Lecture 20: Timed automata and its reachability (cont.)

Huan Zhang

huan@huan-zhang.com

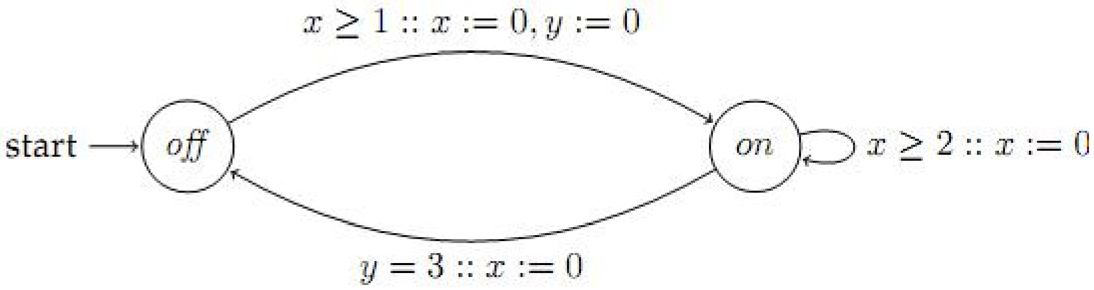
Slides adapted from Prof. Sayan Mitra's slides in Fall 2021

Review: Integral Timed Automata (ITA)

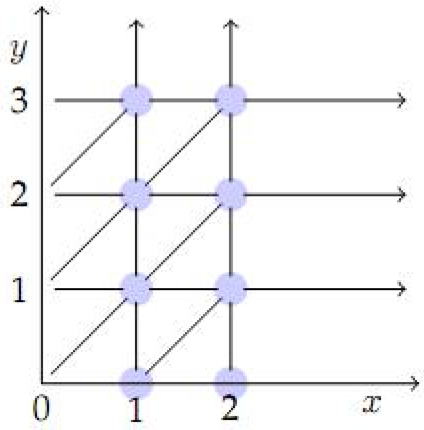
- **Definition.** A **integral timed automaton** $\mathcal{A} = \langle V,\ \Theta,\ A,\ \mathcal{D},\ \mathcal{T} \rangle$ where
  - V = X ∪ $\{l\}$, where X is a set of n clocks and l is a discrete state variable of finite type L**.** The **stata space is** val(V) × L
  - A is a finite set
  - $\mathcal{D}$ is a set of transitions such that
    - The preconditions are described by clock constrains $\Phi(X)$
    - $\langle x, l \rangle_a \rightarrow \langle x', l' \rangle$ implies either $x' = x$ or $x' = 0$ (time is reset to 0, or no change)
  - $\mathcal{T}$ set of clock trajectories for the clock variables in X

Review: Control-state Reachability of ITA: construct a bisimilar FA

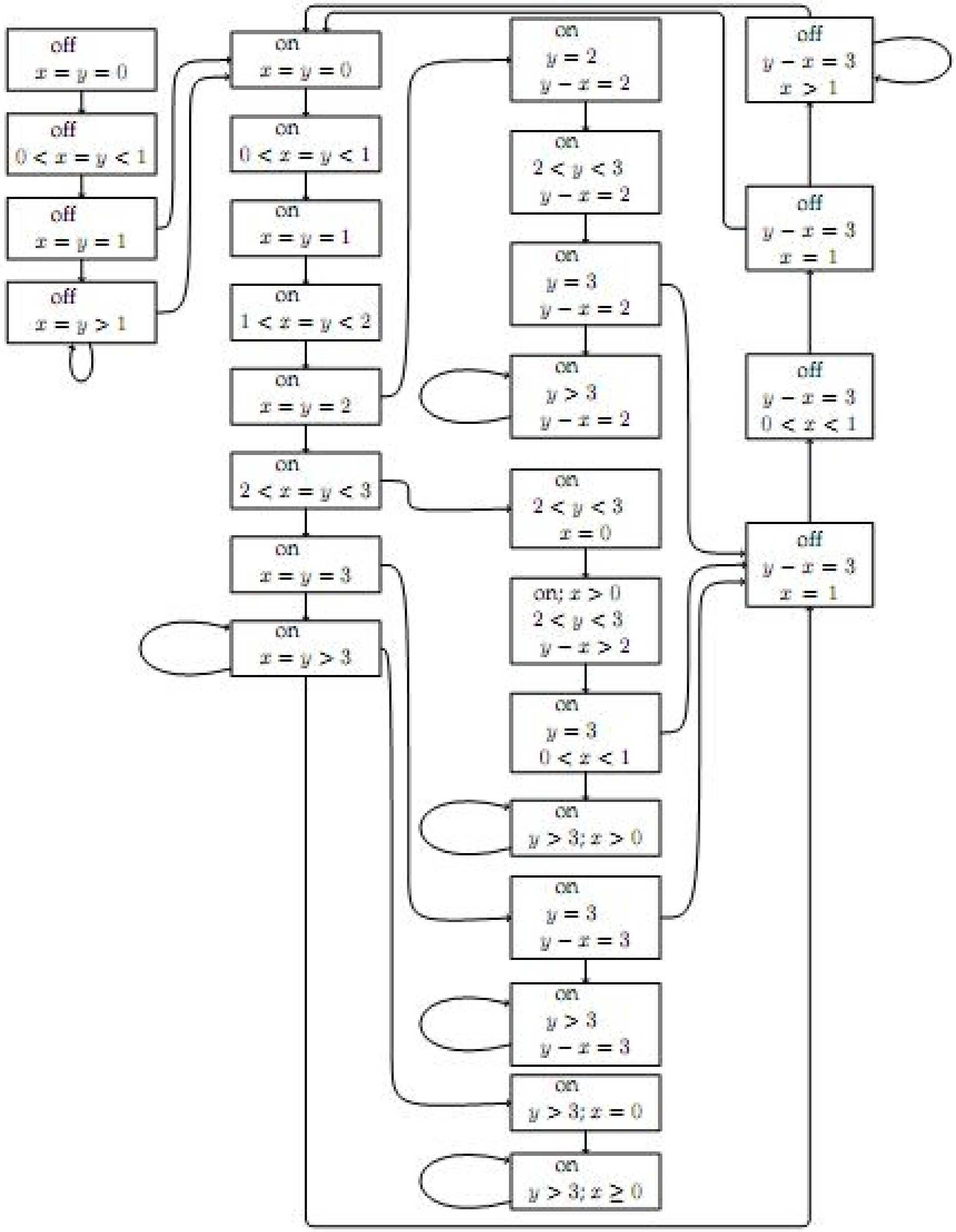
ITA



Clock Regions



Corresponding FA



$$|X|!\; 2^{|X|} \prod_{z \in X} (2c_{\mathcal{A}z} + 2)$$

Drastically increasing with the number of clocks

Rational Timed Automata: Decidable

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 L
- A is a finite set
- $\mathcal{D}$ 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 $x' = x$ or $x = 0$
- $\mathcal{T}$ set of clock trajectories for the clock variables in X

Convert to ITA by multiply clocks by a factor q

Multi-Rate Automaton: decidable

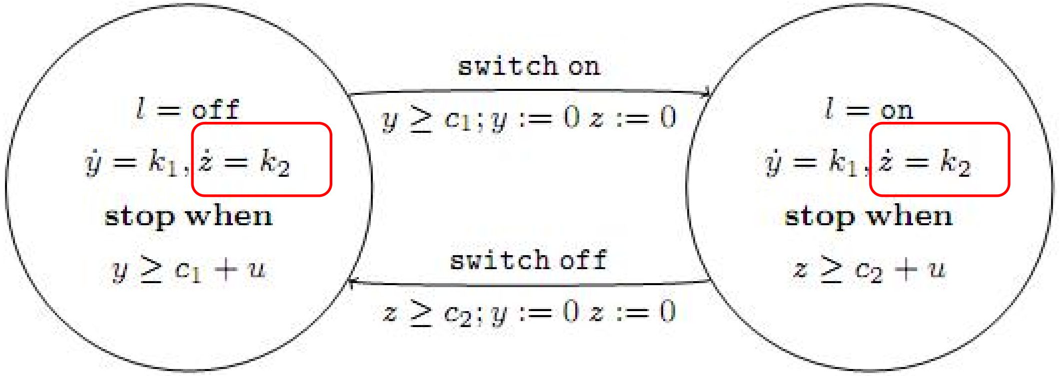
- **Definition.** A **multirate automaton** is $\boldsymbol{\mathcal{A}}$ = $\langle V,\ Q,\ \Theta,\ A,\ \mathcal{D},\ \mathcal{T}\rangle$ where
  - V = X ∪ $\{loc\}$, where X is a set of n **continuous variables** and loc is a discrete state variable of finite type L
  - A is a finite set of actions
  - $\mathcal{D}$ 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

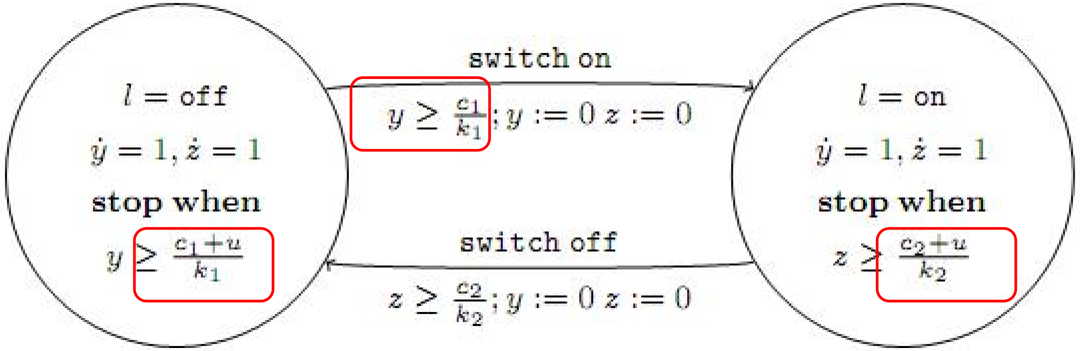
    for each variable $x \in X\ \exists k\ such\ that\ \tau \in \mathcal{T},\ t \in \tau.dom$

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

Convert to RTA by multiply clocks by a factor q

Example: Multi-rate to rational TA





Rectangular HA: undecidable

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

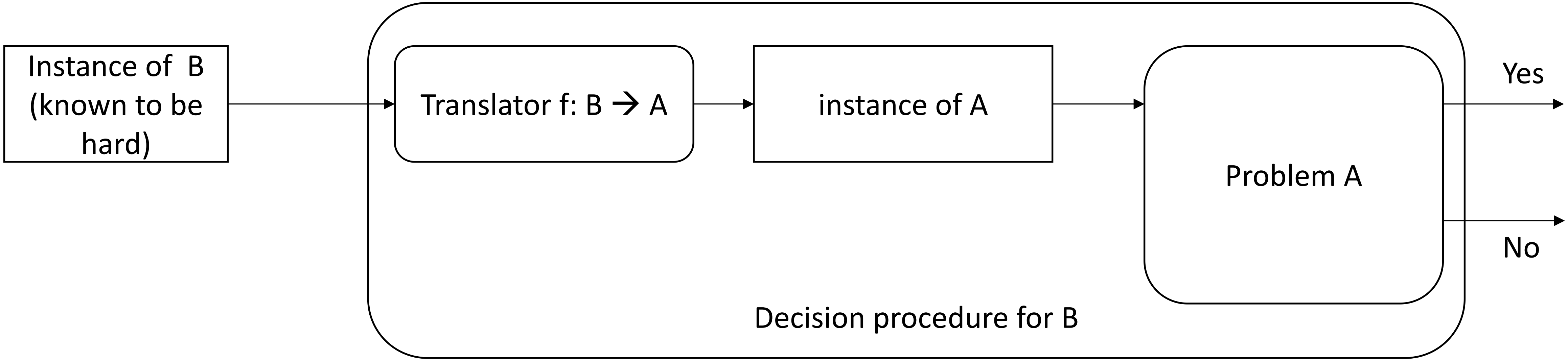
- V = X ∪ $\{loc\}$, where X is a set of n **continuous variables** and loc is a discrete state variable of finite type L
- A is a finite set
- $\mathcal{T} = \cup_\ell \mathcal{T}_\ell$ set of trajectories for X
  - For each $\tau \in \mathcal{T}_\ell$, $x \in X$ either (i) $d(x) = k_\ell$ or (ii) $d(x) \in [k_{\ell 1}, k_{\ell 2}]$
  - 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* $x' \in [c_1, c_2]$

Reachability of Rectangular HA

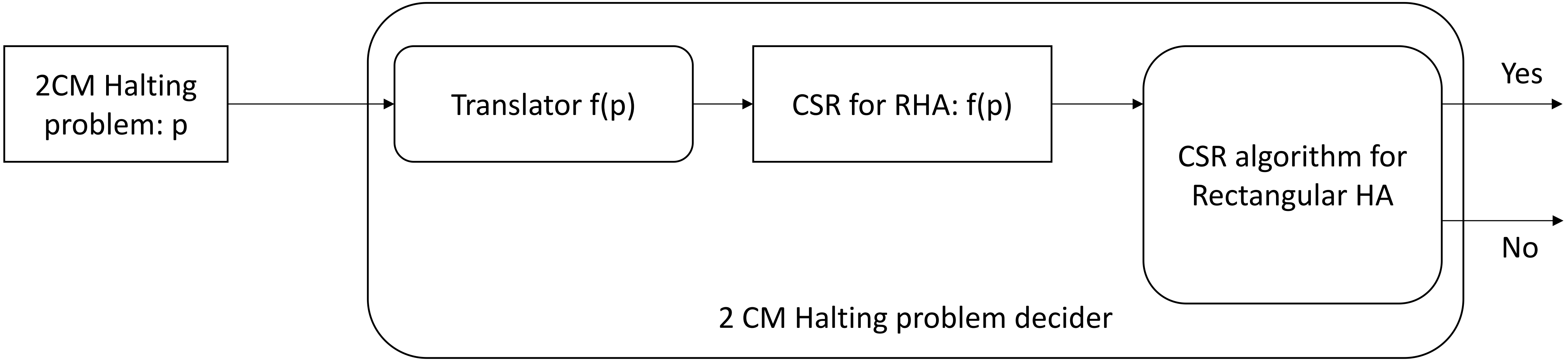
- Is this problem decidable? No
  - **[Henz95]** Thomas Henzinger, Peter Kopke, Anuj Puri, and Pravin Varaiya. What's Decidable About Hybrid Automata?. Journal of Computer and System Sciences, pages 373–382. ACM Press, 1995.
- We will see that the control state reachability (CSR) problem for rectangular hybrid automata (RHA) is undecidable
- This implies that automatic verification of invariants and safety properties is also impossible for this class of models
- The result was shown by Henzinger et al. [1995] through a *reduction from* the Halting problem of two counter machines

General reductions: Using known hard problem B to show hardness of A



Given B is known to be hard
Suppose (for the sake of contradiction) A is solvable
If we can construct a reduction f: B→A (from B to A) then B becomes easy, which is a contradiction

Reduction from Halting Problem for 2CM



Suppose CSR for RHA is decidable
If we can construct a reduction from 2CM Halting Problem to CSR for RHA then 2CM Halting problem is also decidable

Counter Machines

An n-counter machine is an elementary computer with n-unbounded counters and a finite program written in a minimalistic assembly language.

More precisely: A 2-counter machine (2CM) is a discrete transition system with the following components:

- Two **nonnegative** integer counters C and D. Both are **initialized to 0**.
- A finite program with one of these instructions at each location (or line):
  - INCC, INCD: increments counter C (or D)
  - DECC, DECD: decrements counter C (or D), provided it is not 0,
  - JNZC, JNZD [label]: moves the program control to line *label* provided that counter C (or D) is not zero.

Example 2CM for multiplication

A 2-counter machine for multiplying 2x3 is shown below.

```
INCC;
INCC;          % C = 2
INCD;          % LOOP
INCD;
INCD;
DECC;
JNZC 3;        % Jump to LOOP
               % HALT
```

Exercise: Show that any k-counter machine can be simulated by a 2CM.

Halting problem for 2CM

- A **configuration** of a 2CM is a triple (pc, C, D)
  - pc is the program counter that stores the next line to be executed
  - C, D are values of the counter
- A sequence of configurations (pc0, D0, C0), (pc1, D1, C1), … is an **execution** if the ith configuration goes to the (i+1)st configuration in the sequence executing the instruction in line pci
- Given a 2CM **M** and a special halting location (pc_halt), the Halting problem requires us to decide whether all executions of **M** reach the halting location
- Theorem [Minsky 67]. The Halting problem for 2CMs is undecidable.

Reduction from 2CM to CSR-RHS

We have to construct a function (reduction) that maps instances of 2CM-Halt to instances of CSR-RHA

Reduction from 2CM to CSR-RHS

- Program counter pc
- Counters C, D
- Instructions (program)
- Halting location

- Locations, sequence of locations
- Clocks c, d that can go at some constant rates $k_1, k_2, \dots$
- Transitions: *widgets*
- Particular location / control state (to which we will check CSR)

Idea of reduction (an RHA compiler)

- Two clocks (k_2>k_1)
  - $c\ = k_1\left(\frac{k_2}{k_1}\right)^{C}$
  - d $= k_1\left(\frac{k_2}{k_1}\right)^{D}$

- INCC
  - $k_1\left(\frac{k_2}{k_1}\right)^{C+1} = c\left(\frac{k_2}{k_1}\right)$

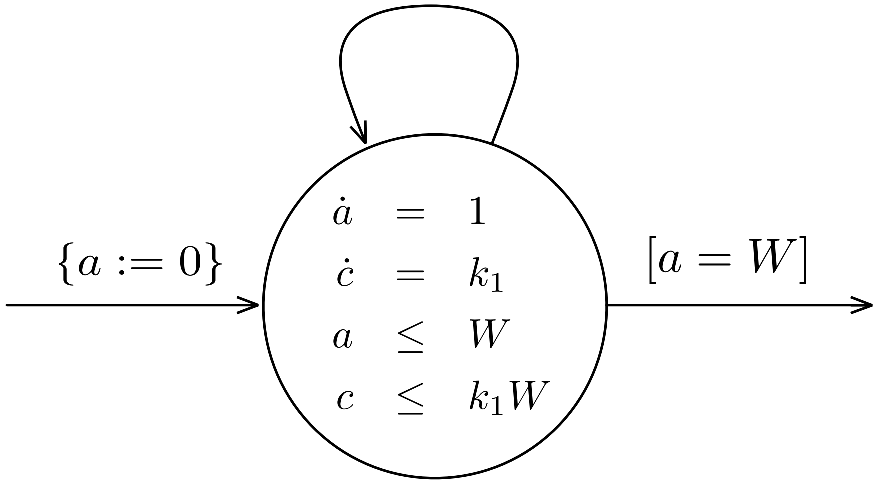
- DECC
  - $k_1\left(\frac{k_2}{k_1}\right)^{C-1} = c\left(\frac{k_1}{k_2}\right)$

- checking nonzero:
  - $c > k_1$

A widget that preserves the value of clock c

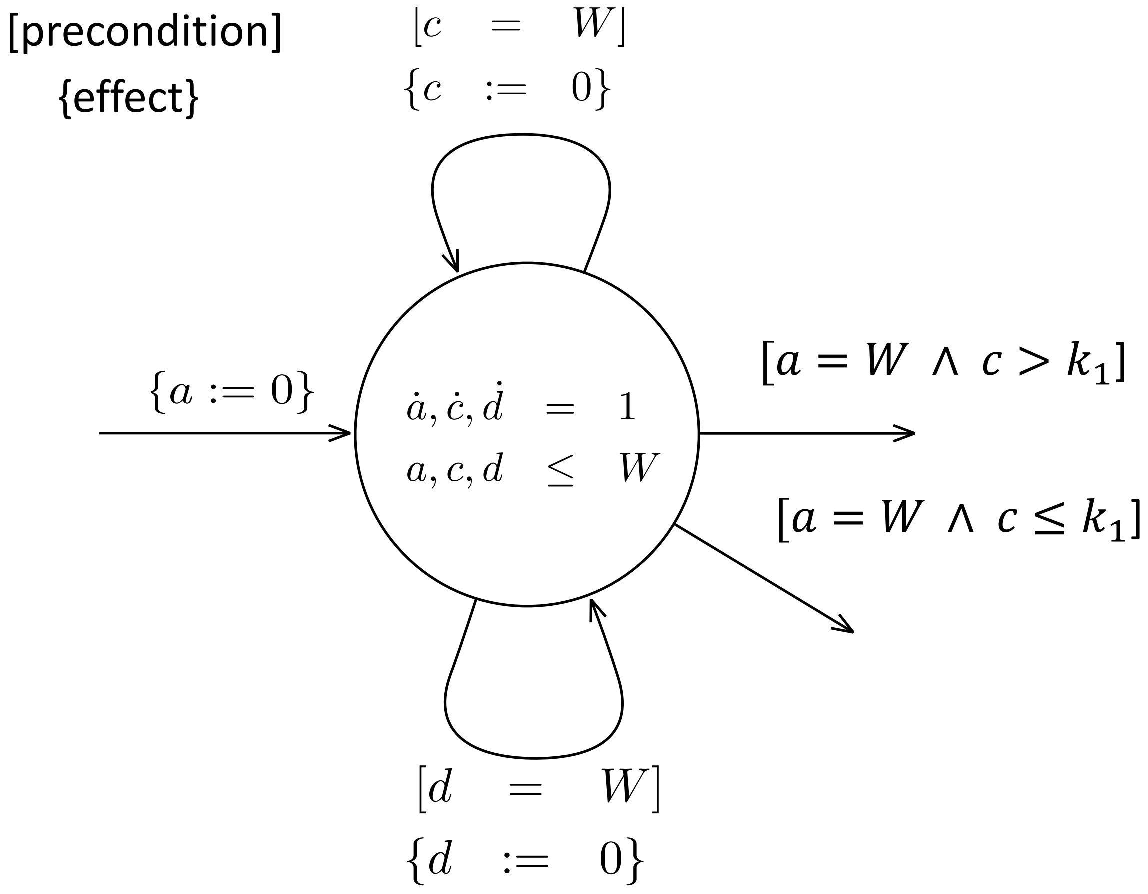
[precondition] $[c \; = \; Wk_1]$

{effect} $\{c \; := \; 0\}$

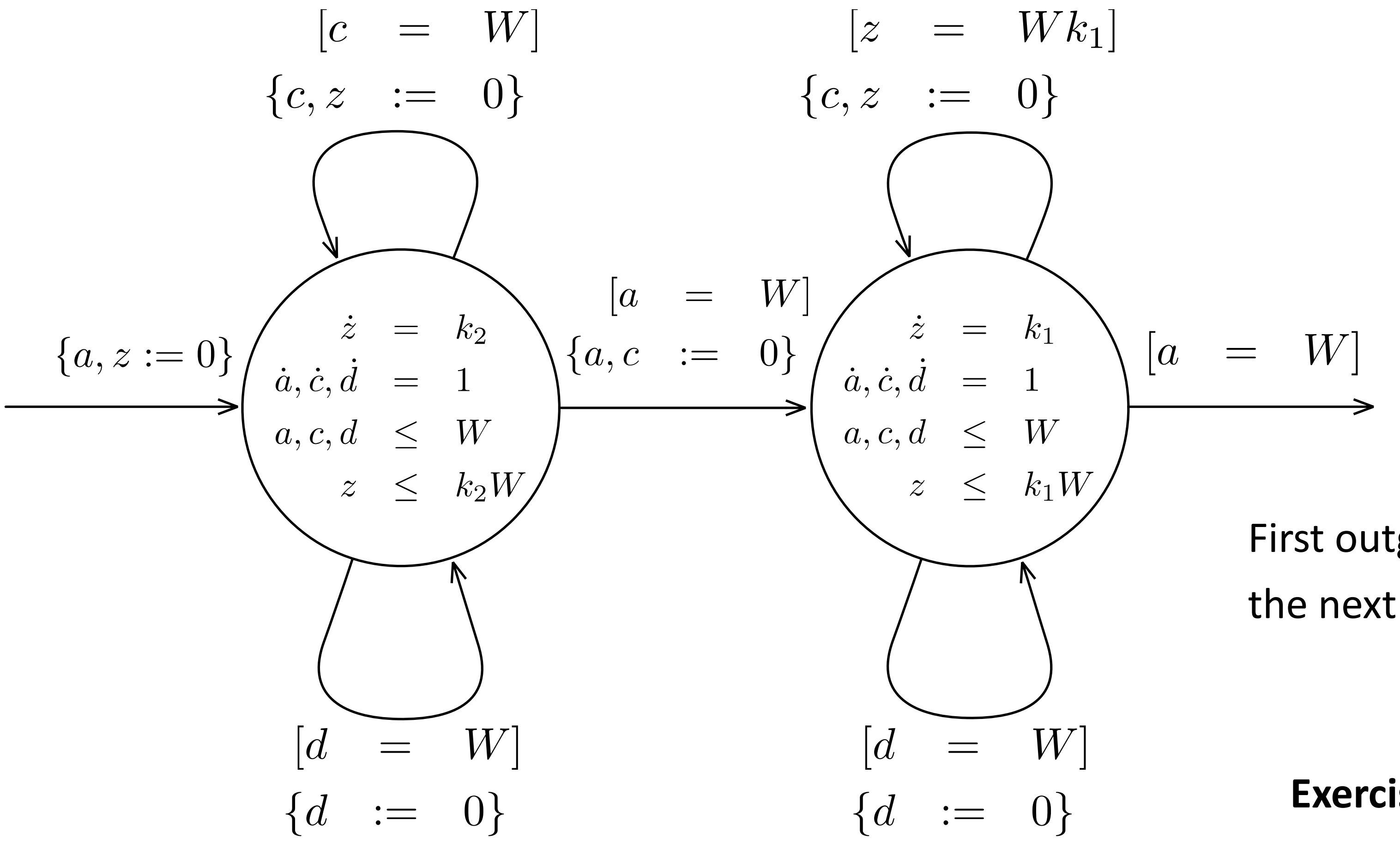


Transitions and clock guards for this control state

A widget for checking JNZC (c < k_1)



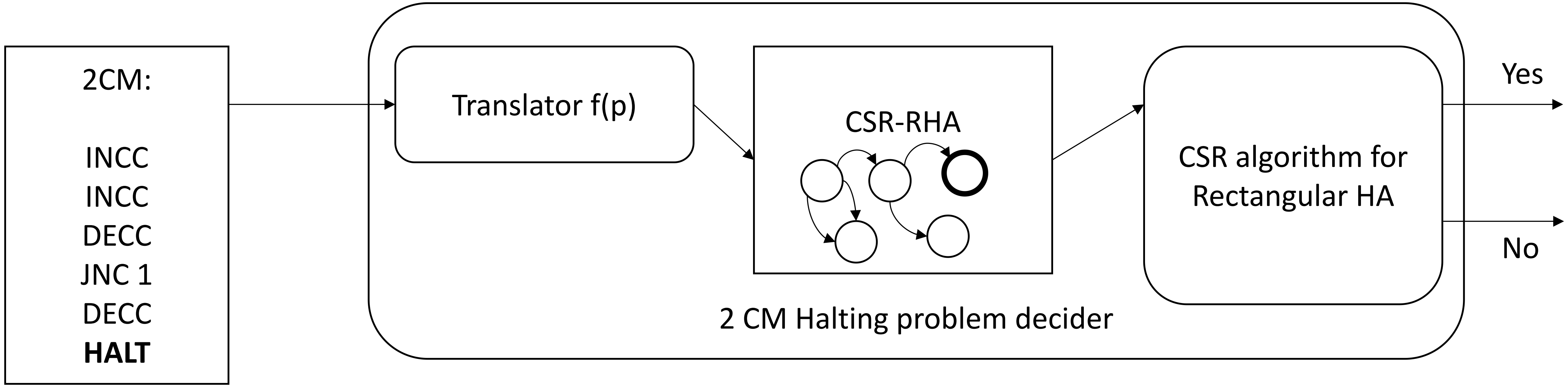
A widget implementing INCC



First outgoing transition sets $z = k_2 c$ and the next outgoing transition sets c $= z * \left(\frac{1}{k_1}\right)$

Exercise: Show the widget for DECC

Putting it all together



Suppose CSR for RHA is decidable
If we can construct a reduction from 2CM Halting Problem to CSR for RHA then 2CM Halting problem is also decidable
Theorem: CSR for RHA is undecidable

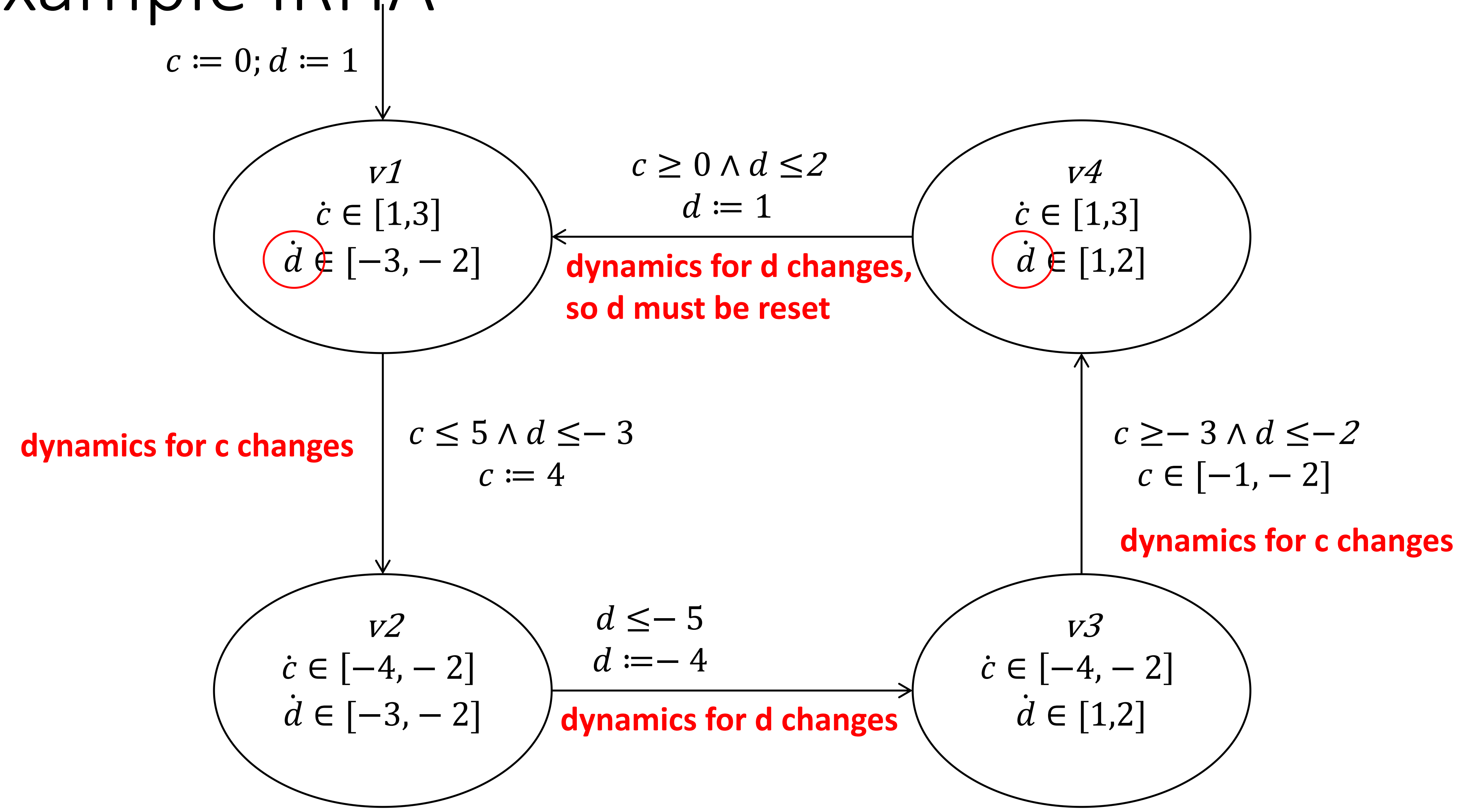
Initialized Rectangular HA

Definition. ***An initialized rectangular hybrid automaton* (IRHA)** is a RHA $\mathcal{A}$ 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** $d(x)$ **changes** from ℓ to ℓ' then $x' \in [c_1, c_2]$, otherwise $x' = x$ if $d(x)$ is not changed

Example IRHA



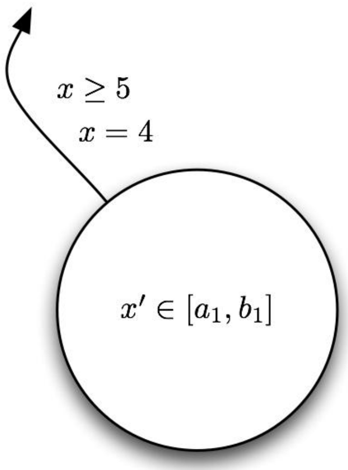
CSR Decidable for IRHA?

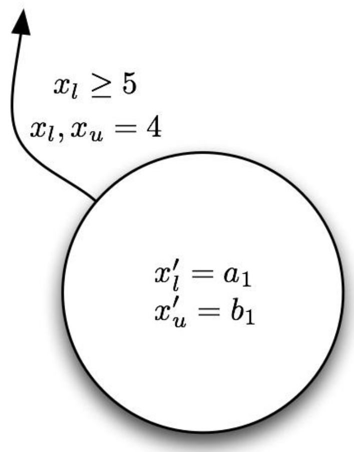
- Given an IRHA, check if a particular location is reachable from the initial states
- Is this problem decidable? Yes
- Key idea:
  - Construct a 2n-dimensional **initialized multi-rate automaton** that is bisimilar to the given IRHA
  - Construct a ITA that is bisimilar to the Singular TA

From IRHA to Singular HA conversion

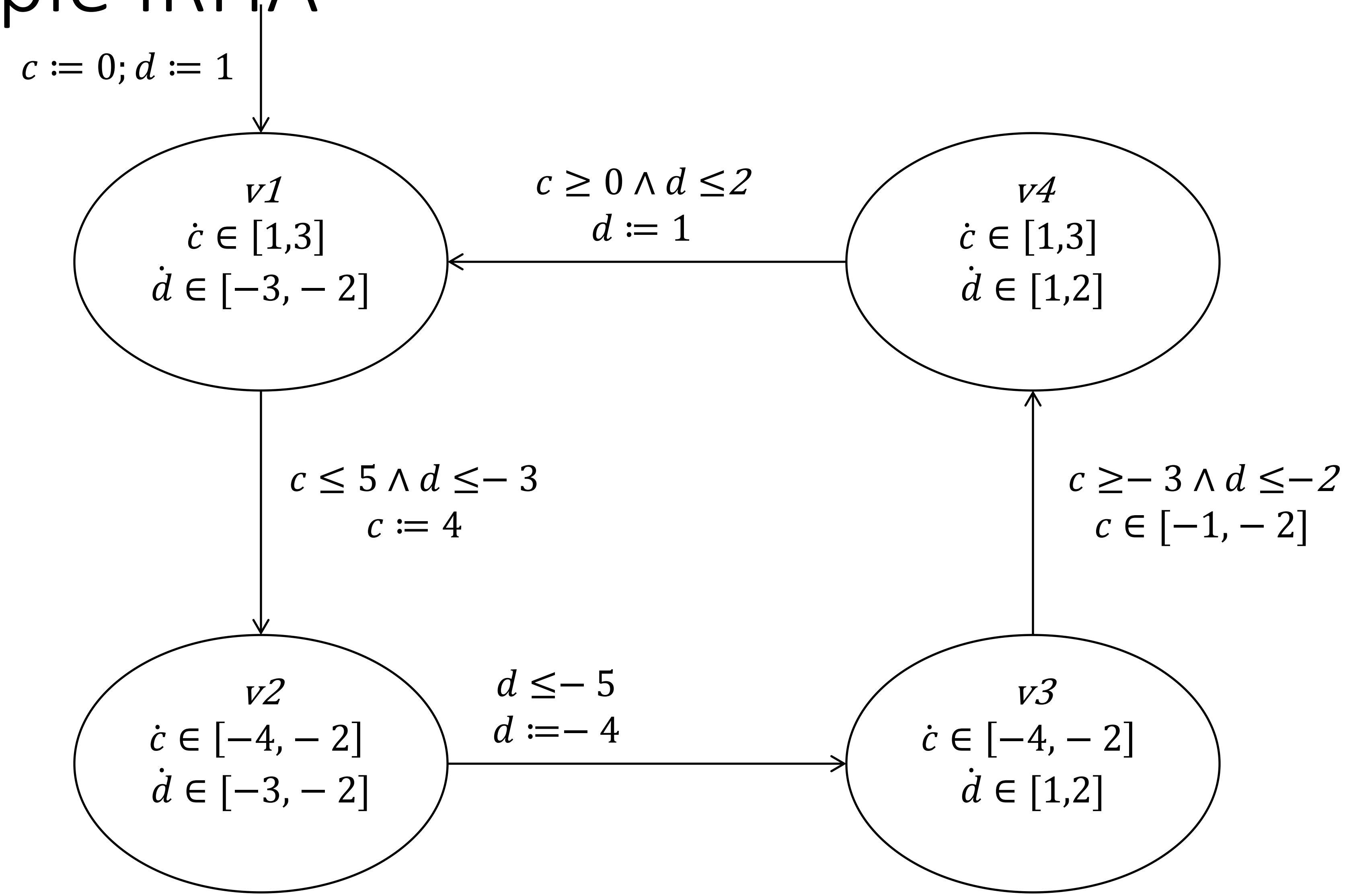
For every variable create two variables---tracking the upper and lower bounds

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 $x_l = 5$

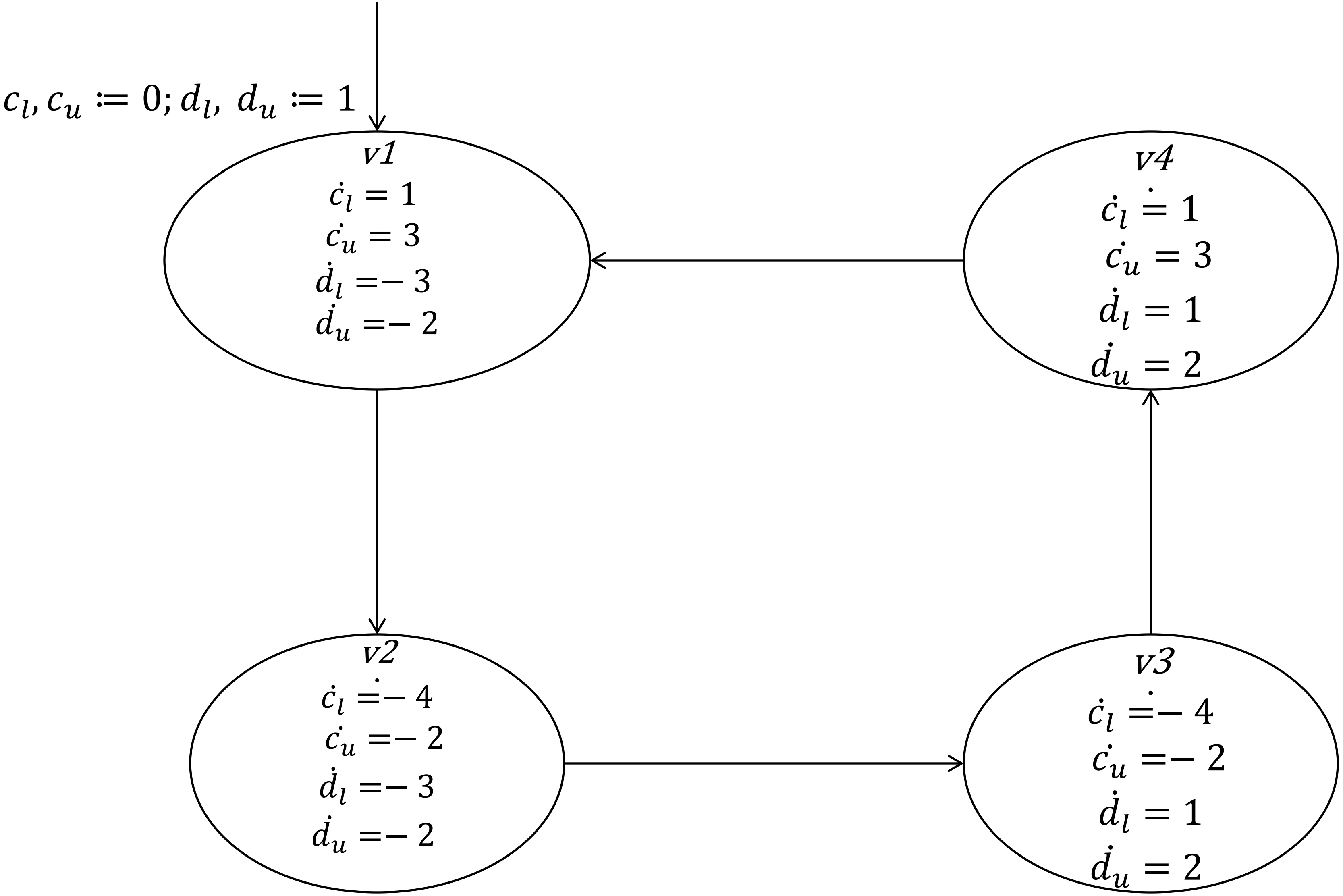




Example IRHA



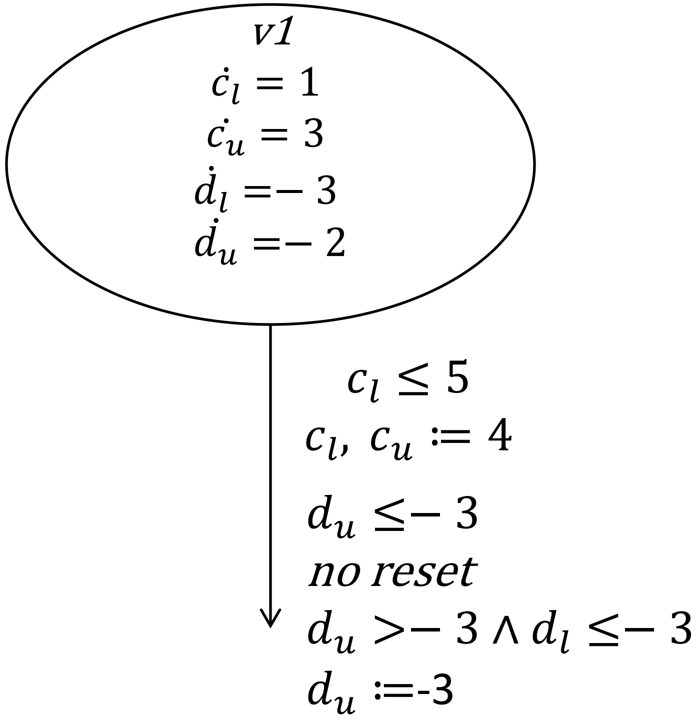
Initialized Singular HA

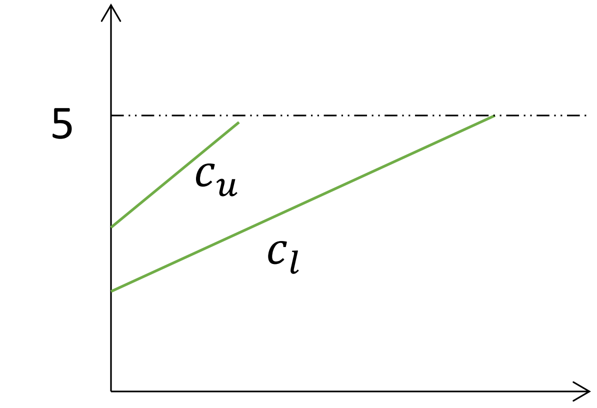


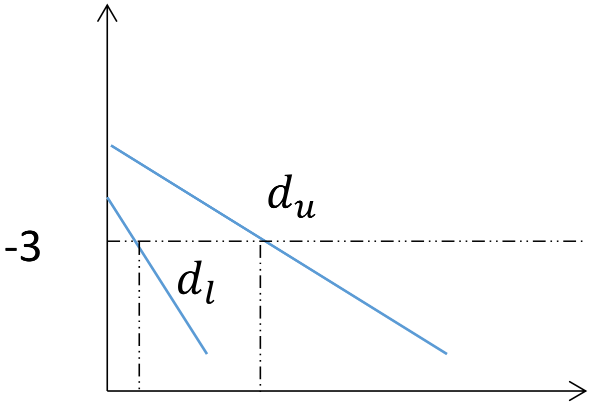
Transitions

From IRHA:

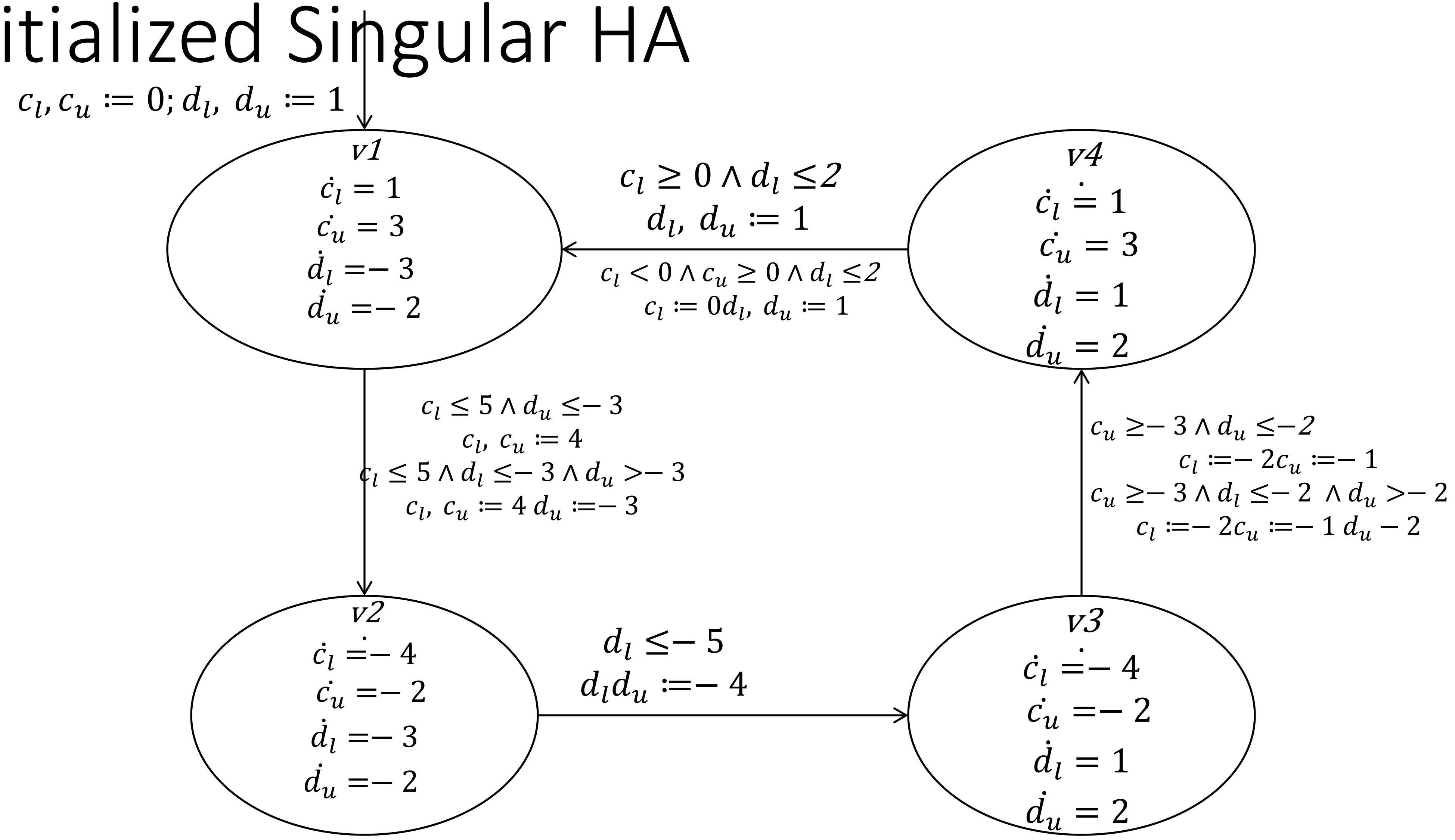
$c \leq 5 \wedge d \leq -3$
$c := 4$







Initialized Singular HA



Practical reachability

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 $(Rt \not\subseteq \text{Reach})$
    $Reach := Reach \cup Rt;$
    $Rt := Rt \cup Post_{\mathbf{D}}(Rt);$
    $Rt := Post_{\mathbf{T}(d)}(Rt);$
Output: $Reach$

Algorithm: Post_{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}

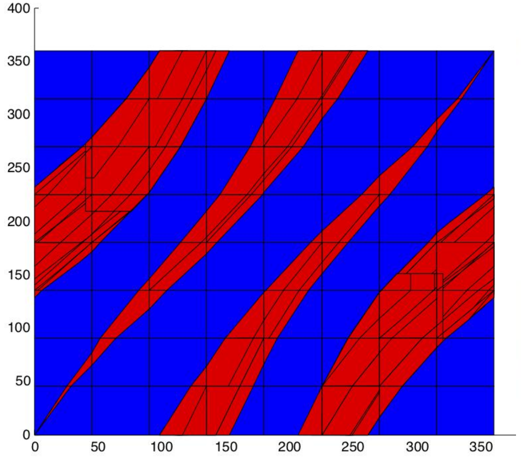
Algorithm: $\text{Post}_{\mathbf{T}(\mathrm{d})}$
\\\\ computes post of all trajectories
Input: $A, \mathbf{T}, S_{in}, d$
  $S_{out} = \emptyset$
  For each $\ell \in L$
    For each $\langle g_1, g_2 \rangle \in S_{in}$
      $P := \cup_{t \leq d} [[g_1, g_2]] \oplus [[t g_{\ell 1}, t g_{\ell 2}]]$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

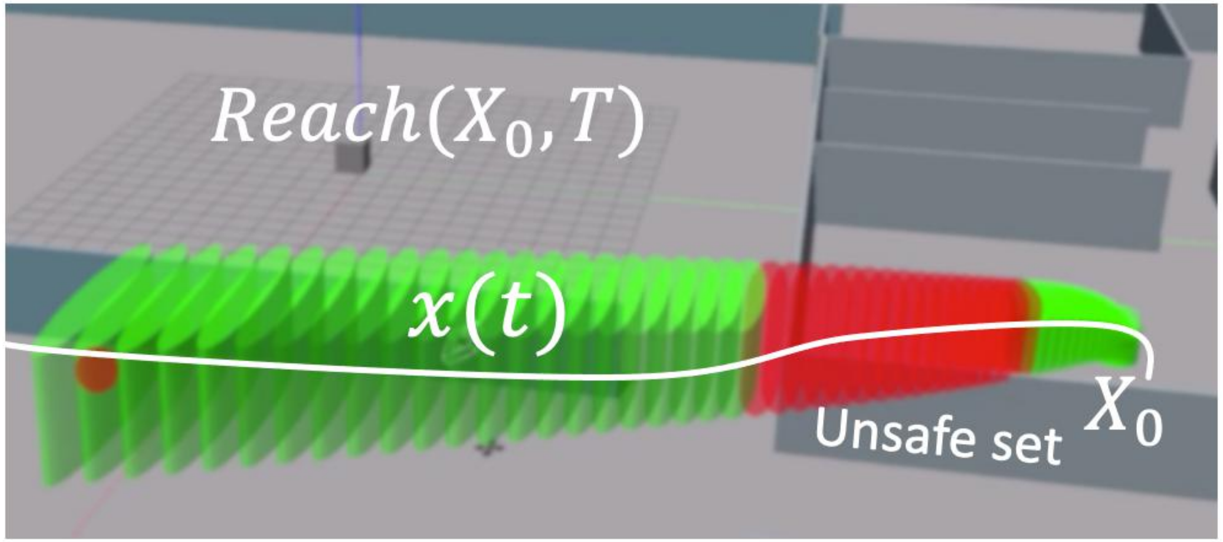
Tools:
SpaceEX
CORA
C2E2
Flow*
DryVR

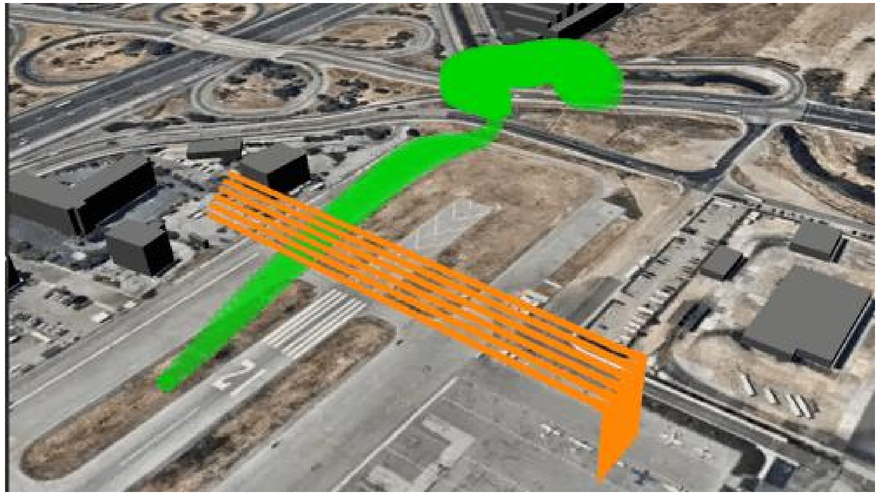
Data structures critical for reachability

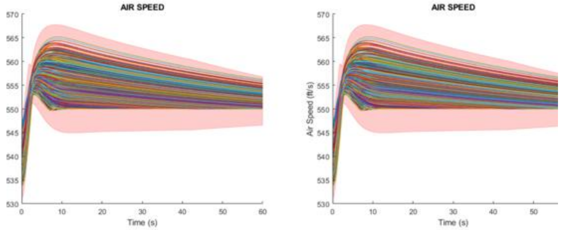
- Hyperrectangles
  - $[[g_1; g_2]] = \{x \in R^n \mid \ ||x - g_1||_\infty \leq ||g_2 - g_1||_\infty\} = \Pi_i[g_{1i}, g_{2i}]$
- Polyhedra
- Zonotopes [Girard 2005]
- Ellipsoids [Kurzhanskiy 2001]
- Support functions [Guernic et al. 2009]
- Generalized star set [Duggirala and Viswanathan 2018]

Reachability in practice

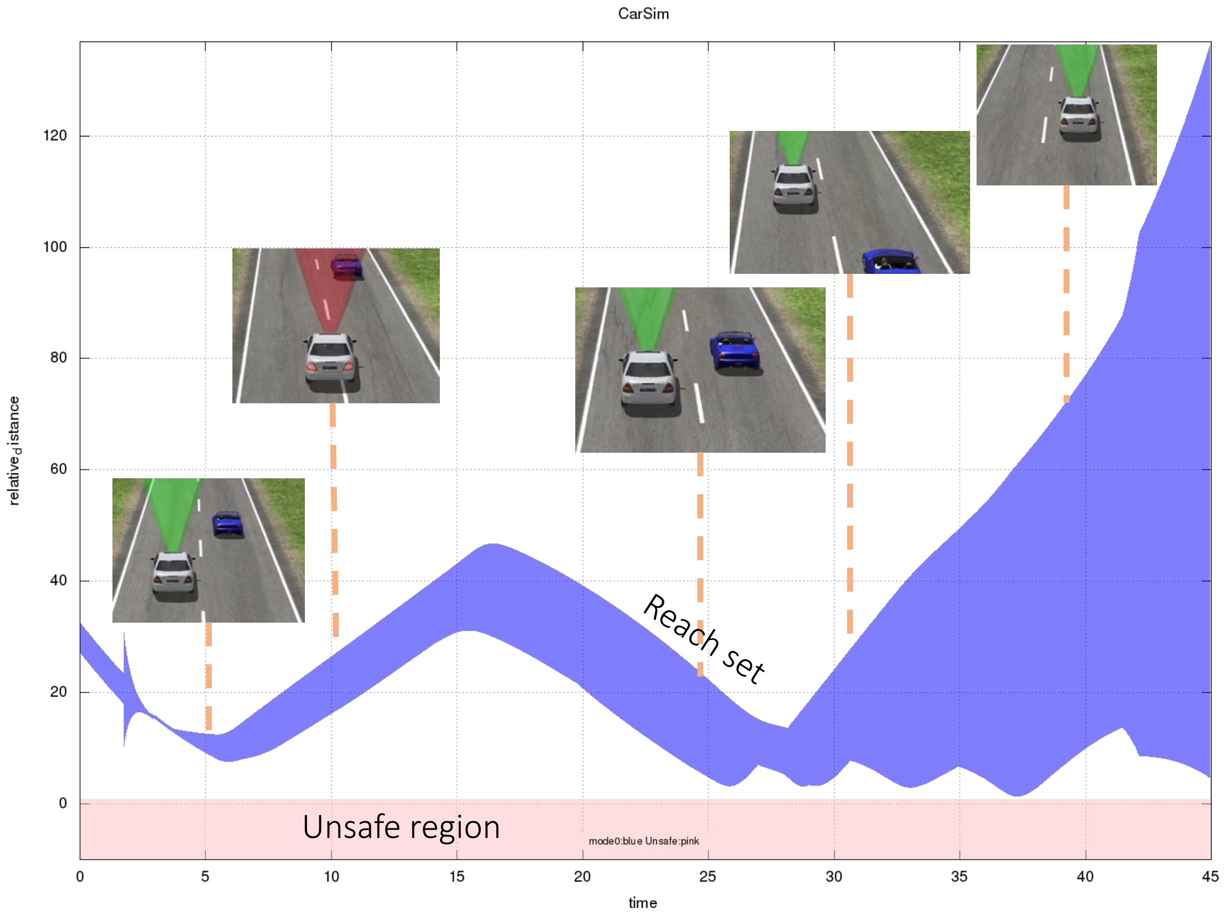


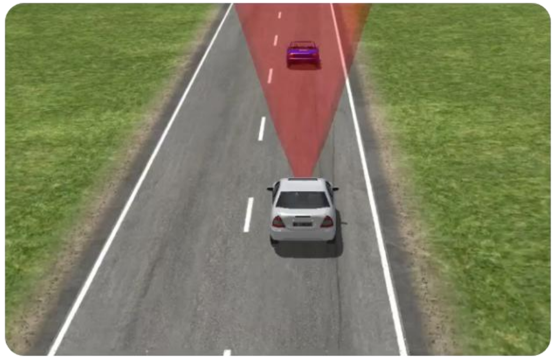






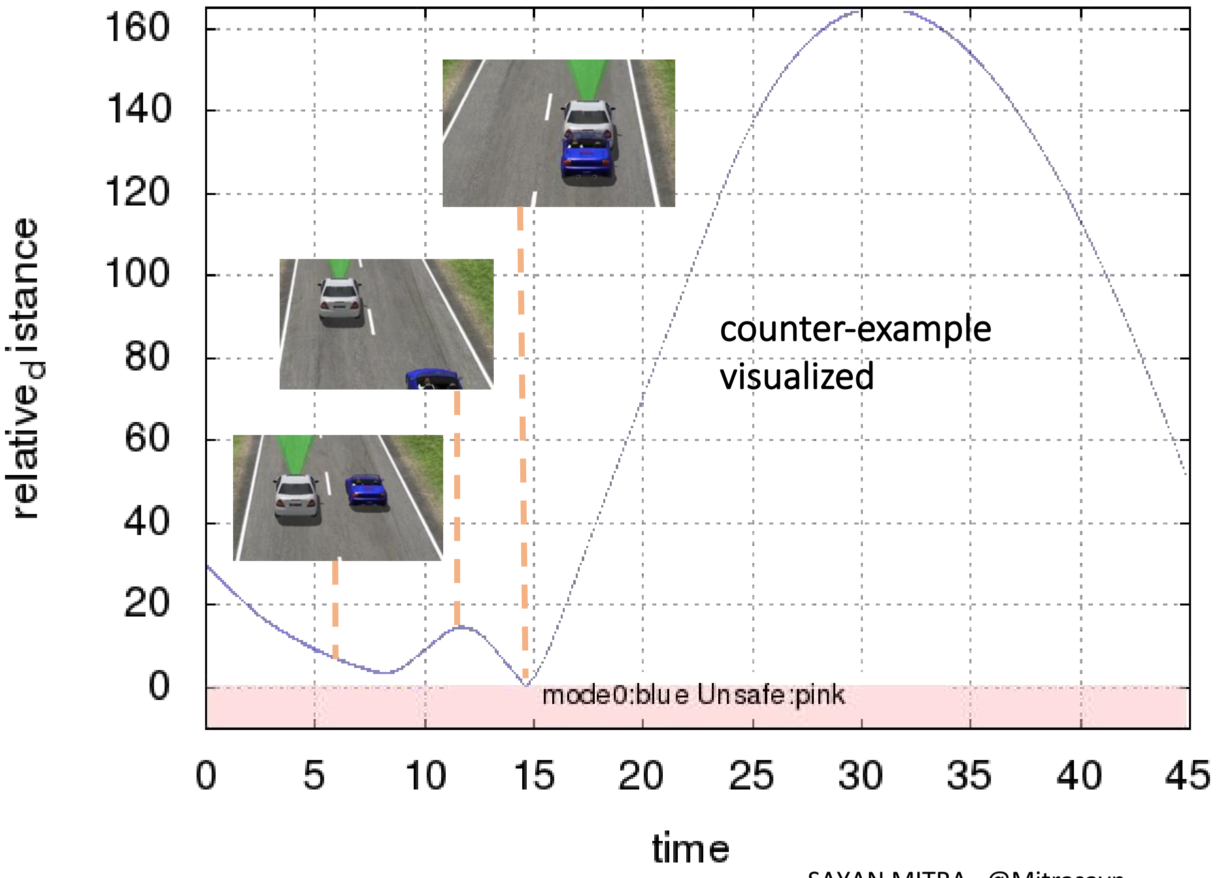
C2E2 generated safety certificate for a given user model

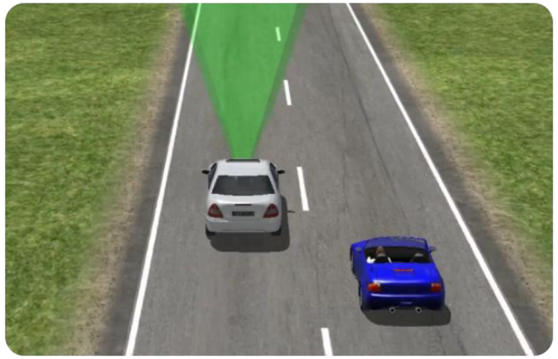




Verify no collision with **uncertainties**: speeds in [70, 85] mph and acceleration range of NPC

For a different user model C2E2 finds a corner case





Verify no collision with **uncertainties** like speeds in [70, 85] mph and **bigger** acceleration range of NPC