Timed and Hybrid Automata

Tom Henzinger
IST Austria

Bertinoro 2010

Two Parallel Stories

Symbolic model checking:

-abstract data type *region algebra*
-termination analysis

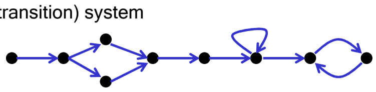
Theory

Timed and hybrid systems:

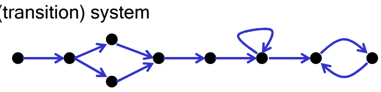
specific region algebra
(e.g. clock regions, polyhedra)

Application

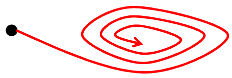
Discrete (transition) system



Discrete (transition) system

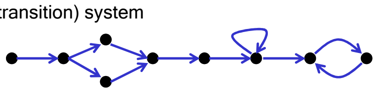


Continuous (dynamical) system

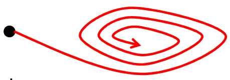


Q = R^n

Discrete (transition) system

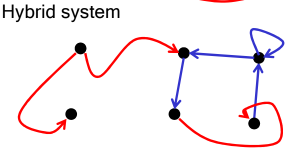


Continuous (dynamical) system



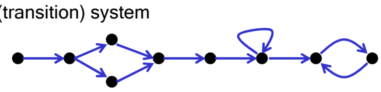
Q = R^n

Hybrid system

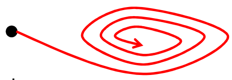


jumps
flows

Discrete (transition) system

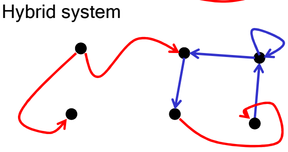


Continuous (dynamical) system



Q = R^n

Hybrid system



jumps
flows

-nondeterministic
-time abstract

A Thermostat

States

$x \in R$	temperature
$h \in \{ on, off \}$	heat

A Thermostat

States

$x \in R$	temperature
$h \in \{ on, off \}$	heat

Flows

f_1	[] h = on	$\rightarrow$ $x' = K \cdot (H-x)$
f_2	[] h = off	$\rightarrow$ $x' = -K \cdot x$

invariants

A Thermostat

States

$x \in R$	temperature
$h \in \{ on, off \}$	heat

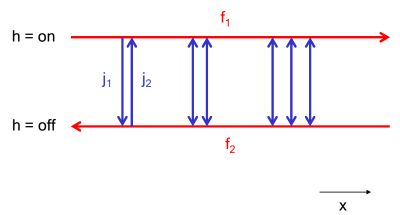
Flows

f_1	[] h = on	$\rightarrow$ $x' = K \cdot (H-x)$
f_2	[] h = off	$\rightarrow$ $x' = -K \cdot x$

Jumps

j_1	[] h = on	$\rightarrow$ h := off
j_2	[] h = off	$\rightarrow$ h := on

guards



A Thermostat

States

$x \in R$	temperature
$h \in \{ on, off \}$	heat
$t \in R$	timer (“clock”)

Flows

f_1	[] h = on	$\rightarrow$ $x' = K \cdot (H-x)$
f_2	[] h = off	$\rightarrow$ $x' = -K \cdot x$

Jumps

j_1	[] h = on	$\rightarrow$ h := off
j_2	[] h = off	$\rightarrow$ h := on

A Thermostat

States

$x \in R$	temperature
$h \in \{ on, off \}$	heat
$t \in R$	timer (“clock”)

Flows

f_1	[] h = on	$\rightarrow$ $x' = K \cdot (H-x)$; $t' = 1$
f_2	[] h = off	$\rightarrow$ $x' = -K \cdot x$; $t' = 1$

Jumps

j_1	[] h = on $\wedge$ $t \geq L$	$\rightarrow$ h := off; $t := 0$
j_2	[] h = off $\wedge$ $t \geq L$	$\rightarrow$ h := on; $t := 0$

A Thermostat

States

$x \in R$	temperature
$h \in \{ on, off \}$	heat
$t \in R$	timer ("clock")

Flows

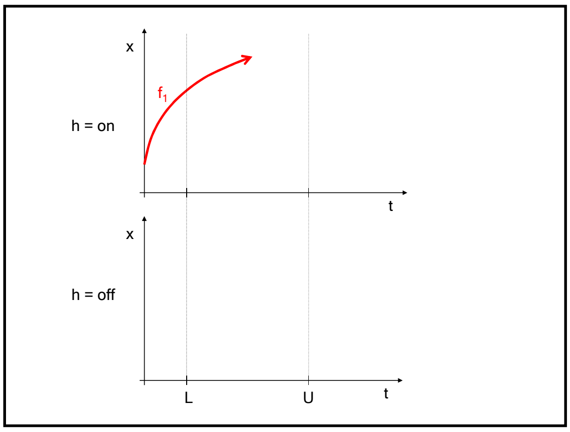
f_1 [] $h = on \wedge t \leq U \rightarrow x' = K \cdot (H - x);\ t' = 1$

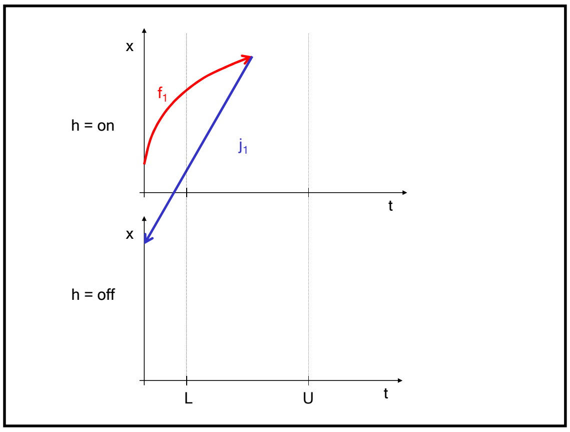
f_2 [] $h = off \wedge t \leq U \rightarrow x' = -K \cdot x;\ t' = 1$

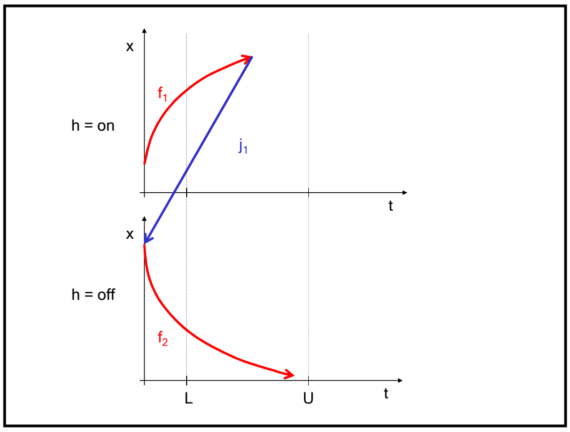
Jumps

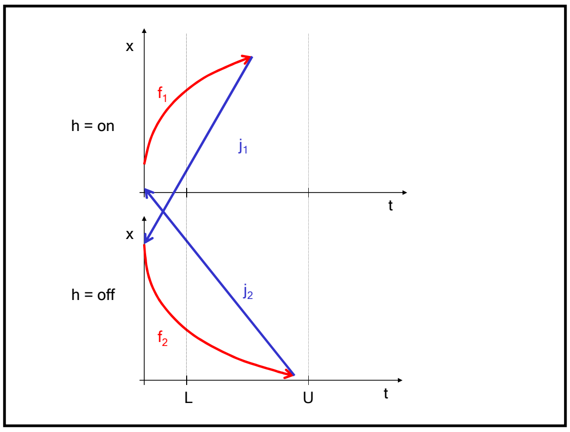
j_1 [] $h = on \wedge t \geq L \rightarrow h := off;\ t := 0$

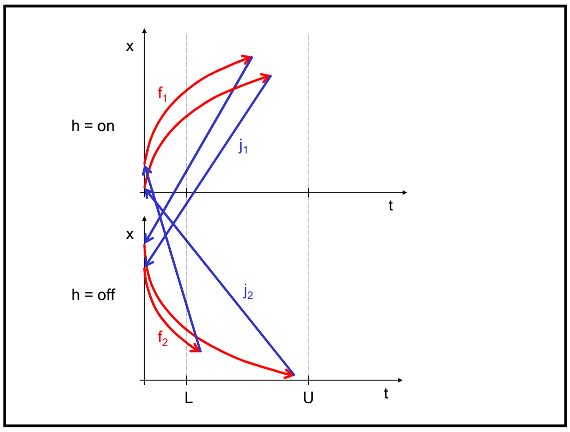
j_2 [] $h = off \wedge t \geq L \rightarrow h := on;\ t := 0$





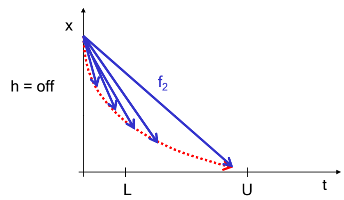






From a Hybrid System to a Symbolic Transition System

1. Discretize: from continuous to discrete
2. Lift: from states to state sets ("regions")
3. Observe: from infinite to finitary



Step 1: Discretize

Transition System

Q	set of states
Σ	set of actions
post: $Q \times \Sigma \to 2^Q$	successor function

Transition System

Q	set of states
Σ	set of actions
post: $Q \times \Sigma \to 2^Q$	successor function

Thermostat

$Q = R^2 \times \{ \text{on}, \text{off} \}$

$\Sigma = \{ f_1, f_2, j_1, j_2 \}$

$$\text{post}(x, t, \text{on}, j_1) = \begin{cases} \{ (x, 0, \text{off}) \} & \text{if } t \geq L \\ \varnothing & \text{if } t < L \end{cases}$$

Transition System

Q	set of states
Σ	set of actions
post: $Q \times \Sigma \to 2^Q$	successor function

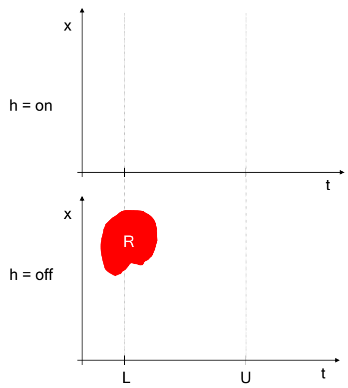
Thermostat

$Q = R^2 \times \{ \text{on}, \text{off} \}$

$\Sigma = \{ f_1, f_2, j_1, j_2 \}$

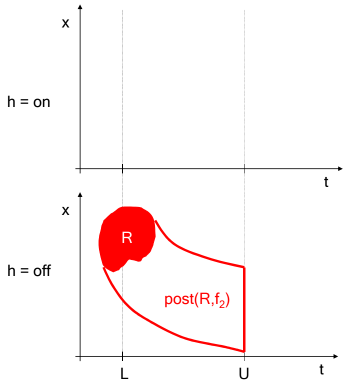
$$\text{post}(x, t, \text{on}, j_1) = \begin{cases} \{ (x, 0, \text{off}) \} & \text{if } t \geq L \\ \varnothing & \text{if } t < L \end{cases}$$

$$\text{post}(x, t, \text{on}, f_1) = \begin{cases} \text{infinite set} & \text{if } t < U \\ \{ (x, t, \text{on}) \} & \text{if } t = U \\ \varnothing & \text{if } t > U \end{cases}$$

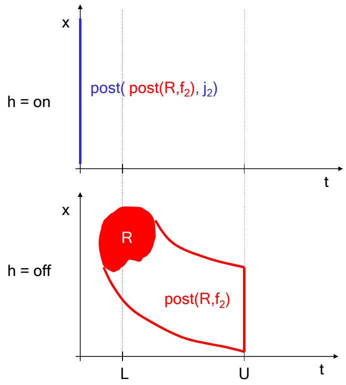


Step 2: Lift

Step 2: Lift



Step 2: Lift

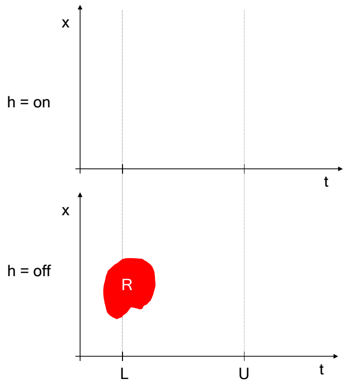


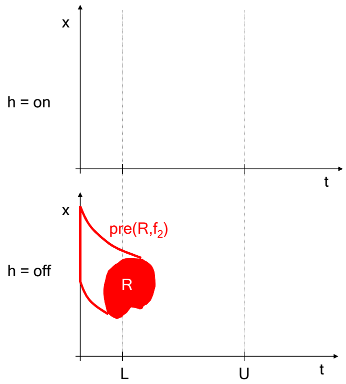
Lifted Transition System

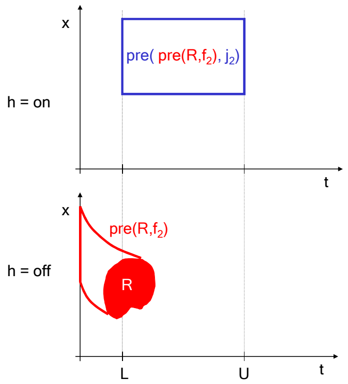
Q
Σ
post: $2^Q \times \Sigma \to 2^Q$ post(R,σ) = $\cup_{q \in R}$ post(q,σ)

Lifted Transition System

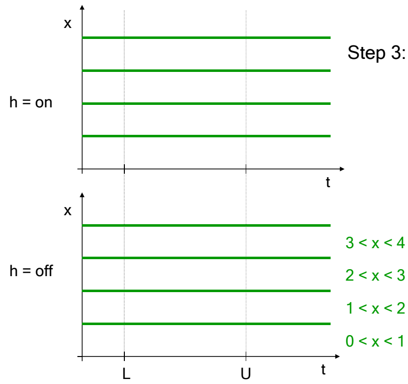
Q
Σ
post: $2^Q \times \Sigma \to 2^Q$ post(R,σ) = $\cup_{q \in R}$ post(q,σ)
pre: $2^Q \times \Sigma \to 2^Q$ pre(R,σ) = $\cup_{q \in R}$ pre(q,σ)





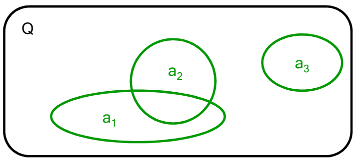


Step 3: Observe



Observed Transition System

Q
Σ
pre, post: $2^Q \times \Sigma \to 2^Q$
A = { a_1, a_2, a_3, ... }   set of observations a_i µ Q



Observed Transition System

Q
Σ
pre, post: $2^Q \times \Sigma \to 2^Q$ A
= { a_1, a_2, a_3, ... }   set of observations a_i µ Q

Thermostat

A = { on, off } [{ x = c, c < x < c+1 | c ∊ Z }

Model Checking: From Finite-state to Hybrid Systems

Graph Algorithms:

-unit operation: access to a vertex (“state”) or edge (“transition”)
-for finite-state systems

Model Checking: From Finite-state to Hybrid Systems

Graph Algorithms:

-unit operation: access to a vertex (“state”) or edge (“transition”)
-for finite-state systems

Symbolic Algorithms:

-unit operation: pre or post on a state set ("region")
-also for infinite-state systems

Model Checking: From Finite-state to Hybrid Systems

Graph Algorithms:

-unit operation: access to a vertex ("state") or edge ("transition")
-for finite-state systems

Symbolic Algorithms:

-unit operation: pre or post on a state set ("region")
-also for infinite-state systems
-two ingredients:

1. region algebra (e.g. BDDs, clock zones, polyhedra)
2. termination analysis

Symbolic Transition System

Q
Σ
pre, post
A
$\mathfrak{R} = \{ R_1, R_2, \dots \}$ set of regions $R_i \subseteq Q$

Symbolic Transition System

Q
Σ
pre, post
A
$\mathfrak{R} = \{ R_1, R_2, \dots \}$ set of regions $R_i \subseteq Q$

Region algebra:

1. $A \subseteq \mathfrak{R}$
2. pre, post: $\mathfrak{R} \times \Sigma \rightarrow \mathfrak{R}$ computable
3. $Å : \mathfrak{R}^2 \rightarrow \mathfrak{R}$
   $\backslash : \mathfrak{R}^2 \rightarrow \mathfrak{R}$ computable
   $\subseteq : \mathfrak{R}^2 \rightarrow \{ t, f \}$

Symbolic Transition System

1. Local computation: Region Operations

Compute pre, post, Å , \ , and ⊆ on regions in $\mathfrak{R}$.

Symbolic Transition System

1. Local computation: Region Operations

Compute pre, post, Å , \ , and ⊆ on regions in $\mathfrak{R}$.

2. Global computation: Symbolic Semi-Algorithms

Starting from the observations in A, compute new regions in $\mathfrak{R}$ by applying the operations pre, post, Å , \ , and ⊆ .

Region Algebras

If -Q is the valuations for a set X:Vals of typed variables,
-the effect of transitions can be expressed using Ops on Vals,
-the first-order theory FO(Vals,Ops) admits quantifier elimination,

Region Algebras

If -Q is the valuations for a set X:Vals of typed variables,
-the effect of transitions can be expressed using Ops on Vals,
-the first-order theory FO(Vals,Ops) admits quantifier elimination,
then the quantifier-free fragment ZO(Vals,Ops) is a region algebra.

This is because each pre and post operation is a quantifier elimination:

pre(R(X)) = ($\exists$X) (Trans(X,X) $\wedge$ R(X))

Region Algebras

If -Q is the valuations for a set X:Vals of typed variables,
-the effect of transitions can be expressed using Ops on Vals,
-the first-order theory FO(Vals,Ops) admits quantifier elimination,
then the quantifier-free fragment ZO(Vals,Ops) is a region algebra.

This is because each pre and post operation is a quantifier elimination:

pre(R(X)) = ($\exists$X) (Trans(X,X) $\wedge$ R(X))

Example: boolean systems
(Vals = B, and $\mathfrak{R}$ = boolean expressions over X)

Example: Polyhedral Hybrid Automata

$Q = B^m \times R^n$

Invariants and guards:
boolean and linear constraints, e.g. $a \wedge (3x_1 + x_2 \leq 7)$

Flows: rectangular differential inclusions, e.g. $x'_1 \in [1,2]$
Jumps: boolean and linear constraints, e.g. $x_2 := 2x_1 + x_2 +1$

Example: Polyhedral Hybrid Automata

$Q = B^m \times R^n$

Invariants and guards:
boolean and linear constraints, e.g. $a \wedge (3x_1 + x_2 \leq 7)$

Flows: rectangular differential inclusions, e.g. $x'_1 \in [1,2]$
Jumps: boolean and linear constraints, e.g. $x_2 := 2x_1 + x_2 +1$

A = set of boolean valuations and integral polyhedra in R^n

Example: Polyhedral Hybrid Automata

$Q = B^m \times R^n$

Invariants and guards:
boolean and linear constraints, e.g. $a \wedge (3x_1 + x_2 \leq 7)$

Flows: rectangular differential inclusions, e.g. $x'_1 \in [1,2]$
Jumps: boolean and linear constraints, e.g. $x_2 := 2x_1 + x_2 +1$

A = set of boolean valuations and integral polyhedra in R^n

$\mathfrak{R}$ = set of boolean valuations and rational polyhedra in R^n

Example: Polyhedral Hybrid Automata

$Q = B^m \times R^n$

Invariants and guards:
boolean and linear constraints, e.g. $a \wedge (3x_1 + x_2 \leq 7)$

Flows: rectangular differential inclusions, e.g. $x'_1 \in [1,2]$
Jumps: boolean and linear constraints, e.g. $x_2 := 2x_1 + x_2 +1$

A = set of boolean valuations and integral polyhedra in R^n

$\mathfrak{R}$ = set of boolean valuations and rational polyhedra in R^n
= ... ZO(Q,$\leq$,+)

Example: Polyhedral Hybrid Automata

FO(Q,≤,+) admits quantifier elimination,
hence ZO(Q,≤,+) is a region algebra.

Example: Polyhedral Hybrid Automata

FO(Q,≤,+) admits quantifier elimination,
hence ZO(Q,≤,+) is a region algebra.

Jump j: ✡ $x_1 \le x_2 \;\to\; x_2 := 2x_1-1$

Example: Polyhedral Hybrid Automata

FO(Q,≤,+) admits quantifier elimination,
hence ZO(Q,≤,+) is a region algebra.

Jump j: ✡ $x_1 \le x_2 \;\to\; x_2 := 2x_1-1$

pre($1 \le x_1 \le x_2 \le 2$, j)

Example: Polyhedral Hybrid Automata

FO(Q,≤,+) admits quantifier elimination,
hence ZO(Q,≤,+) is a region algebra.

Jump j: ✡ $x_1 \le x_2 \;\to\; x_2 := 2x_1-1$

pre($1 \le x_1 \le x_2 \le 2$, j)
$= (\exists \underline{x}_1, \underline{x}_2)\,(x_1 \le x_2 \wedge \underline{x}_1 = x_1 \wedge \underline{x}_2 = 2x_1-1 \wedge 1 \le \underline{x}_1 \le \underline{x}_2 \le 2)$

Example: Polyhedral Hybrid Automata

FO(Q,≤,+) admits quantifier elimination,
hence ZO(Q,≤,+) is a region algebra.

Jump j: ✡ $x_1 \le x_2 \;\to\; x_2 := 2x_1-1$

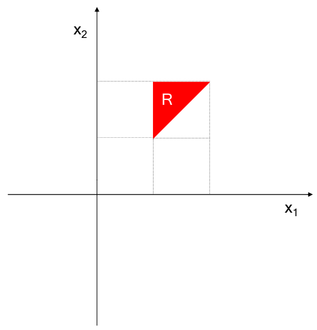
pre($1 \le x_1 \le x_2 \le 2$, j)
$= (\exists \underline{x}_1, \underline{x}_2)\,(x_1 \le x_2 \wedge \underline{x}_1 = x_1 \wedge \underline{x}_2 = 2x_1-1 \wedge 1 \le \underline{x}_1 \le \underline{x}_2 \le 2)$
$= x_1 \le x_2 \wedge 1 \le x_1 \le 2x_1-1 \le 2$

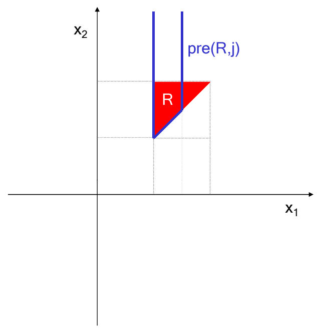
Example: Polyhedral Hybrid Automata

FO(Q,≤,+) admits quantifier elimination,
hence ZO(Q,≤,+) is a region algebra.

Jump j: ✡ $x_1 \le x_2 \;\to\; x_2 := 2x_1-1$

pre($1 \le x_1 \le x_2 \le 2$, j)
$= (\exists \underline{x}_1, \underline{x}_2)\,(x_1 \le x_2 \wedge \underline{x}_1 = x_1 \wedge \underline{x}_2 = 2x_1-1 \wedge 1 \le \underline{x}_1 \le \underline{x}_2 \le 2)$
$= x_1 \le x_2 \wedge 1 \le x_1 \le 2x_1-1 \le 2$
$= x_1 \le x_2 \wedge 1 \le x_1 \le 3/2$





Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \le x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \le x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1 \le x_1 \le x_2 \le 2$, f)

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \le x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1 \le x_1 \le x_2 \le 2$, f)

$= (\exists\, 1 \le k_1 \le 2)\ (\exists\, \delta \ge 0)\ (1 \le x_1 + k_1\delta \le x_2 + \delta \le 2\ \wedge$
$(\forall\, 0 \le \varepsilon \le \delta)\ (x_1 + k_1\varepsilon \le x_2 + \varepsilon))$

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \le x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1 \le x_1 \le x_2 \le 2$, f)

$= (\exists\, 1 \le k_1 \le 2)\ (\exists\, \delta \ge 0)\ (1 \le x_1 + k_1\delta \le x_2 + \delta \le 2\ \wedge$
$(\forall\, 0 \le \varepsilon \le \delta)\ (x_1 + k_1\varepsilon \le x_2 + \varepsilon))$
$= (\exists\, \delta \ge 0)\ (\exists\, \delta \le d_1 \le 2\delta)\ (1 \le x_1 + d_1 \le x_2 + \delta \le 2\ \wedge$

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \leq x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1\leq x_1 \leq x_2 \leq 2$, f)

$= (\exists\, 1\leq k_1\leq 2)\ (\exists\, \delta \geq 0)\ (1\leq x_1+k_1\delta \leq x_2+\delta \leq 2\ \wedge$
$(\forall\, 0 \leq \varepsilon \leq \delta)\ (x_1+k_1\varepsilon \leq x_2+\varepsilon))$

$= (\exists\, \delta \geq 0)\ (\exists\, \delta \leq d_1\leq 2\delta)\ (1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2\ \wedge\ x_1+d_1 \leq x_2+\delta)$

convex guards

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \leq x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1\leq x_1 \leq x_2 \leq 2$, f)

$= (\exists\, 1\leq k_1\leq 2)\ (\exists\, \delta \geq 0)\ (1\leq x_1+k_1\delta \leq x_2+\delta \leq 2\ \wedge$
$(\forall\, 0 \leq \varepsilon \leq \delta)\ (x_1+k_1\varepsilon \leq x_2+\varepsilon))$

$= (\exists\, \delta \geq 0)\ (\exists\, \delta \leq d_1\leq 2\delta)\ (1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2\ \wedge\ x_1+d_1 \leq x_2+\delta)$

$= (\exists\, \delta)\ (9\ d_1)\ (0\cdot \delta \cdot d_1 \cdot 2\delta\ \text{Æ}\ 1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2)$

convex guards

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \leq x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1\leq x_1 \leq x_2 \leq 2$, f)

$= (\exists\, 1\leq k_1\leq 2)\ (\exists\, \delta \geq 0)\ (1\leq x_1+k_1\delta \leq x_2+\delta \leq 2\ \wedge$
$(\forall\, 0 \leq \varepsilon \leq \delta)\ (x_1+k_1\varepsilon \leq x_2+\varepsilon))$

$= (\exists\, \delta \geq 0)\ (\exists\, \delta \leq d_1\leq 2\delta)\ (1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2\ \wedge\ x_1+d_1 \leq x_2+\delta)$

$= (\exists\, \delta)\ (9\ d_1)\ (0\cdot \delta \cdot d_1 \cdot 2\delta\ \text{Æ}\ 1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2)$

$= (9\ \delta)\ (0\cdot \delta\ \text{Æ}\ 1\cdot x_2+\delta \cdot 2\ \text{Æ}\ x_1 \cdot x_2\ \text{Æ}$
$\{\delta,\ 1\text{-}x_1\} \cdot \{2\delta,\ x_2\text{-}x_1\text{+}\delta\})$

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \leq x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1\leq x_1 \leq x_2 \leq 2$, f)

$= (\exists\, 1\leq k_1\leq 2)\ (\exists\, \delta \geq 0)\ (1\leq x_1+k_1\delta \leq x_2+\delta \leq 2\ \wedge$
$(\forall\, 0 \leq \varepsilon \leq \delta)\ (x_1+k_1\varepsilon \leq x_2+\varepsilon))$

$= (\exists\, \delta \geq 0)\ (\exists\, \delta \leq d_1\leq 2\delta)\ (1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2\ \wedge\ x_1+d_1 \leq x_2+\delta)$

$= (\exists\, \delta)\ (9\ d_1)\ (0\cdot \delta \cdot d_1 \cdot 2\delta\ \text{Æ}\ 1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2)$

$= (9\ \delta)\ (0\cdot \delta\ \text{Æ}\ 1\cdot x_2+\delta \cdot 2\ \text{Æ}\ x_1 \cdot x_2\ \text{Æ}$
$\{\delta,\ 1\text{-}x_1\} \cdot \{2\delta,\ x_2\text{-}x_1\text{+}\delta\})$

$= (9\ \delta)\ (0\cdot \delta\ \text{Æ}\ 1\cdot x_2+\delta \cdot 2\ \text{Æ}\ x_1 \cdot x_2\ \text{Æ}\ 1 \cdot x_1+2\delta)$

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \leq x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1\leq x_1 \leq x_2 \leq 2$, f)

$= (\exists\, 1\leq k_1\leq 2)\ (\exists\, \delta \geq 0)\ (1\leq x_1+k_1\delta \leq x_2+\delta \leq 2\ \wedge$
$(\forall\, 0 \leq \varepsilon \leq \delta)\ (x_1+k_1\varepsilon \leq x_2+\varepsilon))$

$= (\exists\, \delta \geq 0)\ (\exists\, \delta \leq d_1\leq 2\delta)\ (1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2\ \wedge\ x_1+d_1 \leq x_2+\delta)$

$= (\exists\, \delta)\ (9\ d_1)\ (0\cdot \delta \cdot d_1 \cdot 2\delta\ \text{Æ}\ 1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2)$

$= (9\ \delta)\ (0\cdot \delta\ \text{Æ}\ 1\cdot x_2+\delta \cdot 2\ \text{Æ}\ x_1 \cdot x_2\ \text{Æ}$
$\{\delta,\ 1\text{-}x_1\} \cdot \{2\delta,\ x_2\text{-}x_1\text{+}\delta\})$

$= (9\ \delta)\ (0\cdot \delta\ \text{Æ}\ 1\cdot x_2+\delta \cdot 2\ \text{Æ}\ x_1 \cdot x_2\ \text{Æ}\ 1 \cdot x_1+2\delta)\ =\ x_1 \cdot$
$x_2\ \text{Æ}\ \{0,\ 1\text{-}x_2,\ (1\text{-}x_1)/2\} \cdot \{2\text{-}x_2\}$

Example: Polyhedral Hybrid Automata

Flow f: ✡ $x_1 \leq x_2 \;\rightarrow\; x'_1 \in [1,2];\; x'_2 = 1$

pre($1\leq x_1 \leq x_2 \leq 2$, f)

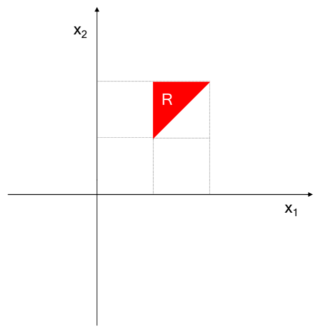
$= (\exists\, 1\leq k_1\leq 2)\ (\exists\, \delta \geq 0)\ (1\leq x_1+k_1\delta \leq x_2+\delta \leq 2\ \wedge$
$(\forall\, 0 \leq \varepsilon \leq \delta)\ (x_1+k_1\varepsilon \leq x_2+\varepsilon))$

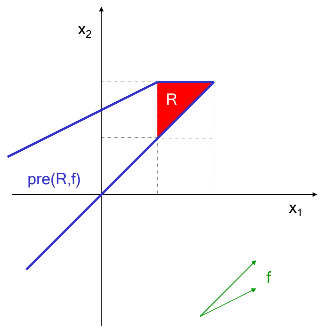
$= (\exists\, \delta \geq 0)\ (\exists\, \delta \leq d_1\leq 2\delta)\ (1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2\ \wedge\ x_1+d_1 \leq x_2+\delta)$

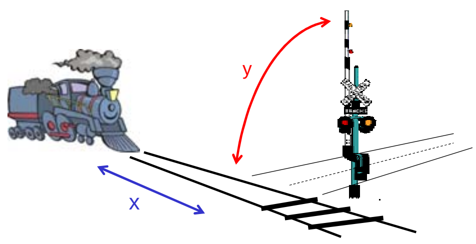
$= (\exists\, \delta)\ (9\ d_1)\ (0\cdot \delta \cdot d_1 \cdot 2\delta\ \text{Æ}\ 1\leq x_1+d_1 \leq x_2+\delta \leq 2\ \wedge$
$x_1 \leq x_2)$

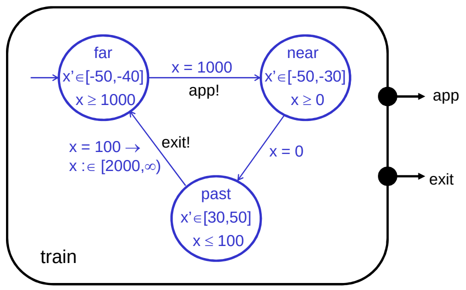
$= (9\ \delta)\ (0\cdot \delta\ \text{Æ}\ 1\cdot x_2+\delta \cdot 2\ \text{Æ}\ x_1 \cdot x_2\ \text{Æ}$
$\{\delta,\ 1\text{-}x_1\} \cdot \{2\delta,\ x_2\text{-}x_1\text{+}\delta\})$

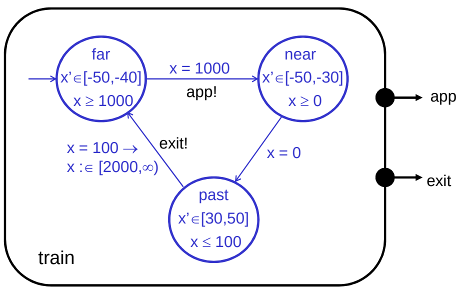
$= (9\ \delta)\ (0\cdot \delta\ \text{Æ}\ 1\cdot x_2+\delta \cdot 2\ \text{Æ}\ x_1 \cdot x_2\ \text{Æ}\ 1 \cdot x_1+2\delta)\ =\ x_1 \cdot$
$x_2\ \text{Æ}\ \{0,\ 1\text{-}x_2,\ (1\text{-}x_1)/2\} \cdot \{2\text{-}x_2\}\quad = x_1 \leq x_2 \leq 2 \wedge$
$2x_2 \leq x_1+3$



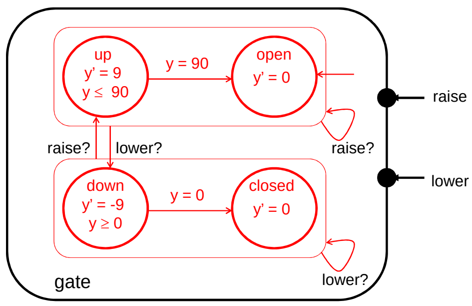


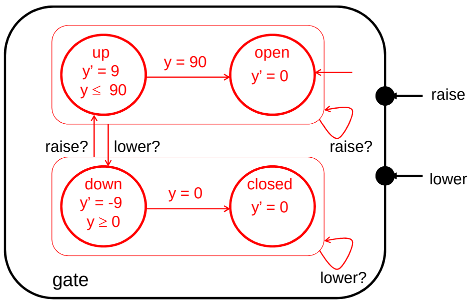




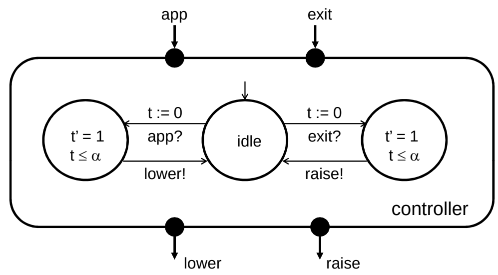


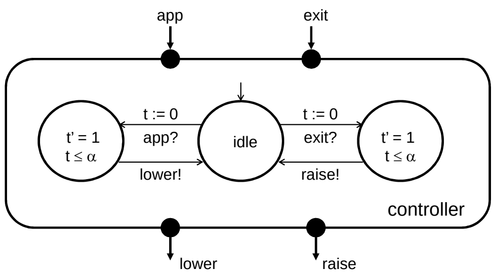
x: initialized rectangular variable





y: uninitialized singular variable





t: clock
α: parameter

Properties

Safety: $\forall\Box$ ($x \leq 10 \Rightarrow$ loc[gate] = closed)

"on all trajectories, always"

For which values of α is this true?

Properties

Safety: $\forall\Box$ ($x \leq 10 \Rightarrow$ loc[gate] = closed)

Liveness: $\forall\Box\ \forall\Diamond$ (loc[gate] = open)

"on all trajectories, eventually"

Properties

Safety: $\forall\Box$ ($x \leq 10 \Rightarrow$ loc[gate] = closed)

Liveness: $\forall\Box\ \forall\Diamond$ (loc[gate] = open)

Real time: $\forall\Box$ $z := 0.$ ($z' = 1 \Rightarrow$
$\forall\Diamond$ (loc[gate] = open $\wedge$ $z \leq 60$))

clock variable

Properties

Safety: $\forall\Box$ ($x \leq 10 \Rightarrow$ loc[gate] = closed)

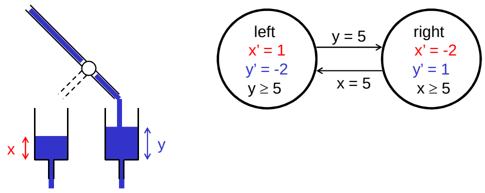
Liveness: $\forall\Box\ \forall\Diamond$ (loc[gate] = open)

Real time: $\forall\Box$ z := 0. (z' = 1 $\Rightarrow$ $\forall\Diamond$ (loc[gate] = open $\wedge$ $z \leq 60$))

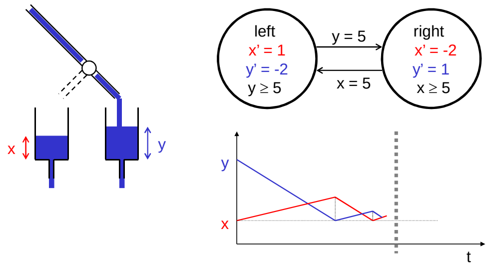
Nonzeno: $\forall\Box$ z := 0. (z' = 1 $\Rightarrow$ $\exists\Diamond$ (z = 1))

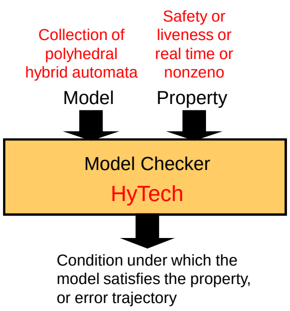
"on some trajectory, eventually"

A Zeno System

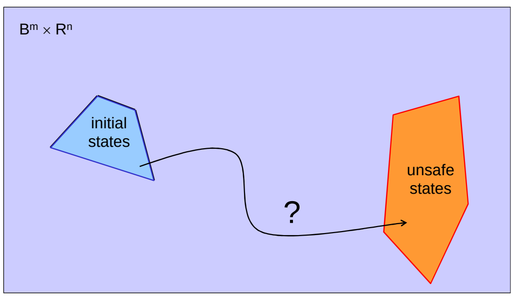


A Zeno System

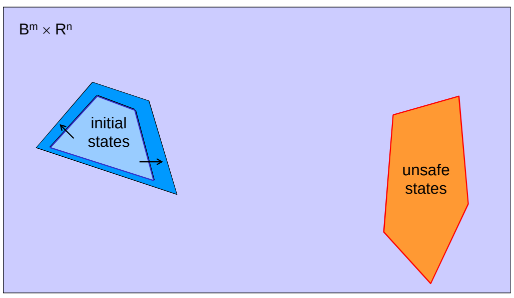




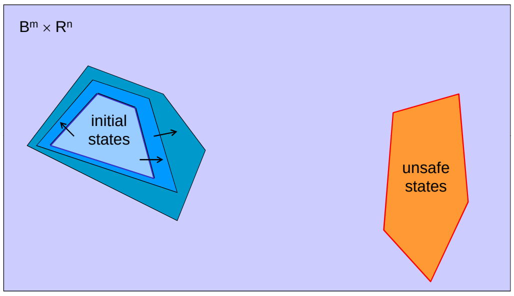
Model Checking for Safety



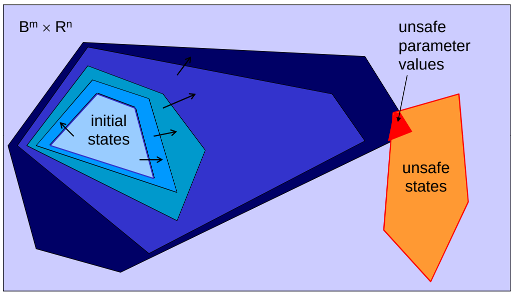
Model Checking for Safety



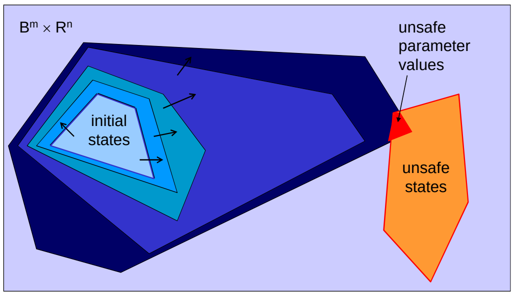
Model Checking for Safety



Model Checking for Safety



Model Checking for Safety



This is guaranteed to terminate if all variables are initialized (e.g. clocks).

The Result

Applications of HyTech and Derivations: polyhedral overapproximation of dynamics

-automotive engine control [Wong-Toi et al.]
-chemical plant control [Preussig et al.]
-flight control [Honeywell; Rockwell-Collins]
-air traffic control [Tomlin et al.]
-robot control [Corbett et al.]

Applications of HyTech and Derivations: polyhedral overapproximation of dynamics

-automotive engine control [Wong-Toi et al.]
-chemical plant control [Preussig et al.]
-flight control [Honeywell; Rockwell-Collins]
-air traffic control [Tomlin et al.]
-robot control [Corbett et al.]

Successor Tools:

1. More expressive region algebras, e.g. FO(R,≤,+,·) still permits quantifier elimination [Pappas et al.]

2. Different approximations, e.g. ellipsoid regions instead of polyhedral regions [Varaiya et al.]

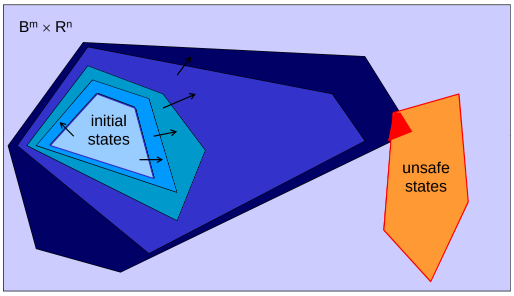
Symbolic Transition System

Q
Σ
pre, post
A
$\mathfrak{R}$

Region algebra:

1. A $\subseteq \mathfrak{R}$
2. pre, post: $\mathfrak{R} \times \Sigma \to \mathfrak{R}$ computable
3. Å : $\mathfrak{R}^2 \to \mathfrak{R}$
   \ : $\mathfrak{R}^2 \to \mathfrak{R}$ computable
   $\subseteq$: $\mathfrak{R}^2 \to$ { t, f }

Model Checking for Safety



Symbolic Semi-Algorithms

Starting from the observations in A, compute new regions in $\mathfrak{R}$ by applying the operations pre, post, Å , \ , and $\subseteq$.

Termination?

Five Verification Questions

V1: Reachability — $\exists\Diamond$ b
Is an invariant always true? — $\exists\Diamond$ unsafe

Five Verification Questions

V1: Reachability — $\exists\Diamond$ b
Is an invariant always true? — $\exists\Diamond$ unsafe

V2: Counting Reachability — $\mu X. (b \vee pre^2(X))$
Conjunction-free μ-calculus
Is an invariant true every other step?

Five Verification Questions

V1: Reachability — $\exists\Diamond$ b
Is an invariant always true? — $\exists\Diamond$ unsafe

V2: Counting Reachability — $\mu X. (b \vee pre^2(X))$
Conjunction-free μ-calculus
Is an invariant true every other step?

V3: Repeated Reachability — $\exists\Box\Diamond$ b
Linear temporal logic (LTL)
Liveness — $\exists(\Box\Diamond \text{ fair} \wedge \neg\Diamond \text{ goal})$

Five Verification Questions

V1: Reachability — $\exists\Diamond$ b
Is an invariant always true? — $\exists\Diamond$ unsafe

V2: Counting Reachability — $\mu X.\ (b \vee pre^2(X))$
Conjunction-free μ-calculus
Is an invariant true every other step?

V3: Repeated Reachability — $\exists\Box\Diamond$ b
Linear temporal logic (LTL)
Liveness — $\exists(\Box\Diamond \text{ fair} \wedge \neg\Diamond \text{ goal})$

V4: Nested Reachability — $\exists\Diamond\ (b \wedge \exists\Diamond\ b_1 \wedge \exists\Diamond\ b_2)$
Half branching temporal logic ($\exists$CTL, $\forall$CTL)

Five Verification Questions

V1: Reachability — $\exists\Diamond$ b
Is an invariant always true? — $\exists\Diamond$ unsafe

V2: Counting Reachability — $\mu X.\ (b \vee pre^2(X))$
Conjunction-free μ-calculus
Is an invariant true every other step?

V3: Repeated Reachability — $\exists\Box\Diamond$ b
Linear temporal logic (LTL)
Liveness — $\exists(\Box\Diamond \text{ fair} \wedge \neg\Diamond \text{ goal})$

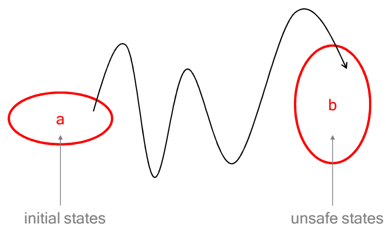
V4: Nested Reachability — $\exists\Diamond\ (b \wedge \exists\Diamond\ b_1 \wedge \exists\Diamond\ b_2)$
Half branching temporal logic ($\exists$CTL, $\forall$CTL)

V5: Negated Reachability — $\forall\Box\ (b \rightarrow \exists\Diamond\ c)$
Full branching temporal logic (CTL)
Nonzeroness — $\forall\Box\ (\text{tick} \rightarrow pre(\exists\Diamond\ \text{tick}))$

V1: Symbolic Reachability

$a \wedge \exists\Diamond\ b$

Given $a,b \in A$, is there a trajectory from a to b?



V1: Symbolic Reachability

$a \wedge \exists\Diamond\ b$

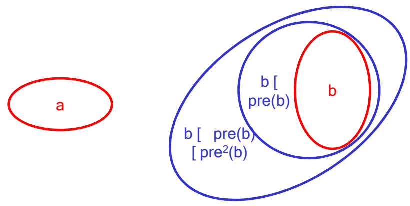
Given $a,b \in A$, is there a trajectory from a to b?

$pre(b) = \cup_{\sigma\in\Sigma}\ pre(b,\sigma)$

V1: Symbolic Reachability

$a \wedge \exists\Diamond\ b$

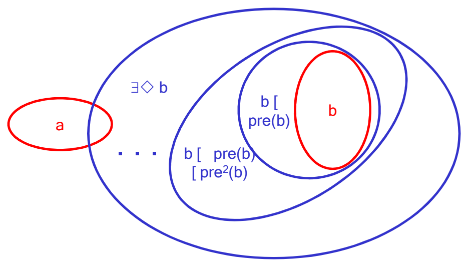
Given $a,b \in A$, is there a trajectory from a to b?



V1: Symbolic Reachability

$a \wedge \exists\Diamond\ b$

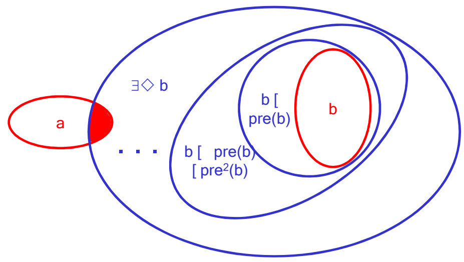
Given $a,b \in A$, is there a trajectory from a to b?



V1: Symbolic Reachability

$a \wedge \exists\Diamond\, b$

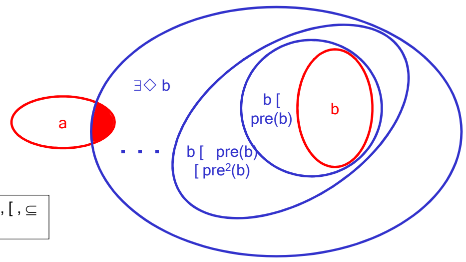
Given a,b∈A, is there a trajectory from a to b?



V1: Symbolic Reachability

$a \wedge \exists\Diamond\, b$

Given a,b∈A, is there a trajectory from a to b?



1. pre, [, ⊆
2. Â a

V2: Symbolic Counting Reachability

$a \wedge \mu X.\ (b \vee pre^2(X))$

Given a,b∈A, is there a trajectory of even length from a to b?

V2: Symbolic Counting Reachability

$a \wedge \mu X.\ (b \vee pre^2(X))$

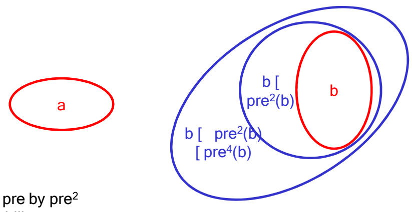
Given a,b∈A, is there a trajectory of even length from a to b?

Replace pre by pre^2
in reachability.

V2: Symbolic Counting Reachability

$a \wedge \mu X.\ (b \vee pre^2(X))$

Given a,b∈A, is there a trajectory of even length from a to b?

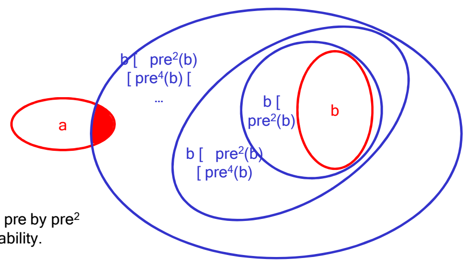


Replace pre by pre^2
in reachability.

V2: Symbolic Counting Reachability

$a \wedge \mu X.\ (b \vee pre^2(X))$

Given a,b∈A, is there a trajectory of even length from a to b?

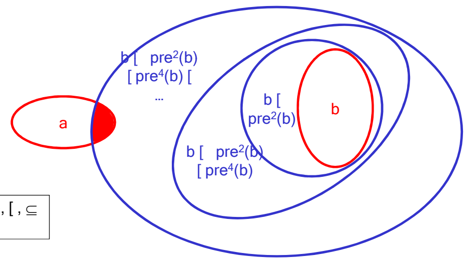


Replace pre by pre^2
in reachability.

V2: Symbolic Counting Reachability

$a \wedge \mu X.\ (b \vee pre^2(X))$

Given a,b∈A, is there a trajectory of even length from a to b?

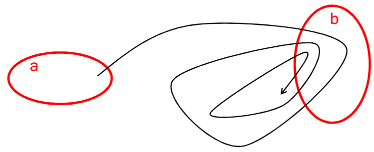


1. pre, [, ⊆
2. Å a

V3: Symbolic Repeated Reachability

$a \wedge \exists\Box\Diamond\ b$

Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?



V3: Symbolic Repeated Reachability

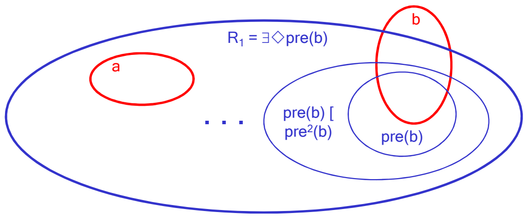
$a \wedge \exists\Box\Diamond\ b$

Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?

V3: Symbolic Repeated Reachability

$a \wedge \exists\Box\Diamond\ b$

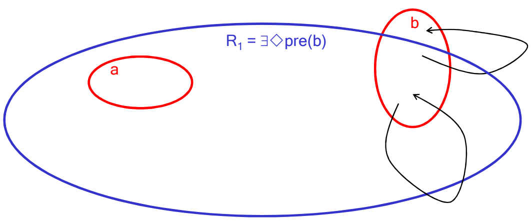
Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?



V3: Symbolic Repeated Reachability

$a \wedge \exists\Box\Diamond\ b$

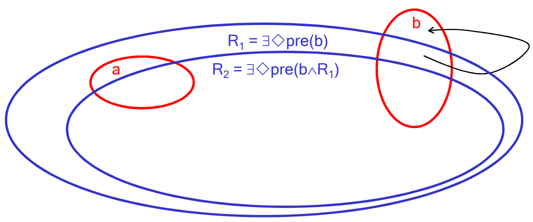
Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?



V3: Symbolic Repeated Reachability

$a \wedge \exists\Box\Diamond\ b$

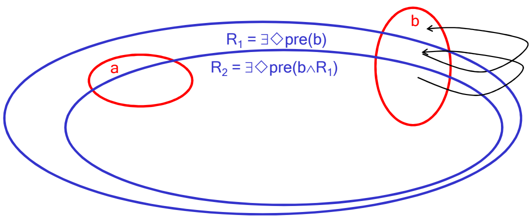
Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?



V3: Symbolic Repeated Reachability

a ∧ ∃□◇ b

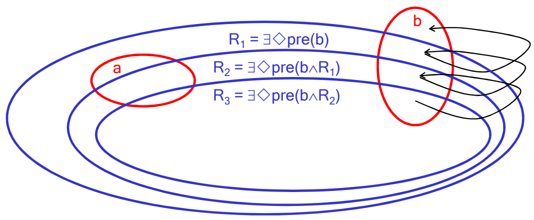
Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?



V3: Symbolic Repeated Reachability

a ∧ ∃□◇ b

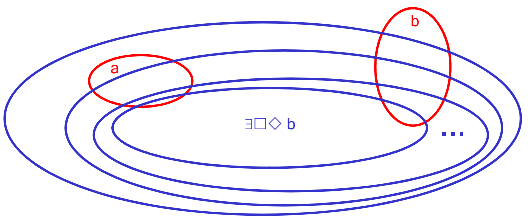
Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?



V3: Symbolic Repeated Reachability

a ∧ ∃□◇ b

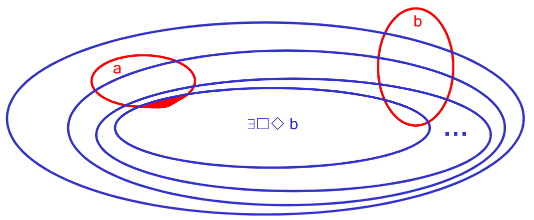
Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?



V3: Symbolic Repeated Reachability

a ∧ ∃□◇ b

Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?

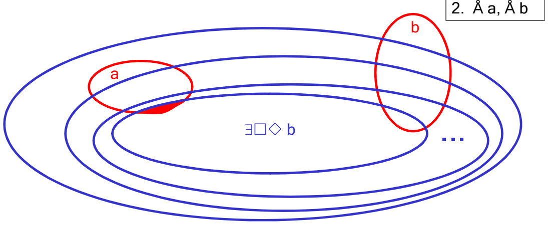


V3: Symbolic Repeated Reachability

a ∧ ∃□◇ b

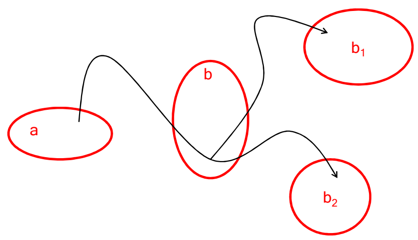
Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?

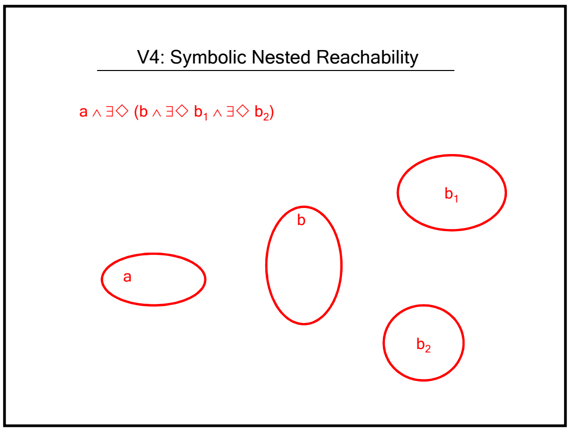
1. pre, [, ⊆
2. Å a, Å b

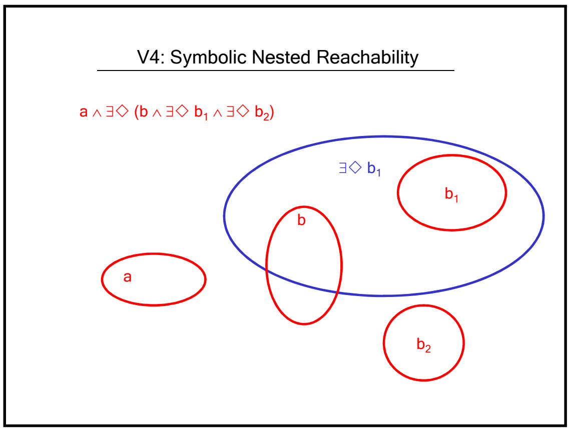


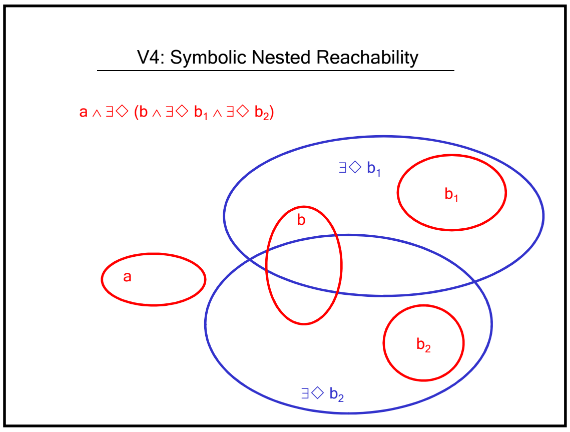
V4: Symbolic Nested Reachability

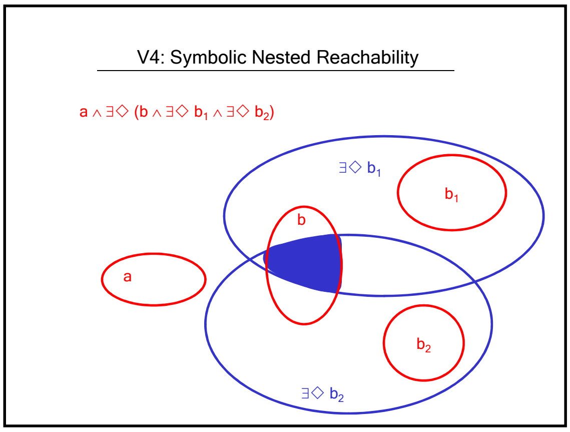
a ∧ ∃◇ (b ∧ ∃◇ b_1 ∧ ∃◇ b_2)

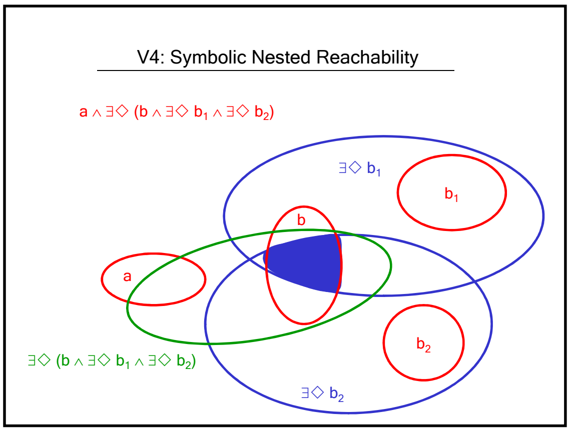


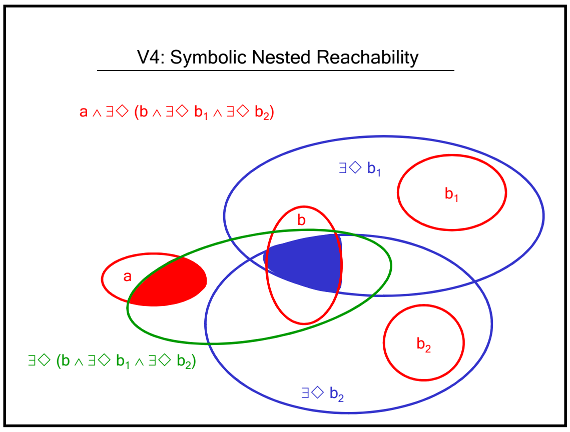

V4: Symbolic Nested Reachability
$a \wedge \exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$
b_1
b
a
b_2


V4: Symbolic Nested Reachability
$a \wedge \exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$
$\exists\Diamond b_1$
b_1
b
a
b_2


V4: Symbolic Nested Reachability
$a \wedge \exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$
$\exists\Diamond b_1$
b_1
b
a
b_2
$\exists\Diamond b_2$

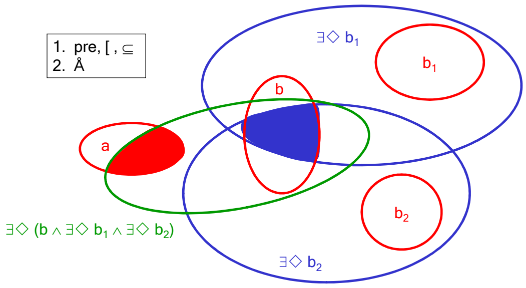

V4: Symbolic Nested Reachability
$a \wedge \exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$
$\exists\Diamond b_1$
b_1
b
a
b_2
$\exists\Diamond b_2$


V4: Symbolic Nested Reachability
$a \wedge \exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$
$\exists\Diamond b_1$
b_1
b
a
$\exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$
b_2
$\exists\Diamond b_2$


V4: Symbolic Nested Reachability
$a \wedge \exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$
$\exists\Diamond b_1$
b_1
b
a
$\exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$
b_2
$\exists\Diamond b_2$

V4: Symbolic Nested Reachability

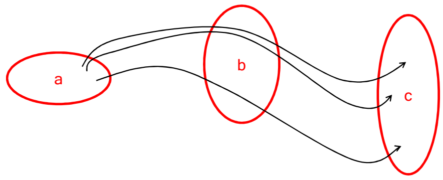
$a \wedge \exists\Diamond (b \wedge \exists\Diamond b_1 \wedge \exists\Diamond b_2)$



V5: Symbolic Negated Reachability

$a \wedge \forall\Box (b \rightarrow \exists\Diamond c)$

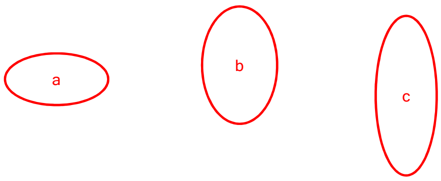
Given a,b,c∈A, can every trajectory from a to b be extended to c?



V5: Symbolic Negated Reachability

$a \wedge \forall\Box (b \rightarrow \exists\Diamond c)$

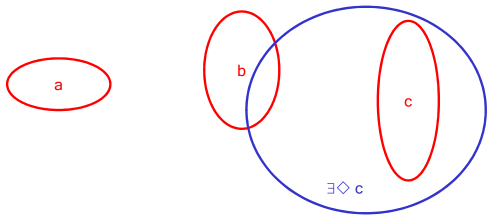
Given a,b,c∈A, can every trajectory from a to b be extended to c?



V5: Symbolic Negated Reachability

$a \wedge \forall\Box (b \rightarrow \exists\Diamond c)$

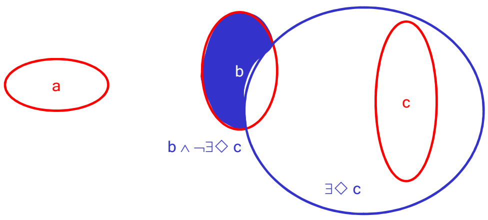
Given a,b,c∈A, can every trajectory from a to b be extended to c?



V5: Symbolic Negated Reachability

$a \wedge \forall\Box (b \rightarrow \exists\Diamond c)$

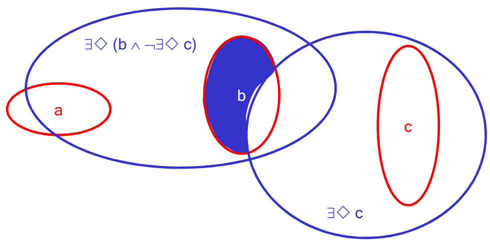
Given a,b,c∈A, can every trajectory from a to b be extended to c?



V5: Symbolic Negated Reachability

$a \wedge \forall\Box (b \rightarrow \exists\Diamond c)$

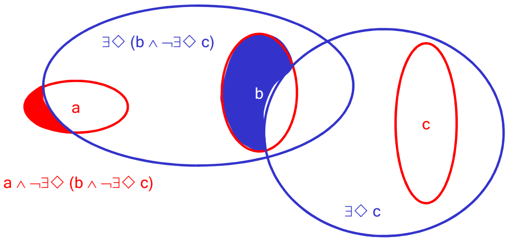
Given a,b,c∈A, can every trajectory from a to b be extended to c?



V5: Symbolic Negated Reachability

a ∧ ∀□ (b → ∃◇ c)

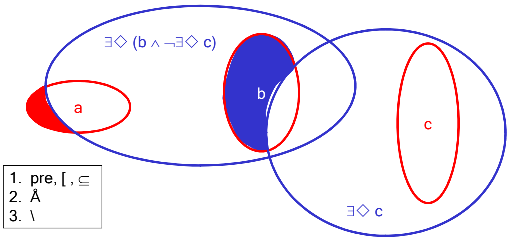
Given a,b,c∈A, can every trajectory from a to b be extended to c?



V5: Symbolic Negated Reachability

a ∧ ∀□ (b → ∃◇ c)

Given a,b,c∈A, can every trajectory from a to b be extended to c?



Five Specification Logics

L1: Reachability Logic

φ := a | φ∨φ | ∃◇φ

Five Specification Logics

L1: Reachability Logic

φ := a | φ∨φ | ∃◇φ

L2: Conjunction-free μ-Calculus

φ := a | X | φ∨φ | pre(φ) | μX.φ
Symbolic model checking: pre, [, ⊆

Five Specification Logics

L1: Reachability Logic

φ := a | φ∨φ | ∃◇φ

L2: Conjunction-free μ-Calculus

φ := a | X | φ∨φ | pre(φ) | μX.φ
Symbolic model checking: pre, [, ⊆

L3: Guarded μ-Calculus (subsumes LTL, omega automata)

φ := a | X | φ∨φ | a∧φ | pre(φ) | μX.φ | νX.φ
Symbolic model checking: pre, [, ⊆ , Å a

Five Specification Logics

L1: Reachability Logic

φ := a | φ∨φ | ∃◇φ

L2: Conjunction-free μ-Calculus

φ := a | X | φ∨φ | pre(φ) | μX.φ
Symbolic model checking: pre, [, ⊆

L3: Guarded μ-Calculus (subsumes LTL, omega automata)

φ := a | X | φ∨φ | a∧φ | pre(φ) | μX.φ | νX.φ
Symbolic model checking: pre, [, ⊆ , Å a

L4: Existential μ-Calculus (subsumes ∃CTL)

φ := a | X | φ∨φ | φ∧φ | pre(φ) | μX.φ | νX.φ
Symbolic model checking: pre, [, ⊆ , Å

Five Specification Logics

L1: Reachability Logic

φ := a | φ∨φ | ∃◇φ

L2: Conjunction-free μ-Calculus

φ := a | X | φ∨φ | pre(φ) | μX.φ
Symbolic model checking: pre, [, ⊆

L3: Guarded μ-Calculus (subsumes LTL, omega automata)

φ := a | X | φ∨φ | a∧φ | pre(φ) | μX.φ | νX.φ
Symbolic model checking: pre, [, ⊆ , Å a

L4: Existential μ-Calculus (subsumes ∃CTL)

φ := a | X | φ∨φ | φ∧φ | pre(φ) | μX.φ | νX.φ
Symbolic model checking: pre, [, ⊆ , Å

L5: μ-Calculus (subsumes CTL) pre(φ)=: pre:(φ)

φ := a | X | φ∨φ | φ∧φ | pre(φ) | pre(φ) | μX.φ | νX.φ
Symbolic model checking: pre, [, ⊆ , Å , \

Five Symbolic Semi-Algorithms

A1: Symbolic backward reachability

for each a∈A do
R_0 := a
for i=1,2,3,... do
R_i := R_{i-1} [pre(R_{i-1})
until R_i = R_{i-1}

Five Symbolic Semi-Algorithms

A1: Symbolic backward reachability
A2: Close A under pre

$\Im_0$:= A
for i=1,2,3,... do
$\Im_i$:= $\Im_{i-1}$ [{ pre(R) | R∈$\Im_i$ }

until $\Im_i$ = $\Im_{i-1}$

A = {a_1, a_2}

A1 computes: a_1[pre(a_1),
a_1[pre(a_1) [pre^2(a_1),
a_1[pre(a_1) [pre^2(a_1) [pre^3(a_1), ...
a_2[pre(a_2),
a_2[pre(a_2) [pre^2(a_2), ...

A2 computes: pre(a_1), pre^2(a_1), pre^3(a_1), ...
pre(a_2), pre^2(a_2), pre^3(a_2), ...

Five Symbolic Semi-Algorithms

A1: Symbolic backward reachability
A2: Close A under pre
A3: Close A under pre, Å a

$\Im_0$:= A
for i=1,2,3,... do
$\Im_i$:= $\Im_{i-1}$ [{ pre(R) | R∈$\Im_i$ }
[{ R Å a | R∈$\Im_i$, a∈A }

until $\Im_i$ = $\Im_{i-1}$

A = {a_1, a_2}

A1 computes: a_1[pre(a_1),
a_1[pre(a_1) [pre^2(a_1),
a_1[pre(a_1) [pre^2(a_1) [pre^3(a_1), ...
a_2[pre(a_2),
a_2[pre(a_2) [pre^2(a_2), ...

A2 computes: pre(a_1), pre^2(a_1), pre^3(a_1), ...
pre(a_2), pre^2(a_2), pre^3(a_2), ...

A3 computes: also pre(a_1)Å a_2 etc.

Five Symbolic Semi-Algorithms

A1: Symbolic backward reachability

A2: Close A under pre

A3: Close A under pre, Å a

A4: Close A under pre, Å

$\Im_0$:= A
for i=1,2,3,… do
    $\Im_i$:= $\Im_{i-1}$ [{ pre(R) | R∈$\Im_i$ }
        [{ R_1 Å R_2 | R_1,R_2∈$\Im_i$ }
until $\Im_i$ = $\Im_{i-1}$

Five Symbolic Semi-Algorithms

A1: Symbolic backward reachability

A2: Close A under pre

A3: Close A under pre, Å a

A4: Close A under pre, Å

A5: Close A under pre, Å , \

$\Im_0$:= A
for i=1,2,3,… do
    $\Im_i$:= $\Im_{i-1}$ [{ pre(R) | R∈$\Im_i$ }
        [{ R_1 Å R_2 | R_1,R_2∈$\Im_i$ }
        [{ R_1\ R_2 | R_1,R_2∈$\Im_i$ }
until $\Im_i$ = $\Im_{i-1}$

Five Symbolic Semi-Algorithms

A1: Symbolic backward reachability

A2: Close A under pre

A3: Close A under pre, Å a

A4: Close A under pre, Å

A5: Close A under pre, Å , \

A_k terminates ($1 \le k \le 5$) $\Rightarrow$
symbolic model checking of L_k terminates.

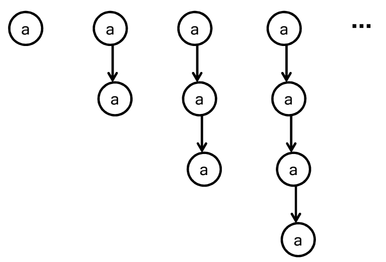
Five State Equivalences

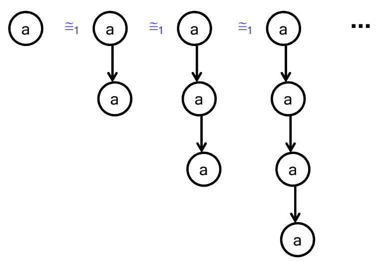
E1: Bounded-Reach Equivalence

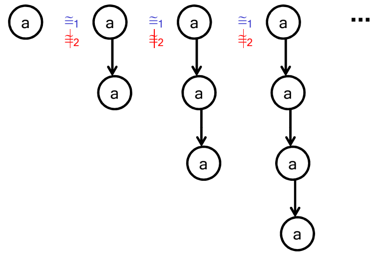
$q_1 \cong_1 q_2$ iff if a∈A can be reached from q_1 in d steps, then a can be reached from q_2 in at most d steps, and vice versa.

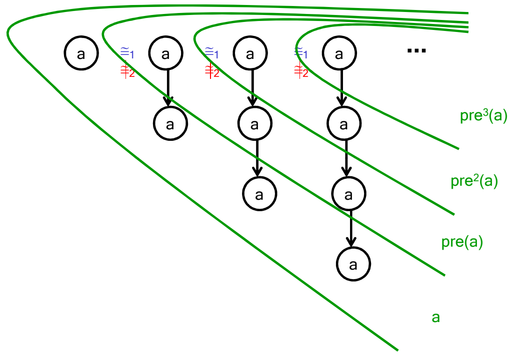
E2: Distance Equivalence

$q_1 \cong_2 q_2$ iff if a∈A can be reached from q_1 in d steps, then a can be reached from q_2 in d steps, and vice versa.









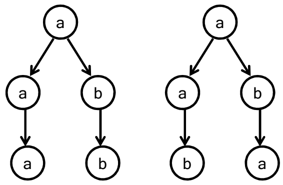
Five State Equivalences

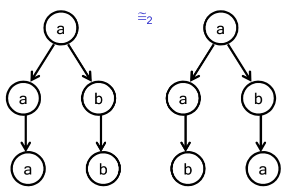
E1: Bounded-Reach Equivalence

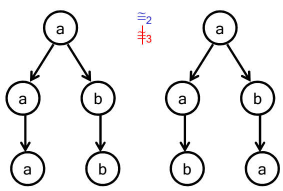
E2: Distance Equivalence

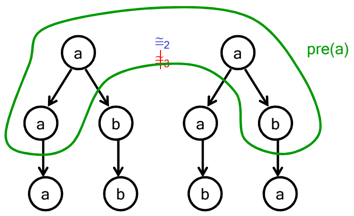
E3: Trace Equivalence

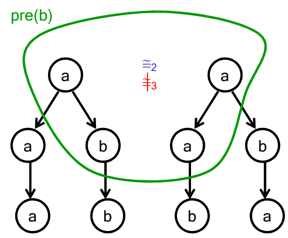
$q_1 \cong_3 q_2$ iff if every finite trace from q_1 is a finite trace from q_2, and vice versa.

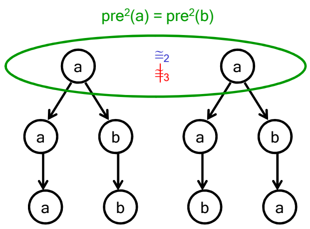


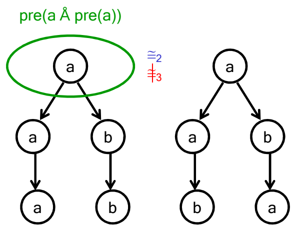












Five State Equivalences

E1: Bounded-Reach Equivalence

E2: Distance Equivalence

E3: Trace Equivalence

E4: Similarity (mutual simulation)

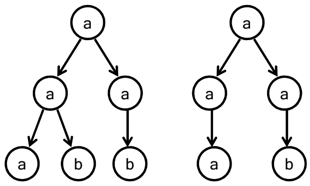
$q_1 \cong_4 q_2$ iff if q_1 simulates q_2, and vice versa.

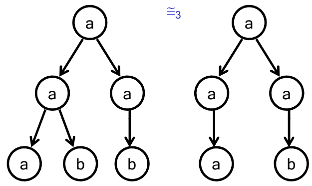
q_1 is simulated by q_2

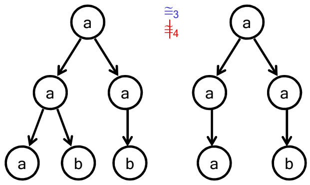
iff

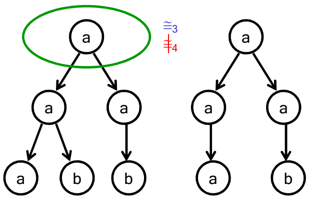
there is a simulation relation S such that

1. $S(q_1,q_2)$
2. if S(p,q) then
   a. (8 a2A) (p2 a iff q2 a)
   b. (8 p') (if p2 pre(p') then (9 q') (q2 pre(q') Æ S(p',q')))









Five State Equivalences

E1: Bounded-Reach Equivalence

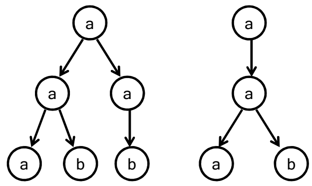
E2: Distance Equivalence

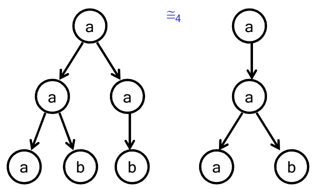
E3: Trace Equivalence

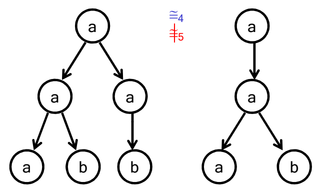
E4: Similarity (mutual simulation)

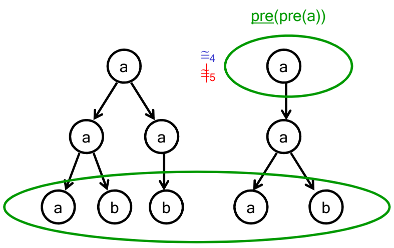
E5: Bisimilarity

$q_1 \cong_5 q_2$ iff if q_1 simulates q_2 via a symmetric simulation relation (this is called a bisimulation relation).









Specification Logics:

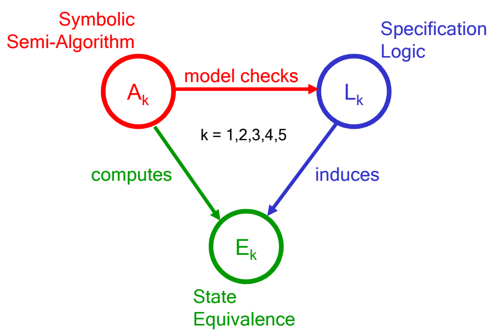
L1	Reachability
L2	Conjunction-free μ-calculus
L3	Guarded μ-calculus / LTL / omega automata
L4	Existential μ-calculus / ∃CTL
L5	μ-Calculus / CTL

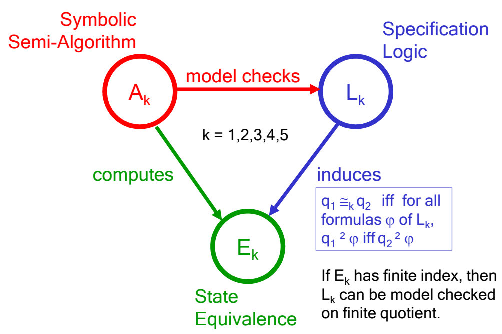
State Equivalences:

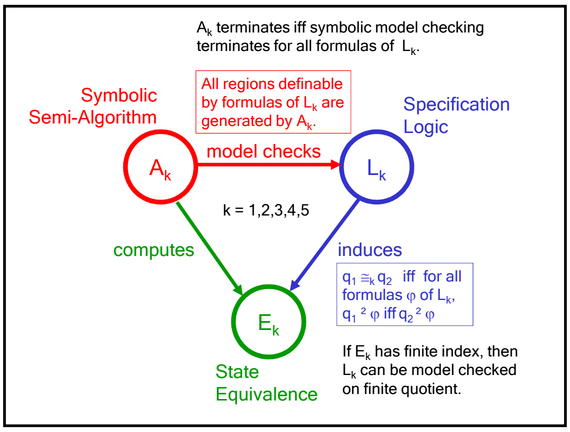
E1	Bounded-reach equivalence
E2	Distance equivalence
E3	Trace equivalence
E4	Similarity
E5	Bisimilarity

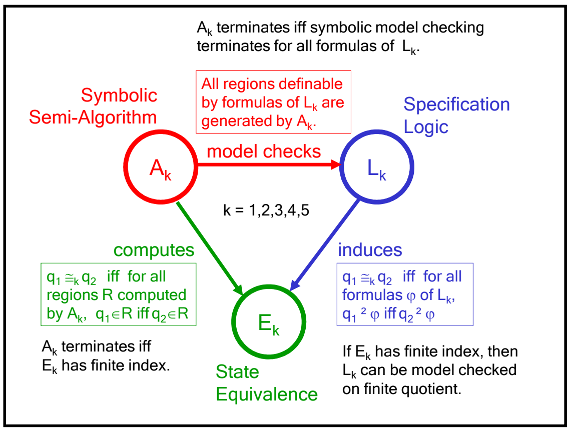
Symbolic Semi-Algorithms:

A1	Backwards pre iteration
A2	Closure under pre
A3	Closure under pre, Å a
A4	Closure under pre, Å
A5	Closure under pre, Å , \ ("partition refinement")









Five Classes of Symbolic Transition Systems

STS1: pre^* terminates ⇔ Finite bounded-reach equiv ⇒
∃◇ decidable

STS2: pre closure terminates ⇔ Finite distance equiv ⇒
conjunction-free μ-calculus decidable

STS3: (pre, Å a) closure terminates ⇔ Finite trace equiv ⇒
guarded μ-calculus (LTL, omega automata) decidable

STS4: (pre, Å) closure terminates ⇔ Finite similarity ⇒
existential μ-calculus (∃CTL, ∀CTL) decidable

STS5: (pre, Å , \ ) closure terminates ⇔ Finite bisimilarity ⇒
μ-calculus (CTL) decidable

Five Classes of Symbolic Transition Systems

STS1: pre^* terminates ⇔ Finite bounded-reach equiv ⇒
∃◇ decidable
Well-structured transition systems of Finkel et al.

STS2: pre closure terminates ⇔ Finite distance equiv ⇒
conjunction-free μ-calculus decidable

STS3: (pre, Å a) closure terminates ⇔ Finite trace equiv ⇒
guarded μ-calculus (LTL, omega automata) decidable
Initialized rectangular hybrid automata

STS4: (pre, Å) closure terminates ⇔ Finite similarity ⇒
existential μ-calculus (∃CTL, ∀CTL) decidable
2D initialized rectangular hybrid automata

STS5: (pre, Å , \ ) closure terminates ⇔ Finite bisimilarity ⇒
μ-calculus (CTL) decidable
Initialized singular hybrid automata

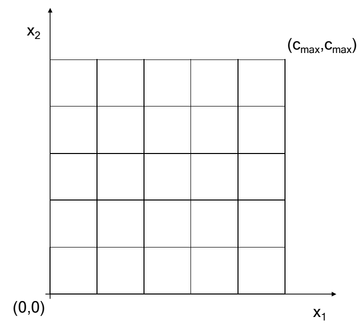
Example: Singular Hybrid Automata

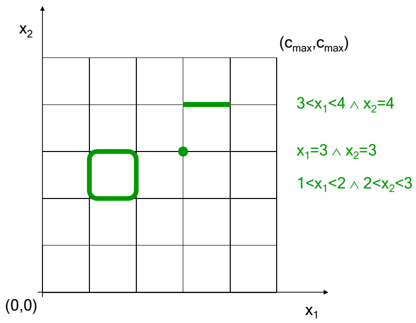
$Q = B^m \times R^n$

Invariants and guards:
integral bounds, e.g. $x_1 < 7 \wedge 1 \le x_2 \le 2$

Flows: constant slopes, e.g. $x'_1 = 1;\ x'_2 = 2$
Jumps: integral assignments, e.g. $x_1 := 0;\ x_2 := 5$

$A = \{ x_i = c,\ c < x_i < c+1 \mid 1 \le i \le n,\ c \in N,\ c < c_{max} \}$





Example: Singular Hybrid Automata

$Q = B^m \times R^n$

Invariants and guards:
integral bounds, e.g. $x_1 < 7 \;\wedge\; 1 \leq x_2 \leq 2$

Flows: constant slopes, e.g. $x'_1 = 1;\; x'_2 = 2$
Jumps: integral assignments, e.g. $x_1 := 0;\; x_2 := 5$

$A = \{ x_i = c,\ c < x_i < c+1 \mid 1 \leq i \leq n,\ c \in N,\ c < c_{max} \}$

Initialized: assignment when slope changes.

Special Case: Timed Automata

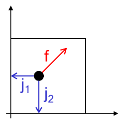
$Q = B^m \times R^n$

Invariants and guards:
integral bounds, e.g. $x_1 < 7 \;\wedge\; 1 \leq x_2 \leq 2$

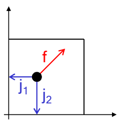
Flows: clocks, e.g. $x'_1 = 1;\; x'_2 = 1$
Jumps: integral assignments, e.g. $x_1 := 0;\; x_2 := 5$

$A = \{ x_i = c,\ c < x_i < c+1 \mid 1 \leq i \leq n,\ c \in N,\ c < c_{max} \}$

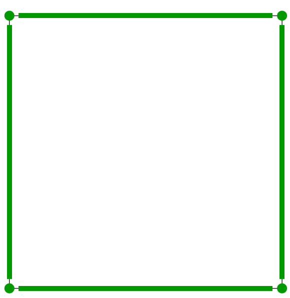
Always initialized.

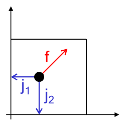


f: $x_1' = 1; x_2' = 1$
j_1: $x_1 := 0$
j_2: $x_2 := 0$

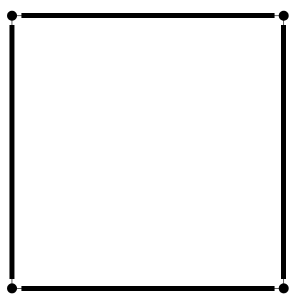


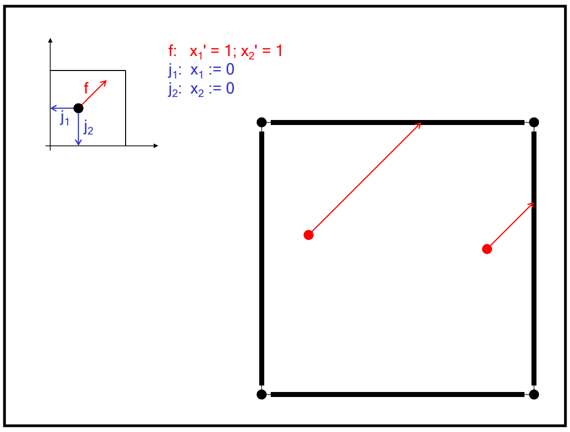
f: $x_1' = 1; x_2' = 1$
j_1: $x_1 := 0$
j_2: $x_2 := 0$

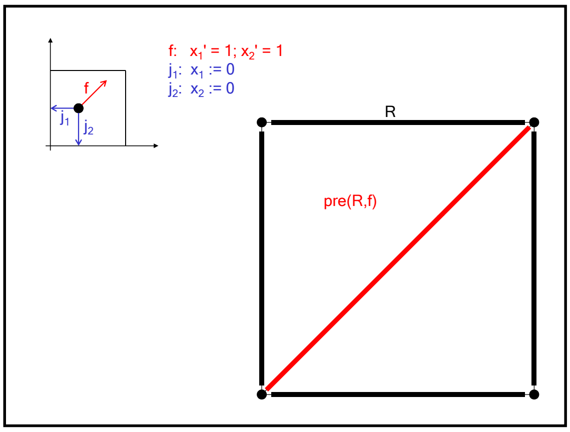


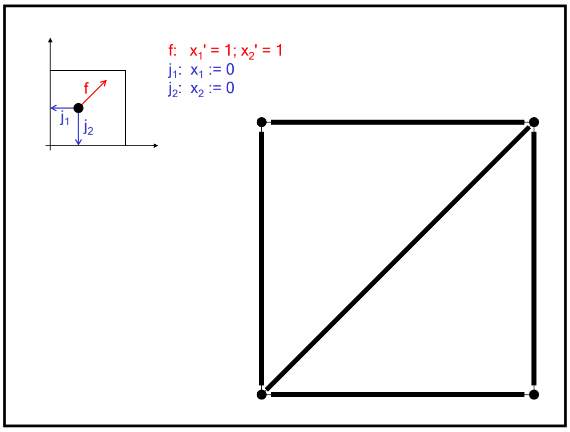


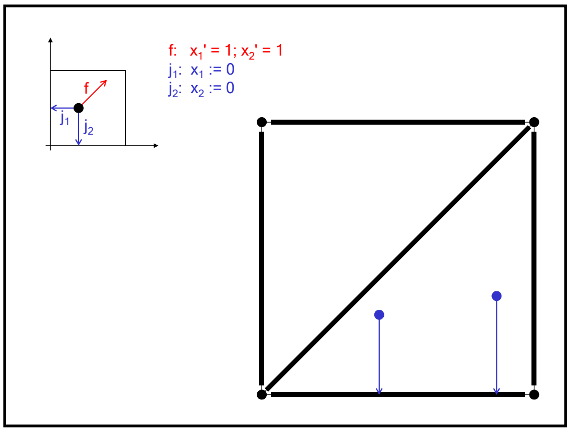
f: $x_1' = 1; x_2' = 1$
j_1: $x_1 := 0$
j_2: $x_2 := 0$

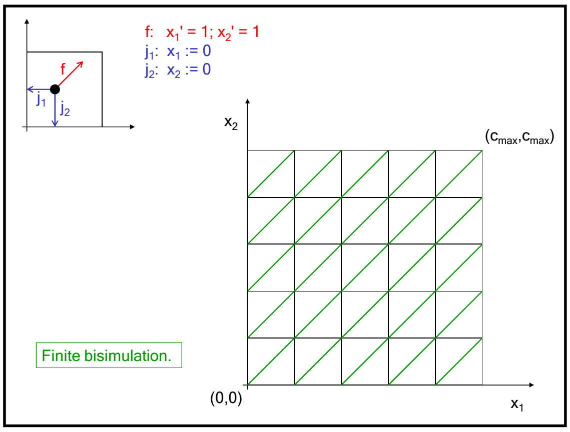


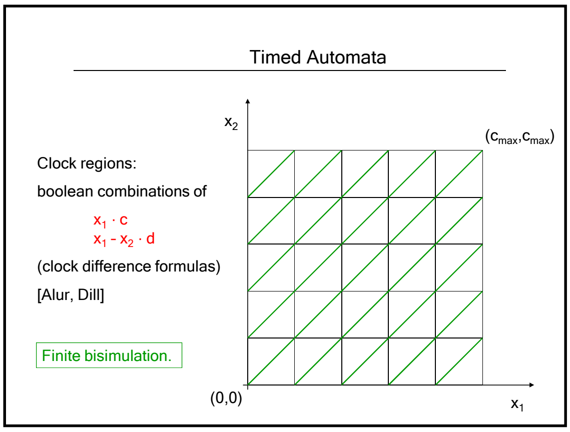

f: $x_1' = 1$; $x_2' = 1$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2


f: $x_1' = 1$; $x_2' = 1$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2
R
pre(R,f)


f: $x_1' = 1$; $x_2' = 1$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2


f: $x_1' = 1$; $x_2' = 1$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2


f: $x_1' = 1$; $x_2' = 1$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2
x_2
(c_{max},c_{max})
Finite bisimulation.
(0,0)
x_1


Timed Automata
x_2
(c_{max},c_{max})
Clock regions:
boolean combinations of
$x_1 \cdot c$
$x_1 - x_2 \cdot d$
(clock difference formulas)
[Alur, Dill]
Finite bisimulation.
(0,0)
x_1

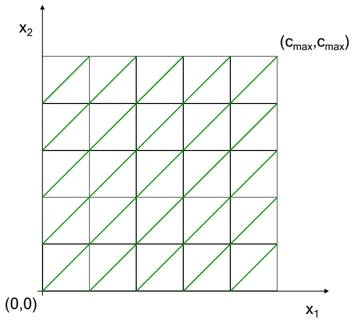
Timed Automata

Corollary:
guards and invariants

$x_1 - x_2 \cdot d$

are permissible.

Finite bisimulation.

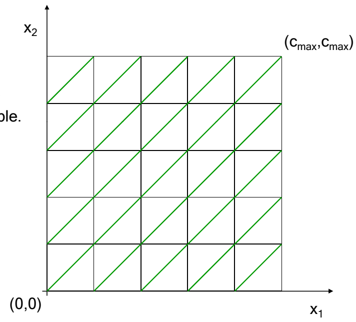


Timed Automata

Corollaries:

CTL model checking over timed automata is decidable.

Finite bisimulation.



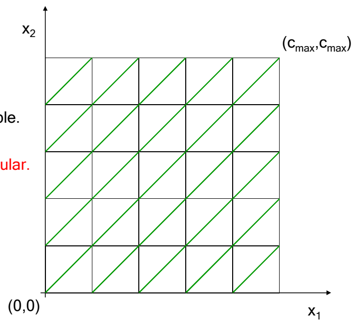
Timed Automata

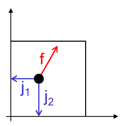
Corollaries:

CTL model checking over timed automata is decidable.

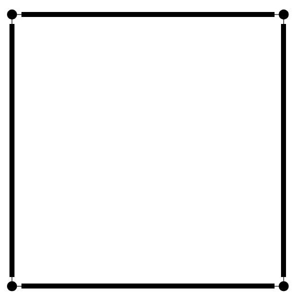
The language of a timed automaton is (omega) regular.

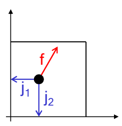
Finite bisimulation.



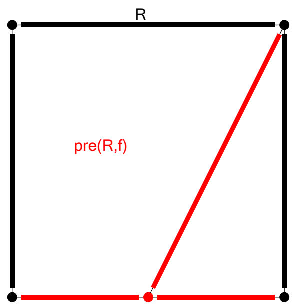


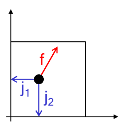
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$



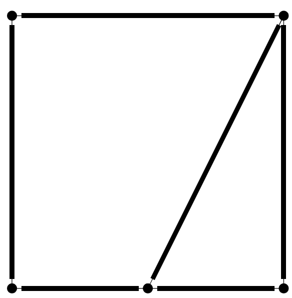


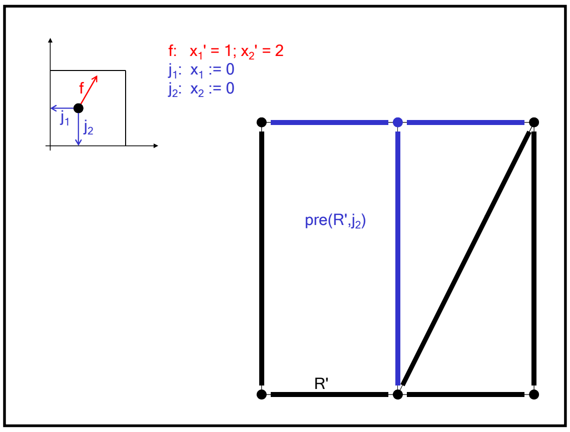
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$

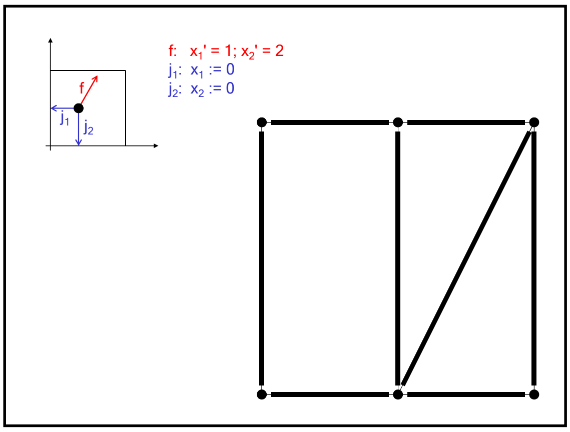


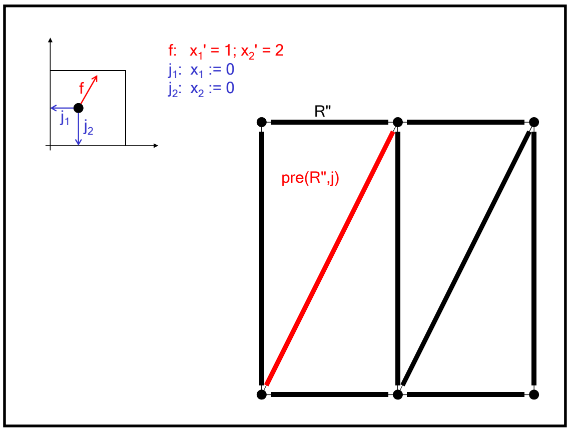


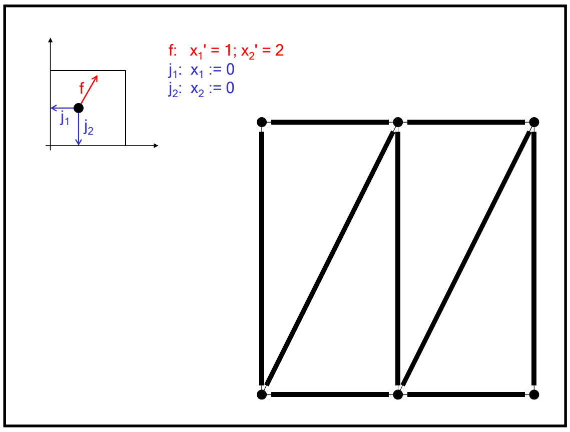
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$

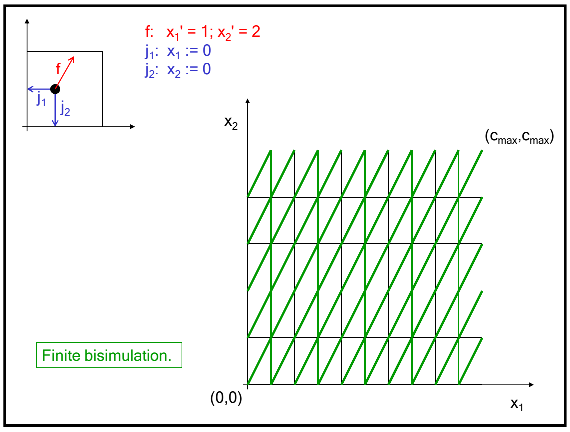


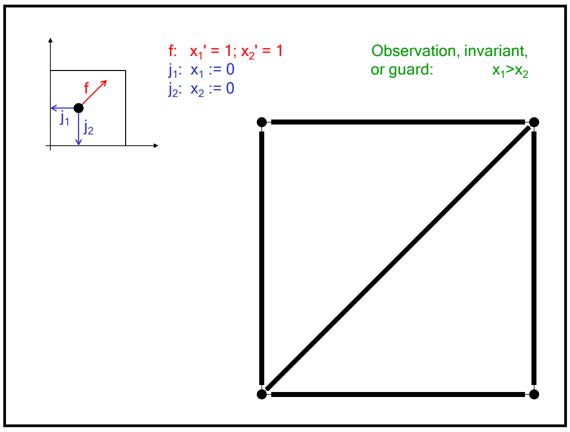

f: x1' = 1; x2' = 2
j1: x1 := 0
j2: x2 := 0
f
j1
j2
pre(R',j2)
R'

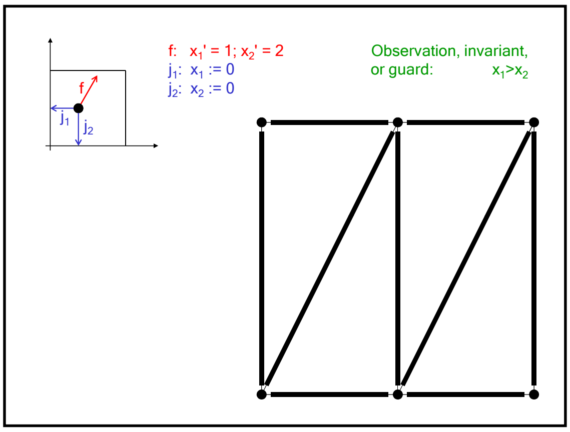

f: x1' = 1; x2' = 2
j1: x1 := 0
j2: x2 := 0
f
j1
j2


f: x1' = 1; x2' = 2
j1: x1 := 0
j2: x2 := 0
f
j1
j2
R"
pre(R",j)

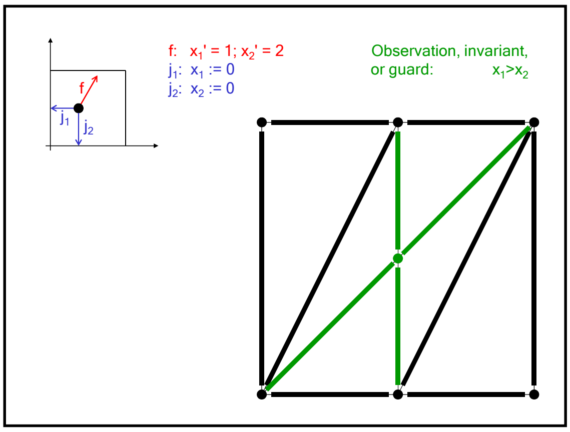

f: x1' = 1; x2' = 2
j1: x1 := 0
j2: x2 := 0
f
j1
j2


f: x1' = 1; x2' = 2
j1: x1 := 0
j2: x2 := 0
f
j1
j2
x2
(cmax,cmax)
Finite bisimulation.
(0,0)
x1

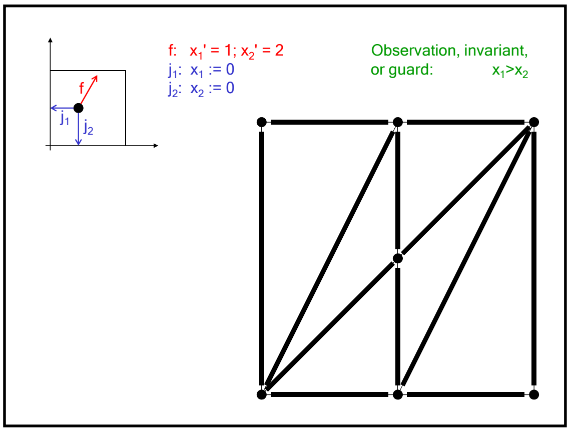

f: x1' = 1; x2' = 1
j1: x1 := 0
j2: x2 := 0
Observation, invariant,
or guard: x1>x2
f
j1
j2



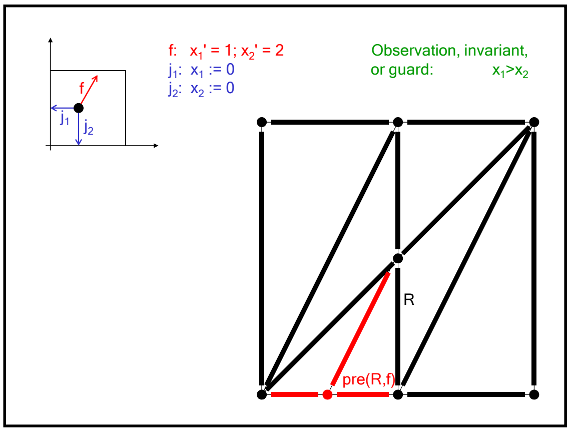
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
Observation, invariant,
or guard: $x_1>x_2$
f
j_1
j_2



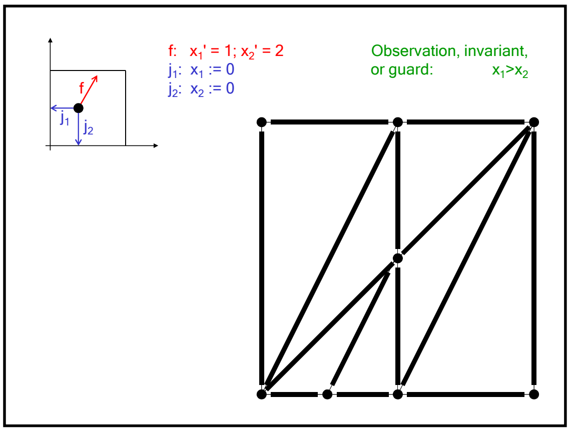
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
Observation, invariant,
or guard: $x_1>x_2$
f
j_1
j_2



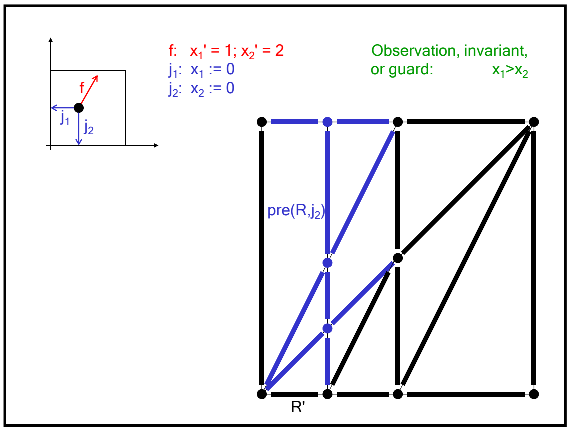
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
Observation, invariant,
or guard: $x_1>x_2$
f
j_1
j_2



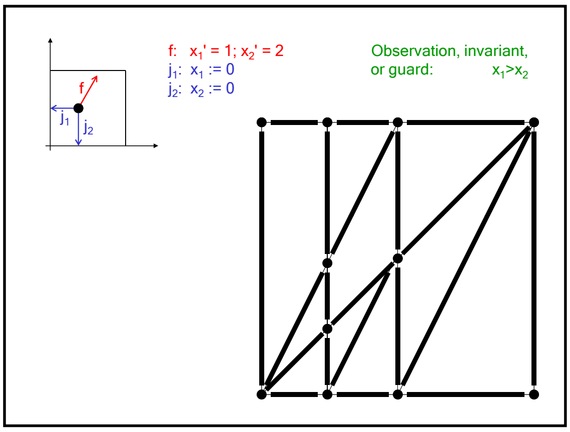
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
Observation, invariant,
or guard: $x_1>x_2$
f
j_1
j_2
R
pre(R,f)

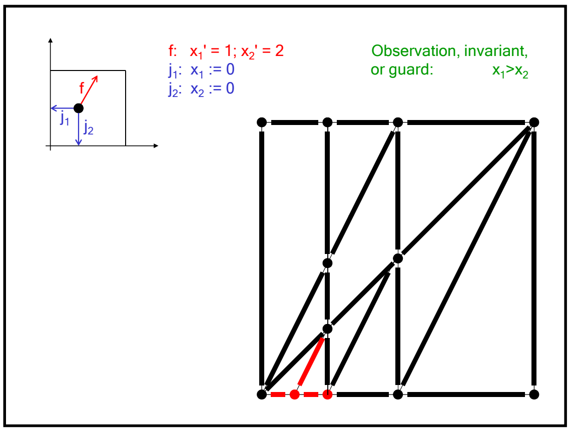


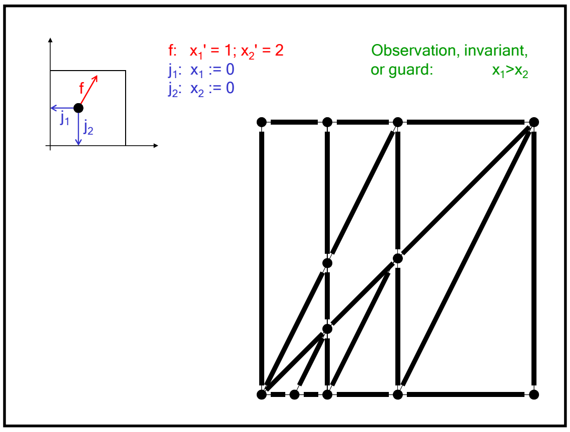
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
Observation, invariant,
or guard: $x_1>x_2$
f
j_1
j_2

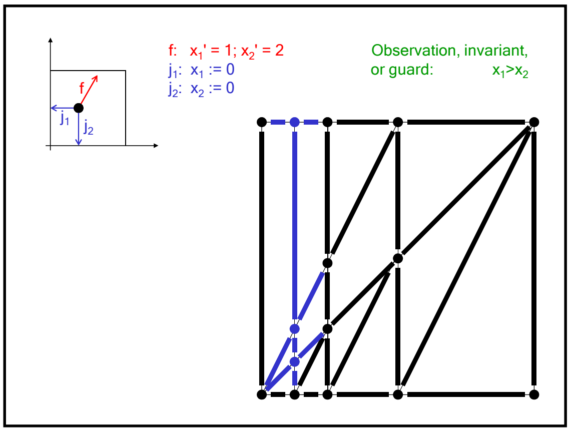


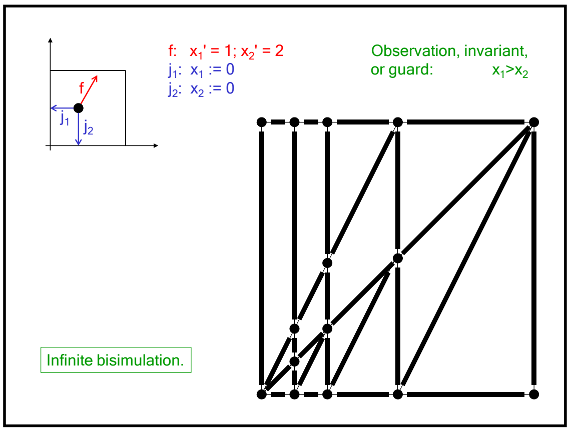
f: $x_1' = 1; x_2' = 2$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
Observation, invariant,
or guard: $x_1>x_2$
f
j_1
j_2
pre(R,j_2)
R'











Undecidability of Reachability for Uninitialized Singular Automata

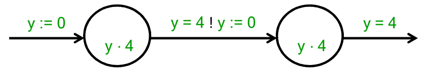
x	clock used for storage of counter value
y	auxiliary clock
z	$z' = 1$ or $z' = 2$

Undecidability of Reachability for Uninitialized Singular Automata

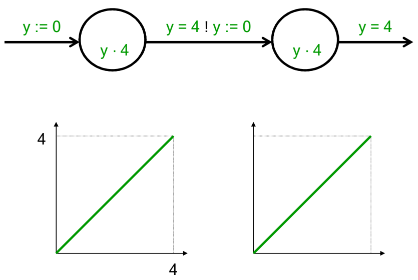
x clock used for storage of counter value
y auxiliary clock
z z' = 1 or z' = 2

Encoding of counter value a:
every 4 time units, $x = 1 / 2^a$

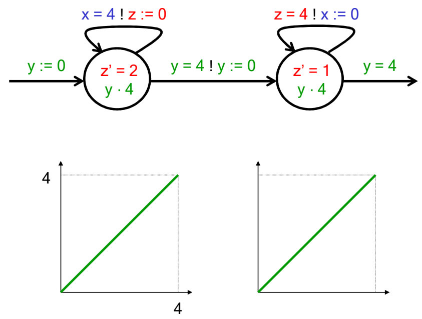
Doubling of Clock Value



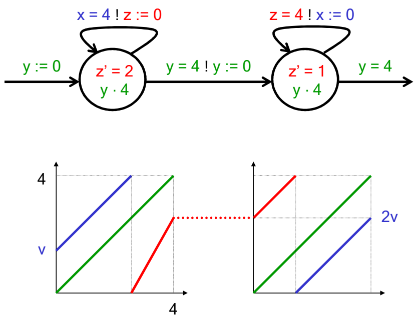
Doubling of Clock Value



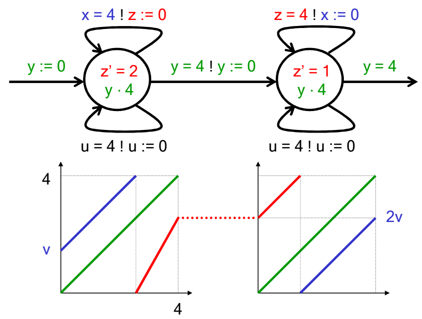
Doubling of Clock Value



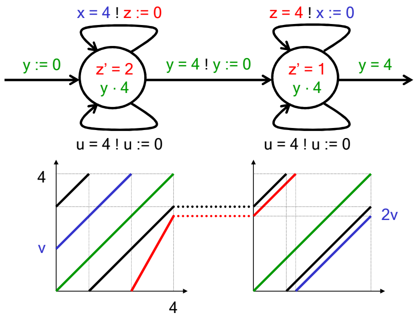
Doubling of Clock Value



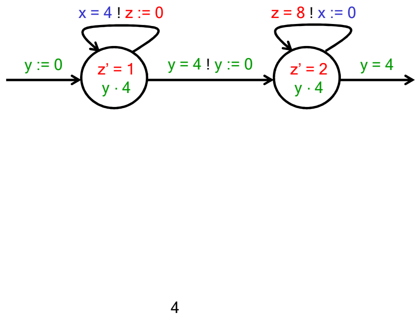
Doubling of Clock Value



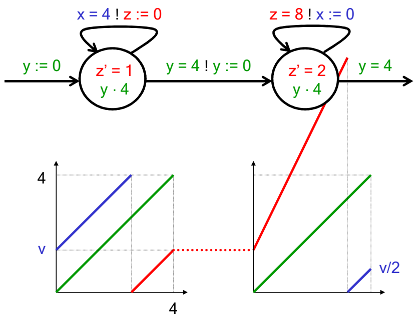
Doubling of Clock Value



Halving of Clock Value



Halving of Clock Value



Example: Rectangular Hybrid Automata

$Q = B^m \times R^n$

Invariants and guards:

integral bounds, e.g. $x_1 < 7 \wedge 1 \leq x_2 \leq 2$

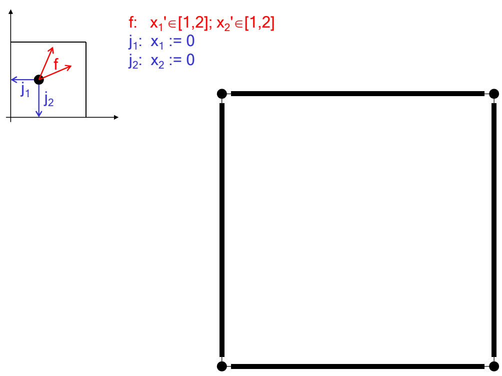
Flows: bounded slopes, e.g. $x'_1 \in [1,2]$; $x'_2 = 1$

Jumps: integral assignments, e.g. $x_1 := 0$; $x_2 := 5$

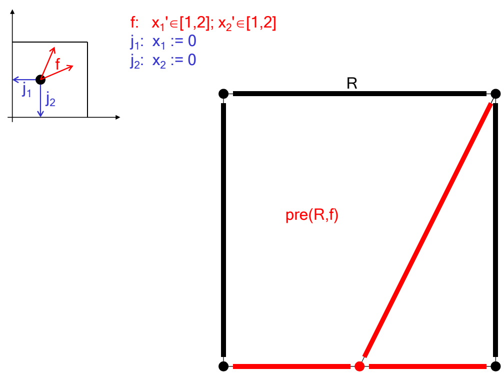
$A = \{ x_i = c,\ c < x_i < c+1 \mid 1 \leq i \leq n,\ c \in N,\ c < c_{max} \}$

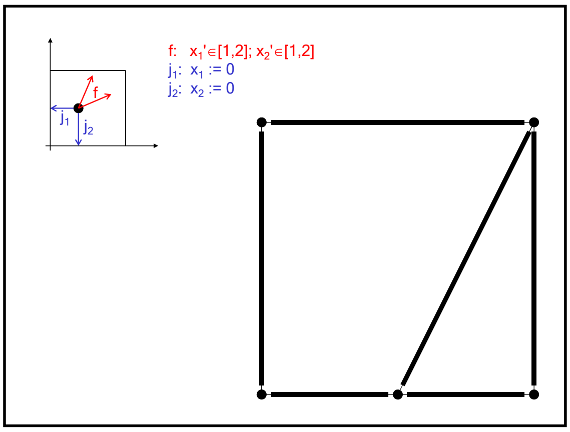
Initialized: assignment when slope bounds change.

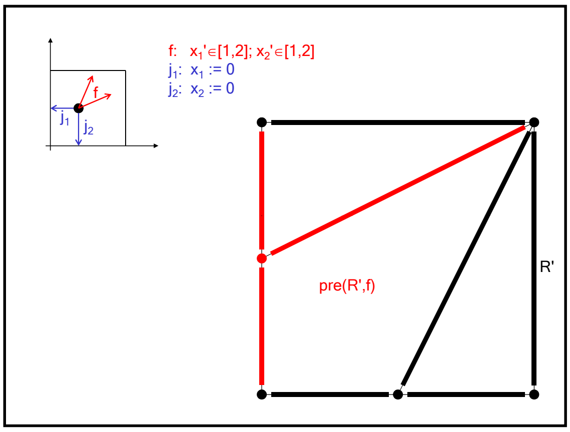
f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$

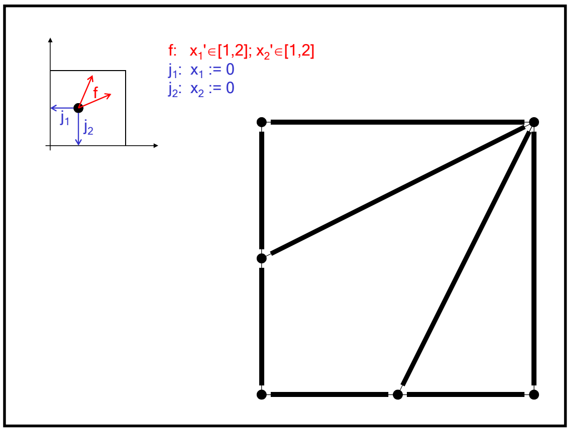


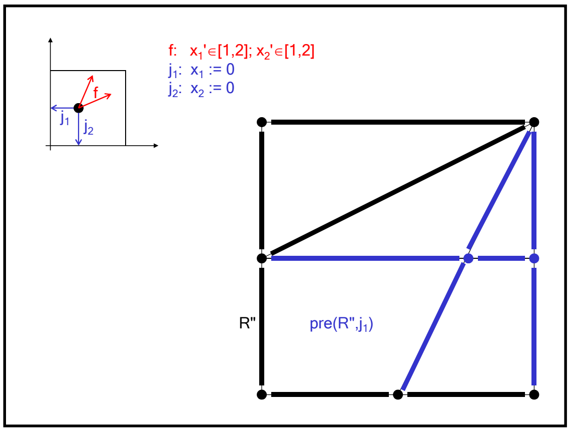
f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$

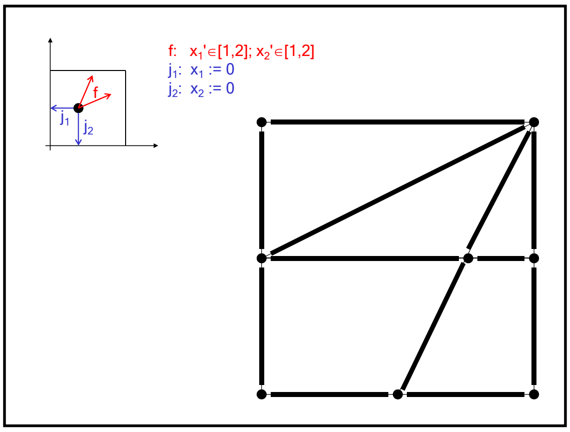


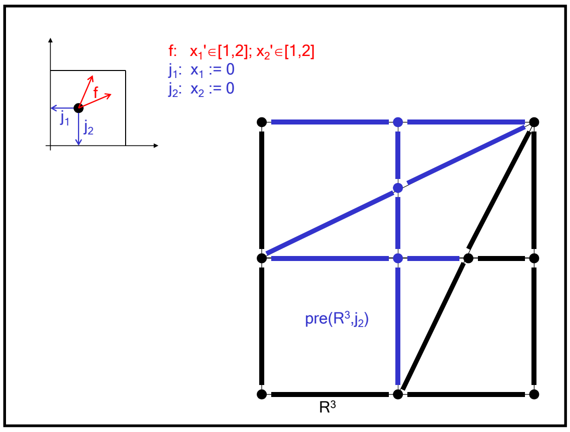

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

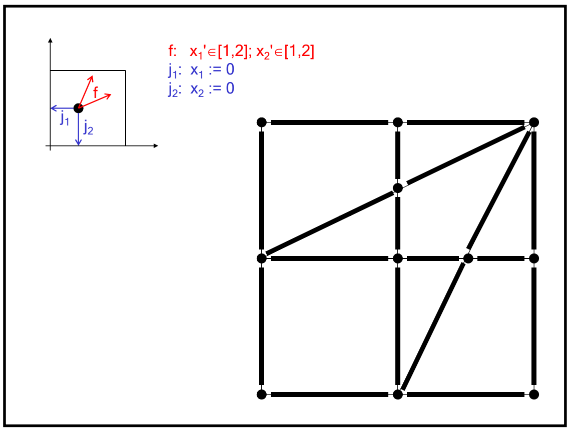

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2
R'
pre(R',f)

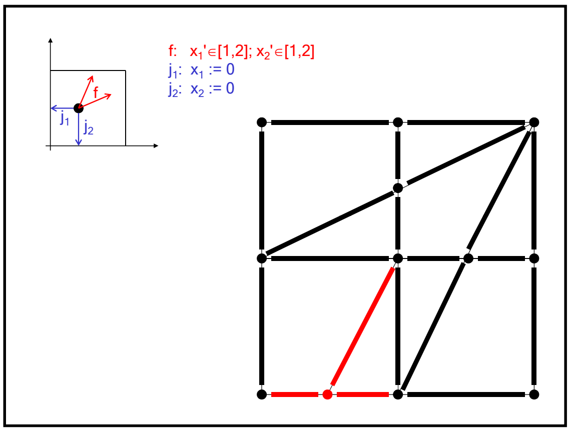

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

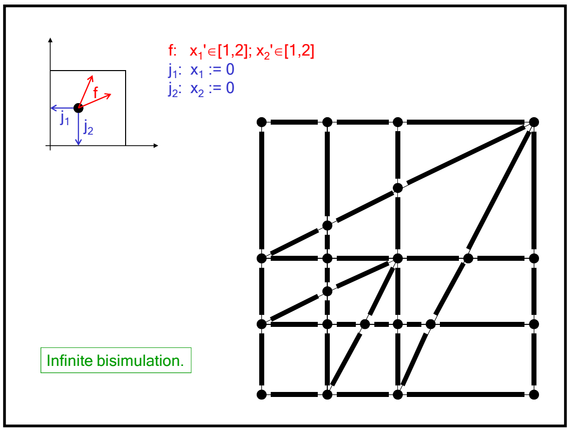

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2
R"
pre(R",j_1)

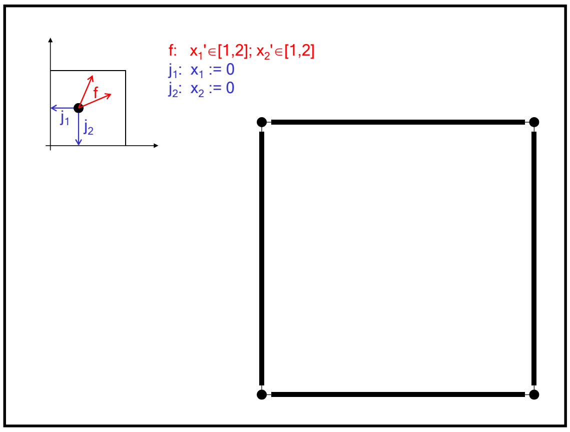

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

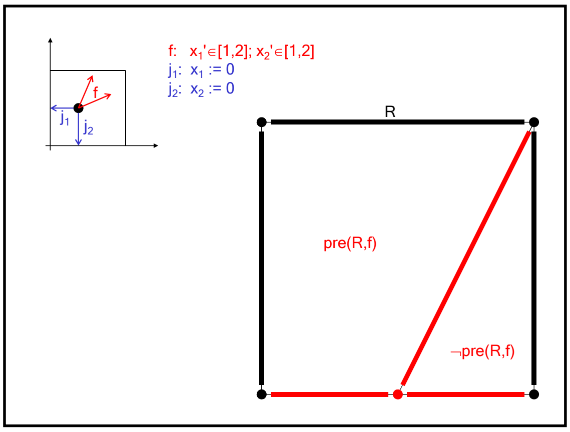

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2
pre(R^3,j_2)
R^3

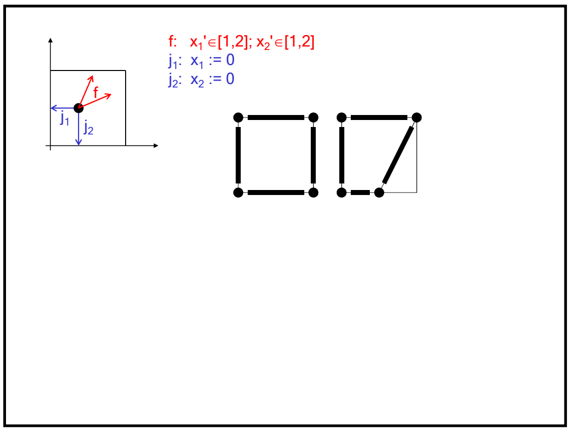

f: x1'∈[1,2]; x2'∈[1,2]
j1: x1 := 0
j2: x2 := 0
f
j1
j2

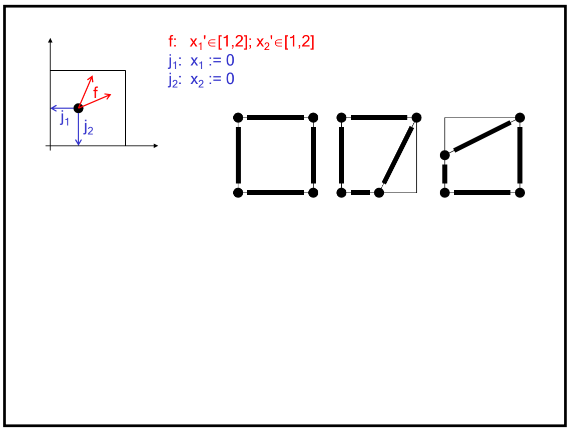

f: x1'∈[1,2]; x2'∈[1,2]
j1: x1 := 0
j2: x2 := 0
f
j1
j2

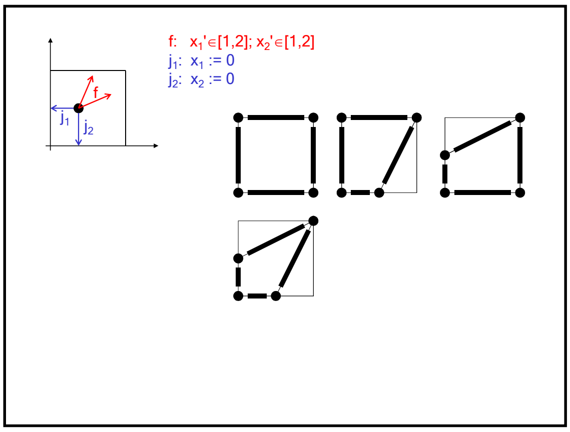

f: x1'∈[1,2]; x2'∈[1,2]
j1: x1 := 0
j2: x2 := 0
f
j1
j2
Infinite bisimulation.

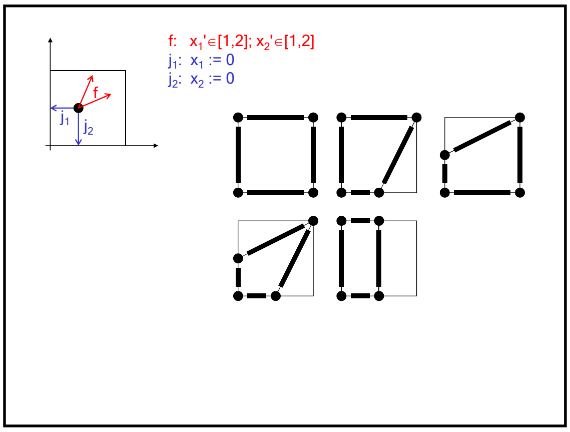

f: x1'∈[1,2]; x2'∈[1,2]
j1: x1 := 0
j2: x2 := 0
f
j1
j2

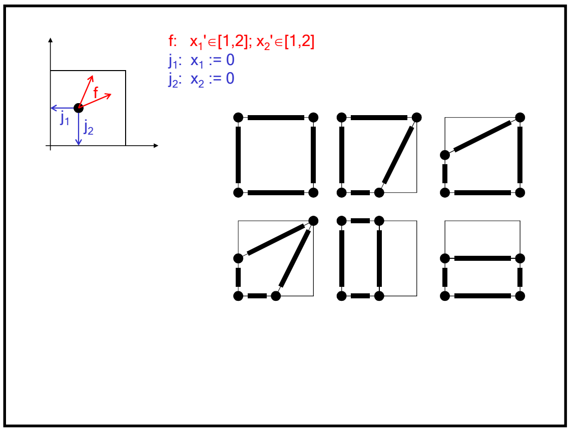

f: x1'∈[1,2]; x2'∈[1,2]
j1: x1 := 0
j2: x2 := 0
f
j1
j2
R
pre(R,f)
¬pre(R,f)

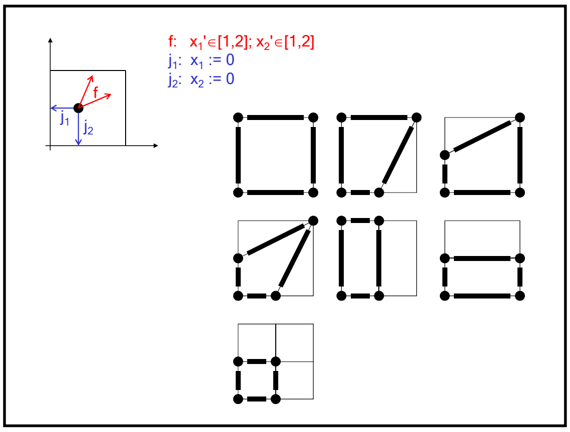

f: x1'∈[1,2]; x2'∈[1,2]
j1: x1 := 0
j2: x2 := 0
f
j1
j2

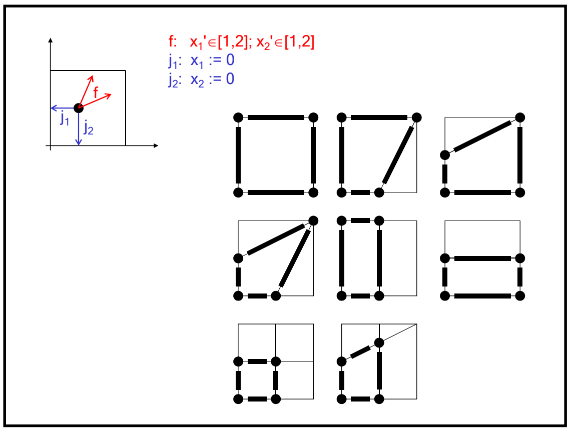

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

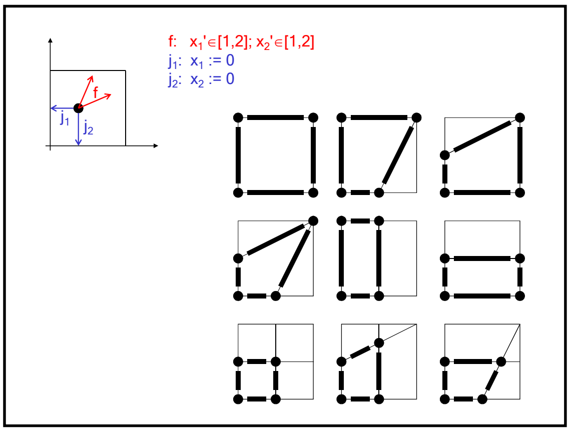

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

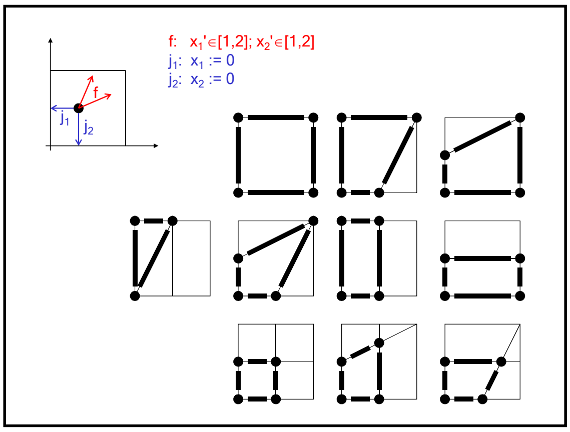

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

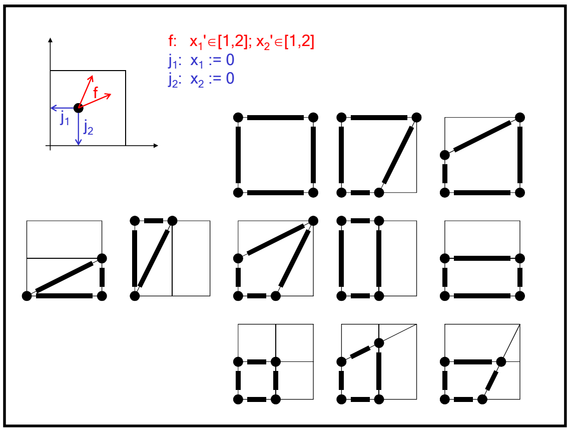

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

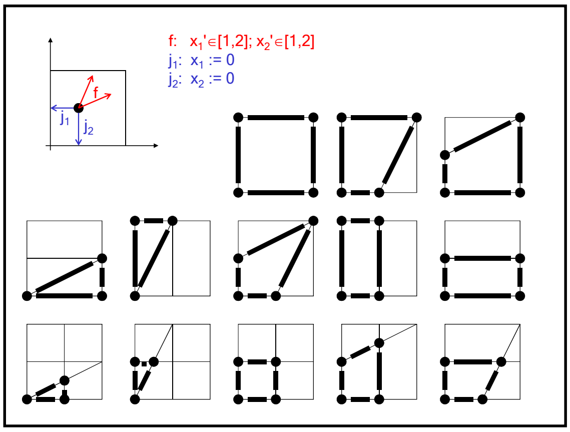

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

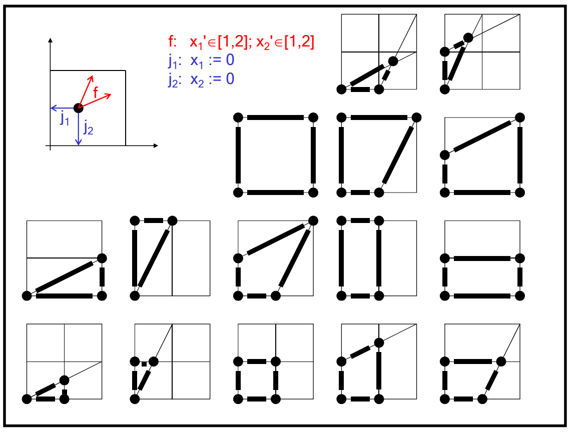

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

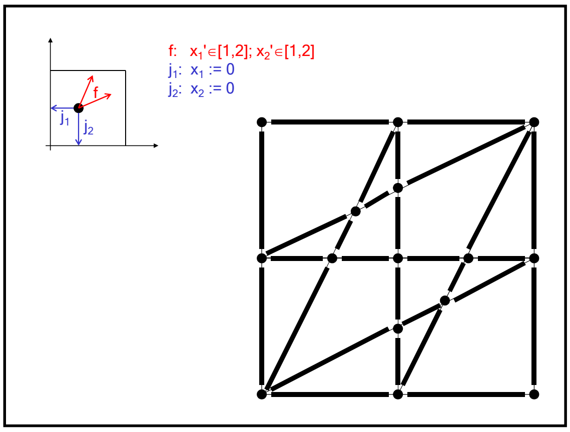

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2

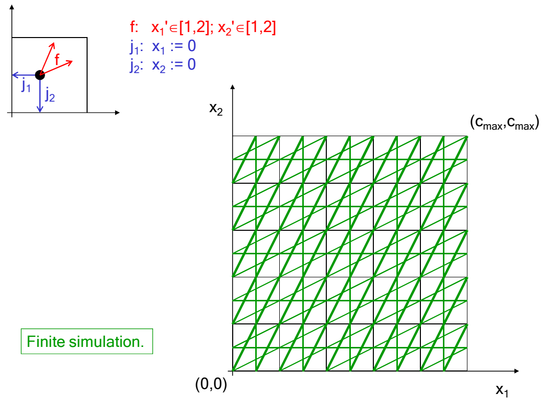

f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2


f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2


f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2


f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2


f: $x_1' \in [1,2]$; $x_2' \in [1,2]$
j_1: $x_1 := 0$
j_2: $x_2 := 0$
f
j_1
j_2



Summary

Timed and initialized singular automata:

STS5 ⇒ CTL model checking
[Alur, Dill; Alur, Courcoubetis, H, Ho]

2D initialized rectangular automata:

STS4 ⇒ ∀CTL model checking
[H, Kopke]

Initialized rectangular automata:

STS3 ⇒ LTL model checking
[H, Kopke, Puri, Varaiya]

Networks of timed automata:

STS1 ⇒ reachability analysis
[Abdullah, Jonsson]

Suppose a hybrid system consists of several components (e.g., controller and plant).

V1-5: Can the components collaborate to achieve an objective?

C1-5: Can a subset of the components (e.g., the controller) achieve the objective no matter how the other components (the plant) behave?

Need model that preserves components:
"players" in a concurrent game.

The Thermostat Revisited

Player 1 (plant):

States

$x \in \mathbb{R}$ temperature

Inputs

$h \in \{ \text{on}, \text{off} \}$ heat

Flows

f_1 ✡ $h = \text{on} \rightarrow x' = K \cdot (H - x)$
f_2 ✡ $h = \text{off} \rightarrow x' = -K \cdot x$

Jumps

The Thermostat Revisited

Payer 2 (controller);

States

$h \in \{ \text{on}, \text{off} \}$ heat
$t \in \mathbb{R}$ timer

Flows

f ✡ $t \leq U \rightarrow t' = 1$

Jumps

j_1 ✡ $h = \text{on} \wedge t \geq L \rightarrow h := \text{off};\ t := 0$
j_2 ✡ $h = \text{off} \wedge t \geq L \rightarrow h := \text{on};\ t := 0$

Concurrent Game

Q	states
Σ_1, Σ_2	moves of both players
post: $Q \times \Sigma_1 \times \Sigma_2 \rightarrow Q$	transitions

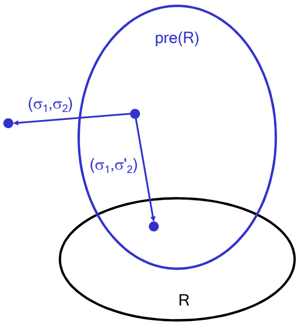
Concurrent Game

Q	states
Σ_1, Σ_2	moves of both players
post: $Q \times \Sigma_1 \times \Sigma_2 \to Q$	transitions

cpre$_1$: $2^Q \times \Sigma_1 \to 2^Q$

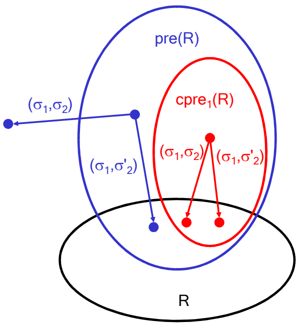
$q \in \text{cpre}_1(R,\sigma_1)$ iff for all $\sigma_2 \in \Sigma_2$, $\text{post}(q,\sigma_1,\sigma_2) \in R$

$\Sigma_1 = \{\sigma_1,\sigma'_1\}$
$\Sigma_2 = \{\sigma_2,\sigma'_2\}$



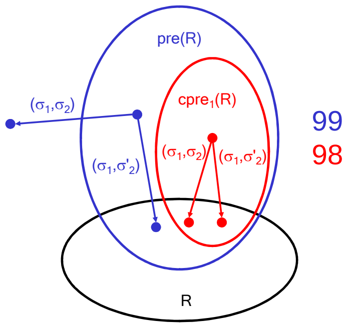
$\Sigma_1 = \{\sigma_1,\sigma'_1\}$
$\Sigma_2 = \{\sigma_2,\sigma'_2\}$

$\text{cpre}_1(R) = \cup_{\sigma_1 \in \Sigma_1} \text{cpre}_1(R,\sigma_1)$



$\Sigma_1 = \{\sigma_1,\sigma'_1\}$
$\Sigma_2 = \{\sigma_2,\sigma'_2\}$

$\text{cpre}_1(R) = \cup_{\sigma_1 \in \Sigma_1} \text{cpre}_1(R,\sigma_1)$



Concurrent Game

Q	states
Σ_1, Σ_2	moves of both players
post: $Q \times \Sigma_1 \times \Sigma_2 \to Q$	transitions

cpre$_1$: $2^Q \times \Sigma_1 \to 2^Q$

$q \in \text{cpre}_1(R,\sigma_1)$ iff for all $\sigma_2 \in \Sigma_2$, $\text{post}(q,\sigma_1,\sigma_2) \in R$

cpre$_2$: $2^Q \times \Sigma_2 \to 2^Q$

$q \in \text{cpre}_2(R,\sigma_2)$ iff for all $\sigma_1 \in \Sigma_1$, $\text{post}(q,\sigma_1,\sigma_2) \in R$

Symbolic Concurrent Game

Q	states
Σ_1, Σ_2	moves of both players
cpre$_1$, cpre$_2$	controllable pre operators
A	observations
$\Re = \{ R_1, R_2, \ldots \}$	regions $R_i \subseteq Q$

Region algebra:

1.	$A \subseteq \Re$	
2.	cpre$_1$: $\Re \times \Sigma_1 \to \Re$ computable cpre$_2$: $\Re \times \Sigma_2 \to \Re$ computable	
3.	Å : $\Re^2 \to \Re$ \ : $\Re^2 \to \Re$ $\subseteq$: $\Re^2 \to \{ t, f \}$	computable

Symbolic Semi-Algorithm

Starting from the observations in A, compute new regions in ℜ by applying the operations $cpre_1$, $cpre_2$, Å , \ , and ⊆ .

V1: Verification for Reachability

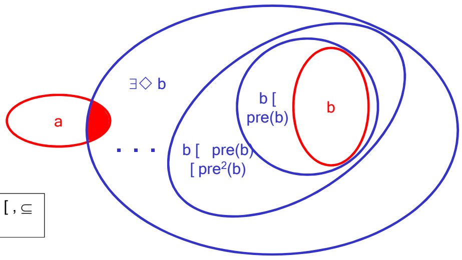
a ∧ ∃◇ b

Given a,b∈A, is there a trajectory from a to b?

V1: Verification for Reachability

a ∧ ∃◇ b

Given a,b∈A, is there a trajectory from a to b?



1. pre, [, ⊆
2. Å a

C1: Control for Reachability

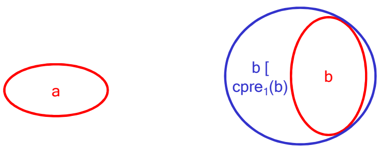
a ∧ ⟨⟨1⟩⟩◇ b

Given a,b∈A, does player 1 have a strategy to force the game from a to b?

C1: Control for Reachability

a ∧ ⟨⟨1⟩⟩◇ b

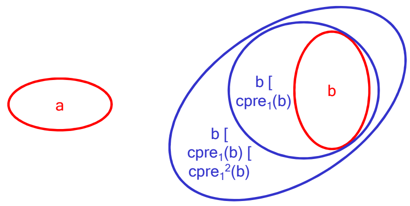
Given a,b∈A, does player 1 have a strategy to force the game from a to b?



C1: Control for Reachability

a ∧ ⟨⟨1⟩⟩◇ b

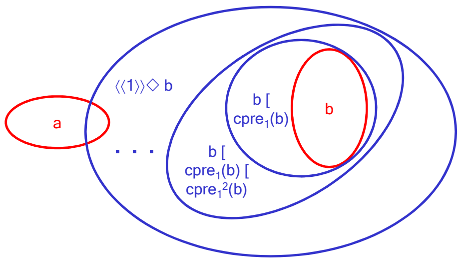
Given a,b∈A, does player 1 have a strategy to force the game from a to b?



C1: Control for Reachability

a ∧ ⟨⟨1⟩⟩◇ b

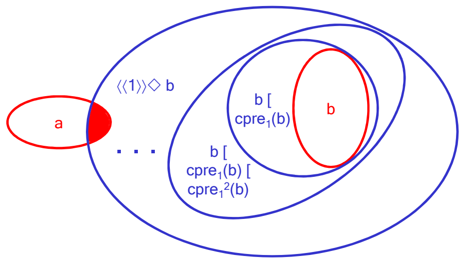
Given a,b∈A, does player 1 have a strategy to force the game from a to b?



C1: Control for Reachability

a ∧ ⟨⟨1⟩⟩◇ b

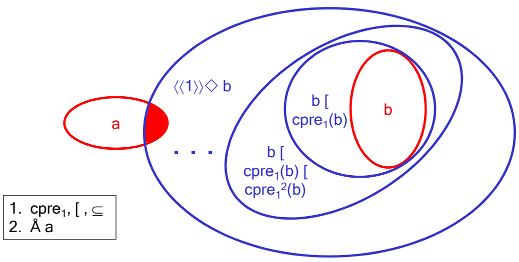
Given a,b∈A, does player 1 have a strategy to force the game from a to b?



C1: Control for Reachability

a ∧ ⟨⟨1⟩⟩◇ b

Given a,b∈A, does player 1 have a strategy to force the game from a to b?



V3: Buechi Verification

a ∧ ∃□◇ b

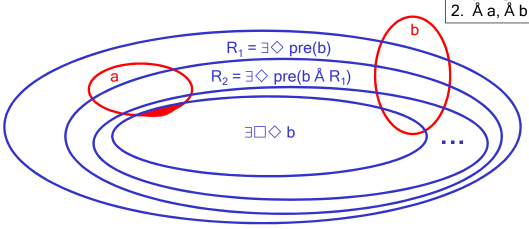
Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?

V3: Buechi Verification

a ∧ ∃□◇ b

Given a,b∈A, is there an infinite trajectory from a that visits b infinitely often?

1. pre, [, ⊆
2. Å a, Å b



C3: Buechi Control

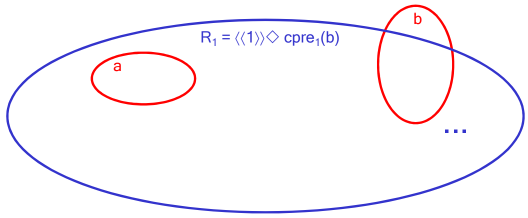
a ∧ ⟨⟨1⟩⟩□◇ b

Given a,b∈A, does player 1 have a strategy from a to visit b infinitely often?

C3: Buechi Control

a ∧ ⟨⟨1⟩⟩□◇ b

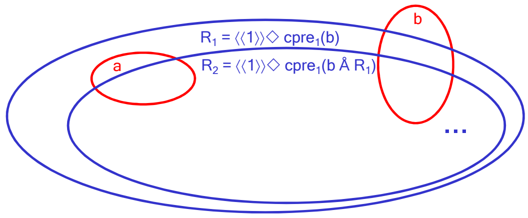
Given a,b∈A, does player 1 have a strategy from a to visit b infinitely often?



C3: Buechi Control

a ∧ ⟨⟨1⟩⟩□◇ b

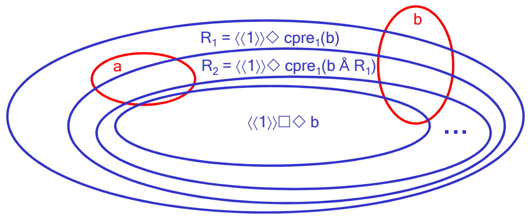
Given a,b∈A, does player 1 have a strategy from a to visit b infinitely often?



C3: Buechi Control

a ∧ ⟨⟨1⟩⟩□◇ b

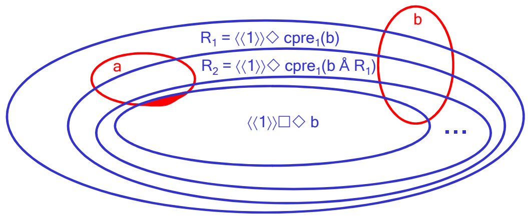
Given a,b∈A, does player 1 have a strategy from a to visit b infinitely often?



C3: Buechi Control

a ∧ ⟨⟨1⟩⟩□◇ b

Given a,b∈A, does player 1 have a strategy from a to visit b infinitely often?

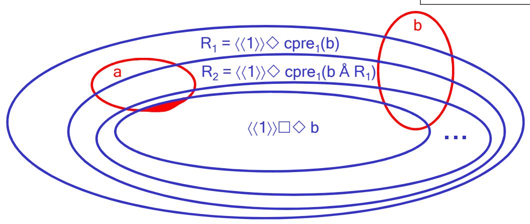


C3: Buechi Control

a ∧ ⟨⟨1⟩⟩□◇ b

Given a,b∈A, does player 1 have a strategy from a to visit b infinitely often?

1. $cpre_1$, [, ⊆
2. Å a, Å b



q_1 is simulated by q_2

iff

there is a simulation relation S such that

1. S(q_1,q_2)
2. if S(p,q) then
   a. (8 a2A) (p2 a iff q2 a)
   b. (8 p') (if p2 pre(p') then
      (9 q') (q2 pre(q') Æ S(p',q')))

q_1 is alternating simulated by q_2 [Alur, H, Kupferman, Vardi]

iff

there is an alternating simulation relation S such that

1. S(q_1,q_2)
2. if S(p,q) then
   a. (8 a2A) (p2 a iff q2 a)
   b. (8 p') (if p2 cpre$_1$(p') then (9 q') (q2 cpre$_1$(q') Æ S(p',q')))

Five Classes of Symbolic Concurrent Games

SCG1: cpre$_1$ iteration terminates ⇒ ⟨⟨1⟩⟩◇ decidable

SCG2: cpre$_1$ closure terminates ⇒ conjunction-free alternating μ-calculus decidable

SCG3: (cpre$_1$, Å a) terminates ⇔ Finite alternating 1-trace equiv ⇒ guarded alternating μ-calculus (LTL, omega games) decidable

SCG4: (cpre$_1$, Å) terminates ⇔ Finite alternating 1-similarity ⇒ existential alternating μ-calculus (⟨⟨1⟩⟩ATL) decidable

SCG5: (cpre$_1$, Å , \ ) terminates ⇔ Finite alternating 1-bisimilarity ⇒ alternating μ-calculus (ATL) decidable

Summary

Timed and initialized singular automata:

SCG5 ⇒ ATL control [de Alfaro, H, Majumdar]

2D initialized rectangular automata:

SCG4 ⇒ ⟨⟨1⟩⟩ATL control [de Alfaro, H, Majumdar]

Initialized rectangular automata:

SCG3 ⇒ LTL control [H, Horowitz, Majumdar]

Networks of timed automata:

SCG1 ⇒ reachability control

Verification vs. Control: Can we use the "same" algorithms?

Vϕ Suppose we have an LTL formula ϕ and a symbolic semi-algorithm A(pre) that computes 9ϕ.

Cϕ Question: does A(cpre$_1$) compute ⟨⟨1⟩⟩ϕ, that is, does it solve the game with player-1 objective ϕ?

Verification vs. Control: Can we use the "same" algorithms?

Vϕ Suppose we have an LTL formula ϕ and a symbolic semi-algorithm A(pre) that computes 9ϕ.

Cϕ Question: does A(cpre$_1$) compute ⟨⟨1⟩⟩ϕ, that is, does it solve the game with player-1 objective ϕ?

Not necessarily!

From Verification to Control

Thm 1: If A(pre) computes 9ϕ and A(pre) computes 8ϕ, then A(cpre$_1$) computes ⟨⟨1⟩⟩ϕ.

From Verification to Control

Thm 1: If A(pre) computes 9ϕ and A(pre) computes 8ϕ,
then A($cpre_1$) computes $\langle\langle 1 \rangle\rangle\phi$.

Example: Since 9◇ a = μ X. (a Ç pre(X))
and 8◇ a = μ X. (a Ç pre(X))
also $\langle\langle 1 \rangle\rangle$◇ a = μ X. (a Ç $cpre_1$(X))

From Verification to Control

Thm 1: If A(pre) computes 9ϕ and A(pre) computes 8ϕ,
then A($cpre_1$) computes $\langle\langle 1 \rangle\rangle\phi$.

Example: Since 9◇ a = μ X. (a Ç pre(X))
and 8◇ a = μ X. (a Ç pre(X))
also $\langle\langle 1 \rangle\rangle$◇ a = μ X. (a Ç $cpre_1$(X))

Thm 2: For every LTL formula ϕ, we can construct a symbolic semi-algorithm (i.e., guarded μ-calculus formula) A_ϕ that satisfies the premise of Thm 1.

[de Alfaro, H, Majumdar: LICS 2001]

Two Messages for Infinite-State Model Checking and Control

1. Separate local (region algebra) from global (symbolic semi-algorithm) concerns
2. Appeal to finite abstractions in the termination argument, not in the algorithm