Automatic Analysis of Hybrid Systems
A Constraint-Solving Perspective

Martin Fränzle

Research Group Hybrid Systems
Department of Computing Science
Carl v. Ossietzky Universität Oldenburg
Germany

Trustworthy Software Systems
Certification
Quality of Service
Component Technology

DFG Graduate School “TrustSoft”
http://trustsoft.uni-oldenburg.de

SFB/TR 14 AVACS
Agenda

1. Embedded systems
   - the term
   - automata-based verification

2. Hybrid discrete-continuous systems
   - the term
   - need for a dedicated theory

3. Automatic methods for hybrid-system analysis
   - Reachability analysis by iterating the transition relation
     - Decidable cases (very confined...)
     - Decidable wrt. depth-bounded reachability
     - Undecidable even in the bounded (the truly interesting stuff...)
   - Some other verification schemes
Embedded computer systems

Cogito, ergo sum!

Estimates for number of embedded systems in use exceed $10^{10}$.

[Rammig 2000, Motorola 2001]
Application domains

- **Consumer & household products:**
  CD players, TV sets, handheld games, electronic pets, cameras, alarm clocks, remote controls, dishwashers, microwave ovens, ...

- **Office, telecommunications, etc.:**
  Printers, network controllers, mobile phones, keyboards, CRTs and flatscreens, ...

- **Environmental control:**
  λ control, programmable heating systems, exhaust control, ...

- **Traffic systems and traffic management:**
  Cars (body, powertrain, suspension, brakes), signalling devices, balises, interlocks, autopilots, traffic information, ...

- **Medicine:**
  Measurement devices (thermometers, RR’s, X-ray, sonographic imaging, EEG, ECG ...), treatment devices (perfusors, respirators, microwave radiation treatment, ...)

- **Supplies:**
  Power plants, distribution networks, ...
Formal methods are *mathematically-based techniques* for the specification, development and verification of software and hardware systems. [R.W. Butler, 2001]

Motivated by the expectation that appropriate *mathematical analyses* can contribute to the reliability and robustness of a design. [M. Holloway, 1997]

Alternative to less exhaustive analyses:

[cartoon]
Automatic Analysis of Embedded Systems

A first idea:
Automata-theoretic verification
Safety requirement: Gate has to be closed whenever a train is in “In”.
The gate model

Opening ~enter?

Closing enter?

Closed leave?

~leave?

Track model

--- safe abstraction ---

Empty enter! Appr.

leave! In
Automatic check
Verification result

Stimuli: Empty, Approach, In
Gate reaction: Open, Closing, Closed, Opening, Open.
## Formal Methods vs. Simulation

<table>
<thead>
<tr>
<th></th>
<th>Environment dynamics exactly known</th>
<th>Environment dynamics partially known</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simulation</strong></td>
<td>efficient, reasonably exact, yet confined coverage of open systems</td>
<td>at most (rough) approximation, i.e. valid over short time frames only</td>
</tr>
<tr>
<td><strong>Formal verific.</strong></td>
<td>full coverage of open systems, at the price of computational complexity</td>
<td>coverage of all possible instances, at the price of computational complexity</td>
</tr>
</tbody>
</table>
Would you like to draw the automaton?
Further problems

- How to obtain the finite-state abstraction of the environment?
- How to justify it?
- How to decide whether a counterexample is real?
  (Could well be an artifact of the abstraction of the environment.)
Let the mass media help us...

"Math is more than formulas or equations; it's logic, it's rationality, it's using your mind to solve the biggest mysteries we know."
Hybrid Systems
What is a hybrid system?


*Hybrid* (from Greece) means *arrogant, presumptuous*. Other interpretations [in particular, in scientific jargon] have been added later.

After H. Menge: Griechisch/Deutsch, Langenscheidt 1984

⇒ when you try to verify hybrid systems, be prepared that they may act like a prima donna!
Hybrid Systems

Plant Control

Loads of continuous computations interleaved with discrete decisions

Plant observable state

Disturbances ("noise")

Environmental influence

Control

Analog switch

Continuous controllers

Discrete supervisor

Setpoints

Active control law

Setpoints part of observable state

Task selection
Hybrid systems

are ensembles of interacting discrete and continuous subsystems:

- **Technical systems:**
  - physical plant + multi-modal control
  - physical plant + embedded digital system
  - mixed-signal circuits
  - multi-objective scheduling problems (computers / distrib. energy management / traffic management / ...)

- **Biological systems:**
  - Delta-Notch signaling in cell differentiation
  - Blood clotting
  - ...

- **Economy:**
  - cash/good flows + decisions
  - ...

- **Medicine/health/epidemiology:**
  - infectious diseases + vaccination strategies
  - ...
Discrete vs. continuous

A discrete system

- operates on a state,
- performs discontinuous state changes at discrete time points,
- state is constant in between

E.g., a program

- Prog. variables, position
- Computation steps: assignments, ctrl. flow
- Stable states

Validation by
- Program verification
- State exploration
Discrete vs. continuous

A continuous system

- operates on a continuous state,
- which evolves continuously.

E.g., a ball
Height, speed
Newtonian mechanics

Validation:
- Analytically
- Simulation + continuity
Interaction of continuous dynamics and discrete mode switch destroys global convergence!
A Formal Model: Hybrid Automata

The ball is moving down and up:

- Vertical position of the ball: $x = 20.0 \land y = 0.0$
- Velocity: $y' = -0.8 \cdot y$

In the diagram:
- $x$: vertical position of the ball
- $y$: velocity
  - $y > 0$: ball is moving up
  - $y < 0$: ball is moving down
A Formal Model: Hybrid Automata

\( x = 20.0 \land y = 0.0 \)

- \( x = y \)
- \( y = -9.81 \)
- \( x \geq 0 \)

\( x = 0.0 \land y \leq 0.0 \) /  
\( y' = -0.8 \cdot y \)

\( x : \) vertical position of the ball  
\( y : \) velocity  
  - \( y > 0 \) ball is moving up  
  - \( y < 0 \) ball is moving down
An Example of HS Analysis

Train Separation in ETCS Level 3
Example: Train Separation in Absolute Braking Distance

Minimal admissible distance $d$ between two successive trains equals braking distance $d_b$ of the second train plus a safety distance $S$.

First train reports position of its tail to the second train every 8 seconds.
Controller in second train automatically initiates braking to maintain a safe distance.
Property to be checked: Does the controller guarantee that collisions don’t occur in any possible scenario of use?
Analysis of Matlab/Simulink Model

Simulation of the Model

Error Trace found by HySAT

- 66 unwindings
- 7809 variables
- 69968 decisions
- 27047 conflicts
- $\approx 5 \times 10^8$ assignments
- $\approx 20$ minutes
The morale

_Tausend mal simuliert,
tausend mal ist nix passiert.
Einmal verifiziert,
und es hat “bumm” gemacht.

(a variation on “Tausend mal berührt” by Klaus Lage)
Hybrid Automata

The formal model
Hybrid automata

\[ \text{Hybrid systems} \quad = \quad \text{Coupled digital \& analog systems} \]

\[ \downarrow \]

\[ \text{Hybrid automata} \quad = \quad \text{Finite automata with} \]
\[ \quad \bullet \text{immediate transitions that are} \]
\[ \quad \bullet \text{triggered by predicates on the (continuous) plant state} \]

\[ + \text{evolution of the continuous plant} \]
\[ \bullet \text{real-valued variables governed by} \]
\[ \bullet \text{a set of (restricted) differential equations that are} \]
\[ \bullet \text{selected by the current automaton state} \]
Hybrid Automaton (w/o input) [after K.H. Johansson]

Def: a **hybrid automaton** $H$ is a tuple $H = (V, X, f, Init, Inv, Jump)$, where:

- $V$ is a *finite* set of **discrete modes**. The elements of $V$ represent the discrete states.

- $X = \{x_1, \ldots, x_n\}$ is an (ordered) finite set of **continuous variables**. A real-valued valuation $z \in \mathbb{R}^n$ of $x_1, \ldots, x_n$ represent a continuous state.

- $f \in V \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ assigns a **vector field** to each mode. The dynamics in mode $m$ is $\frac{dx}{dt} = f(m, x)$.

- $Init \subseteq V \times \mathbb{R}^n$ is the initial condition. $Init$ defines the admissible initial states of $H$.

- $Inv \subseteq V \times \mathbb{R}^n$ specifies the **mode invariants**. $Inv$ defines the admissible states of $H$.

- $Jump \in V \times \mathbb{R}^n \rightarrow \mathcal{P}(V \times \mathbb{R}^n)$ is the jump relation. $Jump$ defines the possible discrete actions of $H$. The jump relation may be non-deterministic and entails both discrete modes and continuous variables.
Generalizations

This definition of a HA is *not* the most general one. Obvious extensions include

- Input / disturbances in the vector field.
- Labeled jumps.
- Nondeterministic continuous evolutions.
- Stochastic effects.
Semantics: Two-Dimensional Time

Discrete activity:
- no progress of physical time involved;
- continuous activity frozen

Continuous phase:
- Phys. time advances, no discrete steps

A discretely perceptible event (threshold, elapse of clock) occurs, starting discrete activity

discrete activity ceases, progress of physical time starts again

An idealization partially justified by differing speeds of ES and environment!
Def: A hybrid time frame is a finite or infinite sequence \( \tau = \langle l_1, l_1, \ldots \rangle \) of time intervals \( l_i \), where

- each \( l_i \) is a non-empty convex subset of \( \mathbb{R}_{\geq 0} \), i.e. a non-empty interval in \( \mathbb{R}_{\geq 0} \),
- \( \inf l_i \in l_i \) for each \( i \), i.e. the intervals are left-closed,
- \( \sup l_i \in l_i \) for each \( i < \text{len} \ \tau \), i.e. all intervals excepts perhaps the rightmost are right-closed,
- \( \max l_i = \min l_{i+1} \) for each \( i < \text{len} \ \tau \), i.e. the intervals are adjacent and overlap exactly in one point.
Def: A hybrid trajectory $E$ is a tuple $E = (\tau, \nu, x)$ such that

- $\tau$ is a hybrid time frame,
- $\nu \in V^* \cup V^\omega$ with $\text{len } \nu = \text{len } \tau$ is a sequence of discrete modes,
- $x \in (\mathbb{R}_{\geq 0} \xrightarrow{\text{part., cont.}} \mathbb{R}^n)^* \cup (\mathbb{R}_{\geq 0} \xrightarrow{\text{part., cont.}} \mathbb{R}^n)^\omega$ with $\text{len } x = \text{len } \tau$ and $\text{dom}(x)_i = \tau_i$ is a sequence of continuous flows of the variables in $X$. 
Executions of a HA

[after K.H. Johansson]

Def: A run $E = (\tau, v, x)$ is an *execution* of the hybrid automaton $H = (V, X, f, Init, Inv, Jump)$ iff

- **Initiation:** $(v_1, x_1(\min \tau_1)) \in Init$,

- **Consecution:** $Jump \left( (v_i, x_i(\max \tau_i)) \ni (v_{i+1}, x_{i+1}(\min \tau_{i+1})) \right)$ holds for all $i < \text{len } \tau$,

- **Continuous evolution:** $x_i$ is a solution of $\frac{dx}{dt} = f(v_i, x)$ for each $i \leq \text{len } \tau$,

- **State consistency:** $(v_i, x_i(t)) \in Inv$ for each $t \in \text{dom}(\tau)_i$ and each $i \leq \text{len } \tau$

hold.
Types of executions

- An execution \((\tau, \nu, x)\) is **finite** iff \(\text{len} \tau < \infty\) and \(\tau_{\text{len}} \tau\) is bounded.
- An execution \((\tau, \nu, x)\) is **infinite** iff \(\text{len} \tau = \infty\) or \(\sup \tau_{\text{len}} \tau = \infty\).
- An execution \((\tau, \nu, x)\) is a **Zeno behavior** iff \(\text{len} \tau = \infty\) but \(\sup_{i \in \mathbb{N}} \sup \tau_i < \infty\).
- An execution \(E\) of \(H\) is **maximal** iff there is no execution \(E'\) of \(H\) with \(E \leq E'\) and \(E \neq E'\).

Note that the maximal executions of \(H\) may *properly* include the infinite executions of \(H\) due to possible maximality = non-extensibility of a finite execution ("blocking").


- **Proof obligation:** Can the system be guaranteed to show desired behaviour, even under disturbances? E.g.,
  - remains in safe states?
  - eventually reaches a desired operational mode?
  - stabilizes, i.e., converges against a setpoint / stable orbit / region of phase space?

  ! involves co-verification of controller and *continuous* environment.
State and Dimension Explosion

Number of **continuous variables linear in number of cars**
- Positions, speeds, accelerations,
- torque, slip, ...

Number of **discrete states exponential in number of cars**
- Operational modes, control modes,
- state of communication subsystem, ...

**Size-dependent dynamics**
- Latency in ctrl. loop depends on number of cars due to communication subsystem.
- Coupled dynamics yields long hidden channels chaining signal transducers.

⇒ Need a scalable approach
⇒ Let’s try to achieve this through strictly symbolic methods.
Further Agenda

1. Reachability analysis by iterating the transition relation
   - Decidable cases (very confined...)
     - Timed automata
     - Initialized rectangular automata
   - Decidable wrt. depth-bounded reachability
     - linear hybrid automata
   - Undecidable even in the bounded (the truly interesting stuff...)
     - Non-polynomial discrete-time hybrid automata
     - Non-linear continuous-time hybrid automata

2. Some other verification schemes
   - safe finite-state approximations
   - Lyapunov criteria
Decidable Case

Timed Automata
Example

- Stays in state opening for exactly 6 time units,
- stays in state open between 4 and 8 time units,
- stays in state closing for exactly 6 time units,
- stays in state closed for exactly 5 time units,
- stays in all other states arbitrarily long.
Formal setup

A timed transition system $TTS = (V, E, L, T, \alpha, G, R, Inv, I)$ over a set $C$ of clocks and alphabet $\Sigma$ has

- a set $V$ of vertices (interpreted as discrete system states, a.k.a. locations),
- a set $E$ of edges (interpreted as possible transitions),
- $L \in V \rightarrow \mathcal{P}(AP)$ labels the vertices with atomic propositions that apply in the individual vertices,
- $I \subseteq V$ is a set of initial states,
- $T : E \rightarrow (V \times V)$ maps edges to location changes,
- $\alpha : E \rightarrow \Sigma$ assigns a communication to transitions,
- $G : E \rightarrow \mathcal{P}(ClockVal)$ gives conditions for a transition to be taken,
- $R : E \rightarrow \mathcal{P}(C)$ states the clocks to be reset upon a transition,
- $Inv : V \rightarrow \mathcal{P}(ClockVal)$ yields state invariants denoting when a state may be held,

where $ClockVal = C \rightarrow \mathbb{R}_{\geq 0}$. 
Given a TTS \((V, E, L, T, \alpha, G, R, \text{Inv}, I)\), a run \(r\) of the TTS is

- an alternating sequence \((v_0, c_0) \xrightarrow{(e_0,t_0)} (v_1, c_1) \xrightarrow{(e_1,t_1)} \ldots\) of
  - state/clock-valuation pairs \((v_i, c_i) \in V \times \text{ClockVal},\)
  - transition/time pairs \((e_i, t_i) \in E \times \mathbb{R}_{\geq 0}\)

- with non-decreasing time stamps: \(t_i \leq t_{i+1}\) for each \(i\)
- that starts in an initial state: \(v_0 \in I\) and \(c_0 \equiv 0\)
- and is state-transition-consistent: \(T(e_i) = (v_i, v_{i+1})\) for each \(i\)
- and satisfies the transition guards: \(c_i + (t_i - t_{i-1}) \in G(e_i)\) for each \(i\), where \(c + t(x) = c(x) + t\) for each clock \(x\) and \(t_{-1} = 0\),
- and invariably satisfies the state invariants: \(c_i + t \in \text{Inv}(v_i)\) for each \(i\) and each \(t\) with \(0 \leq t \leq t_i - t_{i-1}\)
- and obeys clock resets: \(c_{i+1}(x) = \begin{cases} c_i(x) + (t_i - t_{i-1}) & \text{iff } x \not\in R(e_i) \\ 0 & \text{iff } x \in R(e_i) \end{cases}\)

for each \(i\) and each clock \(x\).
The quest

- The set of states of a TTS is $V \times \text{ClockVal}$.
- It is infinite, as $\text{ClockVal} = C \rightarrow \mathbb{R}_{\geq 0}$.
- Naive forward or backward (on the fly or symbolic) state coloring algorithms need not terminate.

Is reachability analysis etc. nevertheless mechanizable?
Simple clock constraints

A clock constraint is simple iff

- it is of the form $x \sim k$, where $x$ is a clock, $k$ an integer constant, and
  \sim one of $<, \leq, =, \geq, >$

- a conjunction of such simple constraints.

From now on, we will concentrate on TTS where

- all guards are simple,

- all invariants are simple.
Clock regions

max. clock constraint
Time-abstract bisimulation

A time-abstract bisimulation between two TTS is a relation

$$\sim \subset (V \times \text{ClockVal}) \times (V' \times \text{ClockVal}')$$

s.t. for each \((v, c) \sim (v', c')\):

1. if there is \((v_1, c_1) \in V \times \text{ClockVal}\) and \((e, t) \in E \times \mathbb{R}_{\geq 0}\) s.t.

   \[(v, c) \overset{(e,t)}{\rightarrow} (v_1, c_1)\]

then there is \((v'_1, c'_1) \in V' \times \text{ClockVal}’\) and \((e', t') \in E' \times \mathbb{R}_{\geq 0}\) s.t.

\[(v', c') \overset{(e',t')}{\rightarrow} (v'_1, c'_1) \quad \text{and} \quad \alpha(e) = \alpha(e') \quad \text{and} \quad (v_1, c_1) \sim (v'_1, c'_1)\]

N.B.: \(t\) and \(t'\) are not related! \(\sim\) time abstraction.
2. if there is \((v_1', c_1') \in V' \times \text{ClockVal}'\) and \((e', t') \in E' \times \mathbb{R}_{\geq 0}\) s.t.

\[(v', c') \xrightarrow{(e', t')} (v_1', c_1')\]

then there is \((v_1, c_1) \in V \times \text{ClockVal}\) and \((e, t) \in E' \times \mathbb{R}_{\geq 0}\) s.t.

\[(v, c) \xrightarrow{(e, t)} (v_1, c_1) \quad \text{and} \quad \alpha(e) = \alpha(e') \quad \text{and} \quad (v_1, c_1) \sim (v_1', c_1')\]
**Clock regions vs. time-abstract bisimulation**

**Thm.:** If ~ is a time-abstract bisimulation on a TTS s.t. ~ does only relate identical vertices (yet with potentially different clock val.s) and if \((v, c) \sim (v', c')\) then a vertex \(w \in V\) is reachable from \((v, c)\) iff \(w\) is reachable from \((v', c')\).

**Thm.:** For any TTS, the relation \(~ \subseteq (V \times \text{ClockVal}) \times (V \times \text{ClockVal})\) defined by \((v, c) \sim (v', c')\) iff

1. \(v = v'\),
2. F.e. clock \(x\), \([c(x)] = [c'(x)]\) or \(c(x) > mc < c'(x)\),
3. F.e. clock \(x\), \(\text{fract}(c(x)) = 0 \iff \text{fract}(c'(x)) = 0\) or \(c(x) > mc < c'(x)\),
4. F.e. clock \(x, y\), \(\text{fract}(c(x)) \leq \text{fract}(c(y)) \iff \text{fract}(c'(x)) \leq \text{fract}(c'(y))\)

is a time-abstract bisimulation on the TTS (i.e., between the states of just that one TTS).

\((mc\) is the maximum time constant in the TTS.\)

**Cor.:** Wrt. vertex reachability (and other time-abstract notions like existence of time-abstract traces), states in the above \(~\) relation are indistinguishable.

**Obs.:** For any TTS, there are only finitely many equivalence classes wrt. \(~\).
Equivalence classes of ~

max. clock constraint
The region automaton

Given the $TTS = (V, E, L, T, \alpha, G, R, Inv, I)$, we define its region “automaton” (like the TTS, it actually lacks an acceptance condition) to be the finite Kripke structure

$A_{TTS} = ([V \times ClockVal]_{\sim}, \rightarrow, L', [I \times \{x \rightarrow 0\}]_{\sim})$ with

- $x \rightarrow y$ iff there is $(v, c) \in x$, $(v', c') \in y$, $t \geq 0$, and $e \in E$ s.t.
  $$(v, c) \xrightarrow{(e, t)} (v', c')$$
- $L'([v, c]) = L(v)$.

😊 This is a finite Kripke structure that can be subjected to CTL model-checking etc.

😢 but its size is exponential in the number of clocks:

$$\#\text{regions} = |C|! \cdot 2^{|\mathcal{C}|} \cdot \prod_{c \in \mathcal{C}} (2 \max(c) + 2)$$

❓ Can we do the state-space traversal more symbolically, representing sets of regions by predicates?
Symbolic Methods for TA

Clock zones
Clock zones

A clock zone is the set of satisfying assignments in $\mathbb{R}_{\geq 0}^n$ of a conjunction of

- inequations that compare a clock to an integer constant and
- inequations that compare the difference of two clocks to an integer constant.

By introduction of a dedicated clock $x_0$ representing the value 0, difference logic formulae of the specific conjunctive form

$$
\phi \ ::= \ \bigwedge_{x \in \mathcal{C}} (x_0 - x \leq 0) \ \land \ \bigwedge_{i=1}^{n} \psi_i
$$

$$
\psi_i \ ::= \ c_{i1} - c_{i2} \sim_i k_i
$$

$$
\sim_i \ ::= \ < \mid \leq
$$

$$
k_i \ ::= \ \mathbb{Z}
$$

form an appropriate symbolic representation of clock zones.
Closure properties of clock zones

If $\phi$ and $\psi$ are symbolic representations of clock zones and $d \in \mathbb{N}$ then symbolic representations

- $\phi \land \psi$ for zone intersection: $\llbracket \phi \land \psi \rrbracket \overset{\text{def}}{=} \{ x \in \mathbb{R}_{\geq 0} \mid x \models \phi \text{ and } x \models \psi \}$
- $\exists x_i. \phi$ for clock hiding:

  $$\llbracket \exists x_i. \phi \rrbracket \overset{\text{def}}{=} \{ (x_1, \ldots, x_n) \mid \text{there is } y \in \mathbb{R}_{\geq 0} \text{ s.t.} \}
  \begin{align*}
  & (x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n) \models \phi
  \end{align*}$$

- $\phi[x_i := 0]$ for clock reset: $\llbracket \phi[x_i := 0] \rrbracket \overset{\text{def}}{=} \llbracket x_i = 0 \land \exists x_i. \phi \rrbracket$
- $\phi \uparrow$ for elapse of time:

  $$\llbracket \phi \uparrow \rrbracket \overset{\text{def}}{=} \{ (x_1 + \delta, \ldots, x_n + \delta) \mid (x_1, \ldots, x_n) \models \phi, \delta \in \mathbb{R}_{\geq 0} \}$$

can be obtained effectively.
TA reachability using zones: the idea

1. Represent **reachable state sets** by lists of pairs of locations and clock zones \( \langle (l_1, z_1), \ldots, (l_m, z_m) \rangle \).

2. For such a pair, compute the set \( Post_t(l, z) \) of successors under a transition \( t \) with \( T(t) = (l, l') \) by
   - Let time elapse starting from \( z \): \( \phi_1 = z \uparrow \) represents states reachable under arbitrary passage of time
   - Intersect \( \phi_1 \) with \( Inv(l) \): \( \phi_2 = \phi_1 \land Inv(l) \) reflects states reachable through time passage consistent with the location invariant (N.B.: invariant is convex due to simplicity)
   - Intersect \( \phi_2 \) with guard \( G(t) \): \( \phi_3 = \phi_2 \land G(t) \) reflects states reachable through time passage which enable the transition \( t \)
   - Reset the clocks in \( R(t) \): \( \phi_4 = \phi_3[r_1 := 0] \ldots [r_j := 0] \), where \( \{r_1, \ldots, r_j\} = R(t) \), reflects the clock readings after performing \( t \)'s resets
   - Intersect with the target loc.'s invariant: \( \phi_5 = \phi_4 \land Inv(l') \)
   - Do the location change: \( Post_t(l, z) = (l', \phi_5) \).
The state-space exploration

1. Start with the state list
   \[ R_0 = I \times \{\text{"} \land_{x \in \mathcal{C}} (x_0 - x \leq 0) \land \land_{x \in \mathcal{C}} (x - x_0 \leq 0)\text{"}\} \].

2. Repeat
   1. select \((l_i, z_i) \in R_k\) and \(t \in E\) with source \(l_i\) s.t. \(Post_t(l_i, z_i)\) is not already subsumed by \(R_k\),
   2. build \(R_{k+1} = R_k \cdot \langle Post_t(l_i, z_i) \rangle\)

until no such \((l_i, z_i) \in R_k\) and \(t \in E\) is found.

N.B. Subsumption test can be performed at various levels of detail.
The problem

- Iterating $Post_t(l, z)$ for all pairs $(l, z)$ in the list of reachable states and all transitions need not terminate:

  \[ x, y := 0 \]

  \[ x \leq 1 \]

  \[ x = 1 / x := 0 \]

- In the region graph, we solved the problem by not distinguishing clock readings above the max. clock constant.

- We can achieve a similar effect by **widening zones** that extend beyond the max. clock constant:
  - Any constraint of the form $x_i - x_j \sim l$ with $l > \text{maxconstant}$ is removed from the symbolic representation when it arises.
Manipulation of Difference Logic Constraints

Difference Bound Matrices
Difference Bound Matrices

Difference bound matrices (DBMs) are a canonizable representation for conjunctive formulae in difference logic

\[
\phi ::= \bigwedge_{i=1}^{n} \psi_i \\
\psi_i ::= c_{i1} - c_{i2} \sim_i k_i \\
\sim_i ::= < | \leq \\
k_i ::= \in \mathbb{Z}
\]

Given a finite clock set \( C \) (in practice containing the pseudo-clock \( x_0 \)), a DBM \( M \) over \( C \) is a mapping

\[
(C \times C) \rightarrow (\{<, \leq\} \times \mathbb{Z} \cup \{(<, \infty)\}) .
\]

clock pairs constraint on diff. unconstrained

Encoding: \( M(x, y) = (\sim, k) \iff x - y \sim k \)
Implied constraints and tightening

Observation: \( x - y \sim_1 k_1 \) and \( y - z \sim_2 k_2 \) implies \( x - z \sim k_1 + k_2 \), where
\[
\sim = \begin{cases} 
\sim_1 & \text{iff } \sim_1 = \sim_2 \\
< & \text{otherwise.}
\end{cases}
\]

Consequence: A DBM may contain constraint pairs which imply constraints that are tighter than the recorded constraints:
\[
M(x, y) = (\sim_1, k_1) \land M(y, z) = (\sim_2, k_2) \land M(x, z) = (\sim, k)
\]
and
1. \( k > k_1 + k_2 \) or
2. \( k = k_1 + k_2 \) but \( \sim = \leq \), yet \( \sim_1 = < \) or \( \sim_2 = < \).

Solution: **Tighten the DBM** by replacing the constraint by the stronger implied constraint.

Repeat this until no implied constraint stronger than a recorded constraint remains. This brings the DBM into a **canonical form**.

Such canonization of DBMs can be done in cubic time using the **Floyd-Warshall algorithm**.
Properties of canonical DBMs

**Thm:** A canonical DBM is unsatisfiable iff there is some \( x \in C \) such that \( M(x, x) = (<, 0) \) or \( M(x, x) = (\sim, k) \) with \( k < 0 \).

**Cor:** Satisfiability test of canonical DBMs runs in \( O(|C|) \) time.
Operations on clock zones using canon. DBMs

Intersection:

\[ M \land N(x, y) = \begin{cases} M(x, y) & \text{if } M(x, y) \text{ is tighter than } N(x, y) \\ N(x, y) & \text{otherwise} \end{cases} \]

Clock reset: When the dedicated clock variable \( x_0 \) is used,

\[ M[z := 0](x, y) = \begin{cases} M(x, y) & \text{if } x \neq z \text{ and } y \neq z \\ M(x, x_0) & \text{if } x \neq z \text{ and } y = z \\ M(x_0, y) & \text{if } x = z \text{ and } y \neq z \\ (\leq, 0) & \text{if } x = y = z \end{cases} \]

Note that canonicity saves an explicit quantifier elimination as the implied constraints are already in place!

These operations do not preserve canonicity!
Operations on clock zones using canon. DBMs

Elapse of time: When the dedicated clock variable $x_0$ is used,

$$M \uparrow (x, y) = \begin{cases} 
M(x, y) & \text{if } x = x_0 \text{ or } y \neq x_0 \\
(\langle, \infty) & \text{if } x \neq x_0 \text{ and } y = x_0
\end{cases}$$

Widening: When the maximum clock constant is $k$,

$$\tilde{M}(x, y) = \begin{cases} 
M(x, y) & \text{if } M(x, y) = (\sim, l) \text{ with } |l| \leq |k| \\
(\langle, \infty) & \text{otherwise}
\end{cases}$$
This technology

- is implemented in mature tools
- extends to some slightly more hybrid domains:
  - Linear Priced Timed Automata
    [Larsen, Rasmussen, Boyer, Brihaye, Bruyère, Raskin, 2005–]
    - Locations and transitions come equipped with costs / cost rates charged for taking the transition / staying in the location
    - Guards and invariants may not query cost variables
    - Problem is to determine infimum cost for reaching some state
  - Initialized rectangular hybrid automata
    [Henzinger, Kopke, Puri, Varaiya, 1995]
    - Rectangular guards, invariants and piecewise constant, rectangular differential inclusions
    - Continuous variables reset whenever differential inclusion changes
Initialized Rectangular Hybrid Automata
Initialized Rectangular Hybrid Automata

Hybrid automata subject to the following restrictions:

- **Being initialized**: Continuous variables are reset whenever the pertinent differential inclusion changes.

- **Being rectangular**:
  - guards and invariants are simple,
  - continuous evolution is governed by (location-dependent) constant differential inclusions.
Reduction to Timed Automata (Sketch)

1. Replace each variable $x$ by two copies $x_l$ and $x_u$:
   - $x_l$ represents lower, $x_u$ upper bound on value
   - replace differential inclusions by DEs accordingly:
     \[
     -2 \leq \frac{dx}{dt} \leq 1 \quad \implies \quad \frac{dx_l}{dt} = -2 \land \frac{dx_u}{dt} = 1
     \]
   - adjust guards and invariants accordingly.

2. Remove all DEs of form $\frac{dy}{dt} = 0$:
   - due to initialization, value upon entering the assoc. location is known,
   - thus, the respective guards and invariants can be evaluated statically.

3. Scale all DEs to form $\frac{dy}{dt} = 1$:
   - Replace DE $\frac{dy}{dt} = a$ by $\frac{dy}{dt} = 1$;
   - multiply constants in the respective guards and invariants by $\frac{1}{a}$,
   - revert inequality signs iff $a < 0$.

4. Make all constants in guards and invariants integer by multiplying with common denominator (doesn’t affect reachability).
Linear Priced Timed Automata
Linear Priced Timed Automata

...extend the concept of timed automata by a \textit{monotonically increasing, linear cost function}:

- each transition \( e \) is associated a cost \( P(e) \in \mathbb{N} \), this cost is charged whenever the transition is taken,
- each location \( v \) is associated a cost rate \( P(v) \in \mathbb{N} \) this cost rate is charged for staying in the location, i.e. staying in \( v \) for \( \delta \) time units costs \( P(v) \cdot \delta \)

Costs accumulate over a run of the TA, yet cannot be queried by the LPTA:

- the cost of a run is the sum of all costs charged along the run, i.e. by its transitions including the delay transitions,
- the cost accumulated so far cannot be queried in guards or invariants,
- neither can it be updated by any other than the above cost accumulation mechanism.
Formal setup

A linear priced timed automaton $A = (V, E, L, T, \alpha, G, R, \text{Inv}, I, P)$ over a set $C$ of clocks and alphabet $\Sigma$ is composed of
- a timed automaton $A' = (V, E, L, T, \alpha, G, R, \text{Inv}, I)$ and
- a cost function $P : (V \cup E) \rightarrow \mathbb{N}$.

A run $r$ of $A$ is an alternating sequence

$$r = (v_0, c_0) \xrightarrow{(e_0,t_0)} (v_1, c_1) \xrightarrow{(e_1,t_1)} \ldots (v_n, c_n)$$

of state/clock-valuation pairs $(v_i, c_i) \in V \times \text{ClockVal}$ such that $r$ is a run of the TA $A'$.

The price $p(r)$ of run $r$ is the accumulated cost

$$p(r) = \sum_{i=0}^{n-1} P(e_i) + \sum_{i=0}^{n-1} (t_i - t_{i-1}) \cdot P(v_i)$$

where $t_{-1} = 0$. 
The goal

Given an LPTA \( A = (V, E, L, T, \alpha, G, R, \text{Inv}, I, P) \) and a location \( g \in V \) (or a set \( G \subseteq V \) of locations),

1. determine \( \text{mincost} = \inf\{p(r) \mid r \text{ is a run of } A \text{ ending in } g\} \),
2. if \( \text{mincost} < \infty \) then find a run \( r \) ending in \( g \) with \( p(r) = \text{mincost} \).

This is called the minimum cost reachability problem.
Generalizing zone-based analysis to LPTA

**Idea:** Extend the concept of a zone into a priced zone
- consists of zone $Z$, i.e. a set of clock valuations,
- plus cost information $c_i$ for each clock valuation in the zone.

**Usage:** Manipulate such priced zones in forward reachability analysis:
- use forward reachability to collect the reachable state in lists of pairs $(v, pz)$, where $v \in V$ and $pz$ is a priced zone,
- for each clock valuation $cv$ in $pz$, the price information records the greatest lower bound of the cost for reaching $(v, cv)$.

**Requires:** Effective manipulation of priced zones:
- intersection with a (unpriced) zone: for guards and invariants,
- reset operations $pz[x := 0]$ on priced zones,
- closure elapse of time $pz \uparrow$ on priced zones.
**Priced zone**

**Def.:** A **priced zone** is a tuple \((Z, c, r)\), where
- \(Z\) is a zone,
- \(c \in \mathbb{N}\) denotes the infimum of the price of all clock valuations in the zone, called **offset of the zone**,
- \(r : C \rightarrow \mathbb{Z}\) assigns a **cost rate** to each clock.

**Def.:** The **cost** associated to a clock valuation \(cv \in Z\) wrt. the priced zone \((Z, c, r)\) is

\[
c + \sum_{x \in C} r(x) \cdot (cv(x) - \Delta_Z(x))
\]

where \(\Delta_Z(x) = \inf\{cv(x) \mid x \in Z\}\) f.e. \(x \in C\) is the “lowermost corner” of \(Z\).

**Linear dependency of cost on clock valuation sufficient?**
Priced zones

\[
Z
\]

\[
\text{cost} = 2y - x + 1
\]
Infimum cost of a priced zone

As the cost associated cost of a clock valuation $cv$ in the priced zone $(Z, c, r)$ is

$$c + \sum_{x \in C} r(x) \cdot (cv(x) - \Delta_Z(x)),$$

the infimum cost $\text{infcost}(Z, c, r)$ of priced zone $(Z, c, r)$ is

$$\inf \left\{ c + \sum_{x \in C} r(x) \cdot (cv(x) - \Delta_Z(x)) \mid cv \in Z \right\}$$
We say that priced zone \((Z, c, r)\) is dominated by priced zone \((Z', c', r')\), denoted \((Z, c, r) \subseteq (Z', c', r')\), iff

- \(Z'\) covers \(Z\), i.e. \(Z \subseteq Z'\), and
- the prices assigned by \((Z', c', r')\) are cheaper, i.e.

\[
c + \sum_{x \in C} r(x) \cdot (cv(x) - \Delta_Z(x)) \geq c' + \sum_{x \in C} r'(x) \cdot (cv(x) - \Delta_{Z'}(x))
\]

for each \(cv \in Z\).
The cost-aware forward reachability algorithm

\[
\begin{align*}
\text{Cost} & \equiv \infty \\
\text{Passed} & \equiv \emptyset \\
\text{Waiting} & \equiv \{(i, (Z, c, r)) \mid i \in I, Z = \bigwedge_{x \in C} c - x_0 = 0, c = 0, r \equiv 0\}
\end{align*}
\]

while Waiting \neq \emptyset do

\[
(v, (Z, c, r)) \in \text{Waiting}; \quad \text{Waiting} \equiv \text{Waiting} \setminus \{(v, (Z, c, r))\}
\]

if \( v = g \) then Cost \equiv \min(\text{Cost}, \infcost(Z, c, r)) \end{if}

if \( \neg \exists (v, (Z', c', r')) \in \text{Passed} \land (Z, c, r) \sqsubseteq (Z', c', r') \) then

\[
\begin{align*}
\text{Passed} & \equiv \text{Passed} \cup \{(v, (Z, c, r))\} \\
\text{Waiting} & \equiv \text{Waiting} \cup \{\text{Post}_t((v, (Z, c, r))) \mid t \in E\}
\end{align*}
\]

fi

return Cost

[G. Behrmann, 2004]
Symbolic operations

To compute $Post_t((v, (Z, c, r)))$ in forward reachability, we need

- increment of cost by a constant,
- intersection with a zone,
- reset operations $pz[x := 0]$ on priced zones,
- closure elapse of time $pz \uparrow$ on priced zones.

**Increment:** Given a priced zone $(Z, c, r)$ and a constant $k \in \mathbb{N}$, the zone’s cost increment by $k$ is the priced zone $(Z, c + k, r)$.

**Intersection:** Given a priced zone $(Z, c, r)$ and an unpriced zone $Y$, the intersection is

$$\left( Z \land Y, c + \sum_{x \in C} r(x) \cdot (\Delta_{Z \land Y}(x) - \Delta_Z(x)), r \right)$$

\[\text{cost of the new “lowermost corner”}\]

which is a priced zone that can be built effectively from $(Z, c, r)$ and $Y$. 
Priced zones: $y := 0$

\[
\begin{align*}
\Delta & & c = 4 & & -1 \\
\Delta & & c = 4 & & 1 \\
\Delta & & c = 2 \\
Z & & c = 4 & & 2
\end{align*}
\]

\[
\begin{align*}
\text{cost} & = 2y - x + 1 \\
\text{cost} & = -x + 5 \\
\text{cost} & = x - 1
\end{align*}
\]

Priced zone has to be split!
Priced zones: passage of time

Priced zone has to be split!
Discussion

- Technology is available in **UPPAAL CORA**
  [G. Behrmann, K.G. Larsen, J.I. Rasmussen, 2002–]

- Various **extensions** to base algorithm discussed in literature:
  - Multiple cost variables
    [Larsen, Rasmussen, 2005]
  - Positive and negative costs, infimal and supremal costs
    [Boyer, Brihaye, Bruyère, Raskin 2007]

- Yet not sufficiently expressive to model feedback between controller and environment:
  - the cost accumulated so far cannot be queried in guards or invariants,
  - neither can it be updated by any other than the accumulation mechanisms.