Abstractions Part II

Sayan Mitra

Verifying cyberphysical systems

mitras@illinois.edu

Outline

- Abstractions
- Simulation relations
- Back to rational timed automata

Abstractions and Simulations

Consider models that are **comparable** (same or overlapping states and/or actions)

We would like to *approximate* one (hybrid) automaton H_1 with another one H_2

- We can over-approximate the reachable states of H_1 with those of H_2
- This would ensure that invariants of H_2 *carry over* to H_1
- Beyond invariants, more general requirements (e.g., CTL) carry over

H_2 should be *simpler* (smaller description, fewer states, transitions, linear dynamics, etc.) and preserve some properties of H_1 (and not others)

Verifying some requirements of H_2 can then carry over requirements to H_1

Finite state examples

A

$$\begin{tikzcd} a_0 \arrow[r, bend left, "0"] & a_1 \arrow[l, bend left, "1"] \arrow[r, bend left, "1"] & a_2 \arrow[l, bend left, "0"] \arrow[r, bend left, "0"] & a_3 \arrow[l, bend left, "1"] \end{tikzcd}$$

B

$$\begin{tikzcd} b_{02} \arrow[r, bend left, "0"] & b_{13} \arrow[l, bend left, "1"] \end{tikzcd}$$

Traces

C

$$\begin{tikzcd} c_0 \arrow[loop right, "{0,1}"] \end{tikzcd}$$

$\text{Traces}_C = \{0,1\}^*$

An execution $\alpha = a_0\, 0\, a_1\, 1\, a_2\, 0\, a_3 \ldots$

Suppose we only care about the sequence of actions (and not states)

Then, we define $\text{trace}(\alpha) = 0\ 1\ 0\ \ \ldots$

Set of all traces of **A** $\text{Traces}_A = (01)^*$

$$Traces_A \subseteq Traces_B \subset Traces_C$$

Finite state examples

A

$$\begin{tikzcd}
a_0 \arrow[r, bend left, "0"] & a_1 \arrow[l, bend left, "1"] \arrow[r, bend left, "1"] & a_2 \arrow[l, bend left, "0"] \arrow[r, bend left, "0"] & a_3 \arrow[l, bend left, "1"]
\end{tikzcd}$$

B simulates A and vice versa.

A and B are bisimilar

B

$$\begin{tikzcd}
b_{02} \arrow[r, bend left, "0"] & b_{13} \arrow[l, bend left, "1"]
\end{tikzcd}$$

Traces$_B$= (01)*

C simulates both A and B.

C is an **abstraction** of both A and B.

C

$$\begin{tikzcd}
c_0 \arrow[loop right, "{0,1}"]
\end{tikzcd}$$

Traces$_C$= {0,1}*

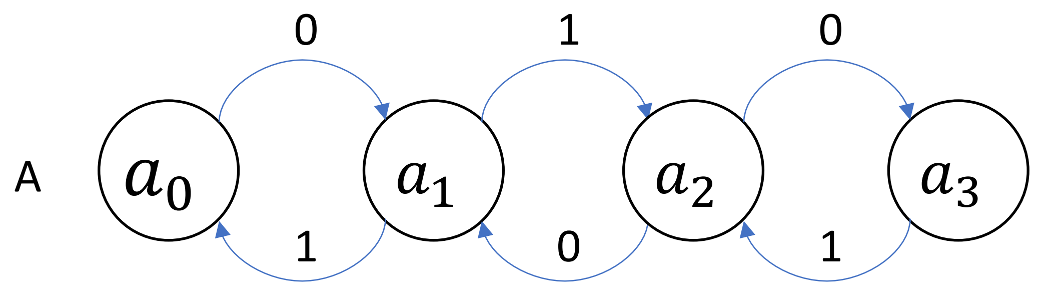
How to prove B simulates A?

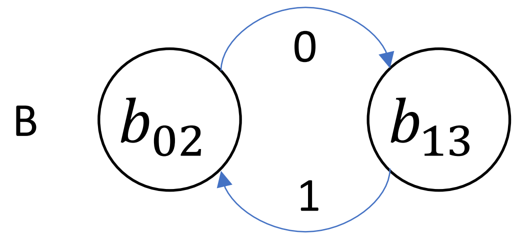
Show there exists a **simulation relation** $R \subseteq Q_A \times Q_B$

Say, $R = \{(a_0, b_{02}), (a_2, b_{02}), (a_1, b_{13}), (a_3, b_{13})\}$

Show that $\forall\, a_i \rightarrow_A a_{i'}$ and $(a_i, b_j) \in R$ there $\exists\, b_{j'}$ such that

1. $b_j \rightarrow_B b_{j'}$
2. $(a_{i'}, b_{j'}) \in R$
3. $trace(b_j \rightarrow_B b_{j'}) = trace(a_i \rightarrow_A a_{i'}) = 0 \; or \; 1$





Forward simulation relation

Consider a pair of automat $\mathcal{A}_1 = \langle Q_1, \Theta_1, A_1, D_1 \rangle$ and $\mathcal{A}_2 = \langle Q_2, \Theta_2, A_2, D_2 \rangle$.

Definition. A **relation** $R \subseteq Q_1 \times Q_2$ is a **forward simulation** relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ if

1. For every $q_1 \in \Theta_1$ there exists a $q_2 \in \Theta_2$ such that $q_1 R q_2$
2. For every transition $q_1 \rightarrow_1^{a_1} q_1'$ and $q_1 R q_2$ there exists q_2', a_2 such that
   - $q_2 \rightarrow_2^{a_2} \; q_2'$
   - $q_1' \; R \; q_2'$
   - $trace(q_1, a_1, q_1') = trace(q_2, a_2, {q_2}')$

Theorem. If $\exists$ forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$ then $Traces_1 \subseteq Traces_2$.

Simulation example continued

$$\begin{tikzcd}
\text{A} & a_0 \arrow[r, bend left, "0"] & a_1 \arrow[l, bend left, "1"] \arrow[r, bend left, "1"] & a_2 \arrow[l, bend left, "0"] \arrow[r, bend left, "0"] & a_3 \arrow[l, bend left, "1"]
\end{tikzcd}$$

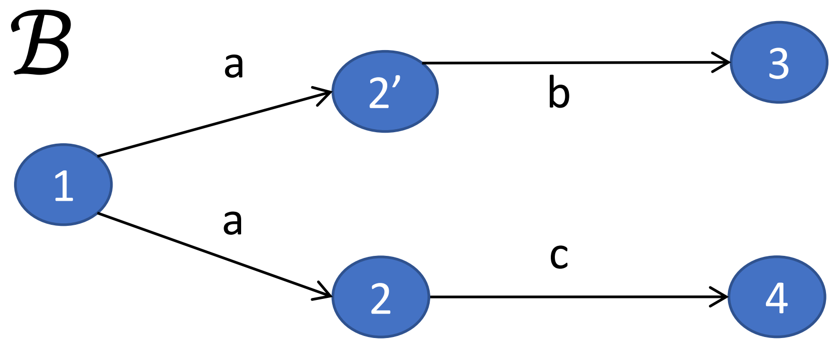
$$\begin{tikzcd}
\text{B} & b_{02} \arrow[r, bend left, "0"] & b_{13} \arrow[l, bend left, "1"]
\end{tikzcd}$$

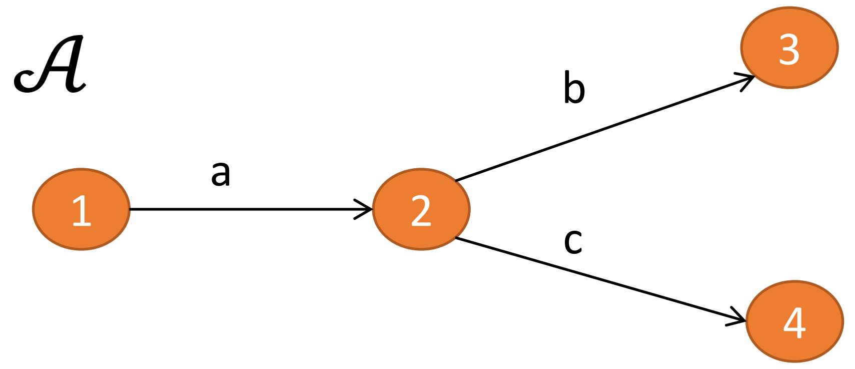
$$\begin{tikzcd}
c_{03} \arrow[loop above, "0"] \arrow[loop right, "1"]
\end{tikzcd}$$

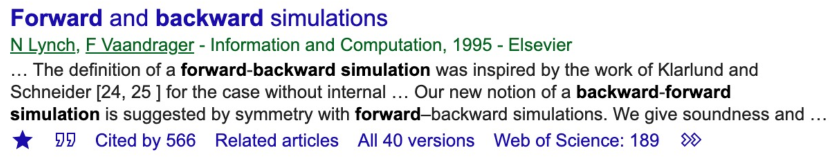
- Check that A also simulates B and that C simulates both A and B.
- Therefore, $\text{Traces}_\text{A} = \text{Traces}_\text{B} \subseteq Traces_C$?
- Does A simulate C?
- Can the simulation relation be checked using an SMT solver?

A Simulation Example

- $Traces_{\mathcal{A}} \subseteq Traces_{\mathcal{B}}$
- Is there a forward simulation from $\mathcal{A}$ to $\mathcal{B}$?
- Consider the forward simulation relation:
- $\mathcal{A}$: $2 \rightarrow_c 4$ cannot be simulated by $\mathcal{B}$ from 2' although (2,2') are related
- Lesson: Forward simulation relation is a **sufficient condition** for proving abstraction (not necessary)
- Other approaches for proving abstraction: Backward simulation, history and prophecy relations


$$\begin{tikzcd}
\mathcal{B} & 2' \arrow[r, "b"] & 3 \\
1 \arrow[ru, "a"] \arrow[r, "a"'] & 2 \arrow[r, "c"] & 4
\end{tikzcd}$$


$$\begin{tikzcd}
\mathcal{A} & & 3 \\
1 \arrow[r, "a"] & 2 \arrow[ru, "b"] \arrow[rd, "c"'] & \\
& & 4
\end{tikzcd}$$


Forward and **backward** simulations
N Lynch, F Vaandrager - Information and Computation, 1995 - Elsevier
… The definition of a **forward-backward simulation** was inspired by the work of Klarlund and Schneider [24, 25] for the case without internal … Our new notion of a **backward-forward simulation** is suggested by symmetry with **forward**–backward simulations. We give soundness and …
☆ 99 Cited by 566 Related articles All 40 versions Web of Science: 189 »

Recall simulations for hybrid systems

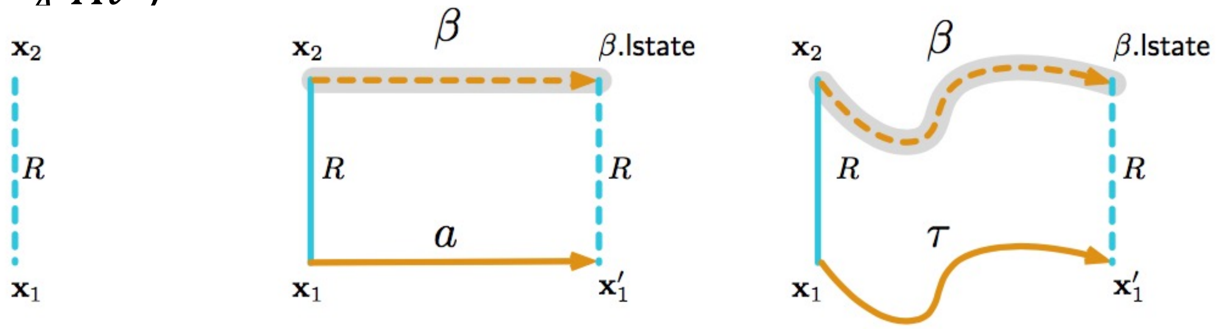
Forward simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ is a relation R $\subseteq val(X_1) \times val(X_2)$ such that

1. For every $\mathbf{x_1} \in \Theta_1$ there exists $\mathbf{x_2} \in \Theta_2$ such that $\mathbf{x_1}$ R $\mathbf{x_2}$
2. For every $\mathbf{x_1} \rightarrow_{\boldsymbol{a_1}} \mathbf{x_1}' \in \mathcal{D}$ and $\mathbf{x_2}$ such that $\mathbf{x_1}$ R $\mathbf{x_2}$, there exists $\mathbf{x_2}'$ such that
   - $x_2 \rightarrow_{\boldsymbol{a_1}} x_2'$ and
   - x_1' R x_2'
3. For every $\boldsymbol{\tau_1} \in \mathcal{T}_1$ and $\mathbf{x_2}$ such that $\tau_1.fstate$ R $\mathbf{x_2}$, there exists $\tau_2 \in \mathcal{T}_2$ that
   - $x_2 = \tau_2.fstate$ and
   - x_1' R $\tau_2.lstate$
   - $\tau_2.\mathrm{dom} = \tau_1.dom$

Theorem. If there exists a forward simulation relation from hybrid automaton $\mathcal{A}_1$ to $\mathcal{A}_2$ then for every execution of $\mathcal{A}_1$ there exists a corresponding execution of $\mathcal{A}_2$.

Simulation relations for hybrid automata

- Recall condition 3 in definition of simulation relation: $Trace(Bj \rightarrow_B Bj') = Trace(Ai \rightarrow_A Ai')$



- Hybrid automata have transitions and trajectories
- Different types of simulation depending on different notions for "Trace"
  - Match for all variable values, action names, and time duration of trajectories (abstraction)
  - Match variables but not time (time abstract simulation)
  - Match a subset (external) of variables and actions (trace inclusion)
  - Match single action/trajectory of A with a sequence of actions and trajectories of B

Timer simulates Ball (w.r.t. timing of bounce actions)

```
Automaton Ball(c,$v_0$,g)
  variables:
    x: Reals := 0
    v: Reals := $v_0$
  actions: bounce
    transitions:
      bounce
        pre $x = 0 \wedge v < 0$
        eff v := -cv
    trajectories:
        evolve d(x) = v; d(v) = -g
        invariant $x \geq 0$
```

```
Automaton Timer(c, $v_0$, g)
  variables: analog
  timer: Reals := $2v_0/g$,
  n:Naturals=0;
  actions: bounce
    transitions:
      bounce
        pre $timer = 0$
        eff n:=n+1; timer := $\frac{2v_0}{gc^n}$
    trajectories:
        evolve d(timer) = -1
        invariant timer $\geq 0$
```

Some nice properties of Forward Simulation

Let $\mathcal{A}, \mathcal{B}, \textbf{and } \mathcal{C}$ be comparable Timed Automatas. If R_1 is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and R_2 is a forward simulation from $\mathcal{B}$ to $\mathcal{C}$, then $R_1 \circ R_2$ is a forward simulation from $\mathcal{A}$ to $\mathcal{C}$

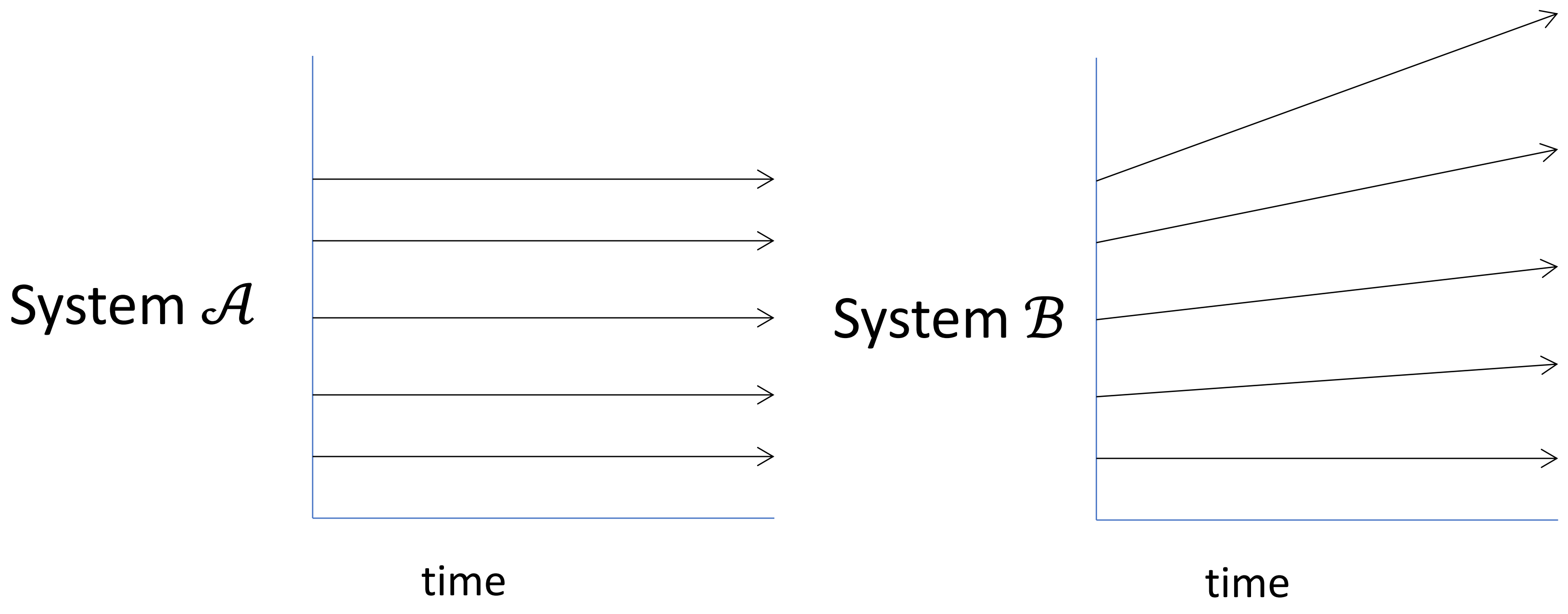
If R is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and R^{-1} is a forward simulation from $\mathcal{B}$ to $\mathcal{A}$ then R is called a bisimulation and $\mathcal{B}$ are $\mathcal{A}$ bisimilar

Bisimilarity is an equivalence relation (reflexive, transitive, and symmetric)

If R is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and I is an invariant of $\mathcal{B}$ then $R^{-1}(I)$ is an invariant of$\mathcal{A}$

Simulations and Stability

But, stability not preserved by ordinary simulations and bisimulations [Prabhakar, et. al 15]



Stability Preserving Simulations and Bisimulations for Hybrid Systems, Prabhakar, Dullerud, Viswanathan IEEE Trans. Automatic Control 2015

Backward Simulations

Backward simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ is a relation R ⊆ $Q_1 \times Q_2$ such that

1. If $\mathbf{x_1} \in \Theta_1$ and $\mathbf{x_1}$ R $\mathbf{x_2}$ then $\mathbf{x_2} \in \Theta_2$ such that
2. If $\mathbf{x'_1}$ R $\mathbf{x'_2}$ and $x_1 \xrightarrow{a} x_1'$ then
   - $x_2 \xrightarrow{\boldsymbol{\beta}} x_2'$ and
   - x_1 R x_2
   - Trace($\boldsymbol{\beta}$) = a_1
3. For every $\boldsymbol{\tau} \in \mathcal{T}$ and $\mathbf{x_2} \in Q_2$ such that $\mathbf{x_1'}$ R $\mathbf{x_2'}$, there exists $\mathbf{x_2}$ such that
   - $x_2 \xrightarrow{\boldsymbol{\beta}} x_2'$ and
   - x_1 R x_2
   - Trace($\boldsymbol{\beta}$) = $\boldsymbol{\tau}$

Theorem. If there exists a backward simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ then ClosedTraces$_1$ ⊆ ClosedTraces$_2$

Special Classes of Hybrid Automata

- Finite Automata
- Integral Timed Automata ←
- Rational time automata
- Multirate automata
- Rectangular Initialized HA

- Rectangular HA

- Linear HA

- Nonlinear HA

ACM NEWS: In Space, No One Can Fix Your Sign Errors---Paul Cheng & Peter Carian

15,000 satellite launches planned for the decade

5.3% satellites are lost in the first year, 42% of those in first 2 months

Most common cause **sign errors**: SW/HW parameter used the wrong way

- fitting acceleration sensors the wrong way
- wrong usage of negative instead of positive parameters
- switching current in wrong direction in a circuit
- inverting the orientation of the electromagnets used for positioning

Genesis (2001) for capturing particles from the solar wind, pounded into the Utah desert unbraked because a pencil-eraser-sized deceleration sensor was mounted upside-down.

Clocks and **Rational** Clock Constraints

- A **clock variable** x is a continuous (analog) variable of type real such that along any trajectory τ of x, for all t ∈ $\tau.dom, (\tau \downarrow x)(t) = t$.
- For a set X of clock variables, the set Φ(X) of ***rational* clock constraints** are expressions defined by the syntax:

  g ::= x $\leq q \mid x \geq q \mid \neg g \mid g_1 \wedge g_2$

  where $x \in X \; and \; q \in \mathbb{Q}$
- Examples: x = 10.125; x ∈ [2.99, 5); true are valid rational clock constraints
- Semantics of clock constraints $[g]$

Step 1. Rational Timed Automata

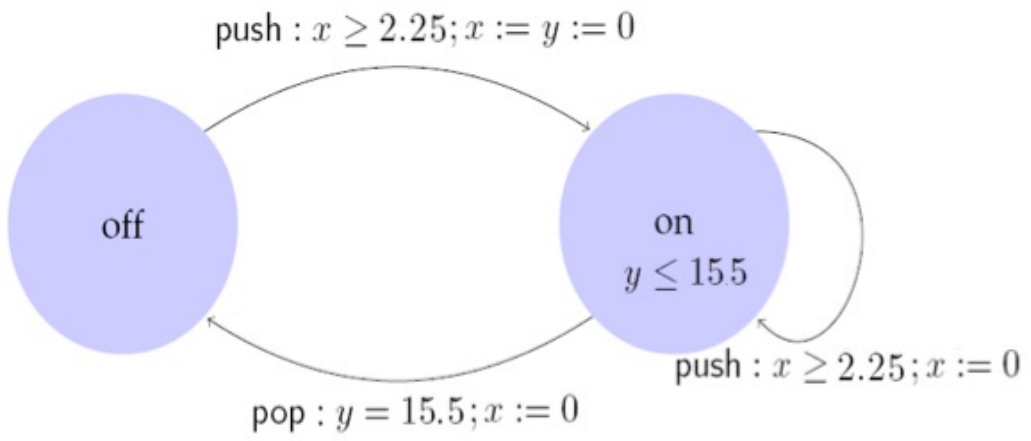
Definition. A ***rational timed automaton*** is a HA $\boldsymbol{\mathcal{A}}$ = $\langle V, \Theta, A, \mathcal{D}, \mathcal{T} \rangle$ where

- V = X ∪ $\{loc\}$, where X is a set of n clocks and l is a discrete state variable of finite type Ł
- A is a finite set
- $\mathcal{D}$ is a set of transitions such that
  - The guards are described by rational clock constraings $\Phi(X)$
  - $\langle x, l \rangle - a \rightarrow \langle x', l' \rangle$ implies either $x' = x$ or $x = 0$
- $\mathcal{T}$ set of clock trajectories for the clock variables in X

Example: Rational Light switch

Switch can be turned on whenever at least 2.25 time units have elapsed since the last turn off or on. Switches off automatically 15.5 time units after the last on.

```
automaton Switch
 internal push; pop
  variables
    internal x, y:Real := 0, loc:{on,off} := off
   transitions
     push
       pre x >=2.25
       eff if loc = on then y := 0 fi; x := 0; loc := off
     pop
       pre y = 15.5 ∧ loc = off
       eff x := 0
   trajectories
     invariant loc = on ∨ loc = off
     stop when y = 15.5 ∧ loc = off
     evolve d(x) = 1; d(y) = 1
```


push : $x \geq 2.25; x := y := 0$
off
on
$y \leq 15.5$
push : $x \geq 2.25; x := 0$
pop : $y = 15.5; x := 0$

Control State (Location) Reachability Problem

- Given an RTA, check if a particular mode is reachable from the initial states
- Is problem decidable?
- Yes
- Key idea:
  - Construct a ITA that has exactly same mode reachability behavior as the given RTA (timing behavior may be different)
  - Check mode reachability for ITA

Construction of ITA from RTA

- Multiply all rational constants by a factor q that make them integral
- Make d(x) = q for all the clocks
- RTA Switch reaches the same control locations as the ITA Iswitch
- Simulation relation R is given by
- (**u**,**s**) $\in R$ iff **u**.x = 4 **s**.x and **u**.y = 4 **s**.y

```
automaton ISwitch
internal push; pop
variables
  internal x, y:Real := 0, loc:{on,off} := off
transitions
  push
    pre x >=  9
    eff if loc = on then y := 0 fi; x := 0; loc := off
   pop
     pre y = 62 ∧ loc = off
     eff x := 0
trajectories
   invariant loc = on ∨ loc = off
   stop when y = 62 ∧ loc = off
   evolve d(x) = 4; d(y) = 4
```

Step 2. Multi-Rate Automaton

Definition. A multirate automaton is $\boldsymbol{\mathcal{A}} = \langle V, Q, \Theta, A, \mathcal{D}, \mathcal{T} \rangle$ where

- V = X ∪ $\{loc\}$, where X is a set of n continuous variables and loc is a discrete state variable of finite type Ł
- A is a finite set of actions
- $\mathcal{D}$ is a set of transitions such that
  - The guards are described by rational clock constrings $\Phi(X)$
  - $\langle x, l \rangle - a \rightarrow \langle x', l' \rangle$ implies either $x' = c \; or \; x' = x$
- $\mathcal{T}$ set of trajectories such that

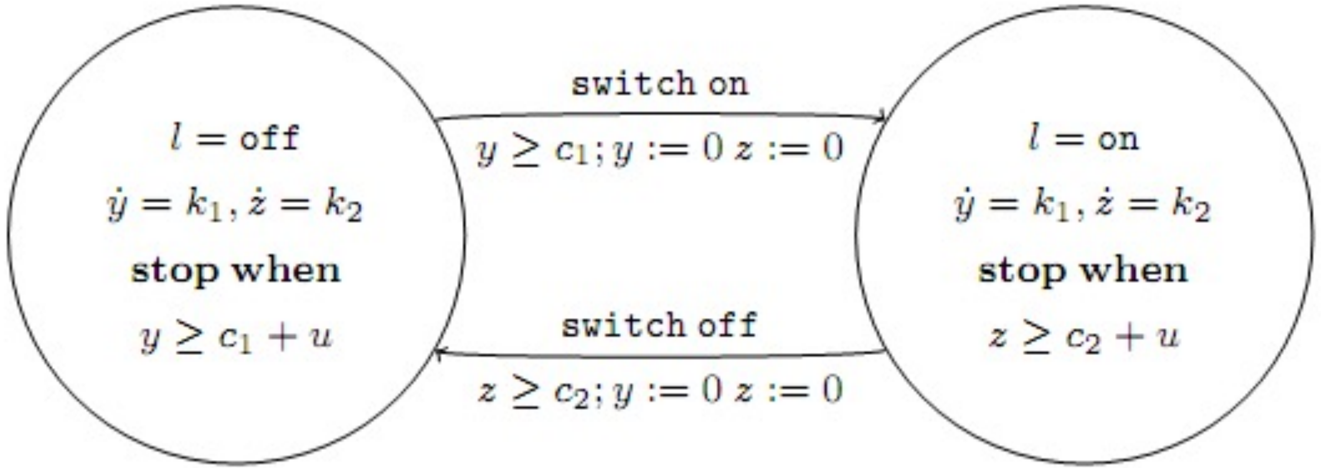
  for each variable $x \in X \; \exists k \; such \; that \; \tau \in \mathcal{T}, t \in \tau.dom$

$$\tau(t).x = \tau(0).x + k\,t$$

Control State (Location) Reachability Problem

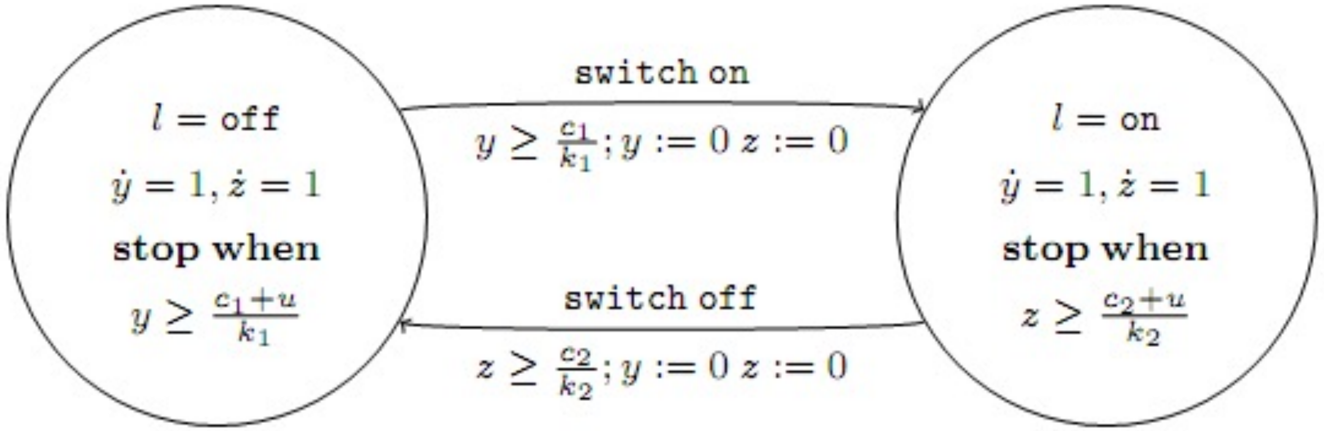
- Given an MRA, check if a particular location is reachable from the initial states
- Is problem is decidable?
- Yes
- Key idea:
    - Construct a RTA that is bisimilar to the given MRA

Example: Multi-rate to rational TA



$l = \text{off}$
$\dot{y} = k_1, \dot{z} = k_2$
stop when
$y \geq c_1 + u$
switch on
$y \geq c_1; y := 0\ z := 0$
switch off
$z \geq c_2; y := 0\ z := 0$
$l = \text{on}$
$\dot{y} = k_1, \dot{z} = k_2$
stop when
$z \geq c_2 + u$

Multirate automaton



$l = \text{off}$
$\dot{y} = 1, \dot{z} = 1$
stop when
$y \geq \frac{c_1+u}{k_1}$
switch on
$y \geq \frac{c_1}{k_1}; y := 0\ z := 0$
switch off
$z \geq \frac{c_2}{k_2}; y := 0\ z := 0$
$l = \text{on}$
$\dot{y} = 1, \dot{z} = 1$
stop when
$z \geq \frac{c_2+u}{k_2}$

Rational timed automaton

Step 3. Rectangular HA

Definition. A rectangular hybrid automaton (RHA) is a HA $\boldsymbol{\mathcal{A}} = \langle V, A, \mathcal{T}, \mathcal{D} \rangle$ where

- V = X ∪ $\{loc\}$, where X is a set of n continuous variables and loc is a discrete state variable of finite type Ł
- A is a finite set
- $\mathcal{T} = \cup_{\ell} \mathcal{T}_{\ell}$ set of trajectories for X
  - For each $\tau \in \mathcal{T}_{\ell}, x \in X$ either (i) $d(x) = k_{\ell}$ or (ii) $d(x) \in [k_{\ell 1}, k_{\ell 2}]$
  - Equivalently, (i) $\tau(t)\lceil x = \tau(0)\lceil x + k_{\ell} t$

    (ii) $\tau(0)\lceil x + k_{\ell 1} t \leq \tau(t)\lceil x \leq \tau(0)\lceil x + k_{\ell 2} t$
- $\mathcal{D}$ is a set of transitions such that
  - Guards are described by rational clock constrings
  - $\langle x, l \rangle \rightarrow_a \langle x', l' \rangle$ implies $x' = x \; or x' \in [c_1, c_2]$

CSR Decidable for RHA?

- Given an RHA, check if a particular location is reachable from the initial states?
- Is this problem decidable? No
  - **[Henz95]** Thomas Henzinger, Peter Kopke, Anuj Puri, and Pravin Varaiya. What's Decidable About Hybrid Automata?. Journal of Computer and System Sciences, pages 373–382. ACM Press, 1995.
  - CSR for RHA reduction to Halting problem for 2 counter machines
  - Halting problem for 2CM known to be undecidable
  - Reduction in next lecture

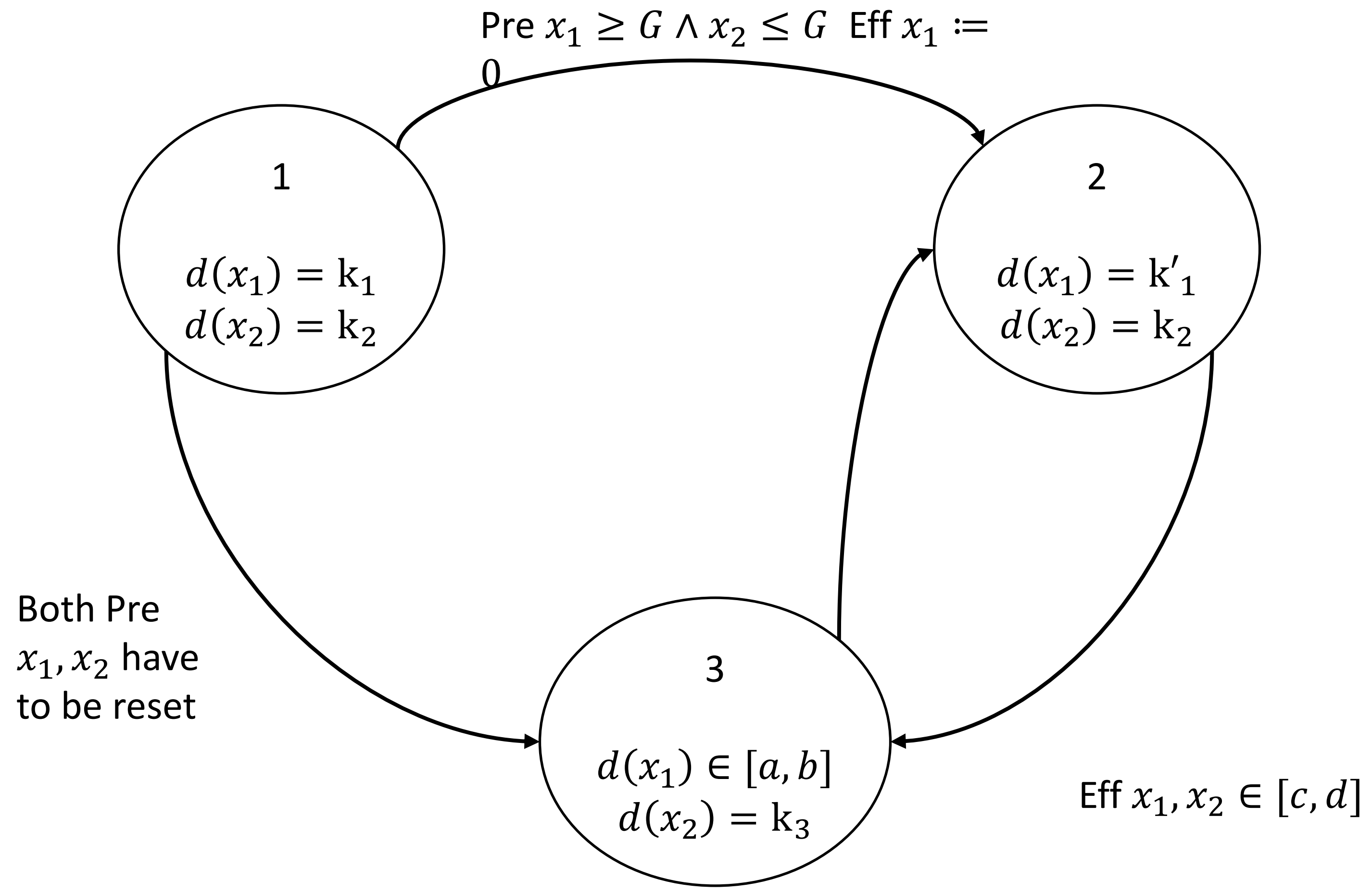
Step 4. Initialized Rectangular HA

Definition. *An initialized rectangular hybrid automaton* (IRHA) is a RHA $\boldsymbol{\mathcal{A}}$ where

- V = X ∪ $\{loc\}$, where X is a set of n continuous variables and $\{loc\}$ is a discrete state variable of finite type Ł
- A is a finite set
- $\mathcal{T} = \cup_{\ell} \mathcal{T}_{\ell}$ set of trajectories for X
    - For each $\tau \in \mathcal{T}_{\ell}, x \in X$ either (i) $d(x) = k_{\ell}$ or (ii) $d(x) \in [k_{\ell 1}, k_{\ell 2}]$
    - Equivalently, (i) $\tau(t)\lceil x = \tau(0)\lceil x + k_{\ell} t$

      (ii) $\tau(0)\lceil x + k_{\ell 1} t \leq \tau(t)\lceil x \leq \tau(0)\lceil x + k_{\ell 2} t$
- $\mathcal{D}$ is a set of transitions such that
    - Guards are described by rational clock constrings
    - $\langle x, l \rangle \rightarrow_{a} \langle x', l' \rangle$ implies if dynamics changes from ℓ to ℓ' then $x' \in [c_1, c_2]$, otherwise $x' = x$

Example: Rectangular Initialized HA



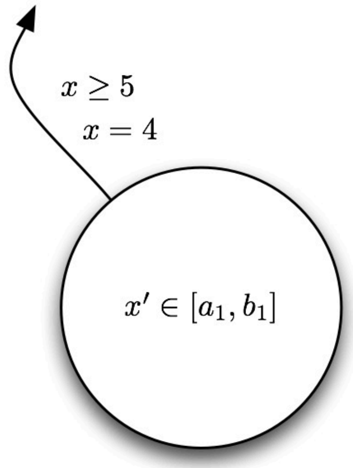
CSR Decidable for IRHA?

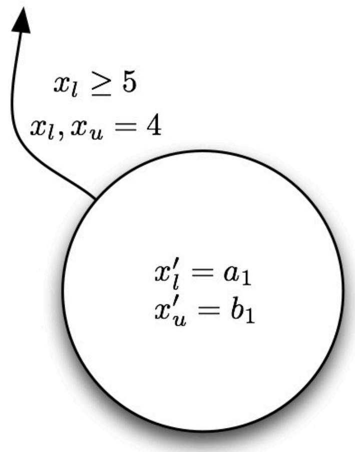
- Given an IRHA, check if a particular location is reachable from the initial states
- Is this problem decidable? Yes
- Key idea:
  - Construct a 2n-dimensional **initialized m**ulti-rate automaton that is bisimilar to the given IRHA
  - Construct a ITA that is bisimilar to the Singular TA

From IRHA to Singular HA conversion

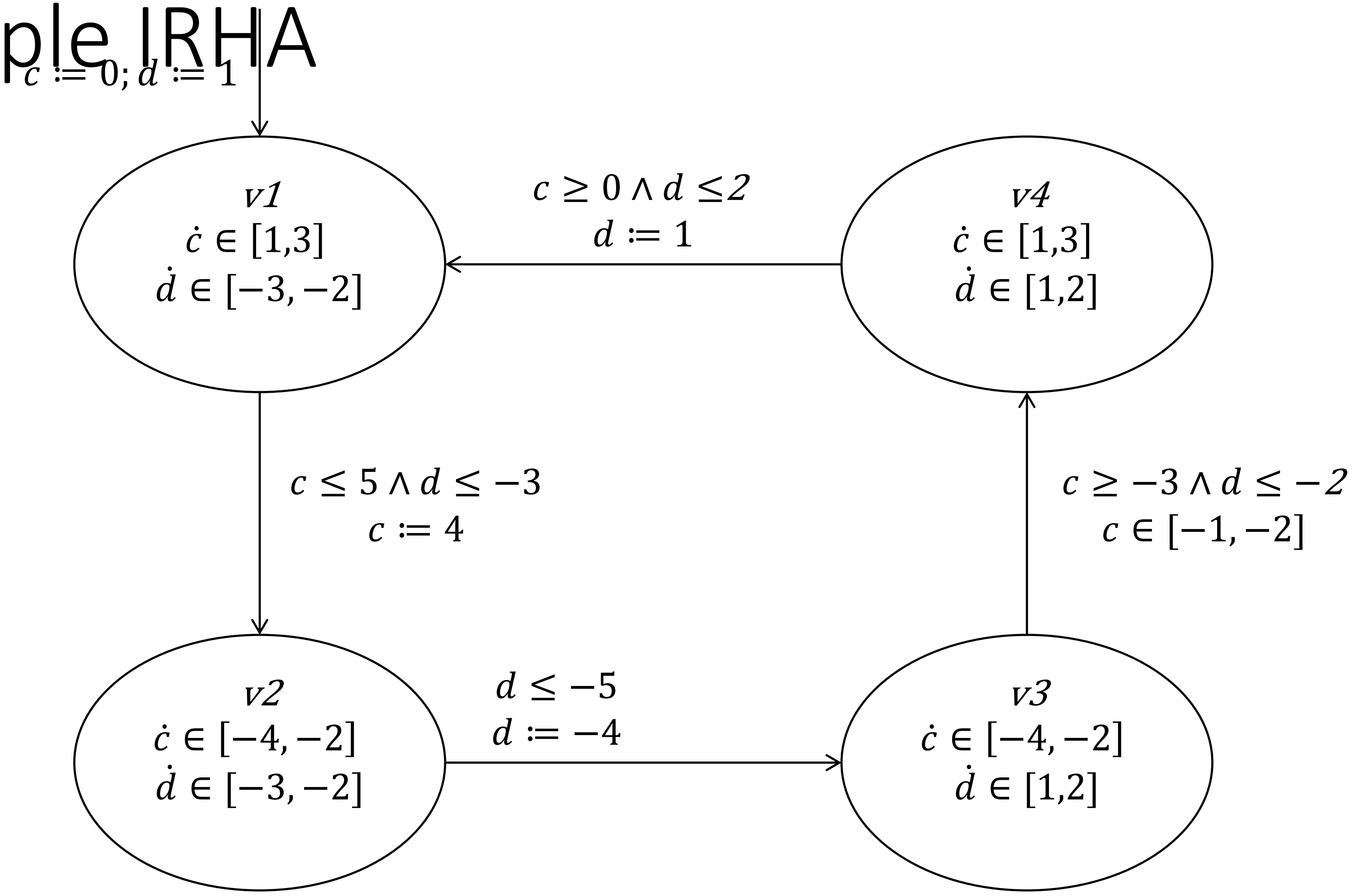
For every variable create two variables---tracking the upper and lower bounds

IRHA	MRA
x	x_ℓ ; x_u
Evolve: $d(x) \in [a_1, b_1]$	Evolve: $d(x_\ell) = a_1; d(x_u) = b_1$
Eff: $x' \in [a_1, b_1]$	Eff: $x_\ell = a_1; x_u = b_1$
$x' = c$	$x_\ell = x_u = c$
Guard: $x \geq 5$	$x_l \geq 5$
	$x_l < 5 \wedge x_u \geq 5$ Eff $x_l = 5$

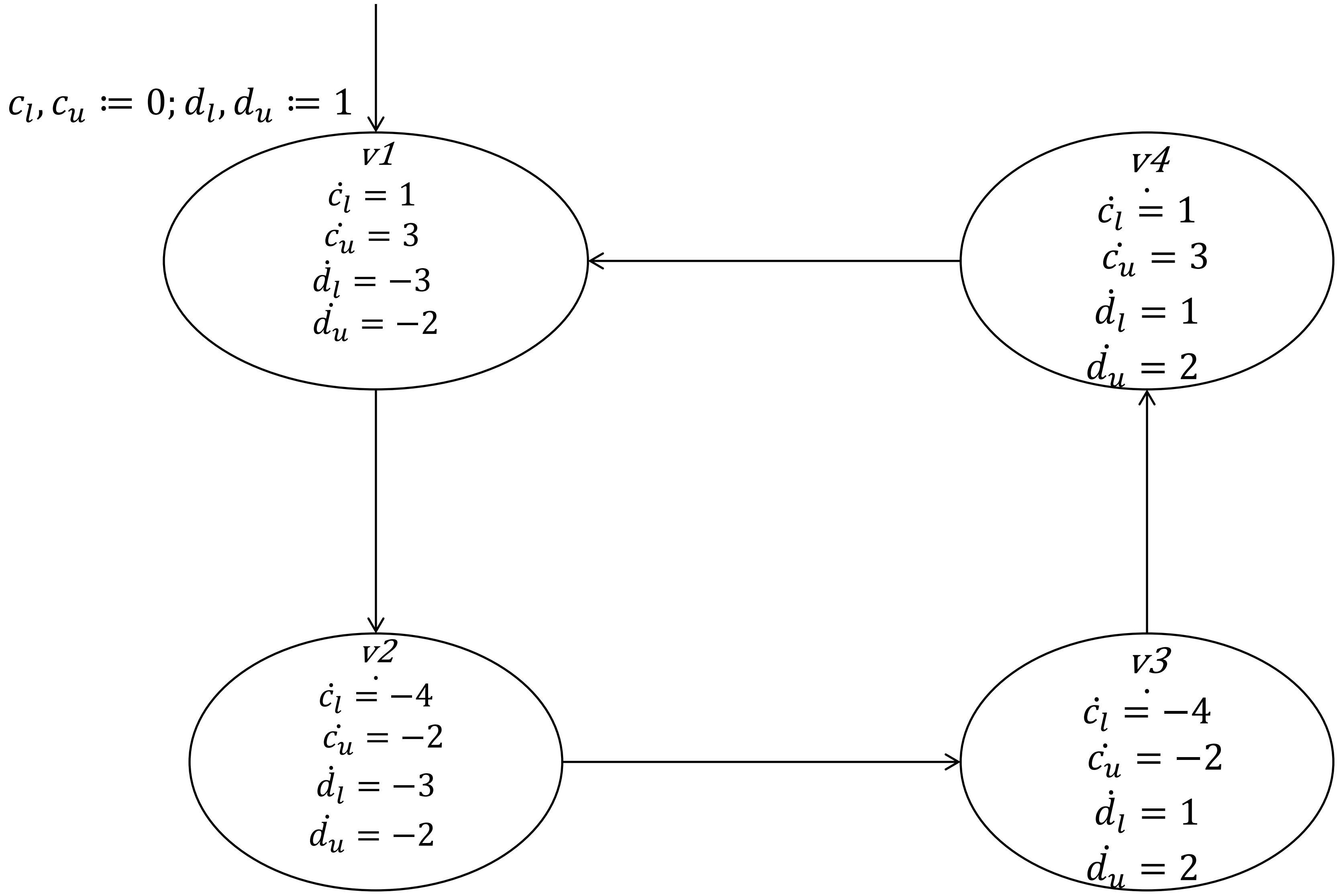




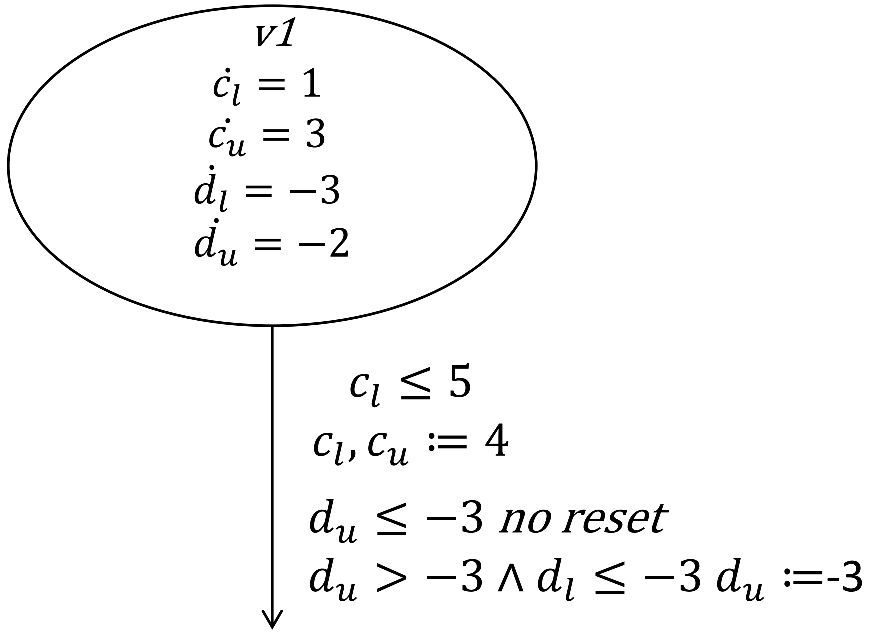
Example IRHA

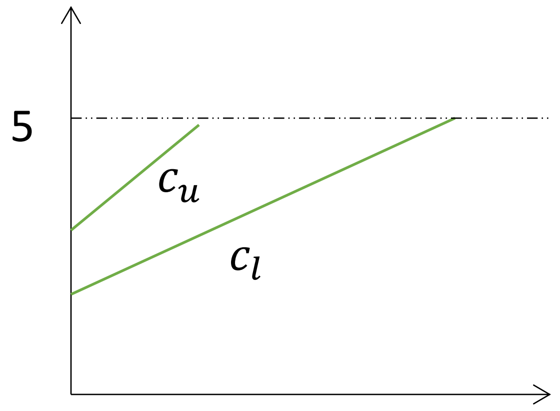


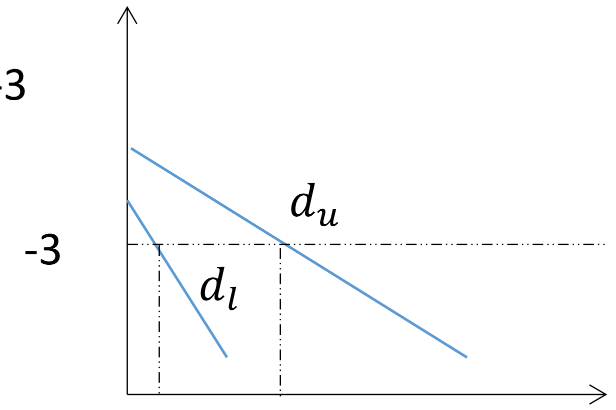
Initialized Singular HA



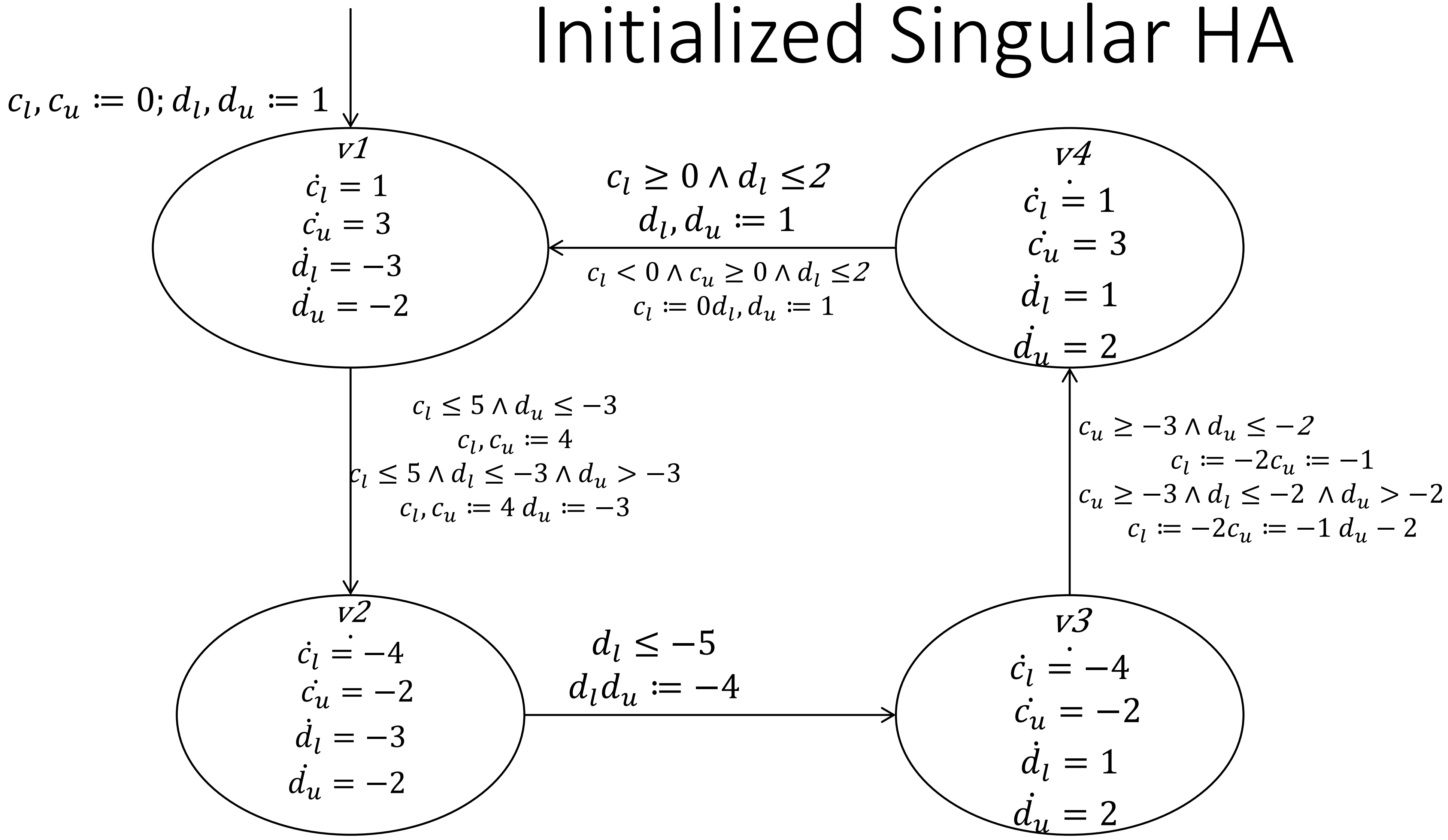
Transitions







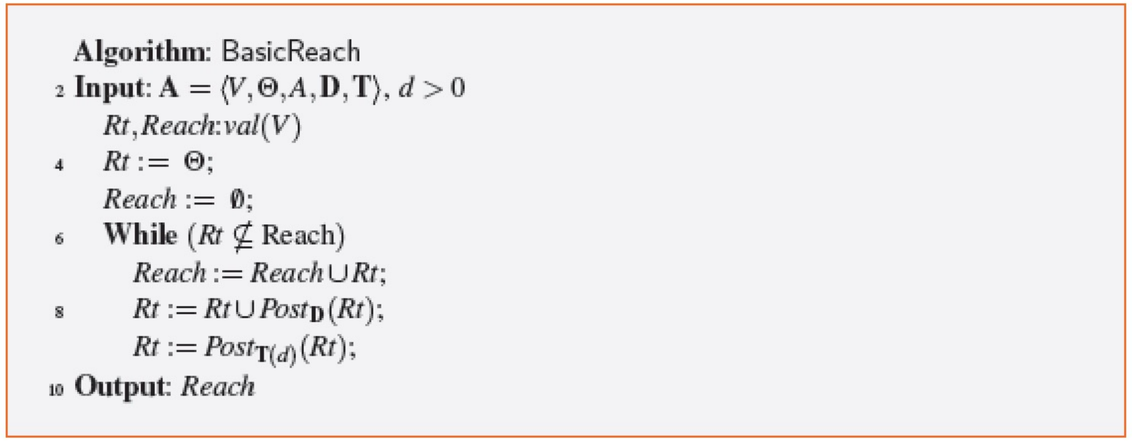
Initialized Singular HA


$c_l, c_u \coloneqq 0; d_l, d_u \coloneqq 1$
$v1$
$\dot{c_l} = 1$
$\dot{c_u} = 3$
$\dot{d_l} = -3$
$\dot{d_u} = -2$
$c_l \geq 0 \wedge d_l \leq 2$
$d_l, d_u \coloneqq 1$
$c_l < 0 \wedge c_u \geq 0 \wedge d_l \leq 2$
$c_l \coloneqq 0 d_l, d_u \coloneqq 1$
$v4$
$\dot{c_l} = 1$
$\dot{c_u} = 3$
$\dot{d_l} = 1$
$\dot{d_u} = 2$
$c_l \leq 5 \wedge d_u \leq -3$
$c_l, c_u \coloneqq 4$
$c_l \leq 5 \wedge d_l \leq -3 \wedge d_u > -3$
$c_l, c_u \coloneqq 4\ d_u \coloneqq -3$
$c_u \geq -3 \wedge d_u \leq -2$
$c_l \coloneqq -2 c_u \coloneqq -1$
$c_u \geq -3 \wedge d_l \leq -2\ \wedge d_u > -2$
$c_l \coloneqq -2 c_u \coloneqq -1\ d_u - 2$
$v2$
$\dot{c_l} = -4$
$\dot{c_u} = -2$
$\dot{d_l} = -3$
$\dot{d_u} = -2$
$d_l \leq -5$
$d_l d_u \coloneqq -4$
$v3$
$\dot{c_l} = -4$
$\dot{c_u} = -2$
$\dot{d_l} = 1$
$\dot{d_u} = 2$

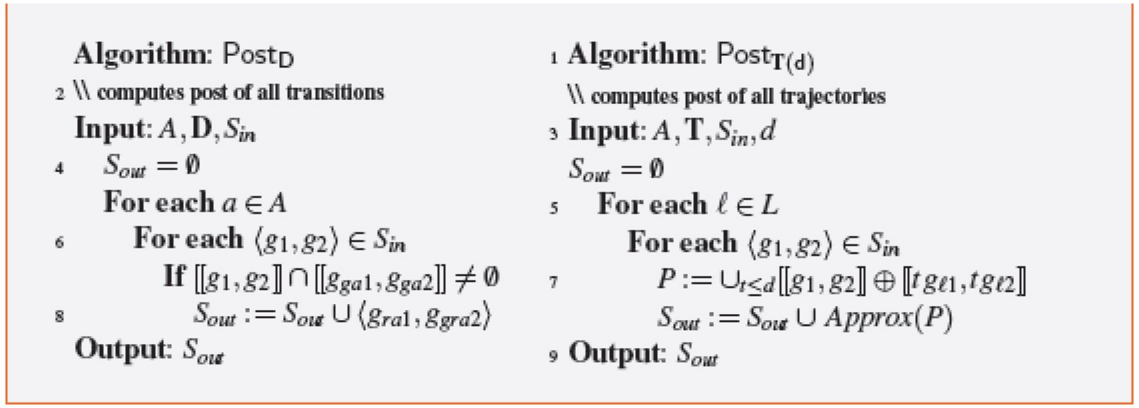
Can this be further generalized ?

- For initialized Rectangular HA, control state reachability is decidable
    - Can we drop the initialization restriction?
        - No, problem becomes undecidable (next time)
    - Can we drop the rectangular restriction?
        - No, problem becomes undecidable

Verification in tools



```
Algorithm: BasicReach
Input: $\mathbf{A} = \langle V, \Theta, A, \mathbf{D}, \mathbf{T} \rangle$, $d > 0$
  $Rt, Reach : val(V)$
  $Rt := \Theta;$
  $Reach := \emptyset;$
  While ($Rt \not\subseteq$ Reach)
    $Reach := Reach \cup Rt;$
    $Rt := Rt \cup Post_{\mathbf{D}}(Rt);$
    $Rt := Post_{\mathbf{T}(d)}(Rt);$
Output: $Reach$
```



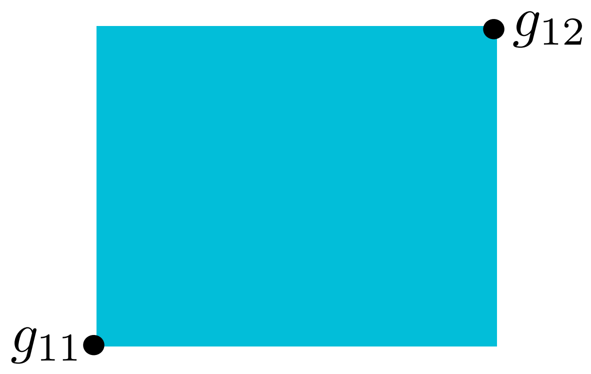
```
Algorithm: $\mathsf{Post}_{\mathsf{D}}$
\\ computes post of all transitions
Input: $A, \mathbf{D}, S_{in}$
  $S_{out} = \emptyset$
  For each $a \in A$
    For each $\langle g_1, g_2 \rangle \in S_{in}$
      If $[[g_1, g_2]] \cap [[g_{ga1}, g_{ga2}]] \neq \emptyset$
        $S_{out} := S_{out} \cup \langle g_{ra1}, g_{gra2} \rangle$
Output: $S_{out}$
```

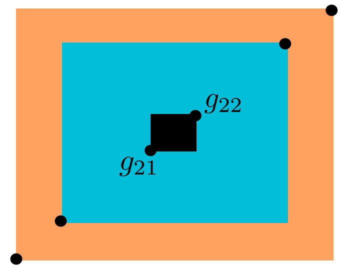
```
Algorithm: $\mathsf{Post}_{\mathbf{T}(\mathsf{d})}$
\\ computes post of all trajectories
Input: $A, \mathbf{T}, S_{in}, d$
$S_{out} = \emptyset$
  For each $\ell \in L$
    For each $\langle g_1, g_2 \rangle \in S_{in}$
      $P := \cup_{t \leq d} [[g_1, g_2]] \oplus [[t g_{\ell 1}, t g_{\ell 2}]]$
      $S_{out} := S_{out} \cup Approx(P)$
Output: $S_{out}$
```

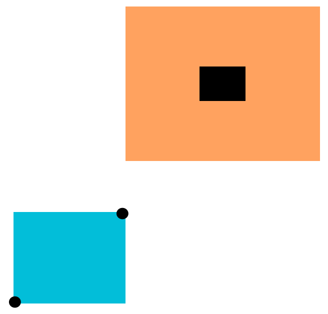
Data structures make reachability go around

- Hyperrectangles
  - $[[g_1; g_2]] = \{x \in R^n \mid \ ||x - g_1||_\infty \leq ||g_2 - g_1||_\infty\} = \Pi_i[g_{1i}, g_{2i}]$
- Polyhedra
- Zonotopes [Girard 2005]
- Ellipsoids [Kurzhanskiy 2001]
- Support functions [Guernic et al. 2009]
- Generalized star set [Duggirala and Viswanathan 2018]

Data structures: rectangles and ellipsoids



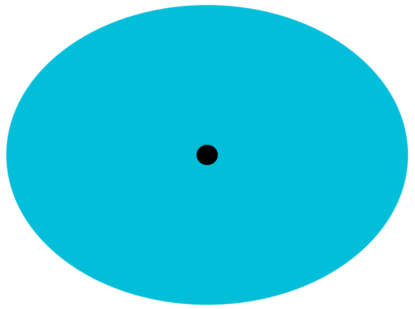


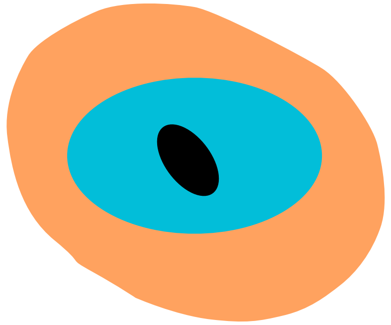


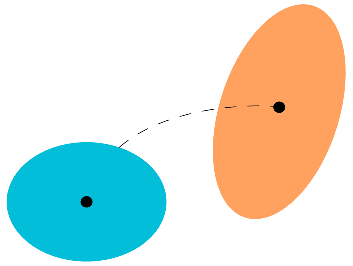
$[[g_{11}, g_{12}]]$

$[[g_{11}, g_{12}]] \oplus [[g_{21}, g_{22}]]$
$= [[g_{11} + g_{12}, g_{21} + g_{22}]]$

$[[g_{11}, g_{12}]] \oplus [[t.g_1, t.g_2]]$





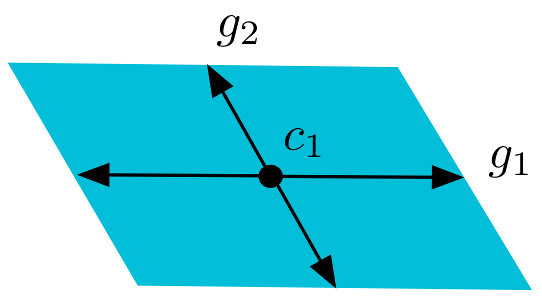


$[[c_1, Q]]$

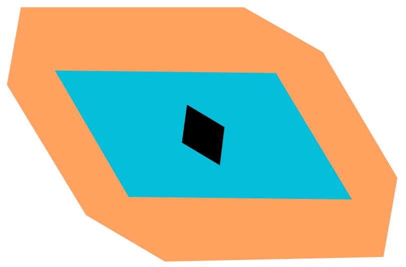
$[[c_1, Q_1]] \oplus [[c_2, Q_2]] \neq [[c_3, Q_3]]$

$[[Ac_1, AQA^T]]$

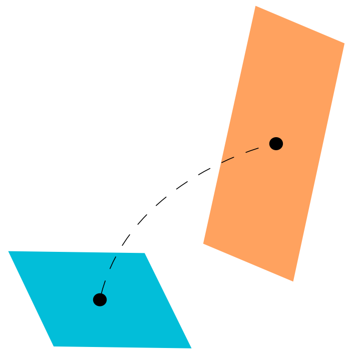
Zonotopes and polytopes



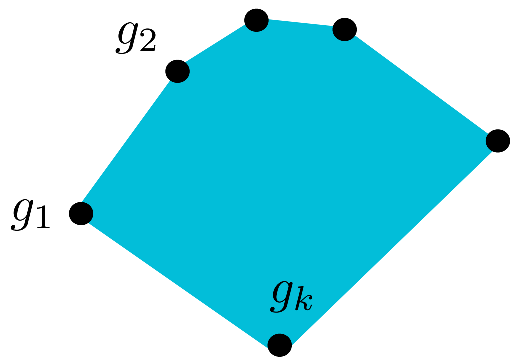
$[[c_1, \langle g_1, g_2 \rangle]]$



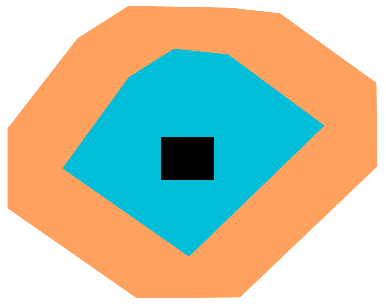
$[[c_1, \langle g_1, g_2 \rangle]] \oplus [[c_2, \langle g'_1, g'_2 \rangle]]$
$= [[c_1 + c_2, \langle g_1, g'_1, g_2, g'_2 \rangle]]$

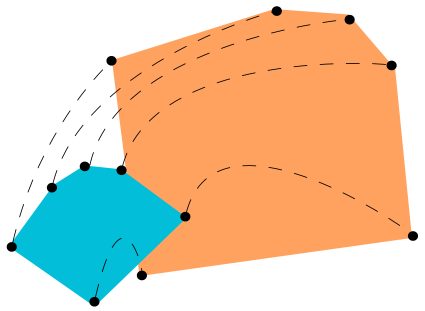


$[[Ac_1, \langle Ag_1, Ag_2 \rangle]]$



$[[A, b]]$
$[[g_1, \ldots, g_k]]$





$[[\xi(g_1, t), \ldots, \xi(g_k, t)]]$

Summary

- ITA: Restricted class of hybrid automata
  - Clocks, integer constraints
  - No clock comparison, linear
- Control state reachability with Alur-Dill’s algorithm (region automaton construction)
- Rational coefficients; multirate Automata
- Initialized Rectangular Hybrid Automata
- HyTech, PHAVer use polyhedral reachability computations