
- Compute “discrete” successors of \((l,p)\)
- Compute “continuous” successor of \((l,p)\)
- Check if \(p\) intersects with “bad” region
- Check if newly found \(p\) is covered by already visited polyhedra \(p_1, \ldots, p_k\) (expensive!)
Computing Discrete Successors

Discrete successor of (l,p)
- Intersect p with g (result r is a polyhedron)
- Apply linear transformation A to r (result r' is a polyhedron)
- Intersect r' with the invariant of l' (result r'' is a polyhedron)
- Successor is (l',r'')

\[ g(x) \rightarrow x := a(x) \]
Thm: If initial set $p$, invariant $I$, and rate constraint $r$, are polyhedra, then set of reachable states is a polyhedron (and computable)

Basically, apply extremal rates to vertices of $p$
Summary: Linear Hybrid Automata

- HyTech implements this strategy
- Core computation: manipulation of polyhedra
- Bottlenecks
  - proliferation of polyhedra (unions)
  - computing with higher dimensional polyhedra
- Many case studies (active structure control, Philips audio control protocol, steam boiler...)

Outline: Part 4

✓ Symbolic Reachability Analysis
✓ Linear Hybrid Automata (HyTech)
Fat Polyhedral Flow-pipe Approximations (CheckMate)
Beyond LHA

- Exact computation with polyhedra is limiting.
- If dynamics is $dX = AX$, and $P$ is a polyhedron, $\text{Reach}(P)$ is not a polyhedron.
- Solutions:
  - Approximate Reach($P$) with an enclosing convex polyhedron: Checkmate (Krogh)
  - Approximate Reach($P$) with an enclosing (non-convex) orthogonal polyhedron: $d/dt$ (Dang/Maler)
  - Level sets method (Greenstreet, Tomlin)
  - Use ellipsoids for representation of sets (Kurzhanski)
• divide $R_{[0,T]}(X_0)$ into $[t_k,t_{k+1}]$ segments

• enclose each segment with a convex polytope

• $R^M_{[0,T]}(X_0) = \text{union of polytopes}$
Wrapping Hyperplanes Around a Set

Step 1:
Choose normal vectors, \( c_1, \ldots, c_m \)
Wrapping Hyperplanes Around a Set

Step 2:
Compute optimal $d$ in $Cx \leq d$, $C^T = [c_1 \cdots c_m]$:

$$d_i = \max_{x \in S} c_i^T x$$
Wrapping a Flow Pipe Segment

Given normal vectors $c_i$, we wrap $R_{[t_k,t_{k+1}]}(X_0)$ in a polytope by solving for each $i$

$\begin{align*}
    d_i &= \max_{x_0,t} c_i^T x(t,x_0) \\
    &\text{s.t. } x_0 \in X_0 \\
    &t \in [t_k,t_{k+1}]
\end{align*}$

Optimization problem is solved by embedding simulation into objective function computation.
Improvements for Linear Systems

- \( \dot{x} = Ax \Rightarrow x(t, x_0) = e^{At}x_0 \)
- No longer need to embed simulation into optimization
- Flow pipe segment computation depends only on time step \( \Delta t \)
- A segment can be obtained by applying \( e^{At} \) to another segment of the same \( \Delta t \)

\[
\hat{R}_{[t,t+\Delta t]}(X_0) = e^{At} \hat{R}_{[0,\Delta t]}(X_0)
\]
Example: Van der Pol Equation

Van der Pol Equation
\[ \dot{x}_1 = x_2 \]
\[ \dot{x}_2 = -0.2(x_1^2 - 1)x_2 - x_1 \]

Initial Set
\[ X_0 = \{0.8 \leq x_1 \leq 1, x_2 = 0\} \]

Uniform time step
\[ \Delta t_k = 0.5 \]
Summary: Flow Pipe Approximation

- Applies in arbitrary dimensions
- Approximation error doesn't grow with time
- Estimation error (Hausdorff distance) can be made arbitrarily small with $\Delta t < \delta$ and size of $X_0 < \delta$
- Integrated into a complete verification tool (CheckMate)