Automated Reasoning and Formal Verification
Module II: Formal Verification
Ch. 09: **Timed and Hybrid Systems**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
URL: https://disi.unitn.it/rseba/DIDATTICA/arfv2025/
Teaching assistant: Gabriele Masina – gabriele.masina@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems
Academic year 2024-2025

last update: Friday 21st February, 2025, 11:21

Outline

Outline

Acknowledgments

Thanks for providing material to:

- Rajeev Alur & colleagues (Penn University)
- Paritosh Pandya (IIT Bombay)
- Andrea Mattioli, Yusi Ramadian (Univ. Trento)
- Marco Di Natale (Scuola Superiore S.Anna, Italy)

Disclaimer

- very introductory
- very-partial coverage
- mostly computer-science centric

Acknowledgments

Thanks for providing material to:

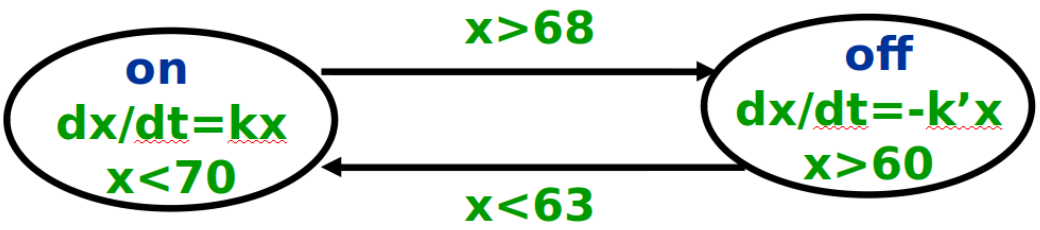
- Rajeev Alur & colleagues (Penn University)
- Paritosh Pandya (IIT Bombay)
- Andrea Mattioli, Yusi Ramadian (Univ. Trento)
- Marco Di Natale (Scuola Superiore S.Anna, Italy)

Disclaimer

- very introductory
- very-partial coverage
- mostly computer-science centric

Hybrid Modeling

Hybrid machines = State machines + Dynamic Systems



Hybrid Modeling: Examples

- Automotive Applications
- Vehicle Coordination Protocols
- Interacting Autonomous Robots
- Bio-molecular Regulatory Networks

Hybrid Modeling: Examples

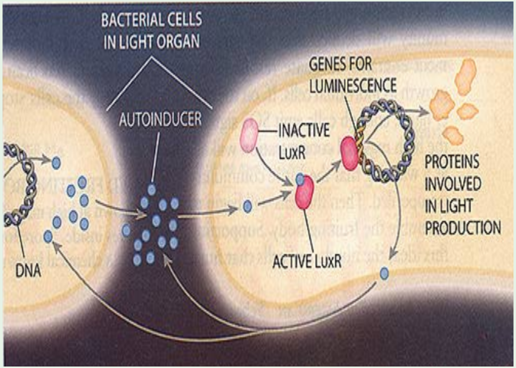
- Automotive Applications
- Vehicle Coordination Protocols
- Interacting Autonomous Robots
- Bio-molecular Regulatory Networks

Hybrid Modeling: Examples

- Automotive Applications
- Vehicle Coordination Protocols
- Interacting Autonomous Robots
- Bio-molecular Regulatory Networks

Hybrid Modeling: Examples

- Automotive Applications
- Vehicle Coordination Protocols
- Interacting Autonomous Robots
- Bio-molecular Regulatory Networks

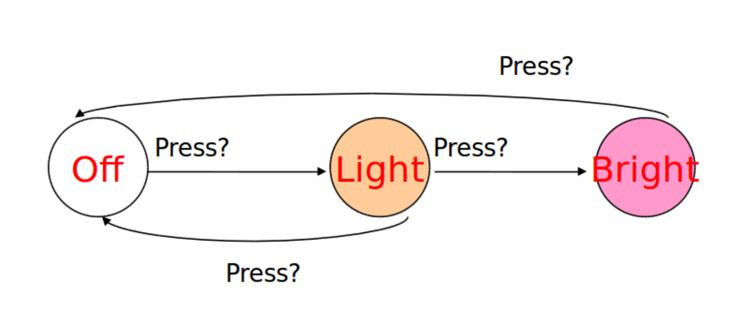


Outline

Outline

Timed Automata

Example: Simple light control

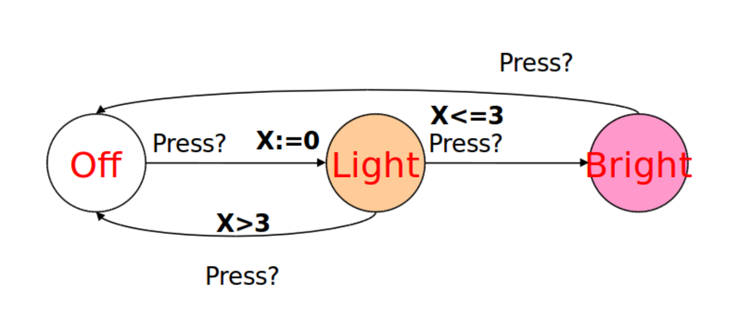


Requirement:

- if Off and press is issued once, then the light switches on;
- if Off and press is issued twice quickly, then the light gets brighter;
- if Light/Bright and press is issued once, then the light switches off;

$\Longrightarrow$ Cannot be achieved with standard automata

Example: Simple light control



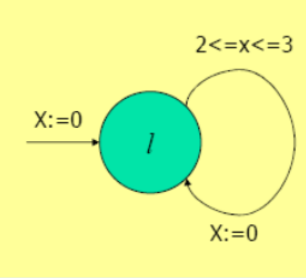
Solution: add real-valued clock x

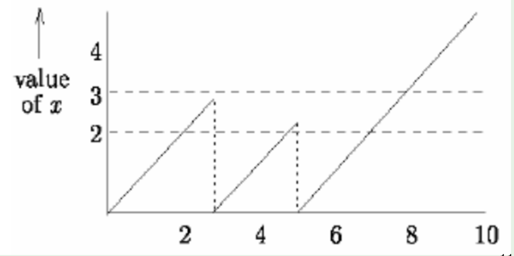
- x reset at first press
- if next press before x reaches 3 time units, then the light will get brighter;
- otherwise the light is turned off

Modeling: timing constraints

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

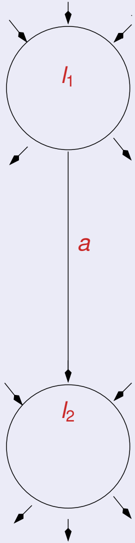
- Vertexes are locations
  - Time can elapse there
  - Constraints (invariants)
- Edges are switches
  - Subject to constraints
  - Reset clocks





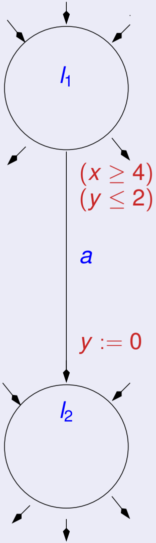
Timed Automata

- Locations l_1, l_2, ... (like in standard automata)
  - discrete part of the state
  - may be implemented by discrete variables
- Switches (discrete transitions like in standard aut.)
- Labels, aka events, actions,... (like in standard aut.)
  - used for synchronization
- Clocks: x, y,... $\in \mathbb{Q}^+$
  - value: time elapsed since the last time it was reset
- Guards: $(x \bowtie C)$ s.t. $\bowtie \in \{\leq, <, \geq, >\}$, $C \in \mathbb{N}$
  - set of clock comparisons against positive integer bounds
  - constrain the execution of the switch
- Resets $(x := 0)$
  - set of clock assignments to 0
- Invariants: $(x \bowtie C)$ s.t. $\bowtie \in \{\leq, <, \geq, >\}$, $C \in \mathbb{N}$
  - set of clock comparisons against positive integer bounds
  - ensure progress



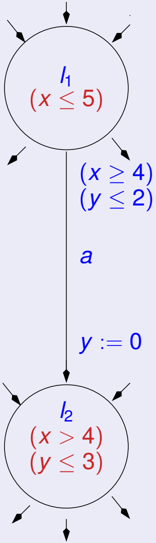
Timed Automata

- Locations $l_1, l_2, \ldots$ (like in standard automata)
  - discrete part of the state
  - may be implemented by discrete variables
- Switches (discrete transitions like in standard aut.)
- Labels, aka events, actions,... (like in standard aut.)
  - used for synchronization
- Clocks: x, y,... $\in \mathbb{Q}^+$
  - value: time elapsed since the last time it was reset
- Guards: $(x \bowtie C)$ s.t. $\bowtie \in \{\leq, <, \geq, >\}$, $C \in \mathbb{N}$
  - set of clock comparisons against positive integer bounds
  - constrain the execution of the switch
- Resets $(x := 0)$
  - set of clock assignments to 0
- Invariants: $(x \bowtie C)$ s.t. $\bowtie \in \{\leq, <, \geq, >\}$, $C \in \mathbb{N}$
  - set of clock comparisons against positive integer bounds
  - ensure progress



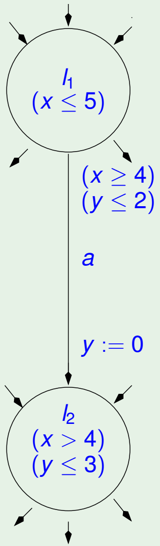
Timed Automata

- Locations l_1, l_2, ... (like in standard automata)
  - discrete part of the state
  - may be implemented by discrete variables
- Switches (discrete transitions like in standard aut.)
- Labels, aka events, actions,... (like in standard aut.)
  - used for synchronization
- Clocks: x, y,... $\in \mathbb{Q}^+$
  - value: time elapsed since the last time it was reset
- Guards: $(x \bowtie C)$ s.t. $\bowtie \in \{\leq, <, \geq, >\}$, $C \in \mathbb{N}$
  - set of clock comparisons against positive integer bounds
  - constrain the execution of the switch
- Resets $(x := 0)$
  - set of clock assignments to 0
- Invariants: $(x \bowtie C)$ s.t. $\bowtie \in \{\leq, <, \geq, >\}$, $C \in \mathbb{N}$
  - set of clock comparisons against positive integer bounds
  - ensure progress



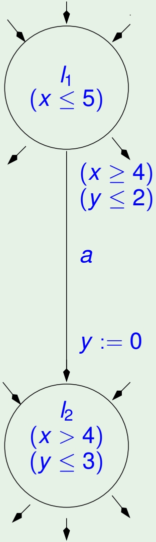
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$



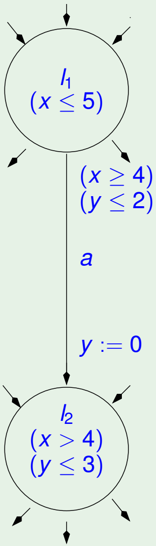
Timed Automata: Example

- State: $\langle l_i, x, y\rangle$
  - $\langle l_1, 4, 7\rangle$:



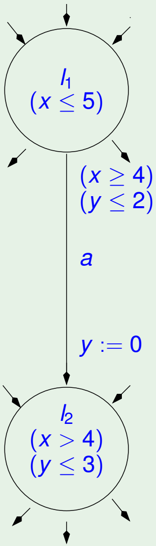
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!



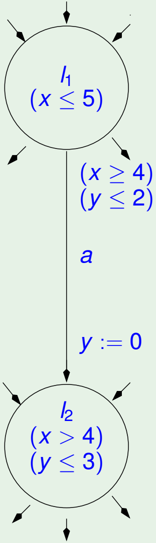
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$:



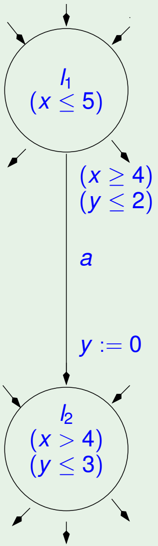
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)



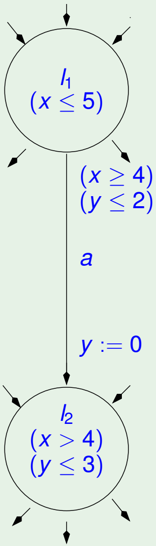
Timed Automata: Example

- State: $\langle l_i, x, y\rangle$
  - $\langle l_1, 4, 7\rangle$: OK!
  - $\langle l_2, 2, 4\rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y\rangle \xrightarrow{a} \langle l_j, x', y'\rangle$



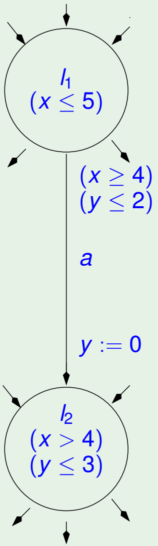
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$:



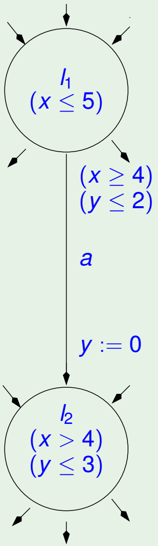
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!



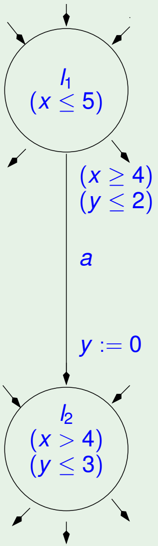
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$:



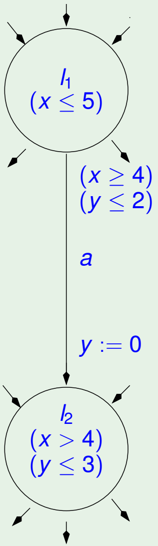
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)



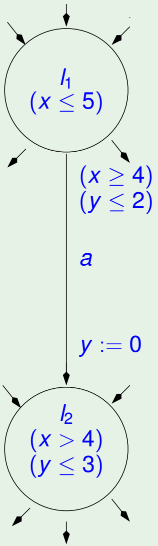
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$:



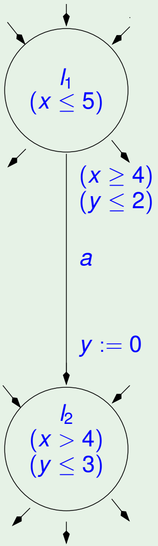
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$: not OK! (violates guard & invar. in l_2)



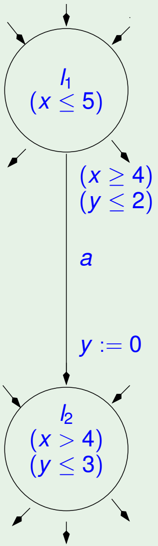
Timed Automata: Example

- State: $\langle l_i, x, y\rangle$
  - $\langle l_1, 4, 7\rangle$: OK!
  - $\langle l_2, 2, 4\rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y\rangle \xrightarrow{a} \langle l_j, x', y'\rangle$
  - $\langle l_1, 4.5, 2\rangle \xrightarrow{a} \langle l_2, 4.5, 0\rangle$: OK!
  - $\langle l_1, 6, 2\rangle \xrightarrow{a} \langle l_2, 6, 0\rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2\rangle \xrightarrow{a} \langle l_2, 3, 0\rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2\rangle \xrightarrow{a} \langle l_2, 4.5, 2\rangle$:



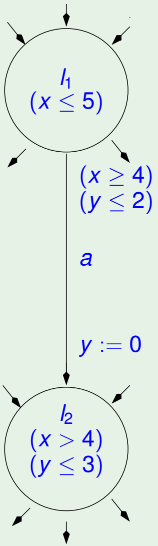
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 2 \rangle$: not OK! (violates reset)



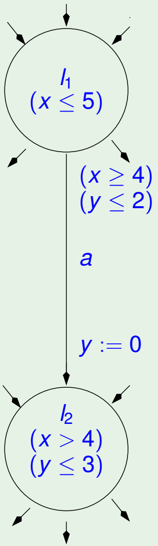
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 2 \rangle$: not OK! (violates reset)
  - $\langle l_1, 4, 2 \rangle \xrightarrow{a} \langle l_2, 4, 0 \rangle$:



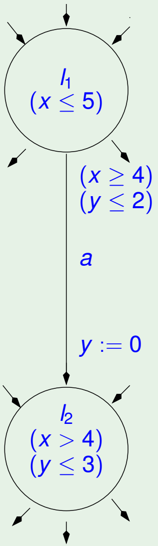
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 2 \rangle$: not OK! (violates reset)
  - $\langle l_1, 4, 2 \rangle \xrightarrow{a} \langle l_2, 4, 0 \rangle$: not OK! (violates invar. in l_2)



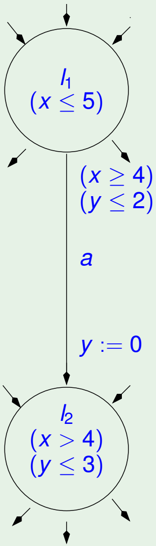
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 2 \rangle$: not OK! (violates reset)
  - $\langle l_1, 4, 2 \rangle \xrightarrow{a} \langle l_2, 4, 0 \rangle$: not OK! (violates invar. in l_2)
- Wait (time elapse): $\langle l_i, x, y \rangle \xrightarrow{\delta} \langle l_i, x + \delta, y + \delta \rangle$



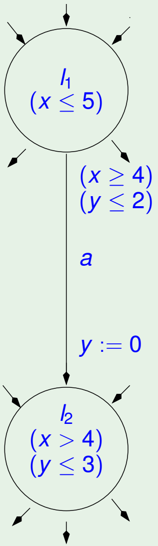
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 2 \rangle$: not OK! (violates reset)
  - $\langle l_1, 4, 2 \rangle \xrightarrow{a} \langle l_2, 4, 0 \rangle$: not OK! (violates invar. in l_2)
- Wait (time elapse): $\langle l_i, x, y \rangle \xrightarrow{\delta} \langle l_i, x + \delta, y + \delta \rangle$
  - $\langle l_1, 3, 0 \rangle \xrightarrow{2} \langle l_1, 5, 2 \rangle$:



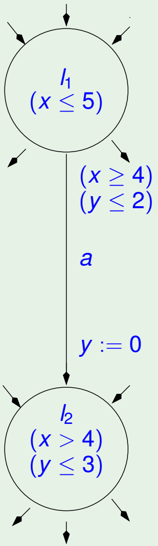
Timed Automata: Example

- State: $\langle l_i, x, y\rangle$
  - $\langle l_1, 4, 7\rangle$: OK!
  - $\langle l_2, 2, 4\rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y\rangle \xrightarrow{a} \langle l_j, x', y'\rangle$
  - $\langle l_1, 4.5, 2\rangle \xrightarrow{a} \langle l_2, 4.5, 0\rangle$: OK!
  - $\langle l_1, 6, 2\rangle \xrightarrow{a} \langle l_2, 6, 0\rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2\rangle \xrightarrow{a} \langle l_2, 3, 0\rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2\rangle \xrightarrow{a} \langle l_2, 4.5, 2\rangle$: not OK! (violates reset)
  - $\langle l_1, 4, 2\rangle \xrightarrow{a} \langle l_2, 4, 0\rangle$: not OK! (violates invar. in l_2)
- Wait (time elapse): $\langle l_i, x, y\rangle \xrightarrow{\delta} \langle l_i, x+\delta, y+\delta\rangle$
  - $\langle l_1, 3, 0\rangle \xrightarrow{2} \langle l_1, 5, 2\rangle$: OK!



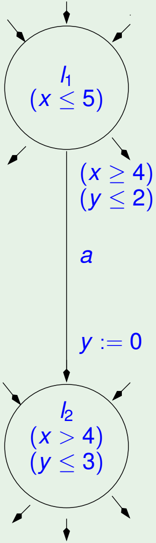
Timed Automata: Example

- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 2 \rangle$: not OK! (violates reset)
  - $\langle l_1, 4, 2 \rangle \xrightarrow{a} \langle l_2, 4, 0 \rangle$: not OK! (violates invar. in l_2)
- Wait (time elapse): $\langle l_i, x, y \rangle \xrightarrow{\delta} \langle l_i, x + \delta, y + \delta \rangle$
  - $\langle l_1, 3, 0 \rangle \xrightarrow{2} \langle l_1, 5, 2 \rangle$: OK!
  - $\langle l_1, 3, 0 \rangle \xrightarrow{3} \langle l_1, 6, 3 \rangle$:



Timed Automata: Example

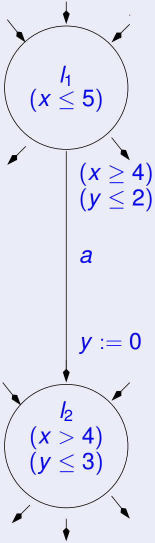
- State: $\langle l_i, x, y \rangle$
  - $\langle l_1, 4, 7 \rangle$: OK!
  - $\langle l_2, 2, 4 \rangle$: not OK! (violates invariant in l_2)
- Switch: $\langle l_i, x, y \rangle \xrightarrow{a} \langle l_j, x', y' \rangle$
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 0 \rangle$: OK!
  - $\langle l_1, 6, 2 \rangle \xrightarrow{a} \langle l_2, 6, 0 \rangle$: not OK! (violates invar. in l_1)
  - $\langle l_1, 3, 2 \rangle \xrightarrow{a} \langle l_2, 3, 0 \rangle$: not OK! (violates guard & invar. in l_2)
  - $\langle l_1, 4.5, 2 \rangle \xrightarrow{a} \langle l_2, 4.5, 2 \rangle$: not OK! (violates reset)
  - $\langle l_1, 4, 2 \rangle \xrightarrow{a} \langle l_2, 4, 0 \rangle$: not OK! (violates invar. in l_2)
- Wait (time elapse): $\langle l_i, x, y \rangle \xrightarrow{\delta} \langle l_i, x + \delta, y + \delta \rangle$
  - $\langle l_1, 3, 0 \rangle \xrightarrow{2} \langle l_1, 5, 2 \rangle$: OK!
  - $\langle l_1, 3, 0 \rangle \xrightarrow{3} \langle l_1, 6, 3 \rangle$: not OK! (violates invar. in l_1)



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E \rangle$

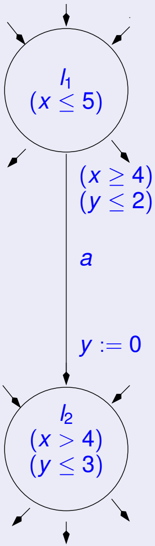
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l' \rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E \rangle$

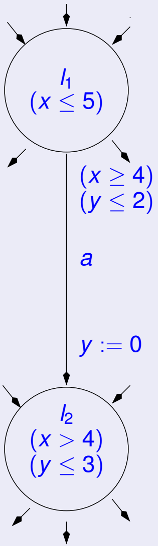
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l' \rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E\rangle$

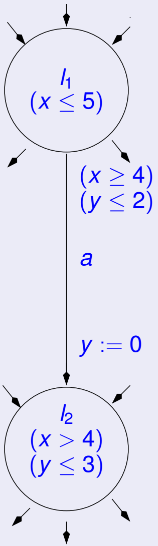
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l'\rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E\rangle$

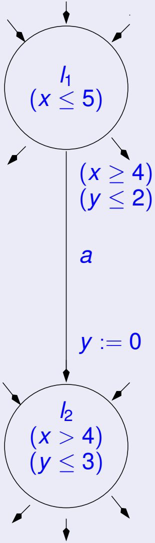
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l'\rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E \rangle$

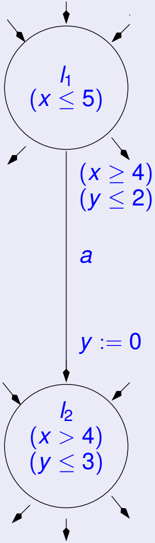
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l' \rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E \rangle$

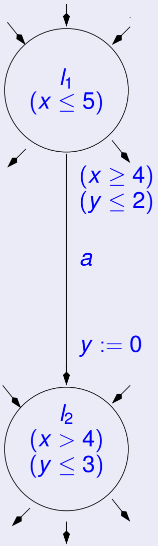
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l' \rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E\rangle$

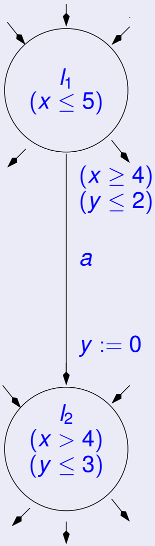
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l'\rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E\rangle$

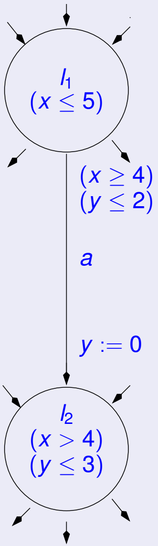
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l'\rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E \rangle$

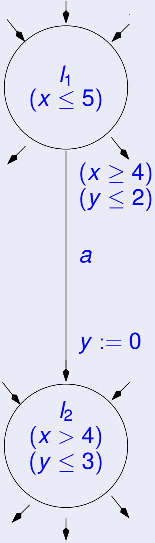
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l' \rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E \rangle$

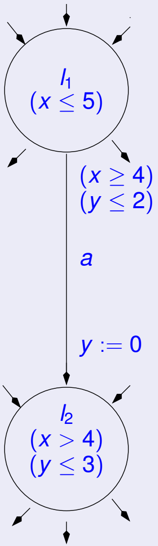
- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l' \rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Timed Automata: Formal Syntax

Timed Automaton $\langle L, L^0, \Sigma, X, \Phi(X), E \rangle$

- L: Set of locations
- $L^0 \subseteq L$: Set of initial locations
- Σ: Set of labels
- X: Set of clocks
- $\Phi(X)$: Set of invariants
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$: Set of switches
  A switch $\langle l, a, \varphi, \lambda, l' \rangle$ s.t.
  - l: source location
  - a: label
  - φ: clock constraints
  - $\lambda \subseteq X$: clocks to be reset
  - l': target location



Clock constraints and clock interpretations

- Grammar of clock constraints:

$$\varphi ::= x \leq C \mid x < C \mid x \geq C \mid x > C \mid \varphi \wedge \varphi$$

s.t. C positive integer values.
$\Longrightarrow$ allow only comparison of a clock with a constant

- clock interpretation: ν

$$X = \langle x, y, z \rangle, \quad \nu = \langle 1.0, 1.5, 0 \rangle$$

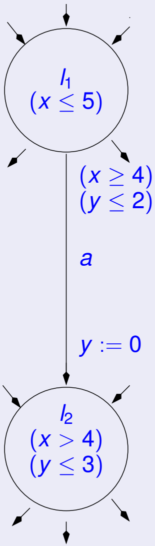
- clock interpretation ν after δ time: $\nu + \delta$

$$\delta = 0.2, \quad \nu + \delta = \langle 1.2, 1.7, 0.2 \rangle$$

- clock interpretation ν after reset λ: $\nu[\lambda]$

$$\lambda = \{y\}, \quad \nu[y := 0] = \langle 1.0, 0, 0 \rangle$$

A state for a timed automaton is a pair $\langle l, \nu \rangle$,
where l is a location and ν is a clock interpretation



Clock constraints and clock interpretations

- Grammar of clock constraints:

$$\varphi ::= x \leq C \mid x < C \mid x \geq C \mid x > C \mid \varphi \wedge \varphi$$

  s.t. C positive integer values.
  $\Longrightarrow$ allow only comparison of a clock with a constant

- clock interpretation: ν

$$X = \langle x, y, z \rangle, \quad \nu = \langle 1.0, 1.5, 0 \rangle$$

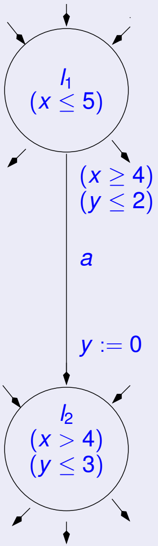
- clock interpretation ν after δ time: $\nu + \delta$

$$\delta = 0.2, \quad \nu + \delta = \langle 1.2, 1.7, 0.2 \rangle$$

- clock interpretation ν after reset λ: $\nu[\lambda]$

$$\lambda = \{y\}, \quad \nu[y := 0] = \langle 1.0, 0, 0 \rangle$$

A state for a timed automaton is a pair $\langle l, \nu \rangle$,
where l is a location and ν is a clock interpretation



Clock constraints and clock interpretations

- Grammar of clock constraints:

$$\varphi ::= x \leq C \mid x < C \mid x \geq C \mid x > C \mid \varphi \wedge \varphi$$

  s.t. C positive integer values.
  $\Longrightarrow$ allow only comparison of a clock with a constant

- clock interpretation: ν

$$X = \langle x, y, z \rangle, \quad \nu = \langle 1.0, 1.5, 0 \rangle$$

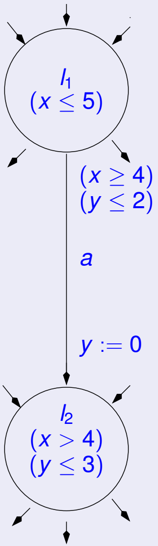
- clock interpretation ν after δ time: $\nu + \delta$

$$\delta = 0.2, \quad \nu + \delta = \langle 1.2, 1.7, 0.2 \rangle$$

- clock interpretation ν after reset λ: $\nu[\lambda]$

$$\lambda = \{y\}, \quad \nu[y := 0] = \langle 1.0, 0, 0 \rangle$$

A state for a timed automaton is a pair $\langle l, \nu \rangle$,
where l is a location and ν is a clock interpretation



Clock constraints and clock interpretations

- Grammar of clock constraints:

$$\varphi ::= x \leq C \mid x < C \mid x \geq C \mid x > C \mid \varphi \wedge \varphi$$

  s.t. C positive integer values.
  $\Longrightarrow$ allow only comparison of a clock with a constant

- clock interpretation: ν

$$X = \langle x, y, z \rangle, \quad \nu = \langle 1.0, 1.5, 0 \rangle$$

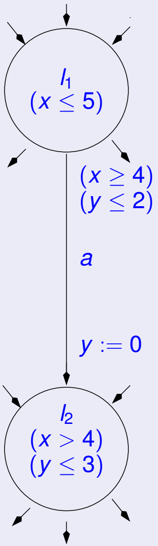
- clock interpretation ν after δ time: $\nu + \delta$

$$\delta = 0.2, \quad \nu + \delta = \langle 1.2, 1.7, 0.2 \rangle$$

- clock interpretation ν after reset λ: $\nu[\lambda]$

$$\lambda = \{y\}, \quad \nu[y := 0] = \langle 1.0, 0, 0 \rangle$$

A state for a timed automaton is a pair $\langle l, \nu \rangle$,
where l is a location and ν is a clock interpretation



Clock constraints and clock interpretations

- Grammar of clock constraints:

$$\varphi ::= x \leq C \mid x < C \mid x \geq C \mid x > C \mid \varphi \wedge \varphi$$

  s.t. C positive integer values.
  $\Longrightarrow$ allow only comparison of a clock with a constant
- clock interpretation: ν

$$X = \langle x, y, z \rangle, \quad \nu = \langle 1.0, 1.5, 0 \rangle$$

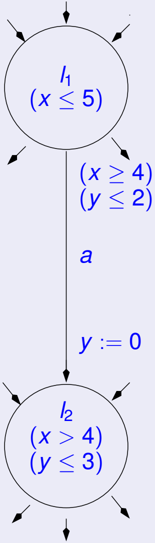
- clock interpretation ν after δ time: $\nu + \delta$

$$\delta = 0.2, \quad \nu + \delta = \langle 1.2, 1.7, 0.2 \rangle$$

- clock interpretation ν after reset λ: $\nu[\lambda]$

$$\lambda = \{y\}, \quad \nu[y := 0] = \langle 1.0, 0, 0 \rangle$$

A state for a timed automaton is a pair $\langle l, \nu \rangle$,
where l is a location and ν is a clock interpretation

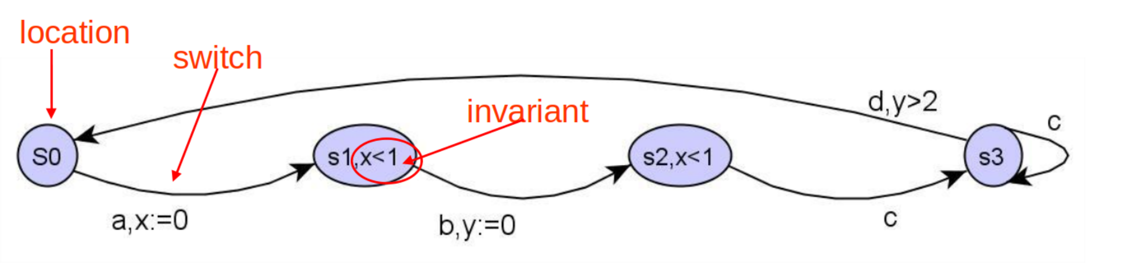


Remark: why integer constants in clock constraints?

The constant in clock constraints are assumed to be integer w.l.o.g.:

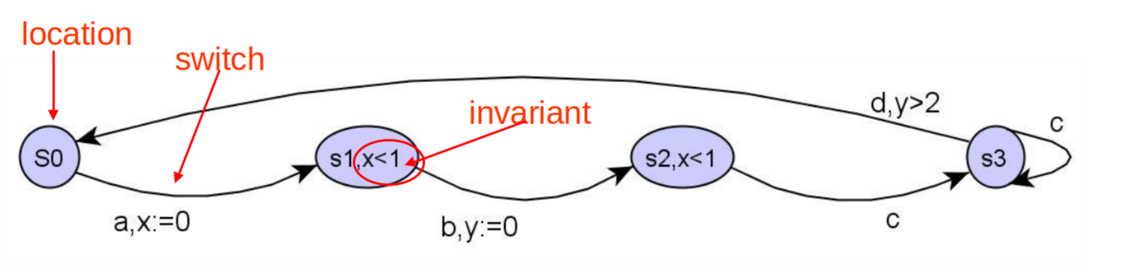
- if rationals, multiply them for their greatest common denominator, and change the time unit accordingly
- in practice, multiply by 10^k (resp 2^k), k being the number of precision digits (resp. bits), and change the time unit accordingly
  Ex: 1.345, 0.78, 102.32 seconds
  $\Longrightarrow 1,345,\ 780,\ 102,320$ milliseconds

Example



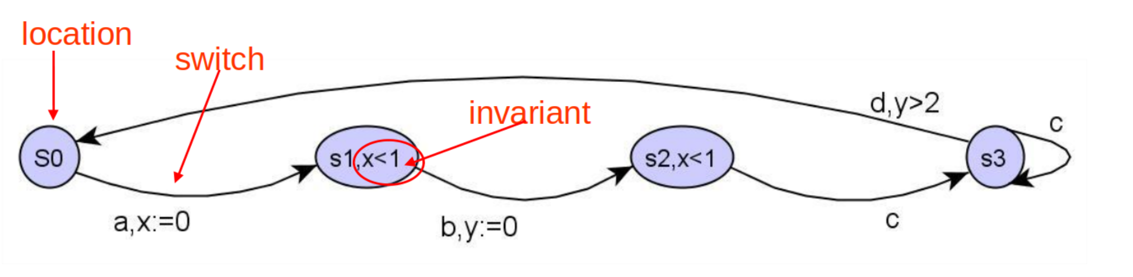
- clocks $\{x, y\}$ can be set/reset independently
- x is reset to 0 from s_0 to s_1 on a
- switches b and c happen within 1 time-unit from a because of constraints in s_1 and s_2
- delay between b and the following d is > 2
- no explicit bounds on time difference between event $c - d$

Example



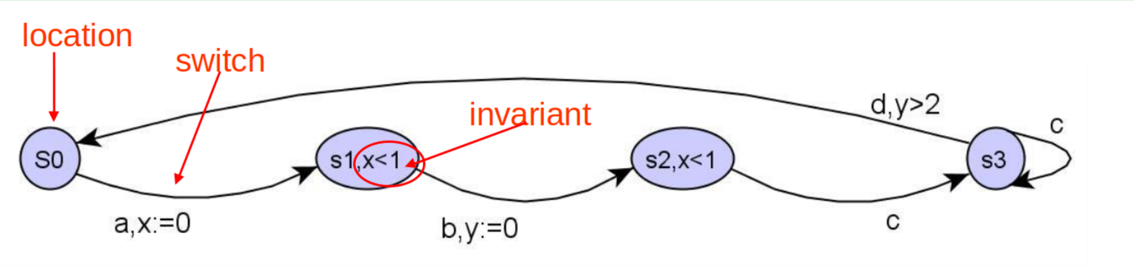
- clocks $\{x, y\}$ can be set/reset independently
- x is reset to 0 from s_0 to s_1 on a
- switches b and c happen within 1 time-unit from a because of constraints in s_1 and s_2
- delay between b and the following d is > 2
- no explicit bounds on time difference between event $c - d$

Example



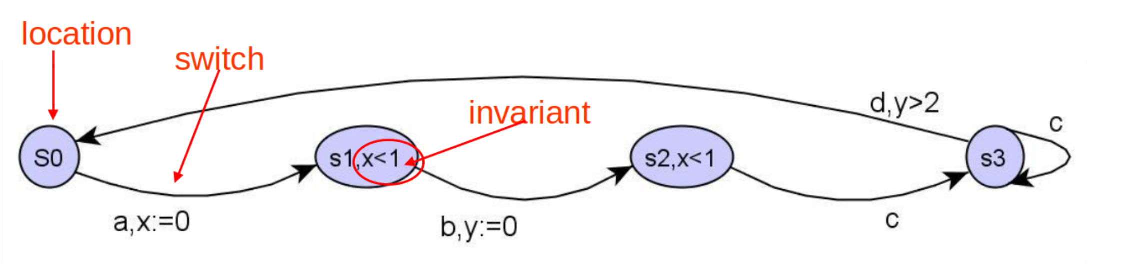
- clocks $\{x, y\}$ can be set/reset independently
- x is reset to 0 from s_0 to s_1 on a
- switches b and c happen within 1 time-unit from a because of constraints in s_1 and s_2
- delay between b and the following d is > 2
- no explicit bounds on time difference between event $c - d$

Example



- clocks $\{x, y\}$ can be set/reset independently
- x is reset to 0 from s_0 to s_1 on a
- switches b and c happen within 1 time-unit from a because of constraints in s_1 and s_2
- delay between b and the following d is > 2
- no explicit bounds on time difference between event $c - d$

Example



- clocks $\{x, y\}$ can be set/reset independently
- x is reset to 0 from s_0 to s_1 on a
- switches b and c happen within 1 time-unit from a because of constraints in s_1 and s_2
- delay between b and the following d is > 2
- no explicit bounds on time difference between event $c - d$

Outline

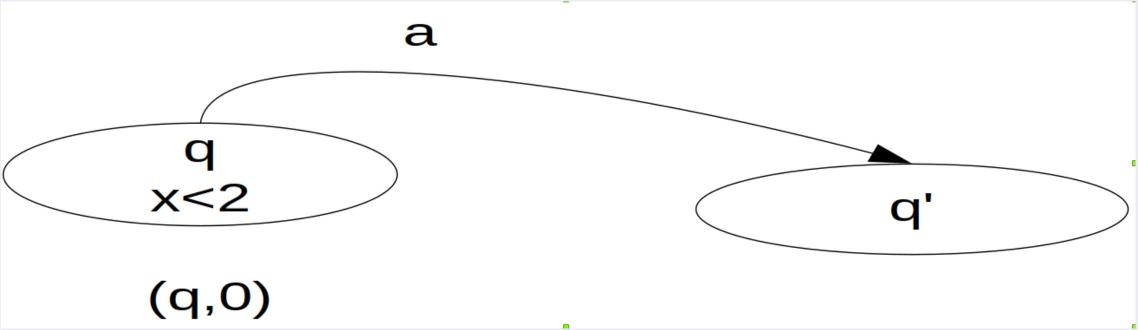
Timed Automata: Semantics

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

$$S_A \stackrel{\text{def}}{=} \langle Q, Q^0, \rightarrow, \Sigma \rangle$$

- Q: $\{\langle l, \nu \rangle\}$ s.t. l location and ν clock evaluation
- Q^0: $\{\langle l, \nu \rangle\}$ s.t. $l \in L^0$ location and $\nu(X) = 0$
- $\rightarrow$:
  - state change due to location switch
  - state change due to time elapse
- Σ: set of labels of $\Sigma \cup \mathbb{Q}^+$

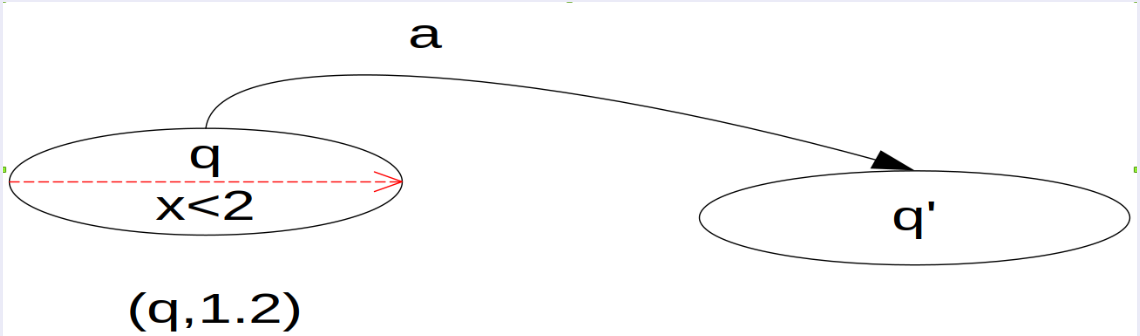
State change in transition system



Initial State

- $\langle q, 0 \rangle$
- Initial state

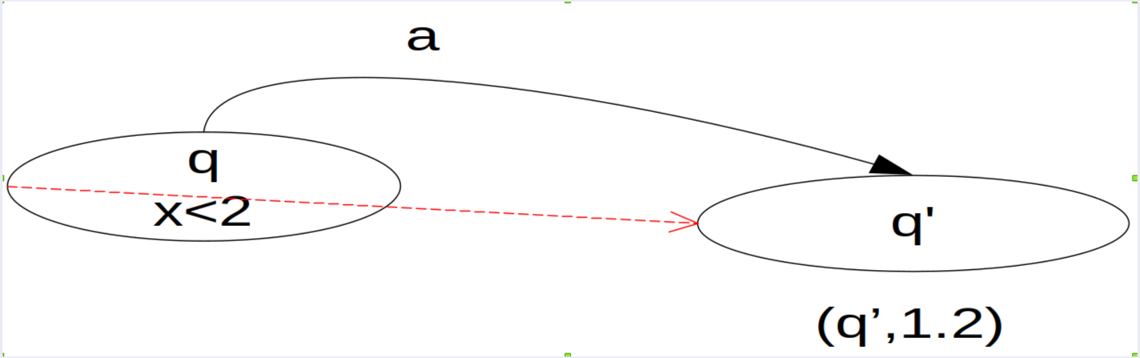
State change in transition system



Time elapse

- $\langle q, 0\rangle \xrightarrow{1.2} \langle q, 1.2\rangle$
- state change due to elapse of time

State change in transition system

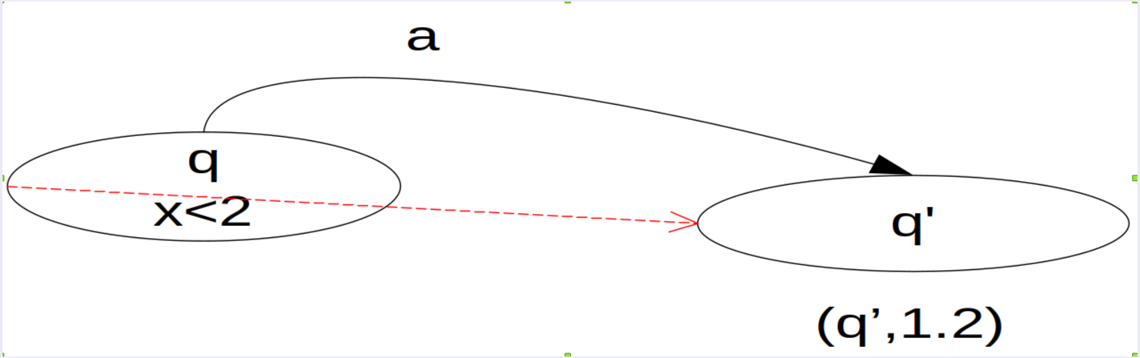


Time Elapse, Switch and their Concatenation

- $\langle q, 0\rangle \xrightarrow{1.2} \langle q, 1.2\rangle \xrightarrow{a} \langle q', 1.2\rangle$ "wait δ; switch;"

$\Longrightarrow$ $\langle q, 0\rangle \xrightarrow{1.2+a} \langle q', 1.2\rangle$ "wait δ and switch;"

State change in transition system

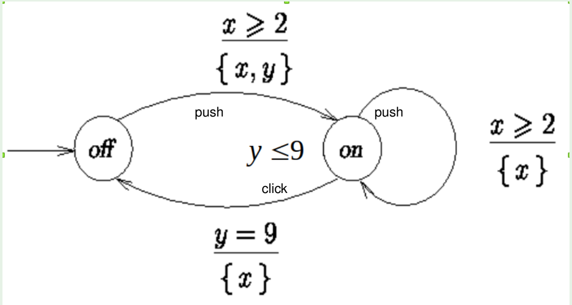


Time Elapse, Switch and their Concatenation

- $\langle q, 0\rangle \xrightarrow{1.2} \langle q, 1.2\rangle \xrightarrow{a} \langle q', 1.2\rangle$ "wait δ; switch;"

$\Longrightarrow$ $\langle q, 0\rangle \xrightarrow{1.2+a} \langle q', 1.2\rangle$ "wait δ and switch;"

Example

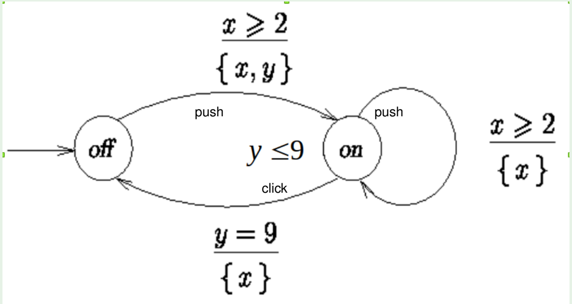


- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Example execution

$\langle \mathit{off}, 0, 0\rangle \xrightarrow{3.5} \langle \mathit{off}, 3.5, 3.5\rangle \xrightarrow{push} \langle \mathit{on}, 0, 0\rangle \xrightarrow{3.14} \langle \mathit{on}, 3.14, 3.14\rangle$
$\xrightarrow{push} \langle \mathit{on}, 0, 3.14\rangle \xrightarrow{3} \langle \mathit{on}, 3, 6.14\rangle \xrightarrow{2.86} \langle \mathit{on}, 5.86, 9\rangle \xrightarrow{click} \langle \mathit{off}, 0, 9\rangle$

Example

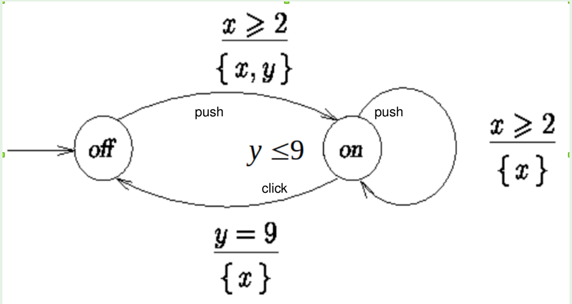


- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Example execution

$\langle \mathit{off}, 0, 0 \rangle \xrightarrow{3.5} \langle \mathit{off}, 3.5, 3.5 \rangle \xrightarrow{push} \langle \mathit{on}, 0, 0 \rangle \xrightarrow{3.14} \langle \mathit{on}, 3.14, 3.14 \rangle$
$\xrightarrow{push} \langle \mathit{on}, 0, 3.14 \rangle \xrightarrow{3} \langle \mathit{on}, 3, 6.14 \rangle \xrightarrow{2.86} \langle \mathit{on}, 5.86, 9 \rangle \xrightarrow{click} \langle \mathit{off}, 0, 9 \rangle$

Example

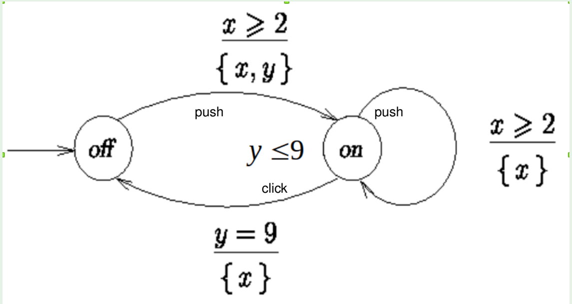


- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Example execution

$\langle off, 0, 0\rangle \xrightarrow{3.5} \langle off, 3.5, 3.5\rangle \xrightarrow{push} \langle on, 0, 0\rangle \xrightarrow{3.14} \langle on, 3.14, 3.14\rangle$
$\xrightarrow{push} \langle on, 0, 3.14\rangle \xrightarrow{3} \langle on, 3, 6.14\rangle \xrightarrow{2.86} \langle on, 5.86, 9\rangle \xrightarrow{click} \langle off, 0, 9\rangle$

Example

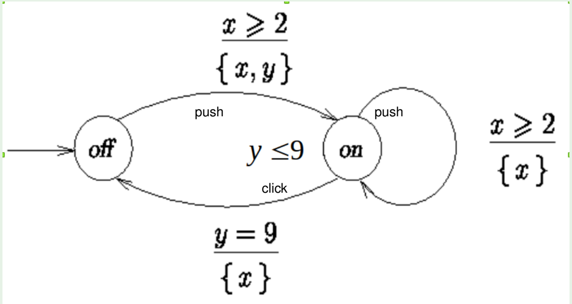


- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Example execution

$\langle \mathit{off}, 0, 0 \rangle \xrightarrow{3.5} \langle \mathit{off}, 3.5, 3.5 \rangle \xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 0 \rangle \xrightarrow{3.14} \langle \mathit{on}, 3.14, 3.14 \rangle$
$\xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 3.14 \rangle \xrightarrow{3} \langle \mathit{on}, 3, 6.14 \rangle \xrightarrow{2.86} \langle \mathit{on}, 5.86, 9 \rangle \xrightarrow{\mathit{click}} \langle \mathit{off}, 0, 9 \rangle$

Example

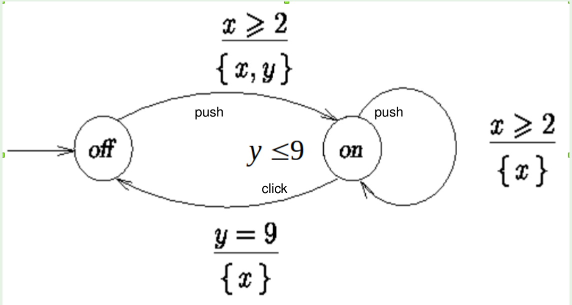


- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Example execution

$\langle \mathit{off}, 0, 0\rangle \xrightarrow{3.5} \langle \mathit{off}, 3.5, 3.5\rangle \xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 0\rangle \xrightarrow{3.14} \langle \mathit{on}, 3.14, 3.14\rangle$
$\xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 3.14\rangle \xrightarrow{3} \langle \mathit{on}, 3, 6.14\rangle \xrightarrow{2.86} \langle \mathit{on}, 5.86, 9\rangle \xrightarrow{\mathit{click}} \langle \mathit{off}, 0, 9\rangle$

Example

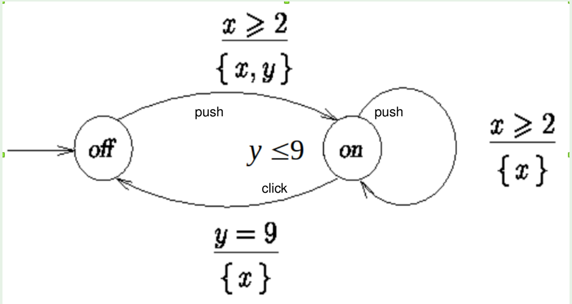


- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Example execution

$\langle \mathit{off}, 0, 0\rangle \xrightarrow{3.5} \langle \mathit{off}, 3.5, 3.5\rangle \xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 0\rangle \xrightarrow{3.14} \langle \mathit{on}, 3.14, 3.14\rangle$
$\xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 3.14\rangle \xrightarrow{3} \langle \mathit{on}, 3, 6.14\rangle \xrightarrow{2.86} \langle \mathit{on}, 5.86, 9\rangle \xrightarrow{\mathit{click}} \langle \mathit{off}, 0, 9\rangle$

Example

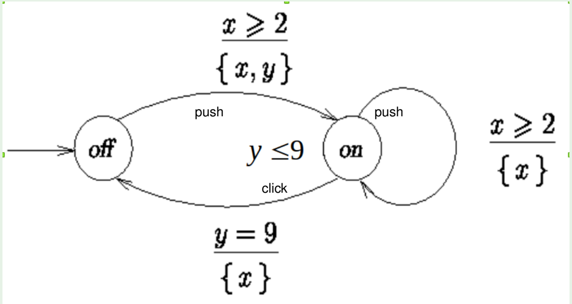


- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Example execution

$\langle \mathit{off}, 0, 0\rangle \xrightarrow{3.5} \langle \mathit{off}, 3.5, 3.5\rangle \xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 0\rangle \xrightarrow{3.14} \langle \mathit{on}, 3.14, 3.14\rangle$
$\xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 3.14\rangle \xrightarrow{3} \langle \mathit{on}, 3, 6.14\rangle \xrightarrow{2.86} \langle \mathit{on}, 5.86, 9\rangle \xrightarrow{\mathit{click}} \langle \mathit{off}, 0, 9\rangle$

Example

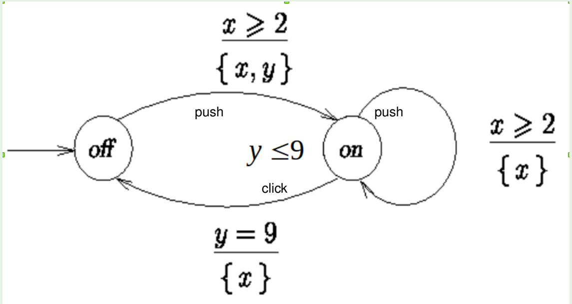


- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Example execution

$\langle off, 0, 0\rangle \xrightarrow{3.5} \langle off, 3.5, 3.5\rangle \xrightarrow{push} \langle on, 0, 0\rangle \xrightarrow{3.14} \langle on, 3.14, 3.14\rangle$
$\xrightarrow{push} \langle on, 0, 3.14\rangle \xrightarrow{3} \langle on, 3, 6.14\rangle \xrightarrow{2.86} \langle on, 5.86, 9\rangle \xrightarrow{click} \langle off, 0, 9\rangle$

Example



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

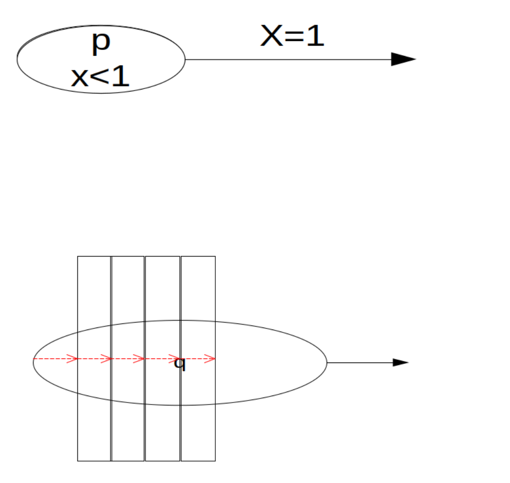
Example execution

$\langle \mathit{off}, 0, 0\rangle \xrightarrow{3.5} \langle \mathit{off}, 3.5, 3.5\rangle \xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 0\rangle \xrightarrow{3.14} \langle \mathit{on}, 3.14, 3.14\rangle$
$\xrightarrow{\mathit{push}} \langle \mathit{on}, 0, 3.14\rangle \xrightarrow{3} \langle \mathit{on}, 3, 6.14\rangle \xrightarrow{2.86} \langle \mathit{on}, 5.86, 9\rangle \xrightarrow{\mathit{click}} \langle \mathit{off}, 0, 9\rangle$

Remark: Non-Zenoness

Beware of Zeno! (paradox)

- When the invariant is violated some edge must be enabled
- Automata should admit the possibility of time to diverge



Outline

Combination of Timed Automata

- Complex system = product of interacting systems
- Let $A_1 \stackrel{\text{def}}{=} \langle L_1, L_1^0, \Sigma_1, X_1, \Phi_1(X_1), E_1 \rangle$, $A_2 \stackrel{\text{def}}{=} \langle L_2, L_2^0, \Sigma_2, X_2, \Phi_2(X_2), E_2 \rangle$
- Product: $A_1 || A_2 \stackrel{\text{def}}{=} \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, \Phi_1(X_1) \cup \Phi_2(X_2), E_1 || E_2 \rangle$
- Transition iff:
  - Label a belongs to both alphabets $\Longrightarrow$ synchronized
    blocking synchronization: a-labeled switches cannot be shot alone
  - Label a only in the alphabet of $A_1 \Longrightarrow$ asynchronized
  - Label a only in the alphabet of $A_2 \Longrightarrow$ asynchronized

Combination of Timed Automata

- Complex system = product of interacting systems
- Let $A_1 \stackrel{\text{def}}{=} \langle L_1, L_1^0, \Sigma_1, X_1, \Phi_1(X_1), E_1 \rangle$, $A_2 \stackrel{\text{def}}{=} \langle L_2, L_2^0, \Sigma_2, X_2, \Phi_2(X_2), E_2 \rangle$
- Product: $A_1 || A_2 \stackrel{\text{def}}{=} \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, \Phi_1(X_1) \cup \Phi_2(X_2), E_1 || E_2 \rangle$
- Transition iff:
  - Label a belongs to both alphabets $\Longrightarrow$ synchronized
    blocking synchronization: a-labeled switches cannot be shot alone
  - Label a only in the alphabet of $A_1 \Longrightarrow$ asynchronized
  - Label a only in the alphabet of $A_2 \Longrightarrow$ asynchronized

Combination of Timed Automata

- Complex system = product of interacting systems
- Let $A_1 \stackrel{\text{def}}{=} \langle L_1, L_1^0, \Sigma_1, X_1, \Phi_1(X_1), E_1 \rangle$, $A_2 \stackrel{\text{def}}{=} \langle L_2, L_2^0, \Sigma_2, X_2, \Phi_2(X_2), E_2 \rangle$
- Product: $A_1 || A_2 \stackrel{\text{def}}{=} \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, \Phi_1(X_1) \cup \Phi_2(X_2), E_1 || E_2 \rangle$
- Transition iff:
  - Label a belongs to both alphabets $\Longrightarrow$ synchronized
    blocking synchronization: a-labeled switches cannot be shot alone
  - Label a only in the alphabet of $A_1 \Longrightarrow$ asynchronized
  - Label a only in the alphabet of $A_2 \Longrightarrow$ asynchronized

Combination of Timed Automata

- Complex system = product of interacting systems
- Let $A_1 \stackrel{\text{def}}{=} \langle L_1, L_1^0, \Sigma_1, X_1, \Phi_1(X_1), E_1 \rangle$, $A_2 \stackrel{\text{def}}{=} \langle L_2, L_2^0, \Sigma_2, X_2, \Phi_2(X_2), E_2 \rangle$
- Product: $A_1 || A_2 \stackrel{\text{def}}{=} \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, \Phi_1(X_1) \cup \Phi_2(X_2), E_1 || E_2 \rangle$
- Transition iff:
  - Label a belongs to both alphabets $\Longrightarrow$ synchronized
    blocking synchronization: a-labeled switches cannot be shot alone
  - Label a only in the alphabet of $A_1 \Longrightarrow$ asynchronized
  - Label a only in the alphabet of $A_2 \Longrightarrow$ asynchronized

Combination of Timed Automata

- Complex system = product of interacting systems
- Let $A_1 \stackrel{\text{def}}{=} \langle L_1, L_1^0, \Sigma_1, X_1, \Phi_1(X_1), E_1 \rangle$, $A_2 \stackrel{\text{def}}{=} \langle L_2, L_2^0, \Sigma_2, X_2, \Phi_2(X_2), E_2 \rangle$
- Product: $A_1 || A_2 \stackrel{\text{def}}{=} \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, \Phi_1(X_1) \cup \Phi_2(X_2), E_1 || E_2 \rangle$
- Transition iff:
  - Label a belongs to both alphabets $\Longrightarrow$ synchronized
    blocking synchronization: a-labeled switches cannot be shot alone
  - Label a only in the alphabet of $A_1 \Longrightarrow$ asynchronized
  - Label a only in the alphabet of $A_2 \Longrightarrow$ asynchronized

Combination of Timed Automata

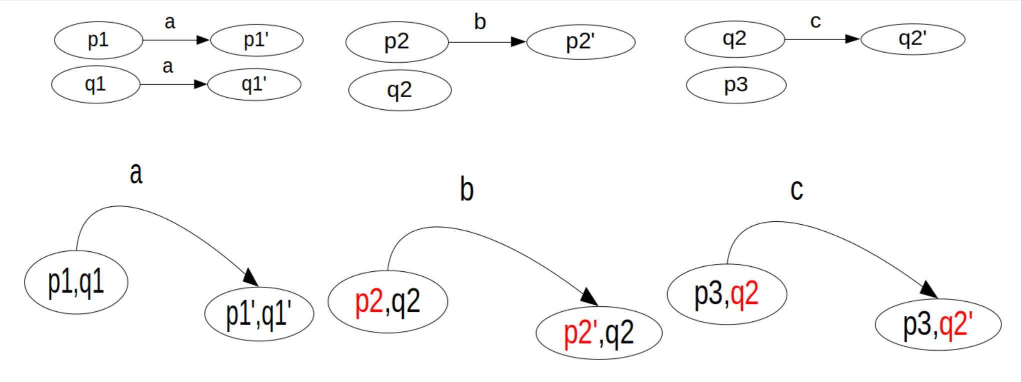
- Complex system = product of interacting systems
- Let $A_1 \stackrel{\text{def}}{=} \langle L_1, L_1^0, \Sigma_1, X_1, \Phi_1(X_1), E_1 \rangle$, $A_2 \stackrel{\text{def}}{=} \langle L_2, L_2^0, \Sigma_2, X_2, \Phi_2(X_2), E_2 \rangle$
- Product: $A_1 || A_2 \stackrel{\text{def}}{=} \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, \Phi_1(X_1) \cup \Phi_2(X_2), E_1 || E_2 \rangle$
- Transition iff:
  - Label a belongs to both alphabets $\Longrightarrow$ synchronized
    blocking synchronization: a-labeled switches cannot be shot alone
  - Label a only in the alphabet of $A_1 \Longrightarrow$ asynchronized
  - Label a only in the alphabet of $A_2 \Longrightarrow$ asynchronized

Combination of Timed Automata

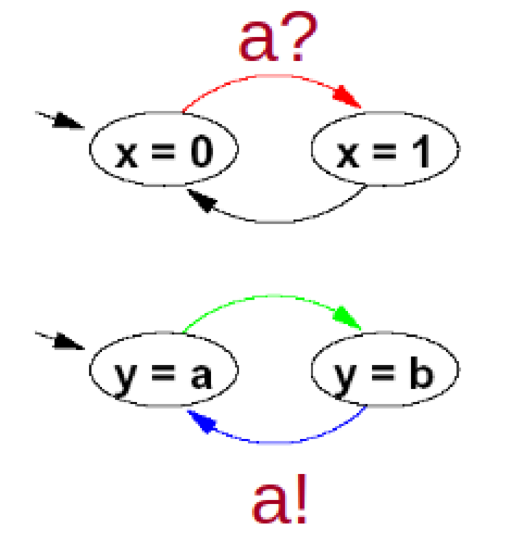
- Complex system = product of interacting systems
- Let $A_1 \stackrel{\text{def}}{=} \langle L_1, L_1^0, \Sigma_1, X_1, \Phi_1(X_1), E_1 \rangle$, $A_2 \stackrel{\text{def}}{=} \langle L_2, L_2^0, \Sigma_2, X_2, \Phi_2(X_2), E_2 \rangle$
- Product: $A_1 || A_2 \stackrel{\text{def}}{=} \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, \Phi_1(X_1) \cup \Phi_2(X_2), E_1 || E_2 \rangle$
- Transition iff:
  - Label a belongs to both alphabets $\Longrightarrow$ synchronized
    blocking synchronization: a-labeled switches cannot be shot alone
  - Label a only in the alphabet of $A_1 \Longrightarrow$ asynchronized
  - Label a only in the alphabet of $A_2 \Longrightarrow$ asynchronized

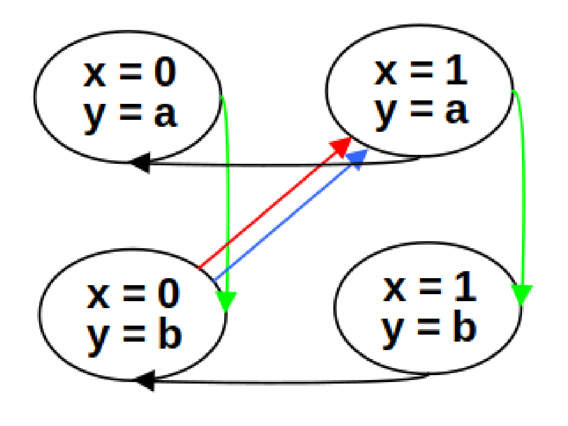
Transition Product

$\Sigma_1 \stackrel{\text{def}}{=} \{a, b\}$
$\Sigma_2 \stackrel{\text{def}}{=} \{a, c\}$



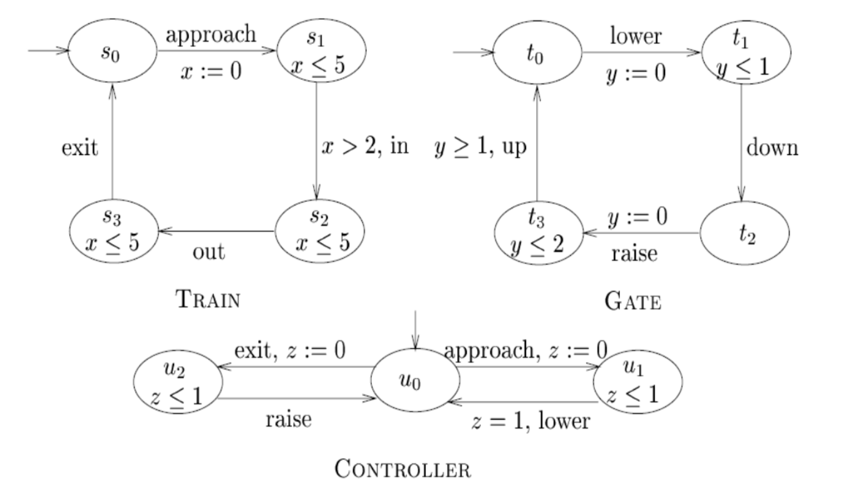
Transition Product: Example





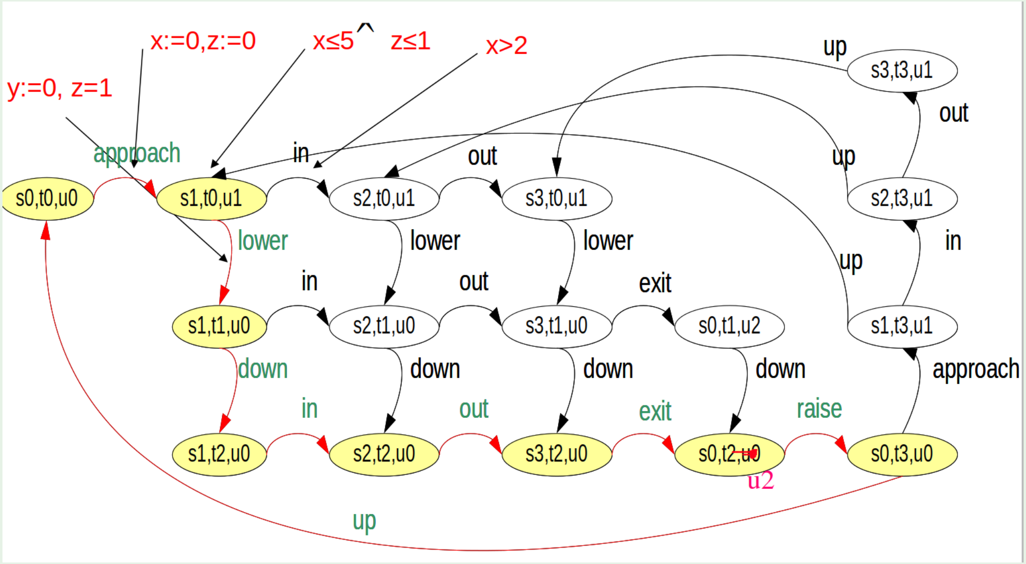
Courtesy of prof. Marco di Natale, Scuola S.Anna, Pisa, Italy

Example: Train-gate controller [Alur CAV'99]



Desired property: $G(s_2 \rightarrow t_2)$

Train-gate controller: Product



Outline

Outline

Reachability Analysis

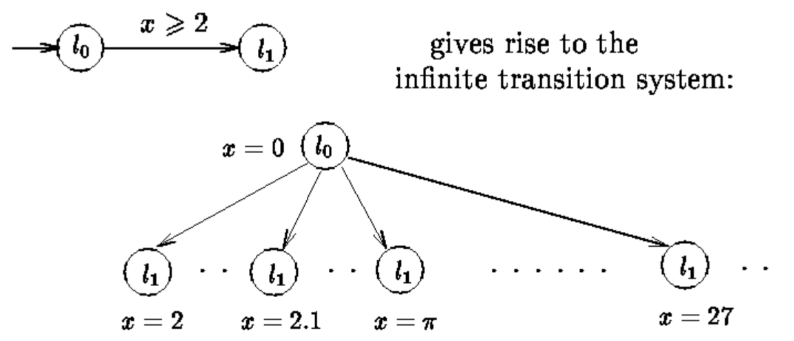
- Verification of safety requirement: reachability problem
- Input: a timed automaton A and a set of target locations $L^F \subseteq L$
- Problem: Determining whether L^F is reachable in a timed automaton A
- A location l of A is reachable if some state q with location component l is a reachable state of the transition system S_A

Timed/hybrid Systems: problem

Problem

The system S_A associated to A has infinitely-many states & symbols.

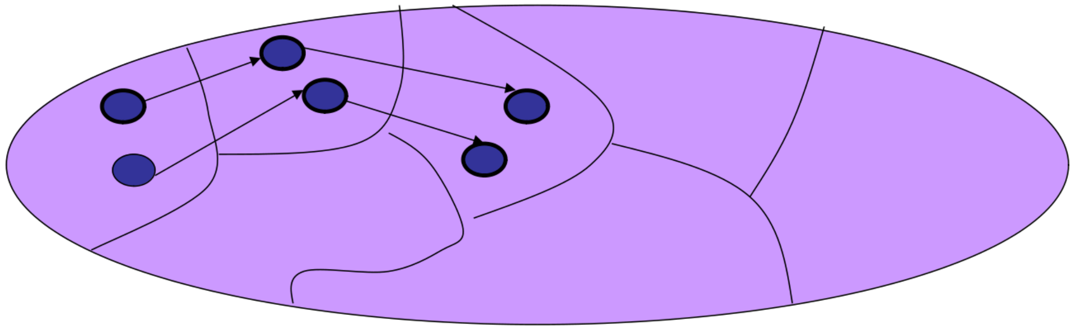
- Is finite state analysis possible?
- Is reachability problem decidable?



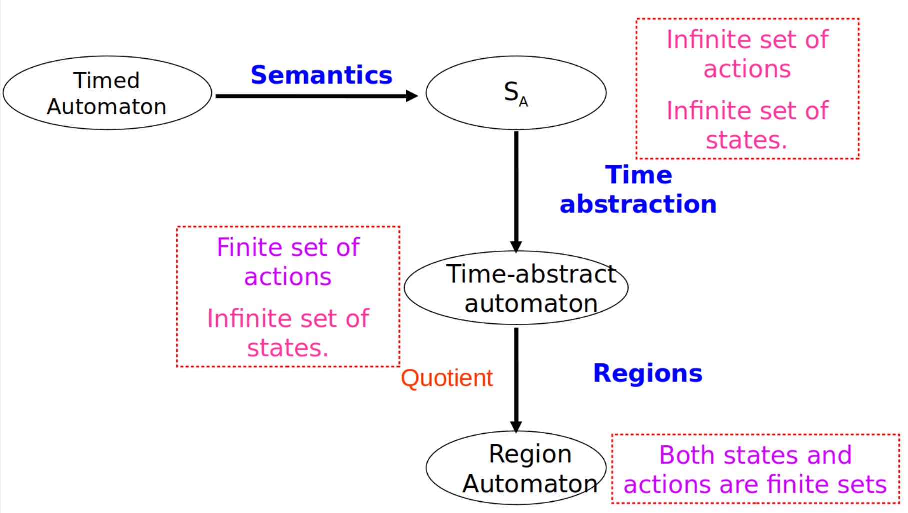
Idea: Finite Partitioning

Goal

Partition the state space into finitely-many equivalence classes, so that equivalent states exhibit (bi)similar behaviors



Reachability analysis



Timed Vs Time-Abstract Relations

Idea

Infinite transition system associated with a timed/hybrid automaton A:

- S_A: Labels on continuous steps are delays in $\mathbb{Q}^+$
- U_A (time-abstract): actual delays are suppressed
  $\Longrightarrow$ all continuous steps have same label
- from "wait δ and switch" to "wait (sometime) and switch"

Time-abstract transition system U_A

U_A (time-abstract): actual delays are suppressed

- Only the change due to location switch is stated explicitly

$\Longrightarrow$ Cuts system into finitely many labels

- U_A (instead of S_A) allows for capturing untimed properties (e.g., reachability, safety)

Example

A: ("wait δ; switch;")

$\langle l_0, 0, 0\rangle \xrightarrow{1.2} \langle l_0, 1.2, 1.2\rangle \xrightarrow{a} \langle l_1, 0, 1.2\rangle \xrightarrow{0.7} \langle l_1, 0.7, 1.9\rangle \xrightarrow{b} \langle l_2, 0.7, 0\rangle$

S_A: ("wait δ and switch;")

$\langle l_0, 0, 0\rangle \xrightarrow{1.2+a} \langle l_1, 0, 1.2\rangle \xrightarrow{0.7+b} \langle l_2, 0.7, 0\rangle$

U_A: ("wait (sometime) and switch;")

$\langle l_0, 0, 0\rangle \xrightarrow{a} \langle l_1, 0, 1.2\rangle \xrightarrow{b} \langle l_2, 0.7, 0\rangle$

Time-abstract transition system U_A

U_A (time-abstract): actual delays are suppressed

- Only the change due to location switch is stated explicitly

$\Longrightarrow$ Cuts system into finitely many labels

- U_A (instead of S_A) allows for capturing untimed properties (e.g., reachability, safety)

Example

A: ("wait δ; switch;")

$\langle l_0, 0, 0\rangle \xrightarrow{1.2} \langle l_0, 1.2, 1.2\rangle \xrightarrow{a} \langle l_1, 0, 1.2\rangle \xrightarrow{0.7} \langle l_1, 0.7, 1.9\rangle \xrightarrow{b} \langle l_2, 0.7, 0\rangle$

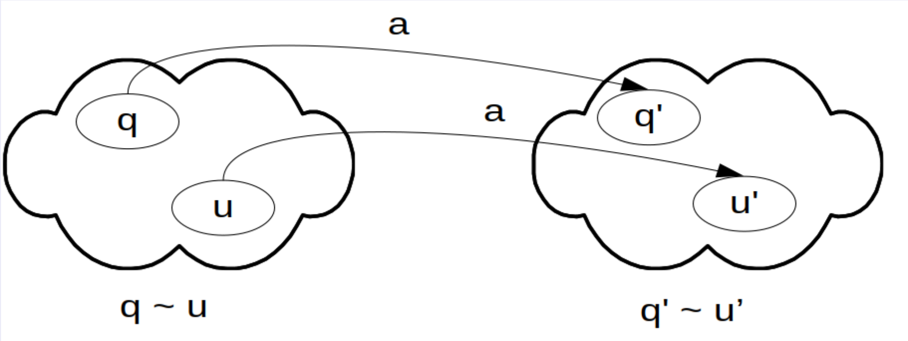
S_A: ("wait δ and switch;")

$\langle l_0, 0, 0\rangle \xrightarrow{1.2+a} \langle l_1, 0, 1.2\rangle \xrightarrow{0.7+b} \langle l_2, 0.7, 0\rangle$

U_A: ("wait (sometime) and switch;")

$\langle l_0, 0, 0\rangle \xrightarrow{a} \langle l_1, 0, 1.2\rangle \xrightarrow{b} \langle l_2, 0.7, 0\rangle$

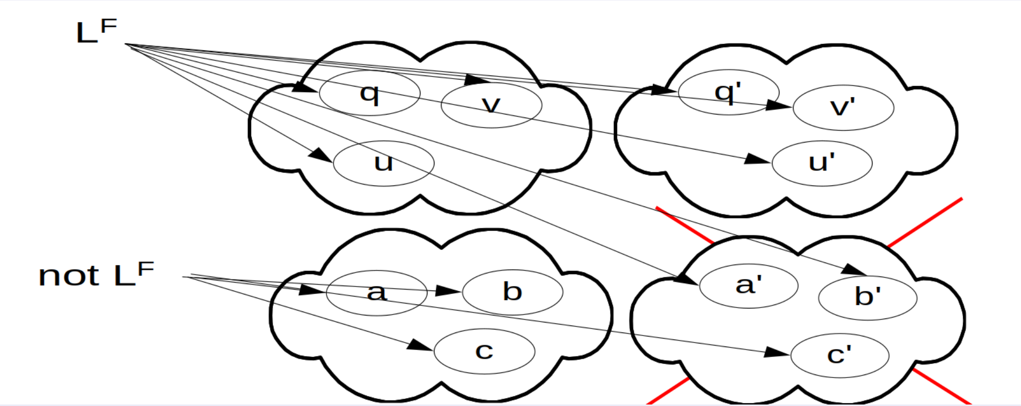
Stable quotients



Idea: Collapse states which are equivalent modulo “wait & switch”

- Cut to finitely many states
- Stable equivalence relation
- Quotient of U_A = transition system $[U_A]$

L^F-sensitive equivalence relation



All equivalent states in a class belong to either L^F or not L^F

- E.g.: states with different labels cannot be equivalent

Stable Quotient: Intuitive example

Task: plan trip from DISI to VR train station

"Take the next #5 bus to TN train station and then the 6pm train to VR"

- Constraints:
  - It is 5.18pm
  - Train to VR leaves at TN train station at 6.00pm
  - it takes 3 minutes to walk from DISI to BUS stop
  - Bus #5 passes at 5.20pm or at 5.40pm
  - Bus #5 takes 15 minutes to reach TN train station
  - it takes 2 minutes to walk from BUS stop to TN train station
- Time-Abstract plan (U_A):
  "walk to bus stop; take 5.40 #5 bus to TN train-station stop;
  walk to train station; take the 6pm train to VR"
- Actual (implicit) plan (A):
  "wait δ_1; walk to bus stop; wait δ_2; take 5.40 #5 bus to TN train-station stop;
  wait δ_3 at bus stop; walk to train station; wait δ_4; take the 6pm train to VR"
  for some $\delta_1, \delta_2, \delta_3, \delta_4$ s.t $\delta_1 + \delta_2 = 19min$ and $\delta_3 + \delta_4 = 3min$
- All executions with distinct values of δ_i are bisimilar

Stable Quotient: Intuitive example

Task: plan trip from DISI to VR train station

"Take the next #5 bus to TN train station and then the 6pm train to VR"

- Constraints:
  - It is 5.18pm
  - Train to VR leaves at TN train station at 6.00pm
  - it takes 3 minutes to walk from DISI to BUS stop
  - Bus #5 passes at 5.20pm or at 5.40pm
  - Bus #5 takes 15 minutes to reach TN train station
  - it takes 2 minutes to walk from BUS stop to TN train station
- Time-Abstract plan (U_A):
  "walk to bus stop; take 5.40 #5 bus to TN train-station stop;
  walk to train station; take the 6pm train to VR"
- Actual (implicit) plan (A):
  "wait δ_1; walk to bus stop; wait δ_2; take 5.40 #5 bus to TN train-station stop;
  wait δ_3 at bus stop; walk to train station; wait δ_4; take the 6pm train to VR"
  for some $\delta_1, \delta_2, \delta_3, \delta_4$ s.t $\delta_1 + \delta_2 = 19min$ and $\delta_3 + \delta_4 = 3min$
- All executions with distinct values of δ_i are bisimilar

Stable Quotient: Intuitive example

Task: plan trip from DISI to VR train station

"Take the next #5 bus to TN train station and then the 6pm train to VR"

- Constraints:
  - It is 5.18pm
  - Train to VR leaves at TN train station at 6.00pm
  - it takes 3 minutes to walk from DISI to BUS stop
  - Bus #5 passes at 5.20pm or at 5.40pm
  - Bus #5 takes 15 minutes to reach TN train station
  - it takes 2 minutes to walk from BUS stop to TN train station
- Time-Abstract plan (U_A):
  "walk to bus stop; take 5.40 #5 bus to TN train-station stop;
  walk to train station; take the 6pm train to VR"
- Actual (implicit) plan (A):
  "wait δ_1; walk to bus stop; wait δ_2; take 5.40 #5 bus to TN train-station stop;
  wait δ_3 at bus stop; walk to train station; wait δ_4; take the 6pm train to VR"
  for some $\delta_1, \delta_2, \delta_3, \delta_4$ s.t $\delta_1 + \delta_2 = 19min$ and $\delta_3 + \delta_4 = 3min$
- All executions with distinct values of δ_i are bisimilar

Stable Quotient: Intuitive example

Task: plan trip from DISI to VR train station

"Take the next #5 bus to TN train station and then the 6pm train to VR"

- Constraints:
  - It is 5.18pm
  - Train to VR leaves at TN train station at 6.00pm
  - it takes 3 minutes to walk from DISI to BUS stop
  - Bus #5 passes at 5.20pm or at 5.40pm
  - Bus #5 takes 15 minutes to reach TN train station
  - it takes 2 minutes to walk from BUS stop to TN train station
- Time-Abstract plan (U_A):
  "walk to bus stop; take 5.40 #5 bus to TN train-station stop; walk to train station; take the 6pm train to VR"
- Actual (implicit) plan (A):
  "wait δ_1; walk to bus stop; wait δ_2; take 5.40 #5 bus to TN train-station stop; wait δ_3 at bus stop; walk to train station; wait δ_4; take the 6pm train to VR"
  for some $\delta_1, \delta_2, \delta_3, \delta_4$ s.t $\delta_1 + \delta_2 = 19min$ and $\delta_3 + \delta_4 = 3min$
- All executions with distinct values of δ_i are bisimilar

Outline

Region Equivalence over clock interpretation

Preliminary definitions & terminology

Given a clock x:

- $\lfloor x \rfloor$ is the integral part of x (ex: $\lfloor 3.7 \rfloor = 3$)
- fr(x) is the fractional part of x (ex: $fr(3.7) = 0.7$)
- C_x is the maximum constant occurring in clock constraints $x \bowtie C_x$

Region Equivalence: $\nu \cong \nu'$

Given a timed automaton A, two clock interpretations ν, ν' are region equivalent ($\nu \cong \nu'$) iff all the following conditions hold:

C1: For every clock x, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or $\lfloor \nu(x) \rfloor, \lfloor \nu'(x) \rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$,
$fr(\nu(x)) \leq fr(\nu(y))$ *iff* $fr(\nu'(x)) \leq fr(\nu'(y))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$
$fr(\nu(x)) = 0$ *iff* $fr(\nu'(x)) = 0$

Region Equivalence over clock interpretation

Preliminary definitions & terminology

Given a clock x:

- $\lfloor x \rfloor$ is the integral part of x (ex: $\lfloor 3.7 \rfloor = 3$)
- fr(x) is the fractional part of x (ex: $\mathrm{fr}(3.7) = 0.7$)
- C_x is the maximum constant occurring in clock constraints $x \bowtie C_x$

Region Equivalence: $\nu \cong \nu'$

Given a timed automaton A, two clock interpretations ν, ν' are region equivalent ($\nu \cong \nu'$) iff all the following conditions hold:

C1: For every clock x, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or $\lfloor \nu(x) \rfloor, \lfloor \nu'(x) \rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$,
$\mathrm{fr}(\nu(x)) \leq \mathrm{fr}(\nu(y))$ *iff* $\mathrm{fr}(\nu'(x)) \leq \mathrm{fr}(\nu'(y))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$
$\mathrm{fr}(\nu(x)) = 0$ *iff* $\mathrm{fr}(\nu'(x)) = 0$

Region Equivalence over clock interpretation

Preliminary definitions & terminology

Given a clock x:

- $\lfloor x \rfloor$ is the integral part of x (ex: $\lfloor 3.7 \rfloor = 3$)
- fr(x) is the fractional part of x (ex: $\mathrm{fr}(3.7) = 0.7$)
- C_x is the maximum constant occurring in clock constraints $x \bowtie C_x$

Region Equivalence: $\nu \cong \nu'$

Given a timed automaton A, two clock interpretations ν, ν' are region equivalent ($\nu \cong \nu'$) iff all the following conditions hold:

C1: For every clock x, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or $\lfloor \nu(x) \rfloor, \lfloor \nu'(x) \rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$,
$\mathrm{fr}(\nu(x)) \leq \mathrm{fr}(\nu(y))$ *iff* $\mathrm{fr}(\nu'(x)) \leq \mathrm{fr}(\nu'(y))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$
$\mathrm{fr}(\nu(x)) = 0$ *iff* $\mathrm{fr}(\nu'(x)) = 0$

Region Equivalence over clock interpretation

Preliminary definitions & terminology

Given a clock x:

- $\lfloor x \rfloor$ is the integral part of x (ex: $\lfloor 3.7 \rfloor = 3$)
- fr(x) is the fractional part of x (ex: $\mathrm{fr}(3.7) = 0.7$)
- C_x is the maximum constant occurring in clock constraints $x \bowtie C_x$

Region Equivalence: $\nu \cong \nu'$

Given a timed automaton A, two clock interpretations ν, ν' are region equivalent ($\nu \cong \nu'$) iff all the following conditions hold:

C1: For every clock x, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or $\lfloor \nu(x) \rfloor, \lfloor \nu'(x) \rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$,
$\mathrm{fr}(\nu(\mathrm{x})) \leq \mathrm{fr}(\nu(\mathrm{y}))$ *iff* $\mathrm{fr}(\nu'(\mathrm{x})) \leq \mathrm{fr}(\nu'(\mathrm{y}))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$
$\mathrm{fr}(\nu(\mathrm{x})) = 0$ *iff* $\mathrm{fr}(\nu'(\mathrm{x})) = 0$

Region Equivalence over clock interpretation

Preliminary definitions & terminology

Given a clock x:

- $\lfloor x \rfloor$ is the integral part of x (ex: $\lfloor 3.7 \rfloor = 3$)
- fr(x) is the fractional part of x (ex: $\mathrm{fr}(3.7) = 0.7$)
- C_x is the maximum constant occurring in clock constraints $x \bowtie C_x$

Region Equivalence: $\nu \cong \nu'$

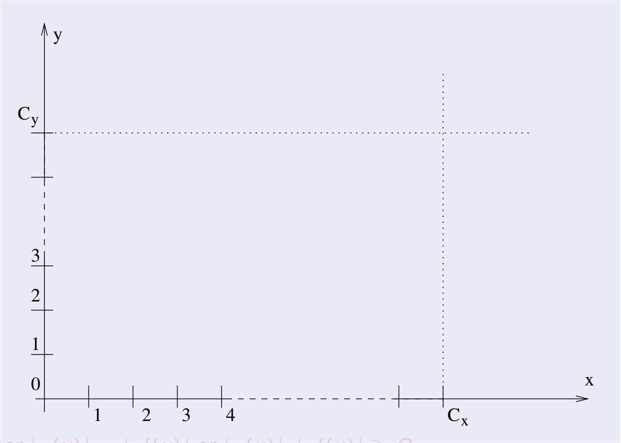
Given a timed automaton A, two clock interpretations ν, ν' are region equivalent ($\nu \cong \nu'$) iff all the following conditions hold:

C1: For every clock x, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or $\lfloor \nu(x) \rfloor, \lfloor \nu'(x) \rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$,
$\mathrm{fr}(\nu(\mathrm{x})) \leq \mathrm{fr}(\nu(\mathrm{y}))$ *iff* $\mathrm{fr}(\nu'(\mathrm{x})) \leq \mathrm{fr}(\nu'(\mathrm{y}))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$
$\mathrm{fr}(\nu(\mathrm{x})) = 0$ *iff* $\mathrm{fr}(\nu'(\mathrm{x})) = 0$

Conditions: C1 + C2 + C3

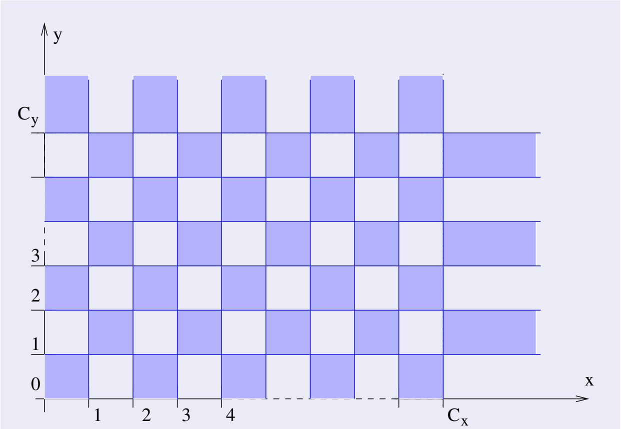


C1: For every clock x, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or $\lfloor \nu(x) \rfloor, \lfloor \nu'(x) \rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$, $\mathrm{fr}(\nu(x)) \leq \mathrm{fr}(\nu(y))$ *iff* $\mathrm{fr}(\nu'(x)) \leq \mathrm{fr}(\nu'(y))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$, $\mathrm{fr}(\nu(x)) = 0$ *iff* $\mathrm{fr}(\nu'(x)) = 0$

Conditions: C1 + C2 + C3

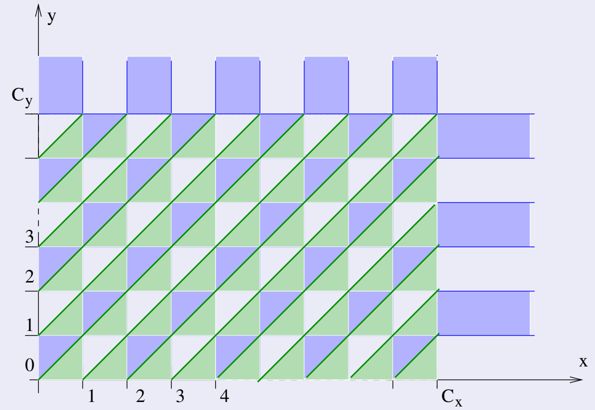


C1: For every clock x, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or $\lfloor \nu(x) \rfloor, \lfloor \nu'(x) \rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$, $\mathrm{fr}(\nu(x)) \leq \mathrm{fr}(\nu(y))$ *iff* $\mathrm{fr}(\nu'(x)) \leq \mathrm{fr}(\nu'(y))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$, $\mathrm{fr}(\nu(x)) = 0$ *iff* $\mathrm{fr}(\nu'(x)) = 0$

Conditions: C1 + C2 + C3

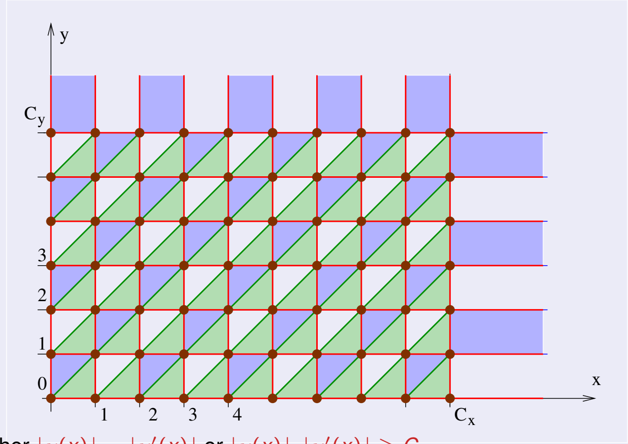


C1: For every clock x, either $\lfloor\nu(x)\rfloor = \lfloor\nu'(x)\rfloor$ or $\lfloor\nu(x)\rfloor, \lfloor\nu'(x)\rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$,
$\mathrm{fr}(\nu(x)) \leq \mathrm{fr}(\nu(y))$ *iff* $\mathrm{fr}(\nu'(x)) \leq \mathrm{fr}(\nu'(y))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$, $\mathrm{fr}(\nu(x)) = 0$ *iff* $\mathrm{fr}(\nu'(x)) = 0$

Conditions: C1 + C2 + C3

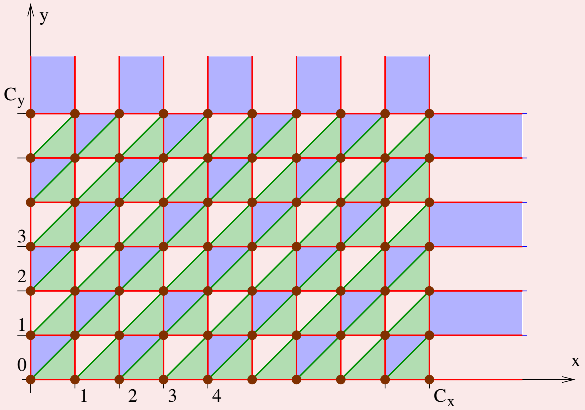


C1: For every clock x, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or $\lfloor \nu(x) \rfloor, \lfloor \nu'(x) \rfloor \geq C_x$

C2: For every clock pair x, y s.t. $\nu(x), \nu'(x) \leq C_x$ and $\nu(y), \nu'(y) \leq C_y$, $\mathrm{fr}(\nu(x)) \leq \mathrm{fr}(\nu(y))$ *iff* $\mathrm{fr}(\nu'(x)) \leq \mathrm{fr}(\nu'(y))$

C3: For every clock x s.t. $\nu(x), \nu'(x) \leq C_x$, $\mathrm{fr}(\nu(x)) = 0$ *iff* $\mathrm{fr}(\nu'(x)) = 0$

Regions, intuitive idea:

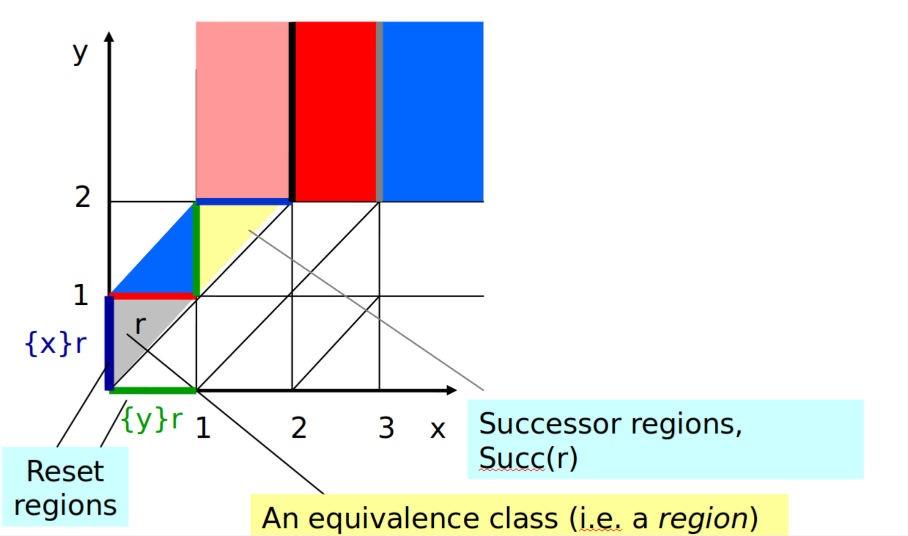


Intuition: $\nu \cong \nu'$ iff they satisfy the same set of constraints in the form

$$x_i < c,\ x_i > c,\ x_i = c,\ x_i - x_j < c,\ x_i - x_j > c,\ x_i - x_j = c$$

s.t. $c \leq C_{x_i}$

Region Operations



Properties of Regions

- The region equivalence relation $\cong$ is a time-abstract bisimulation:
  - Action transitions: if $\nu \cong \mu$ and $\langle l, \nu \rangle \xrightarrow{a} \langle l', \nu' \rangle$ for some l', ν', then there exists μ' s.t. $\nu' \cong \mu'$ and $\langle l, \mu \rangle \xrightarrow{a} \langle l', \mu' \rangle$
  - Wait transitions: if $\nu \cong \mu$, then for every $\delta \in \mathbb{Q}^+$ there exists $\delta' \in \mathbb{Q}^+$ s.t. $\nu + \delta \cong \mu + \delta'$

$\Longrightarrow$ If $\nu \cong \mu$, then $\langle l, \nu \rangle$ and $\langle l, \mu \rangle$ satisfy the same temporal-logic formulas

Properties of Regions

- The region equivalence relation $\cong$ is a time-abstract bisimulation:
  - Action transitions: if $\nu \cong \mu$ and $\langle l, \nu \rangle \xrightarrow{a} \langle l', \nu' \rangle$ for some l', ν', then there exists μ' s.t. $\nu' \cong \mu'$ and $\langle l, \mu \rangle \xrightarrow{a} \langle l', \mu' \rangle$
  - Wait transitions: if $\nu \cong \mu$, then for every $\delta \in \mathbb{Q}^+$ there exists $\delta' \in \mathbb{Q}^+$ s.t. $\nu + \delta \cong \mu + \delta'$

$\Longrightarrow$ If $\nu \cong \mu$, then $\langle l, \nu \rangle$ and $\langle l, \mu \rangle$ satisfy the same temporal-logic formulas

Properties of Regions

- The region equivalence relation $\cong$ is a time-abstract bisimulation:
  - Action transitions: if $\nu \cong \mu$ and $\langle l, \nu\rangle \xrightarrow{a} \langle l', \nu'\rangle$ for some l', ν', then there exists μ' s.t. $\nu' \cong \mu'$ and $\langle l, \mu\rangle \xrightarrow{a} \langle l', \mu'\rangle$
  - Wait transitions: if $\nu \cong \mu$, then for every $\delta \in \mathbb{Q}^+$ there exists $\delta' \in \mathbb{Q}^+$ s.t. $\nu + \delta \cong \mu + \delta'$

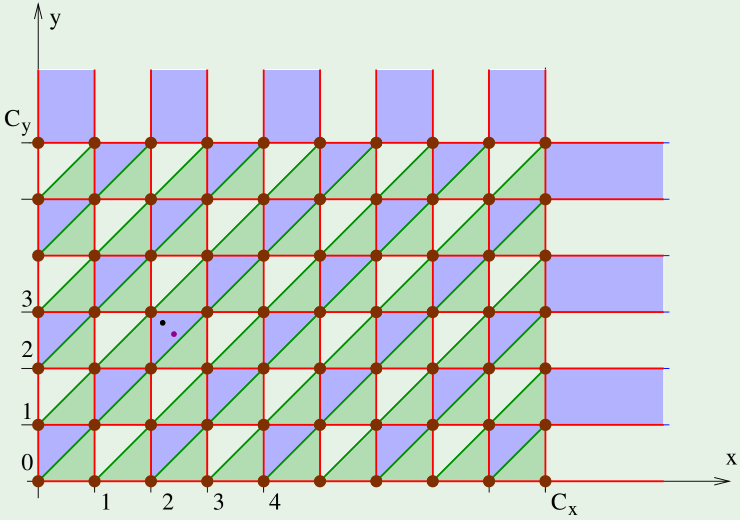
$\Longrightarrow$ If $\nu \cong \mu$, then $\langle l, \nu\rangle$ and $\langle l, \mu\rangle$ satisfy the same temporal-logic formulas

Properties of Regions

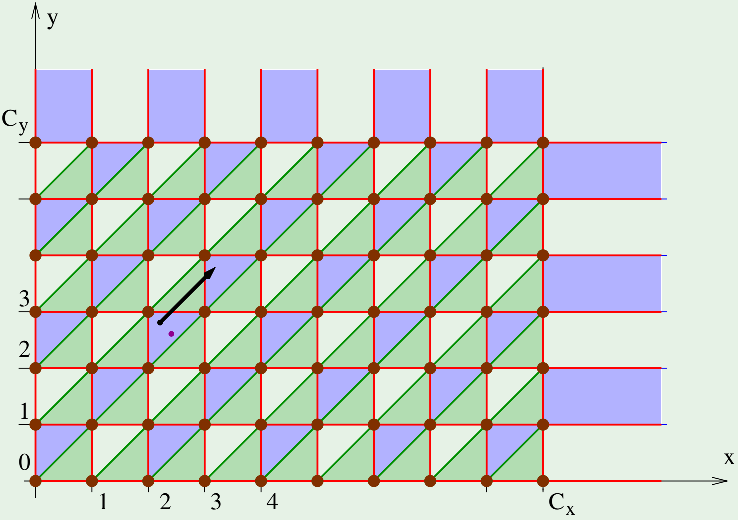
- The region equivalence relation $\cong$ is a time-abstract bisimulation:
  - Action transitions: if $\nu \cong \mu$ and $\langle l, \nu \rangle \xrightarrow{a} \langle l', \nu' \rangle$ for some l', ν', then there exists μ' s.t. $\nu' \cong \mu'$ and $\langle l, \mu \rangle \xrightarrow{a} \langle l', \mu' \rangle$
  - Wait transitions: if $\nu \cong \mu$, then for every $\delta \in \mathbb{Q}^+$ there exists $\delta' \in \mathbb{Q}^+$ s.t. $\nu + \delta \cong \mu + \delta'$

$\Longrightarrow$ If $\nu \cong \mu$, then $\langle l, \nu \rangle$ and $\langle l, \mu \rangle$ satisfy the same temporal-logic formulas

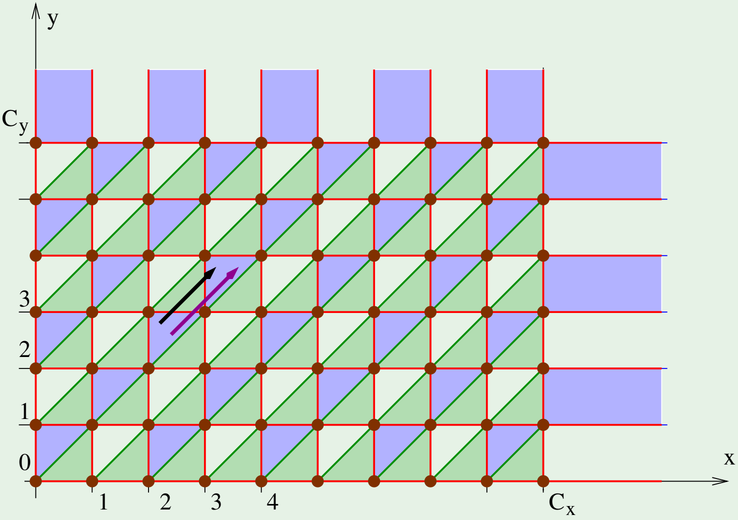
Time-abstract Bisimulation in Regions



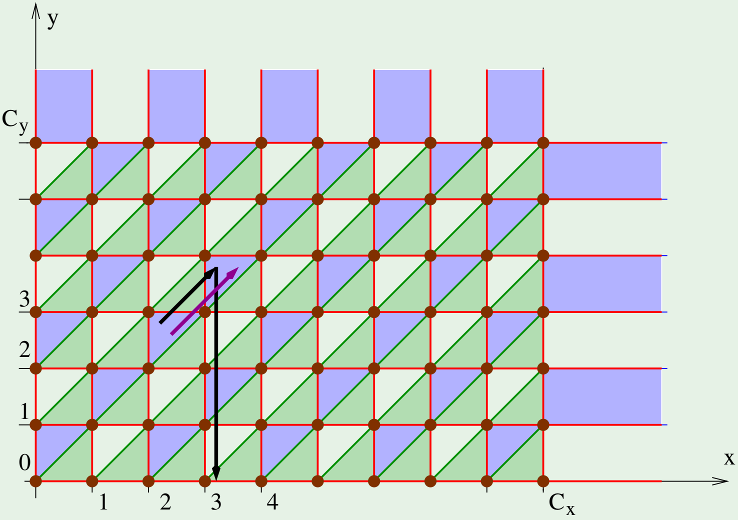
Time-abstract Bisimulation in Regions



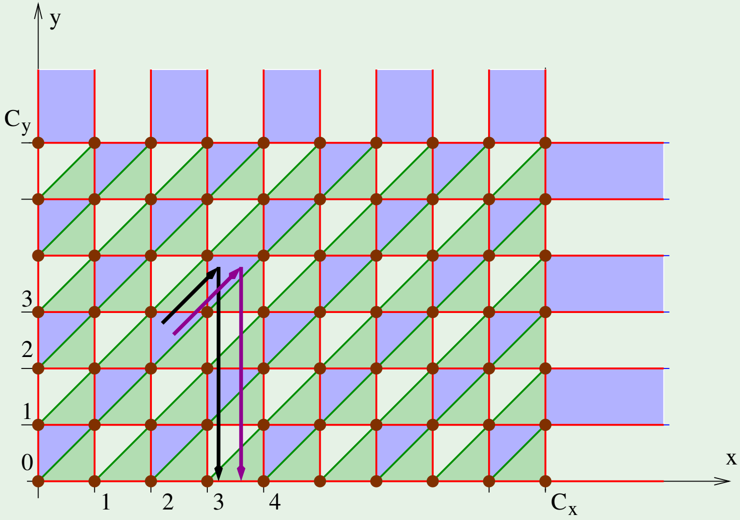
Time-abstract Bisimulation in Regions



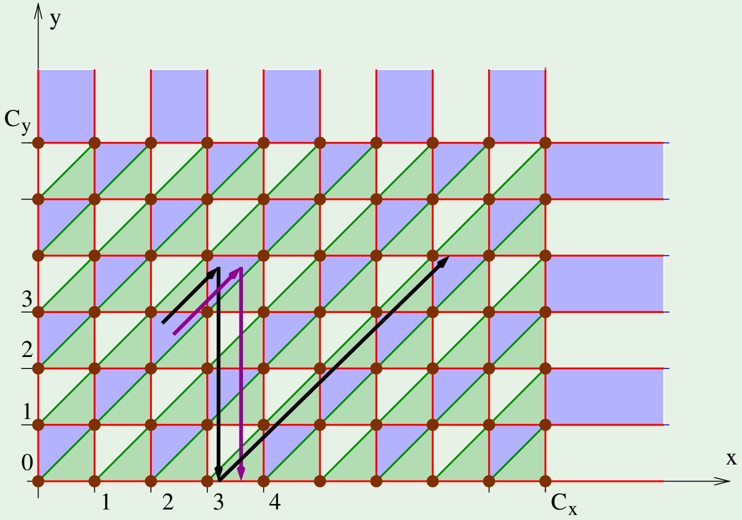
Time-abstract Bisimulation in Regions



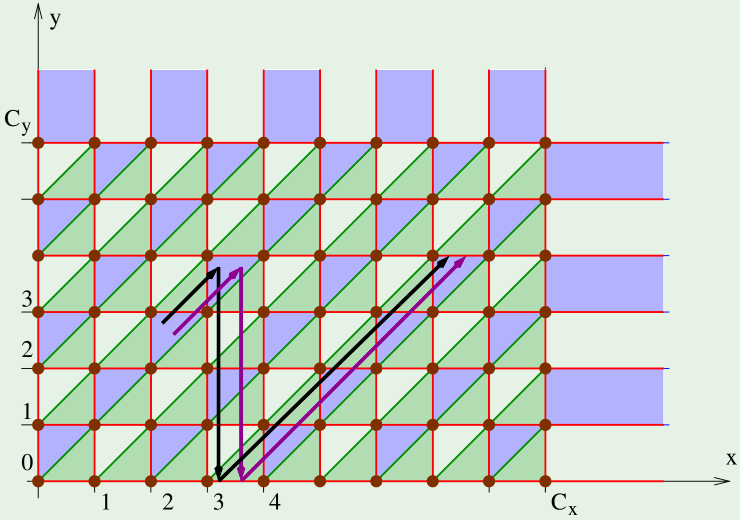
Time-abstract Bisimulation in Regions



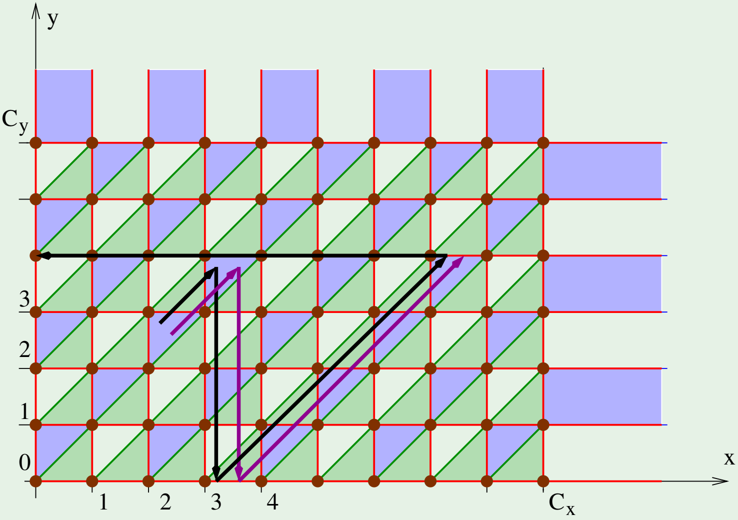
Time-abstract Bisimulation in Regions



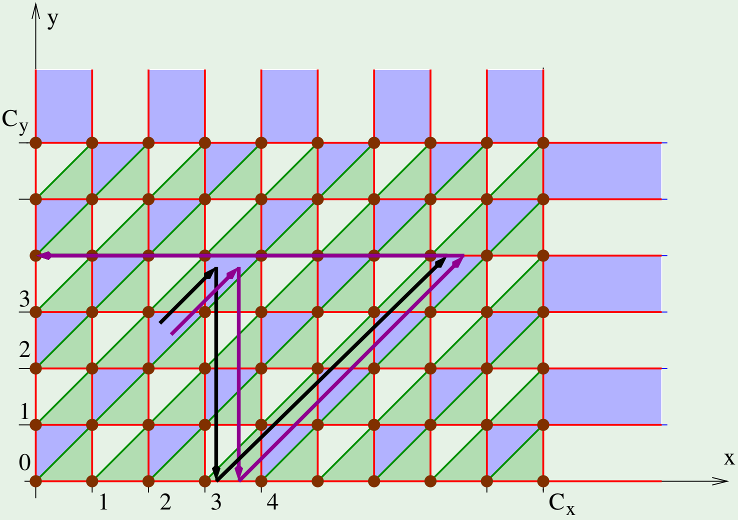
Time-abstract Bisimulation in Regions



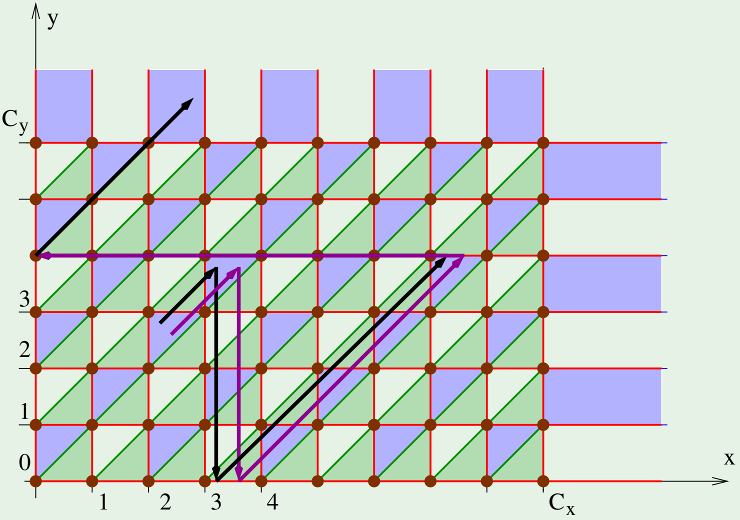
Time-abstract Bisimulation in Regions



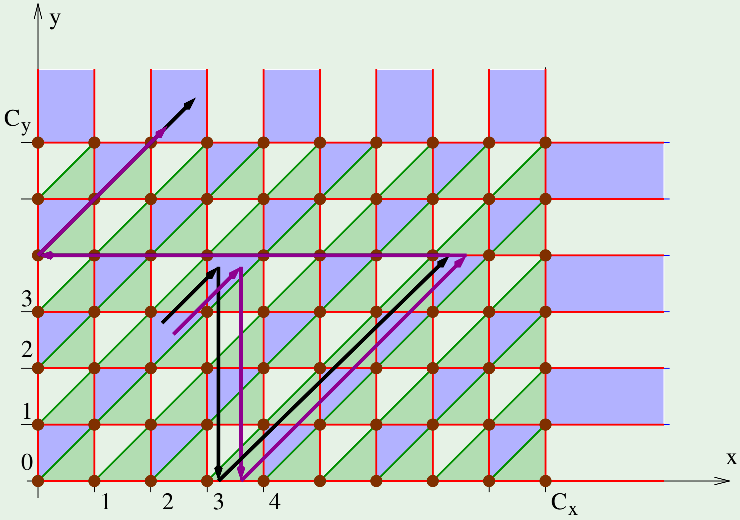
Time-abstract Bisimulation in Regions



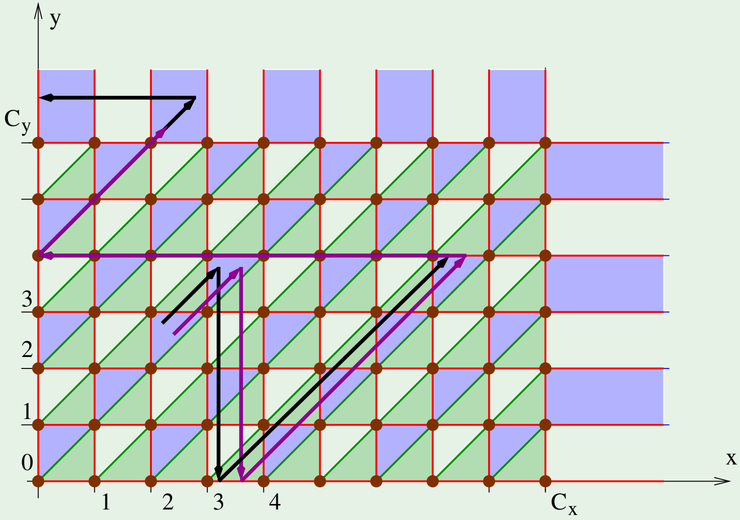
Time-abstract Bisimulation in Regions



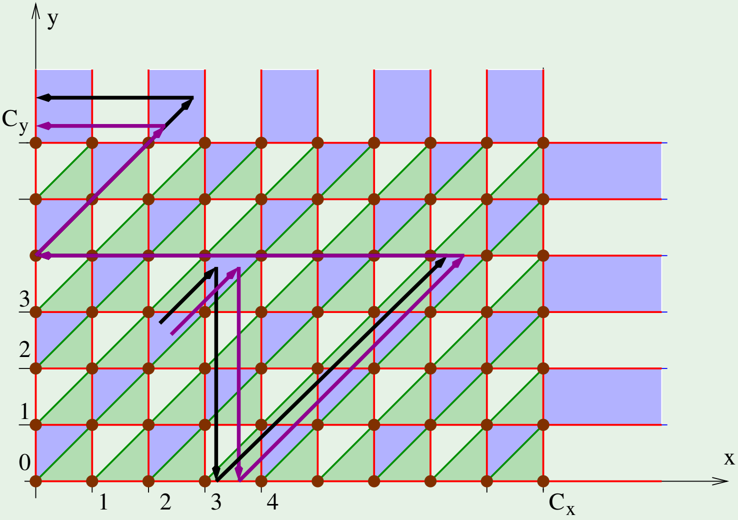
Time-abstract Bisimulation in Regions



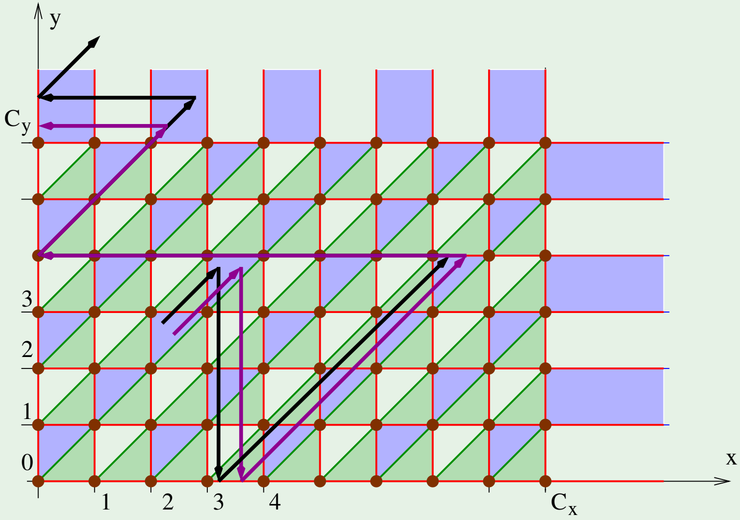
Time-abstract Bisimulation in Regions



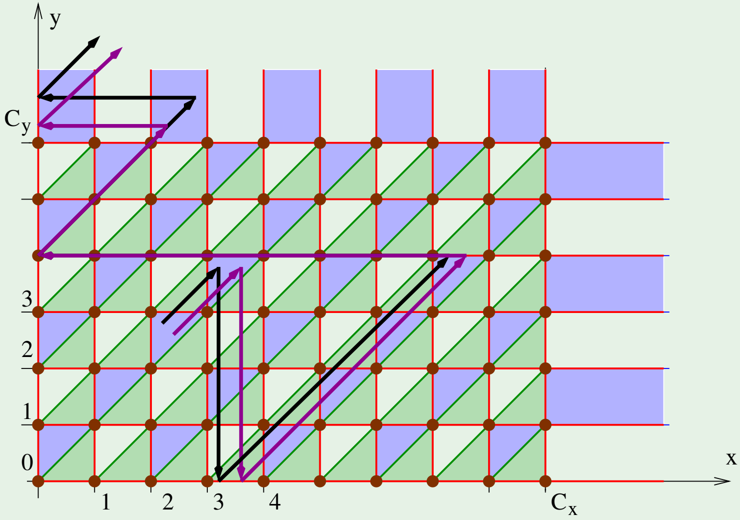
Time-abstract Bisimulation in Regions



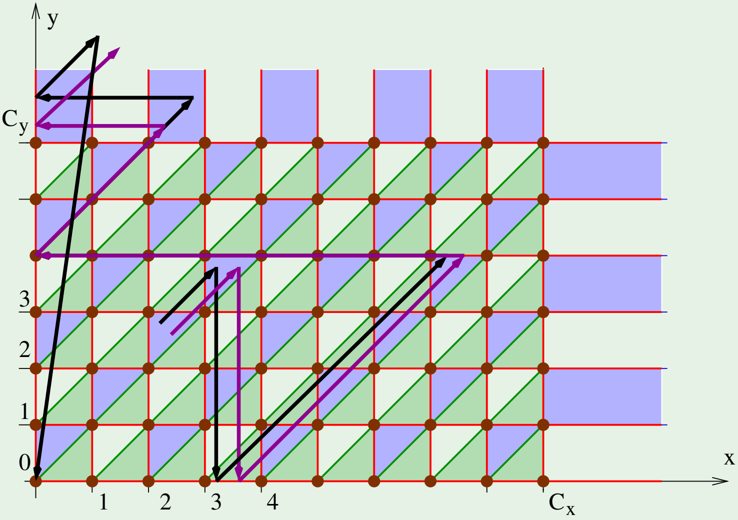
Time-abstract Bisimulation in Regions



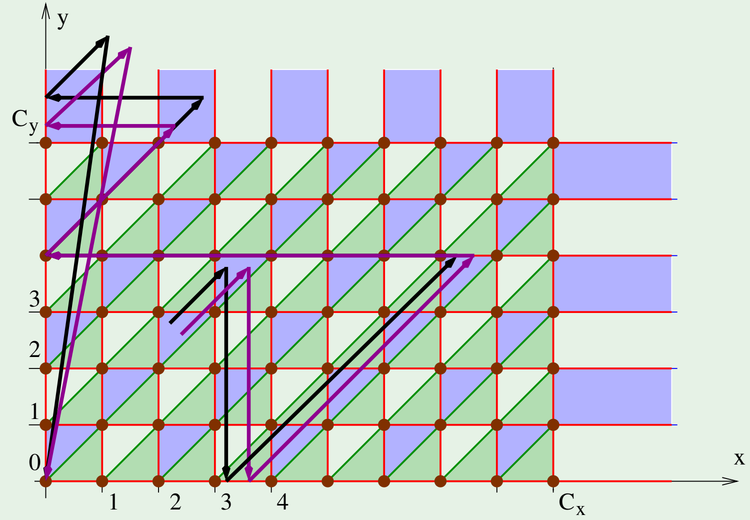
Time-abstract Bisimulation in Regions



Time-abstract Bisimulation in Regions



Time-abstract Bisimulation in Regions



Number of Clock Regions

- Clock region: equivalence class of clock interpretations
- Number of clock regions upper-bounded by

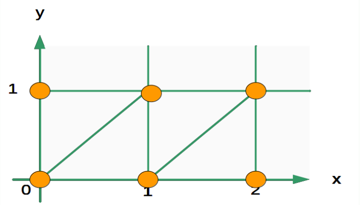
$$k! \cdot 2^k \cdot \Pi_{x \in X}(2 \cdot C_x + 2), \quad s.t.\ k \stackrel{\text{def}}{=} ||X||$$

  - finite!
  - exponential in the number of clocks
  - grows with the values of C_x
  - typically quite pessimistic

Example

- 2 clocks x,y, $C_x = 2$, $C_y = 1$
  - 8 open regions
  - 14 open line segments
  - 6 corner points

  $\Longrightarrow$ 28 regions
  $< 2 \cdot 2^2 \cdot (2 \cdot 2 + 2) \cdot (2 \cdot 1 + 2) = 192$



Number of Clock Regions

- Clock region: equivalence class of clock interpretations
- Number of clock regions upper-bounded by

$$k! \cdot 2^k \cdot \Pi_{x \in X}(2 \cdot C_x + 2), \quad s.t.\ k \stackrel{\text{def}}{=} ||X||$$

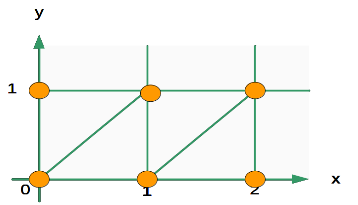
  - finite!
  - exponential in the number of clocks
  - grows with the values of C_x
  - typically quite pessimistic

Example

- 2 clocks x,y, $C_x = 2$, $C_y = 1$
  - 8 open regions
  - 14 open line segments
  - 6 corner points

$\Longrightarrow$ 28 regions

$< 2 \cdot 2^2 \cdot (2 \cdot 2 + 2) \cdot (2 \cdot 1 + 2) = 192$



Region automaton

- Equivalent states = identical location + $\cong$-equivalent evaluations
- Equivalent Classes (regions): finite, stable, L^F-sensitive
- $R(A)$: Region automaton of A
  - States: $\langle l, r(A) \rangle$ s.t. $r(A)$ regions of A

  $\Longrightarrow$ Finite state automaton!
- Reachability problem $\langle A, L^F \rangle \Longrightarrow$ Reachability problem $\langle R(A), L^F \rangle$

$\Longrightarrow$ Reachability in timed automata reduced to that in finite automata!

Region automaton

- Equivalent states = identical location + $\cong$-equivalent evaluations
- Equivalent Classes (regions): finite, stable, L^F-sensitive
- $R(A)$: Region automaton of A
  - States: $\langle l, r(A) \rangle$ s.t. $r(A)$ regions of A

  $\Longrightarrow$ Finite state automaton!
- Reachability problem $\langle A, L^F \rangle \Longrightarrow$ Reachability problem $\langle R(A), L^F \rangle$

$\Longrightarrow$ Reachability in timed automata reduced to that in finite automata!

Region automaton

- Equivalent states = identical location + $\cong$-equivalent evaluations
- Equivalent Classes (regions): finite, stable, L^F-sensitive
- $R(A)$: Region automaton of A
  - States: $\langle l, r(A) \rangle$ s.t. $r(A)$ regions of A

  $\Longrightarrow$ Finite state automaton!
- Reachability problem $\langle A, L^F \rangle \Longrightarrow$ Reachability problem $\langle R(A), L^F \rangle$

$\Longrightarrow$ Reachability in timed automata reduced to that in finite automata!

Region automaton

- Equivalent states = identical location + $\cong$-equivalent evaluations
- Equivalent Classes (regions): finite, stable, L^F-sensitive
- $R(A)$: Region automaton of A
  - States: $\langle l, r(A) \rangle$ s.t. $r(A)$ regions of A

  $\Longrightarrow$ Finite state automaton!
- Reachability problem $\langle A, L^F \rangle \Longrightarrow$ Reachability problem $\langle R(A), L^F \rangle$

$\Longrightarrow$ Reachability in timed automata reduced to that in finite automata!

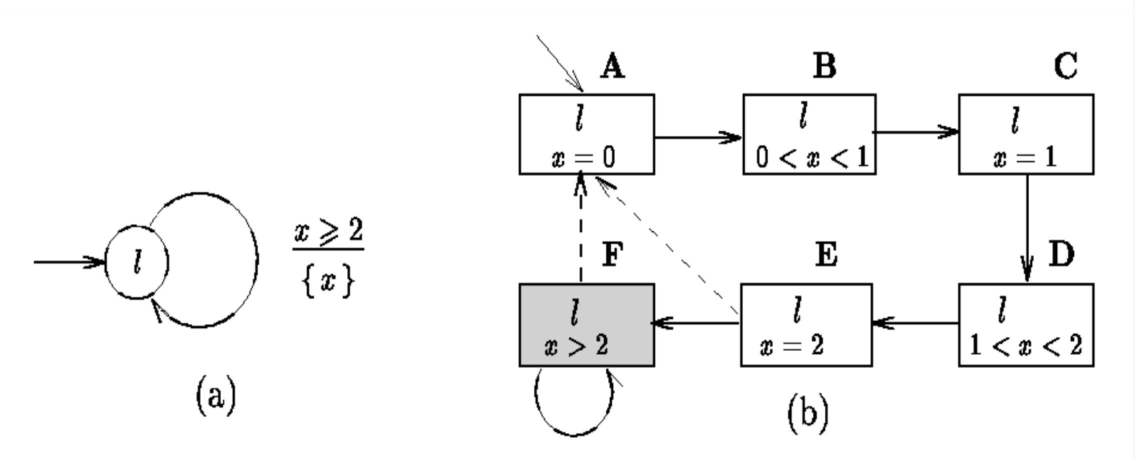
Region automaton

- Equivalent states = identical location + $\cong$-equivalent evaluations
- Equivalent Classes (regions): finite, stable, L^F-sensitive
- $R(A)$: Region automaton of A
  - States: $\langle l, r(A) \rangle$ s.t. $r(A)$ regions of A

  $\Longrightarrow$ Finite state automaton!
- Reachability problem $\langle A, L^F \rangle \Longrightarrow$ Reachability problem $\langle R(A), L^F \rangle$

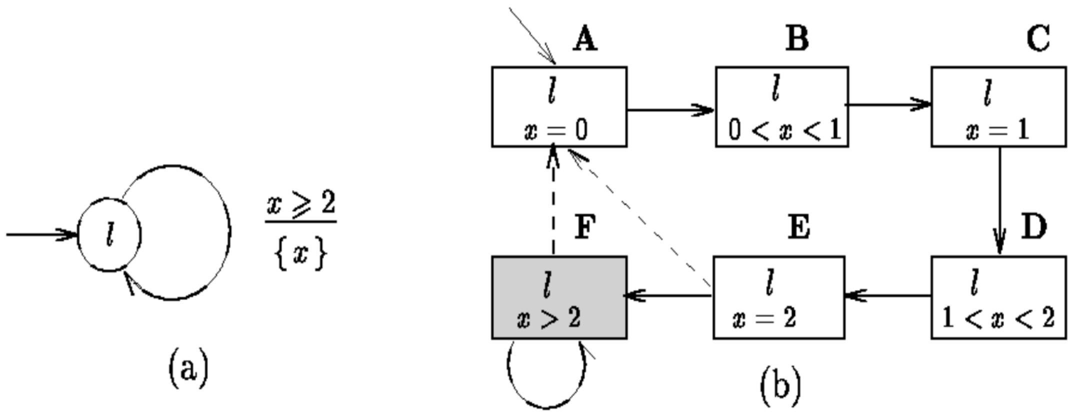
$\Longrightarrow$ Reachability in timed automata reduced to that in finite automata!

Example: Region graph of a simple timed automata



May be further reduced (e.g., collapsing B, C, D into one state)

Example: Region graph of a simple timed automata



May be further reduced (e.g., collapsing B, C, D into one state)

Complexity of Reasoning with Timed Automata

Reachability in Timed Automata

- Decidable!
- Linear with number of locations
- Exponential in the number of clocks
- Grows with the values of C_x
- Overall, PSPACE-Complete

Language-containment with Timed Automata

Undecidable!

Complexity of Reasoning with Timed Automata

Reachability in Timed Automata

- Decidable!
- Linear with number of locations
- Exponential in the number of clocks
- Grows with the values of C_x
- Overall, PSPACE-Complete

Language-containment with Timed Automata

Undecidable!

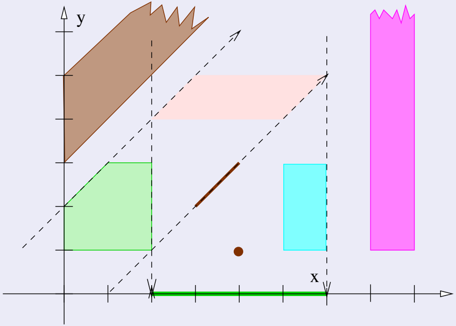
Outline

Zone Automata

- Collapse regions by convex unions of clock regions
- Clock Zone φ: set/conjunction of clock constraints in the form $(x_i \bowtie c)$, $(x_i - x_j \bowtie c)$, $\bowtie \in \{>, <, =, \geq, \leq\}$, $c \in \mathbb{Z}$
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible relationship for all clock value in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: clock zone

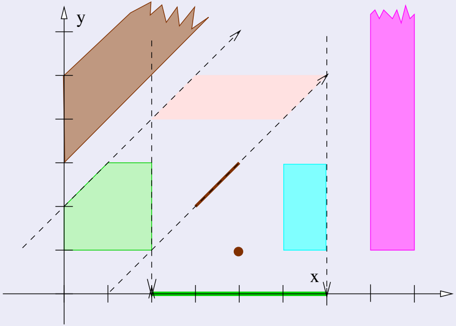


Zone Automata

- Collapse regions by convex unions of clock regions
- Clock Zone φ: set/conjunction of clock constraints in the form $(x_i \bowtie c)$, $(x_i - x_j \bowtie c)$, $\bowtie \in \{>, <, =, \geq, \leq\}$, $c \in \mathbb{Z}$
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible relationship for all clock value in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: clock zone

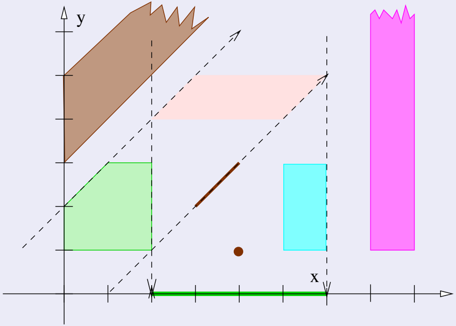


Zone Automata

- Collapse regions by convex unions of clock regions
- Clock Zone φ: set/conjunction of clock constraints in the form $(x_i \bowtie c)$, $(x_i - x_j \bowtie c)$, $\bowtie \in \{>, <, =, \geq, \leq\}$, $c \in \mathbb{Z}$
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible relationship for all clock value in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: clock zone

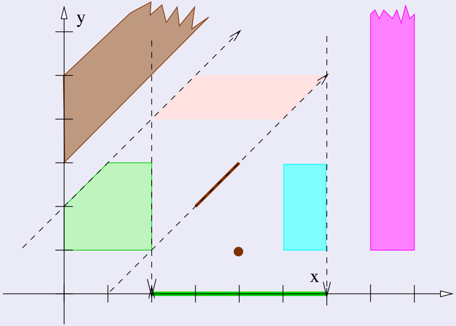


Zone Automata

- Collapse regions by convex unions of clock regions
- Clock Zone φ: set/conjunction of clock constraints in the form $(x_i \bowtie c)$, $(x_i - x_j \bowtie c)$, $\bowtie \in \{>, <, =, \geq, \leq\}$, $c \in \mathbb{Z}$
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible relationship for all clock value in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: clock zone

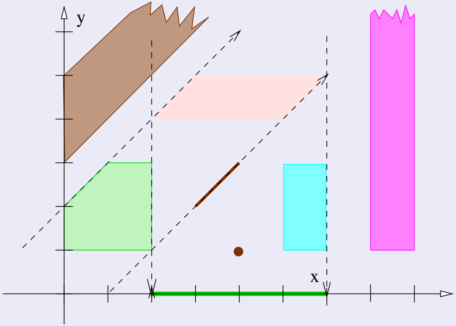


Zone Automata

- Collapse regions by convex unions of clock regions
- Clock Zone φ: set/conjunction of clock constraints in the form $(x_i \bowtie c)$, $(x_i - x_j \bowtie c)$, $\bowtie \in \{>, <, =, \geq, \leq\}$, $c \in \mathbb{Z}$
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible relationship for all clock value in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: clock zone



Zone Automata

Definition: Zone Automaton

- Given a Timed Automaton $A \stackrel{\text{def}}{=} \langle L, L^0, \Sigma, X, \Phi(X), E\rangle$, the Zone Automaton Z(A) is a transition system $\langle Q, Q^0, \Sigma, \longrightarrow\rangle$ s.t.
  - Q: set of all symbolic states of A (a symbolic state is $\langle l, \varphi\rangle$)
  - $Q^0 \stackrel{\text{def}}{=} \{\langle l, [X := 0]\rangle \mid l \in L^0\}$
  - Σ: set of labels/events in A
  - $\longrightarrow$: set of "wait&switch" symbolic transitions, in the form: $\langle l, \varphi\rangle \xrightarrow{a} \langle l', succ(\varphi, e)\rangle$
    $succ(\varphi, e)$: successor of φ after (waiting and) executing the switch $e \stackrel{\text{def}}{=} \langle l, a, \psi, \lambda, l'\rangle$
- $succ(\langle l, \varphi\rangle, e) \stackrel{\text{def}}{=} \langle l', succ(\varphi, e)\rangle$

Zone Automata

Definition: Zone Automaton

- Given a Timed Automaton $A \stackrel{\text{def}}{=} \langle L, L^0, \Sigma, X, \Phi(X), E \rangle$, the Zone Automaton Z(A) is a transition system $\langle Q, Q^0, \Sigma, \longrightarrow \rangle$ s.t.
  - Q: set of all symbolic states of A (a symbolic state is $\langle l, \varphi \rangle$)
  - $Q^0 \stackrel{\text{def}}{=} \{\langle l, [X := 0] \rangle \mid l \in L^0\}$
  - Σ: set of labels/events in A
  - $\longrightarrow$: set of "wait&switch" symbolic transitions, in the form: $\langle l, \varphi \rangle \xrightarrow{a} \langle l', succ(\varphi, e) \rangle$

    $succ(\varphi, e)$: successor of φ after (waiting and) executing the switch $e \stackrel{\text{def}}{=} \langle l, a, \psi, \lambda, l' \rangle$
- $succ(\langle l, \varphi \rangle, e) \stackrel{\text{def}}{=} \langle l', succ(\varphi, e) \rangle$

Zone Automata

Definition: Zone Automaton

- Given a Timed Automaton $A \stackrel{\text{def}}{=} \langle L, L^0, \Sigma, X, \Phi(X), E\rangle$,
  the Zone Automaton Z(A) is a transition system $\langle Q, Q^0, \Sigma, \longrightarrow\rangle$ s.t.
  - Q: set of all symbolic states of A (a symbolic state is $\langle l, \varphi\rangle$)
  - $Q^0 \stackrel{\text{def}}{=} \{\langle l, [X := 0]\rangle \mid l \in L^0\}$
  - Σ: set of labels/events in A
  - $\longrightarrow$: set of "wait&switch" symbolic transitions, in the form: $\langle l, \varphi\rangle \xrightarrow{a} \langle l', succ(\varphi, e)\rangle$
    $succ(\varphi, e)$: successor of φ after (waiting and) executing the switch $e \stackrel{\text{def}}{=} \langle l, a, \psi, \lambda, l'\rangle$
- $succ(\langle l, \varphi\rangle, e) \stackrel{\text{def}}{=} \langle l', succ(\varphi, e)\rangle$

Zone Automata

Definition: Zone Automaton

- Given a Timed Automaton $A \stackrel{\text{def}}{=} \langle L, L^0, \Sigma, X, \Phi(X), E\rangle$, the Zone Automaton Z(A) is a transition system $\langle Q, Q^0, \Sigma, \longrightarrow\rangle$ s.t.
    - Q: set of all symbolic states of A (a symbolic state is $\langle l, \varphi\rangle$)
    - $Q^0 \stackrel{\text{def}}{=} \{\langle l, [X := 0]\rangle \mid l \in L^0\}$
    - Σ: set of labels/events in A
    - $\longrightarrow$: set of "wait&switch" symbolic transitions, in the form: $\langle l, \varphi\rangle \xrightarrow{a} \langle l', succ(\varphi, e)\rangle$
      $succ(\varphi, e)$: successor of φ after (waiting and) executing the switch $e \stackrel{\text{def}}{=} \langle l, a, \psi, \lambda, l'\rangle$
- $succ(\langle l, \varphi\rangle, e) \stackrel{\text{def}}{=} \langle l', succ(\varphi, e)\rangle$

Zone Automata

Definition: Zone Automaton

- Given a Timed Automaton $A \stackrel{\text{def}}{=} \langle L, L^0, \Sigma, X, \Phi(X), E\rangle$, the Zone Automaton Z(A) is a transition system $\langle Q, Q^0, \Sigma, \longrightarrow\rangle$ s.t.
  - Q: set of all symbolic states of A (a symbolic state is $\langle l, \varphi\rangle$)
  - $Q^0 \stackrel{\text{def}}{=} \{\langle l, [X := 0]\rangle \mid l \in L^0\}$
  - Σ: set of labels/events in A
  - $\longrightarrow$: set of "wait&switch" symbolic transitions, in the form: $\langle l, \varphi\rangle \xrightarrow{a} \langle l', succ(\varphi, e)\rangle$
    $succ(\varphi, e)$: successor of φ after (waiting and) executing the switch $e \stackrel{\text{def}}{=} \langle l, a, \psi, \lambda, l'\rangle$
- $succ(\langle l, \varphi\rangle, e) \stackrel{\text{def}}{=} \langle l', succ(\varphi, e)\rangle$

Zone Automata

Definition: Zone Automaton

- Given a Timed Automaton $A \stackrel{\text{def}}{=} \langle L, L^0, \Sigma, X, \Phi(X), E \rangle$, the Zone Automaton Z(A) is a transition system $\langle Q, Q^0, \Sigma, \rightarrow \rangle$ s.t.
  - Q: set of all symbolic states of A (a symbolic state is $\langle l, \varphi \rangle$)
  - $Q^0 \stackrel{\text{def}}{=} \{ \langle l, [X := 0] \rangle \mid l \in L^0 \}$
  - Σ: set of labels/events in A
  - $\rightarrow$: set of "wait&switch" symbolic transitions, in the form: $\langle l, \varphi \rangle \stackrel{a}{\longrightarrow} \langle l', succ(\varphi, e) \rangle$

    $succ(\varphi, e)$: successor of φ after (waiting and) executing the switch $e \stackrel{\text{def}}{=} \langle l, a, \psi, \lambda, l' \rangle$
- $succ(\langle l, \varphi \rangle, e) \stackrel{\text{def}}{=} \langle l', succ(\varphi, e) \rangle$

Zone Automata: Symbolic Transitions

Definition: $succ(\varphi, e)$

- Let $e \stackrel{\text{def}}{=} \langle l, a, \psi, \lambda, l' \rangle$, and ϕ, ϕ' the invariants in l, l'
- Then

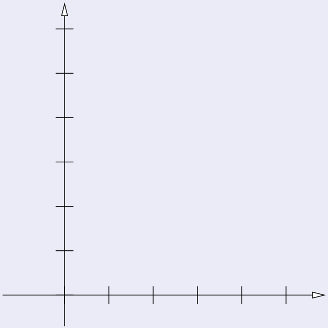
$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \ \wedge \phi) \wedge \psi)[\lambda := 0]$$

  - $\wedge$: standard conjunction/intersection
  - $\Uparrow$: projection to infinity: $\psi\Uparrow \stackrel{\text{def}}{=} \{\nu + \delta \mid \nu \in \psi, \delta \in [0, +\infty)\}$
  - $[\lambda := 0]$: reset projection: $\psi[\lambda := 0] \stackrel{\text{def}}{=} \{\nu[\lambda := 0] \mid \nu \in \psi\}$
- note: φ is considered "immediately before entering l"

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

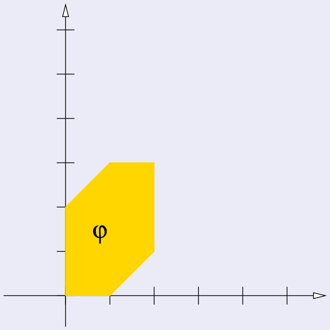


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \wedge \phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

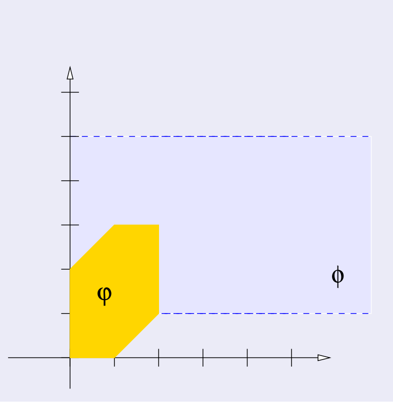


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \wedge\phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

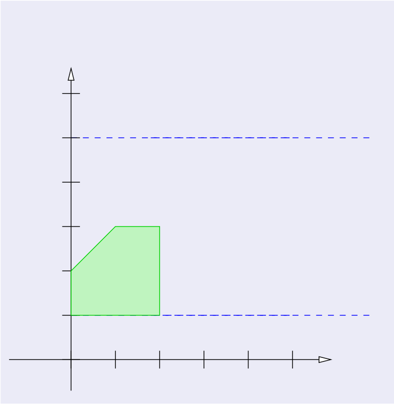


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \wedge \phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

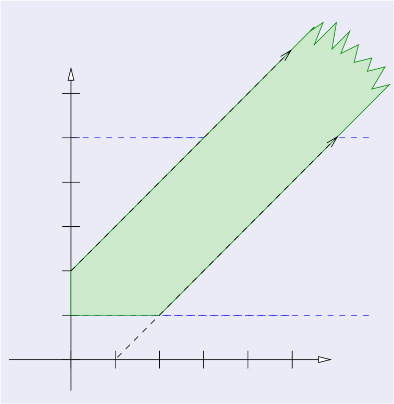


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \wedge \phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

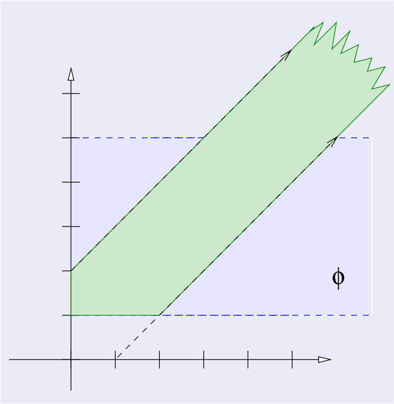


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \ \wedge\phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

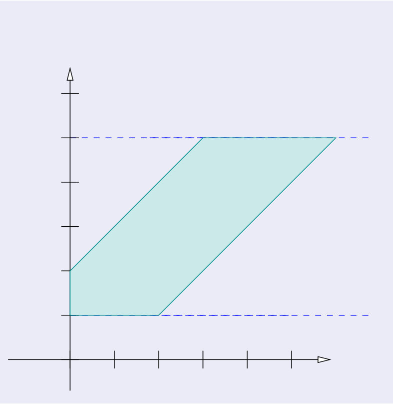


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \wedge\phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

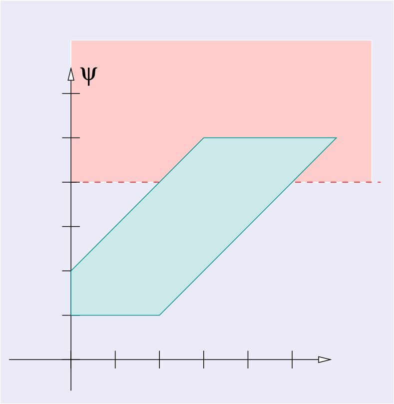


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \wedge \phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

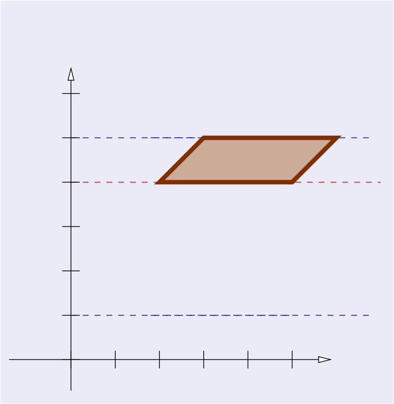


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \; \wedge\phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

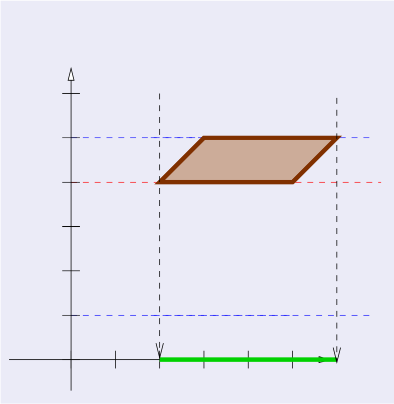


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \land \phi)\Uparrow \ \land \phi) \land \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

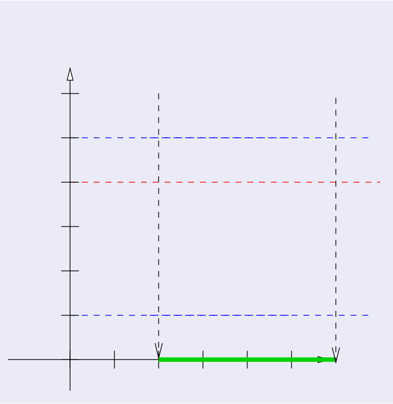


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \ \wedge \phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

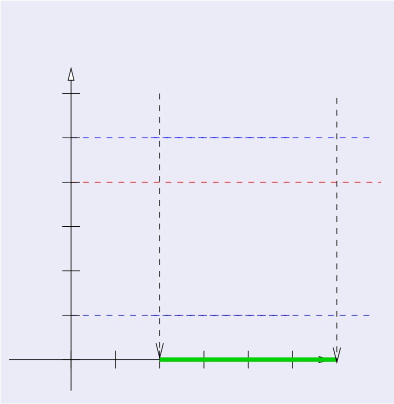


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \ \wedge \phi) \wedge \psi)[\lambda := 0]$$

Zone Automata: Symbolic Transitions (cont.)

- Initial zone: values before entering the location
- Intersection with invariant ϕ: values allowed to enter the location
- Projection to infinity: values allowed to enter the location, after waiting unbounded time
- Intersection with invariant ϕ: values allowed to enter the location, after waiting a legal amount of time
- Intersection with guard ψ: values allowed to enter the location, after waiting a legal amount of time, from which the switch can be shot
- Reset projection λ: values ..., after reset

$\Longrightarrow$ Final!

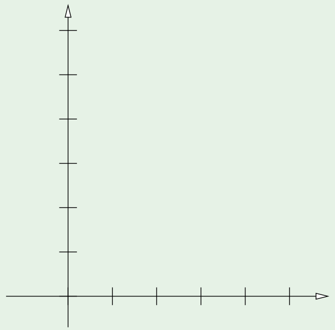


$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \ \wedge \phi) \wedge \psi)[\lambda := 0]$$

Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow \ (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow \ (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

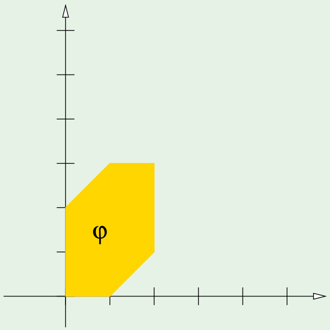
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant $\phi : (y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard $\psi : (y \geq 4)$
  $\Longrightarrow (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

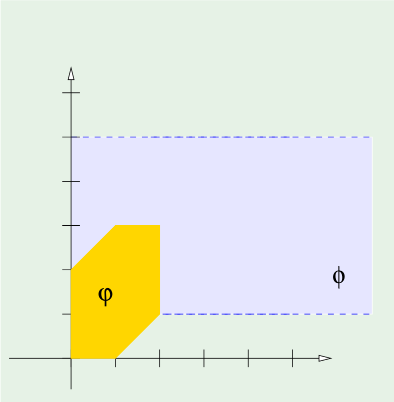
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge (y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge (y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow (y \geq 4) \wedge (y \leq 5) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

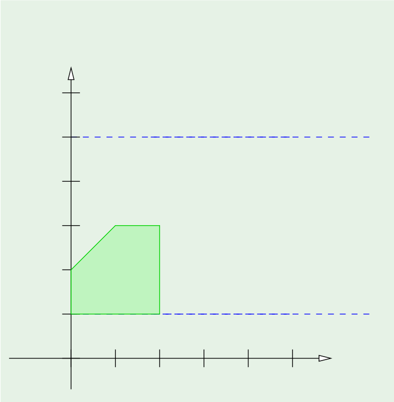
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow \ (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow \ (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

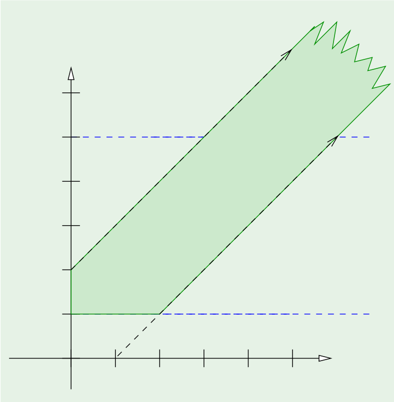
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow \ (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow \ (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

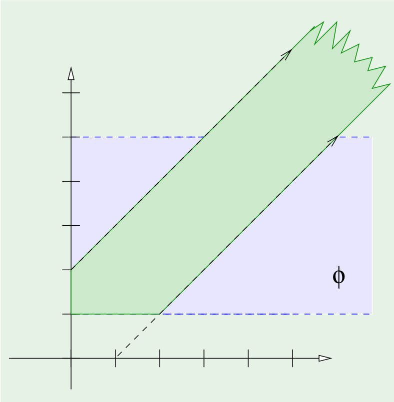
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

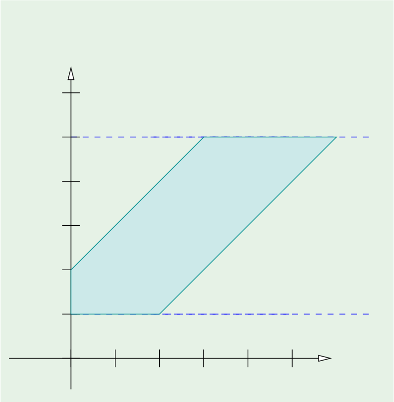
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant $\phi : (y \geq 1) \wedge (y \leq 5)$
  $\implies (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\implies (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\implies (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard $\psi : (y \geq 4)$
  $\implies (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\implies (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

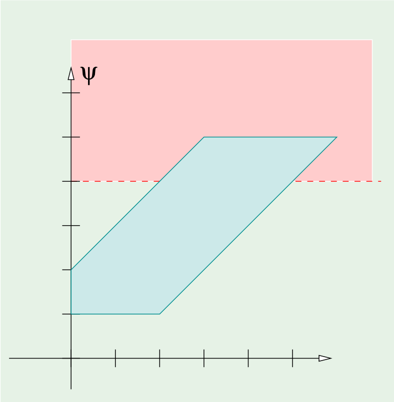
$\implies$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

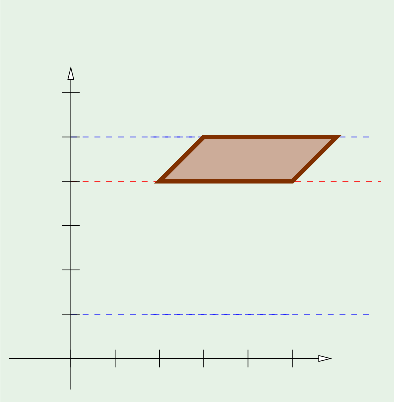
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow \; (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow \; (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow \; (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow \; (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

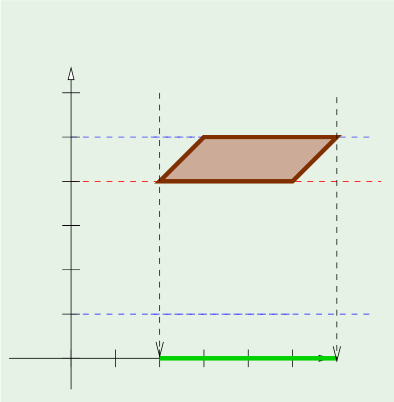
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow \ (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow \ (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

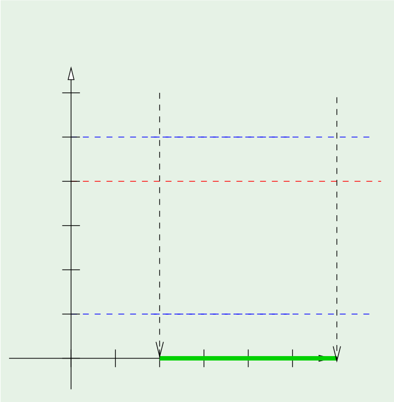
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge (y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge (y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow (y \geq 4) \wedge (y \leq 5) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

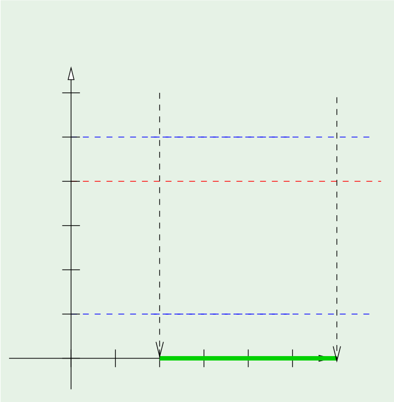
$\Longrightarrow$ Final!



Example: Zone Automata, Symbolic Transitions

- Initial zone: $(x \geq 0) \wedge (x \leq 2) \wedge$
  $(y \geq 0) \wedge (y \leq 3) \wedge (y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ : $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 1) \wedge$
  $(y \leq 3) \wedge (y - x \leq 2)$
- Projection to infinity:
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with invariant ϕ: $(y \geq 1) \wedge (y \leq 5)$
  $\Longrightarrow \ (x \geq 0) \wedge (y \geq 1) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Intersection with guard ψ : $(y \geq 4)$
  $\Longrightarrow \ (y \geq 4) \wedge (y \leq 5) \wedge$
  $(y - x \geq -1) \wedge (y - x \leq 2)$
- Reset projection $\lambda \stackrel{\text{def}}{=} \{y := 0\}$
  $\Longrightarrow \ (x \geq 2) \wedge (x \leq 6) \wedge (y \geq 0) \wedge (y \leq 0)$

$\Longrightarrow$ Final!



Remark on $succ(\varphi, e)$

- In the above definition of $succ(\varphi, e)$, φ is considered "immediately before entering l":

$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \ \wedge \phi) \wedge \psi)[\lambda := 0]$$

- Alternative definition of $succ(\varphi, e)$, φ is considered "immediately after entering l":

$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi\Uparrow \ \wedge \phi) \wedge \psi)[\lambda := 0] \wedge \phi')$$

  - no initial intersection with the invariant ϕ of source location l (here φ is assumed to be already the result of such intersection)
  - final intersection with the invariant ϕ' of target location l'

Remark on $succ(\varphi, e)$

- In the above definition of $succ(\varphi, e)$, φ is considered "immediately before entering l":

$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi \wedge \phi)\Uparrow \ \wedge \phi) \wedge \psi)[\lambda := 0]$$

- Alternative definition of $succ(\varphi, e)$, φ is considered "immediately after entering l":

$$succ(\varphi, e) \stackrel{\text{def}}{=} (((\varphi\Uparrow \ \wedge \phi) \wedge \psi)[\lambda := 0] \wedge \phi')$$

  - no initial intersection with the invariant ϕ of source location l (here φ is assumed to be already the result of such intersection)
  - final intersection with the invariant ϕ' of target location l'

Symbolic Reachability Analysis

function Reachable (A, L^F) $\quad$ // $A \stackrel{\text{def}}{=} \langle L, L^0, \Sigma, X, \Phi(X), E\rangle$
$\quad$ *Reachable* $= \emptyset$
$\quad$ *Frontier* $= \{\langle l_i, \{X = 0\}\rangle \mid l_i \in L^0\}$
$\quad$ **while** (*Frontier* $\neq \emptyset$) **do**
$\qquad$ *extract* $\langle l, \varphi\rangle$ *from* *Frontier*
$\qquad$ **if** ($l \in L^F$ *and* $\varphi \neq \bot$) **then**
$\qquad\quad$ **return** True
$\qquad$ **end if**
$\qquad$ **if** ($\nexists\ \langle l, \varphi'\rangle \in$ *Reachable* $s.t.\ \varphi \subseteq \varphi'$) **then**
$\qquad\quad$ *add* $\langle l, \varphi\rangle$ *to* *Reachable*
$\qquad\quad$ **for** $e \in outcoming(l)$ **do**
$\qquad\qquad$ *add* $succ(\varphi, e)$ *to* *Frontier*
$\qquad\quad$ **end for**
$\qquad$ **end if**
$\quad$ **end while**
$\quad$ **return** False

Canonical Data-structures for Zones: DBMs

Difference-bound Matrices (DBMs)

- Matrix representation of constraints
  - bounds on a single clock
  - differences between 2 clocks
- Reduced form computed by all-pairs shortest path algorithm (e.g. Floyd-Warshall)
- Reduced DBM is canonical:
  equivalent sets of constraints produce the same reduced DBM
- Operations s.a reset, time-successor, inclusion, intersection are efficient

$\Longrightarrow$ Popular choice in timed-automata-based tools

Difference-bound matrices, DBMs

- DBM: matrix $(k+1) \times (k+1)$, k being the number of clocks
  - added an implicit fake variable $x_0 \stackrel{\text{def}}{=} 0$ s.t. $x_i \bowtie c \Longrightarrow x_i - x_0 \bowtie c$
  - each element is a pair (value,$\{0,1\}$), s.t "$\{0,1\}$" means "$\{<, \leq\}$"

Example:

$(0 \leq x_1) \quad \wedge(0 < x_2) \quad \wedge(x_1 < 2) \quad \wedge(x_2 < 1) \quad \wedge(x_1 - x_2 \geq 0)$

$(x_0 - x_1 \leq 0) \quad \wedge(x_0 - x_2 < 0) \quad \wedge(x_1 - x_0 < 2) \quad \wedge(x_2 - x_0 < 1) \quad \wedge(x_2 - x_1 \leq 0)$

Difference-bound matrices, DBMs

- DBM: matrix $(k+1) \times (k+1)$, k being the number of clocks
  - added an implicit fake variable $x_0 \stackrel{\text{def}}{=} 0$ s.t. $x_i \bowtie c \Longrightarrow x_i - x_0 \bowtie c$
  - each element is a pair (value,$\{0,1\}$), s.t "$\{0,1\}$" means "$\{<,\leq\}$"

Example:

$(0 \leq x_1)$ $\wedge(0 < x_2)$ $\wedge(x_1 < 2)$ $\wedge(x_2 < 1)$ $\wedge(x_1 - x_2 \geq 0)$

$(x_0 - x_1 \leq 0)$ $\wedge(x_0 - x_2 < 0)$ $\wedge(x_1 - x_0 < 2)$ $\wedge(x_2 - x_0 < 1)$ $\wedge(x_2 - x_1 \leq 0)$

Difference-bound matrices, DBMs

- DBM: matrix $(k+1) \times (k+1)$, k being the number of clocks
  - added an implicit fake variable $x_0 \stackrel{\text{def}}{=} 0$ s.t. $x_i \bowtie c \Longrightarrow x_i - x_0 \bowtie c$
  - each element is a pair (value,$\{0,1\}$), s.t "$\{0,1\}$" means "$\{<,\leq\}$"

Example:

$$\begin{array}{lllll}
(0 \leq x_1) & \wedge(0 < x_2) & \wedge(x_1 < 2) & \wedge(x_2 < 1) & \wedge(x_1 - x_2 \geq 0) \\
(x_0 - x_1 \leq 0) & \wedge(x_0 - x_2 < 0) & \wedge(x_1 - x_0 < 2) & \wedge(x_2 - x_0 < 1) & \wedge(x_2 - x_1 \leq 0)
\end{array}$$

Difference-bound matrices, DBMs

- DBM: matrix $(k+1) \times (k+1)$, k being the number of clocks
  - added an implicit fake variable $x_0 \stackrel{\text{def}}{=} 0$ s.t. $x_i \bowtie c \implies x_i - x_0 \bowtie c$
  - each element is a pair (value,$\{0,1\}$), s.t "$\{0,1\}$" means "$\{<, \leq\}$"

Example:

$(0 \leq x_1)$ $\wedge(0 < x_2)$ $\wedge(x_1 < 2)$ $\wedge(x_2 < 1)$ $\wedge(x_1 - x_2 \geq 0)$

$(x_0 - x_1 \leq 0)$ $\wedge(x_0 - x_2 < 0)$ $\wedge(x_1 - x_0 < 2)$ $\wedge(x_2 - x_0 < 1)$ $\wedge(x_2 - x_1 \leq 0)$

	Matrix D		
	0	1	2
0	∞	(0,1)	(0,0)
1	(2,0)	∞	∞
2	(1,0)	(0,1)	∞

D_{0i} = lower bound

D_{i0} = upper bound

D_{ij} = upper bound of x_i and x_j difference

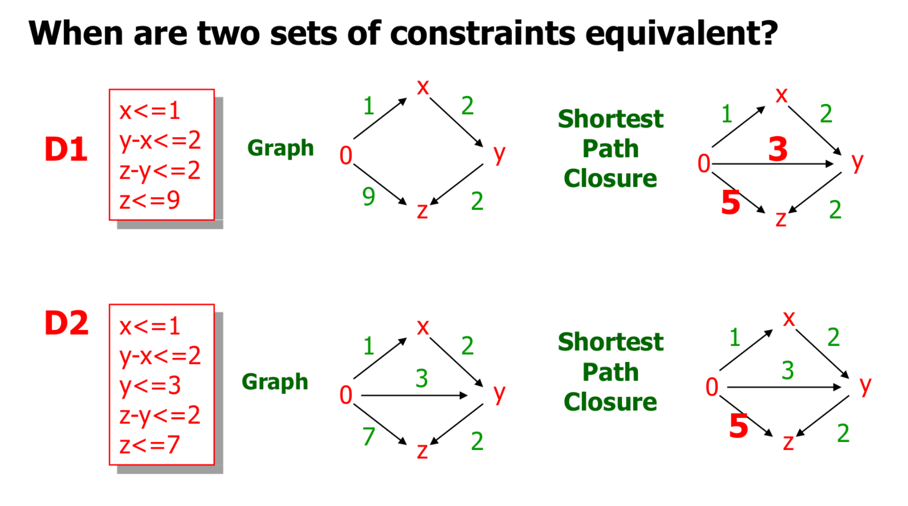
- i,j: $(c,1)$ → Xi-Xj ≤ c
- i,j: $(c,0)$ → Xi-Xj < c
- i,j: ∞ → absence of bound

Difference-bound matrices, DBMs (cont.)

- Use all-pairs shortest paths, check DBM
  - Add $x_i - x_i \leq 0$ for each i
  - Idea: given $x_i - x_j \bowtie c$, $x_i - x_k \bowtie c_1$ and $x_k - x_j \bowtie c_2$ s.t. $\bowtie \in \{\leq, <\}$, then c is updated with $c_1 + c_2$ if $c_1 + c_2 < c$
  - Satisfiable (no negative loops) $\Longrightarrow$ a non-empty clock zone
  - Canonical: matrices with tightest possible constraints
- Canonical DBMs represent clock zones: equivalent sets of constraints $\Longleftrightarrow$ same reduced DBM

	Matrix D			Matrix D'		
	0	1	2	0	1	2
0	∞	(0,1)	(0,0)	(0, 1)	(0,1)	(0,0)
1	(2,0)	∞	∞	(2,0)	(0,1)	(2,0)
2	(1,0)	(0,1)	∞	(1,0)	(0,1)	(0,1)

Canonical Data-structures for Zones: DBMs



$\Longrightarrow$ they have the same reduced DBM

Complexity Issues

- In theory:
  - Zone automaton might be exponentially bigger than the region automaton
- In practice:
  - Fewer reachable vertices $\Longrightarrow$ performances much improved

Timed Automata: summary

- Only continuous variables are timers
- Invariants and Guards: $x \bowtie const$, $\bowtie \in \{<, >, \leq, \geq\}$
- Actions: x:=0
- Reachability is decidable
- Clustering of regions into zones desirable in practice
- Tools: Uppaal, Kronos, RED ...
- Symbolic representation: matrices

Decidable Problems with Timed Automata

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

Outline

Hybrid Systems

Hybrid (Dynamical) System

- A dynamical system that exhibits both continuous and discrete dynamic behavior

$\Longrightarrow$ Can both:

  - flow (described by differential equations) and
  - jump (described by a state machine or automaton).

- Mostly used to model Cyber-Physical Systems (CPSs)
  - a physical (chemical, biological...) mechanism is controlled by computer-based algorithms
  - physical and software components are deeply intertwined
- Most popular formalism: Hybrid Automata and variants

Hybrid Systems

Hybrid (Dynamical) System

- A dynamical system that exhibits both continuous and discrete dynamic behavior

$\Longrightarrow$ Can both:

  - flow (described by differential equations) and
  - jump (described by a state machine or automaton).
- Mostly used to model Cyber-Physical Systems (CPSs)
  - a physical (chemical, biological...) mechanism is controlled by computer-based algorithms
  - physical and software components are deeply intertwined
- Most popular formalism: Hybrid Automata and variants

Hybrid Systems

Hybrid (Dynamical) System

- A dynamical system that exhibits both continuous and discrete dynamic behavior

$\Longrightarrow$ Can both:

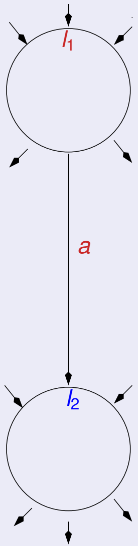
  - flow (described by differential equations) and
  - jump (described by a state machine or automaton).
- Mostly used to model Cyber-Physical Systems (CPSs)
  - a physical (chemical, biological...) mechanism is controlled by computer-based algorithms
  - physical and software components are deeply intertwined
- Most popular formalism: Hybrid Automata and variants

Hybrid Sysmem: Example

Outline

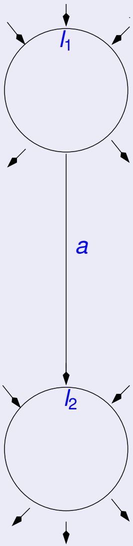
Hybrid Automata

- Locations, Switches, Labels (like in standard aut.)
- Continuous variables: $X \stackrel{\text{def}}{=} \{x_1, x_2, ..., x_k\} \in \mathbb{R}$
  - value evolves with time
  - e.g., distance, speed, pressure, temperature, ...
- Guards: $g(X) \geq 0$
  - sets of inequalities (equalities) on functions on X
  - constrain the execution of the switch
- Jump Transformations $J(X, X')$
  - discrete transformation on the values of X
- Invariants: $X \in Inv_l(X)$
  - set of invariant constraints on X
  - ensure progress
- Continuous Flow: $\frac{dX}{dt} \in flow_l(X)$
  - set of degree-1 differential (in)equalities
  - describe continuous dynamics
- Initial: $X \in Init_l(X)$
  - initial conditions ($Init_l(X) = \perp$ iff $l \notin L^0$)



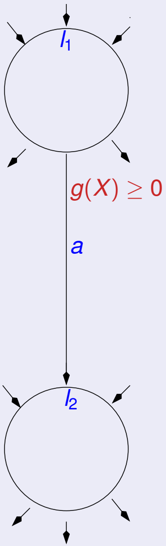
Hybrid Automata

- Locations, Switches, Labels (like in standard aut.)
- Continuous variables: $X \stackrel{\text{def}}{=} \{x_1, x_2, ..., x_k\} \in \mathbb{R}$
  - value evolves with time
  - e.g., distance, speed, pressure, temperature, ...
- Guards: $g(X) \geq 0$
  - sets of inequalities (equalities) on functions on X
  - constrain the execution of the switch
- Jump Transformations $J(X, X')$
  - discrete transformation on the values of X
- Invariants: $X \in Inv_l(X)$
  - set of invariant constraints on X
  - ensure progress
- Continuous Flow: $\frac{dX}{dt} \in flow_l(X)$
  - set of degree-1 differential (in)equalities
  - describe continuous dynamics
- Initial: $X \in Init_l(X)$
  - initial conditions ($Init_l(X) = \perp$ iff $l \notin L^0$)



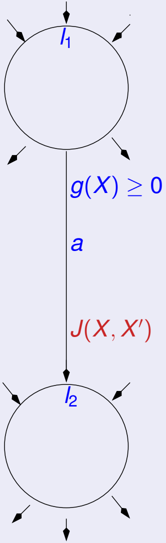
Hybrid Automata

- Locations, Switches, Labels (like in standard aut.)
- Continuous variables: $X \stackrel{\text{def}}{=} \{x_1, x_2, ..., x_k\} \in \mathbb{R}$
  - value evolves with time
  - e.g., distance, speed, pressure, temperature, ...
- Guards: $g(X) \geq 0$
  - sets of inequalities (equalities) on functions on X
  - constrain the execution of the switch
- Jump Transformations $J(X, X')$
  - discrete transformation on the values of X
- Invariants: $X \in Inv_l(X)$
  - set of invariant constraints on X
  - ensure progress
- Continuous Flow: $\frac{dX}{dt} \in flow_l(X)$
  - set of degree-1 differential (in)equalities
  - describe continuous dynamics
- Initial: $X \in Init_l(X)$
  - initial conditions ($Init_l(X) = \perp$ iff $l \notin L^0$)



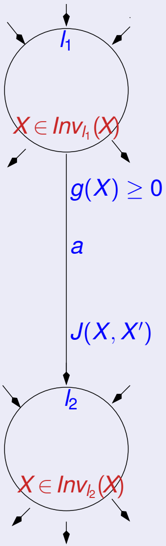
Hybrid Automata

- Locations, Switches, Labels (like in standard aut.)
- Continuous variables: $X \stackrel{\text{def}}{=} \{x_1, x_2, ..., x_k\} \in \mathbb{R}$
  - value evolves with time
  - e.g., distance, speed, pressure, temperature, ...
- Guards: $g(X) \geq 0$
  - sets of inequalities (equalities) on functions on X
  - constrain the execution of the switch
- Jump Transformations $J(X, X')$
  - discrete transformation on the values of X
- Invariants: $X \in Inv_l(X)$
  - set of invariant constraints on X
  - ensure progress
- Continuous Flow: $\frac{dX}{dt} \in flow_l(X)$
  - set of degree-1 differential (in)equalities
  - describe continuous dynamics
- Initial: $X \in Init_l(X)$
  - initial conditions ($Init_l(X) = \perp$ iff $l \not\in L^0$)



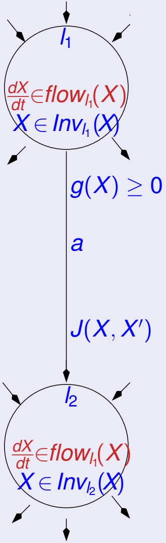
Hybrid Automata

- Locations, Switches, Labels (like in standard aut.)
- Continuous variables: $X \stackrel{\text{def}}{=} \{x_1, x_2, ..., x_k\} \in \mathbb{R}$
  - value evolves with time
  - e.g., distance, speed, pressure, temperature, ...
- Guards: $g(X) \geq 0$
  - sets of inequalities (equalities) on functions on X
  - constrain the execution of the switch
- Jump Transformations $J(X, X')$
  - discrete transformation on the values of X
- Invariants: $X \in Inv_l(X)$
  - set of invariant constraints on X
  - ensure progress
- Continuous Flow: $\frac{dX}{dt} \in flow_l(X)$
  - set of degree-1 differential (in)equalities
  - describe continuous dynamics
- Initial: $X \in Init_l(X)$
  - initial conditions ($Init_l(X) = \perp$ iff $l \notin L^0$)



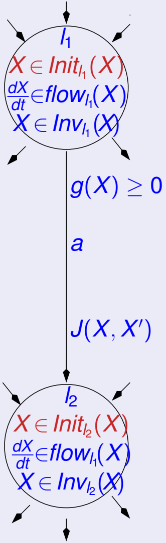
Hybrid Automata

- Locations, Switches, Labels (like in standard aut.)
- Continuous variables: $X \stackrel{\text{def}}{=} \{x_1, x_2, ..., x_k\} \in \mathbb{R}$
  - value evolves with time
  - e.g., distance, speed, pressure, temperature, ...
- Guards: $g(X) \geq 0$
  - sets of inequalities (equalities) on functions on X
  - constrain the execution of the switch
- Jump Transformations $J(X, X')$
  - discrete transformation on the values of X
- Invariants: $X \in Inv_l(X)$
  - set of invariant constraints on X
  - ensure progress
- Continuous Flow: $\frac{dX}{dt} \in flow_l(X)$
  - set of degree-1 differential (in)equalities
  - describe continuous dynamics
- Initial: $X \in Init_l(X)$
  - initial conditions ($Init_l(X) = \perp$ iff $l \notin L^0$)



Hybrid Automata

- Locations, Switches, Labels (like in standard aut.)
- Continuous variables: $X \stackrel{\text{def}}{=} \{x_1, x_2, ..., x_k\} \in \mathbb{R}$
  - value evolves with time
  - e.g., distance, speed, pressure, temperature, ...
- Guards: $g(X) \geq 0$
  - sets of inequalities (equalities) on functions on X
  - constrain the execution of the switch
- Jump Transformations $J(X, X')$
  - discrete transformation on the values of X
- Invariants: $X \in Inv_l(X)$
  - set of invariant constraints on X
  - ensure progress
- Continuous Flow: $\frac{dX}{dt} \in flow_l(X)$
  - set of degree-1 differential (in)equalities
  - describe continuous dynamics
- Initial: $X \in Init_l(X)$
  - initial conditions ($Init_l(X) = \bot$ iff $l \notin L^0$)



Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E\rangle$

- L: Set of locations,
- $L^0 \in L$: Set of initial locations (s.t. $Init_l(X) = \perp$ iff $l \notin L_0$)
- X: Set of k continuous variables
- $\Phi(X)$: Set of Constraints on X
- Σ: Set of synchronization labels (alphabet)
- E: Set of edges
- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi\rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - region ψ: subset of $\mathbb{R}^k$
- For each location l:
  - Initial states: region $Init_l(X)$
  - Invariant: region $Inv_l(X)$
  - Continuous dynamics: $\frac{dX}{dt} \in flow_l(X)$
- For each edge e from location l to location l'
  - Guard: region $g(X) \geq 0$
  - Update relation "Jump" $J(X, X')$ over $\mathbb{R}^k \times \mathbb{R}^k$
  - Synchronization label $a \in \Sigma$ (communication information)

Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E \rangle$

- L: Set of locations,
- $L^0 \in L$: Set of initial locations (s.t. $Init_l(X) = \perp$ iff $l \notin L_0$)
- X: Set of k continuous variables
- $\Phi(X)$: Set of Constraints on X
- Σ: Set of synchronization labels (alphabet)
- E: Set of edges
- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi \rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - region ψ: subset of $\mathbb{R}^k$
- For each location l:
  - Initial states: region $Init_l(X)$
  - Invariant: region $Inv_l(X)$
  - Continuous dynamics: $\frac{dX}{dt} \in flow_l(X)$
- For each edge e from location l to location l'
  - Guard: region $g(X) \geq 0$
  - Update relation "Jump" $J(X, X')$ over $\mathbb{R}^k \times \mathbb{R}^k$
  - Synchronization label $a \in \Sigma$ (communication information)

Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E\rangle$

- L: Set of locations,
- $L^0 \in L$: Set of initial locations (s.t. $Init_l(X) = \perp$ iff $l \notin L_0$)
- X: Set of k continuous variables
- $\Phi(X)$: Set of Constraints on X
- Σ: Set of synchronization labels (alphabet)
- E: Set of edges
- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi\rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - region ψ: subset of $\mathbb{R}^k$
- For each location l:
  - Initial states: region $Init_l(X)$
  - Invariant: region $Inv_l(X)$
  - Continuous dynamics: $\frac{dX}{dt} \in flow_l(X)$
- For each edge e from location l to location l'
  - Guard: region $g(X) \geq 0$
  - Update relation "Jump" $J(X, X')$ over $\mathbb{R}^k \times \mathbb{R}^k$
  - Synchronization label $a \in \Sigma$ (communication information)

Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E \rangle$

- L: Set of locations,
- $L^0 \in L$: Set of initial locations (s.t. $Init_l(X) = \perp$ iff $l \notin L_0$)
- X: Set of k continuous variables
- $\Phi(X)$: Set of Constraints on X
- Σ: Set of synchronization labels (alphabet)
- E: Set of edges
- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi \rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - region ψ: subset of $\mathbb{R}^k$
- For each location l:
  - Initial states: region $Init_l(X)$
  - Invariant: region $Inv_l(X)$
  - Continuous dynamics: $\frac{dX}{dt} \in flow_l(X)$
- For each edge e from location l to location l'
  - Guard: region $g(X) \geq 0$
  - Update relation "Jump" $J(X, X')$ over $\mathbb{R}^k \times \mathbb{R}^k$
  - Synchronization label $a \in \Sigma$ (communication information)

Remark: Degree of $flow_l(X)$

- Continuous dynamics described w.l.o.g. with sets of degree-1 differential (in)equalities $flow_l(X)$
- Sets/conjunctions of higher-degree differential (in)equalities can be reduced to degree 1 by renaming
- Ex:

$$\left(a_1 \frac{d^2 s}{dt^2} + a_2 \frac{ds}{dt} + a_3 s + a_4 \bowtie 0\right), \quad \bowtie \in \{\leq, <, \geq, >, =\}$$
$$\Downarrow$$
$$\left(v = \frac{ds}{dt}\right) \wedge \left(a_1 \frac{dv}{dt} + a_2 v + a_3 s + a_4 \bowtie 0\right), \quad \bowtie \in \{\leq, <, \geq, >, =\}$$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X\rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X\rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X\rangle \xrightarrow{a} \langle l', X'\rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X'\rangle$ satisfies the jump condition of e
  - Continuous flows: $\langle l, X\rangle \xrightarrow{f} \langle l, X'\rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), \dots, f_k(t)\rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X \rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X \rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X \rangle \xrightarrow{a} \langle l', X' \rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X' \rangle$ satisfies the jump condition of e (i.e., $\langle X, X' \rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X \rangle \xrightarrow{f} \langle l, X' \rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t) \rangle \; : \; [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X \rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X \rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X \rangle \xrightarrow{a} \langle l', X' \rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X' \rangle$ satisfies the jump condition of e (i.e., $\langle X, X' \rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X \rangle \xrightarrow{f} \langle l, X' \rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t) \rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X\rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X\rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X\rangle \xrightarrow{a} \langle l', X'\rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X'\rangle$ satisfies the jump condition of e (i.e., $\langle X, X'\rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X\rangle \xrightarrow{f} \langle l, X'\rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t)\rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X \rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X \rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X \rangle \xrightarrow{a} \langle l', X' \rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X' \rangle$ satisfies the jump condition of e (i.e., $\langle X, X' \rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X \rangle \xrightarrow{f} \langle l, X' \rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t) \rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X\rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X\rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X\rangle \xrightarrow{a} \langle l', X'\rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X'\rangle$ satisfies the jump condition of e (i.e., $\langle X, X'\rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X\rangle \xrightarrow{f} \langle l, X'\rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t)\rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X\rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X\rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X\rangle \xrightarrow{a} \langle l', X'\rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X'\rangle$ satisfies the jump condition of e (i.e., $\langle X, X'\rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X\rangle \xrightarrow{f} \langle l, X'\rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t)\rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X \rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X \rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X \rangle \xrightarrow{a} \langle l', X' \rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X' \rangle$ satisfies the jump condition of e (i.e., $\langle X, X' \rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X \rangle \xrightarrow{f} \langle l, X' \rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t) \rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X \rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X \rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X \rangle \xrightarrow{a} \langle l', X' \rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X' \rangle$ satisfies the jump condition of e (i.e., $\langle X, X' \rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X \rangle \xrightarrow{f} \langle l, X' \rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t) \rangle \; : \; [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X\rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X\rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X\rangle \xrightarrow{a} \langle l', X'\rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X'\rangle$ satisfies the jump condition of e (i.e., $\langle X, X'\rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X\rangle \xrightarrow{f} \langle l, X'\rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t)\rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

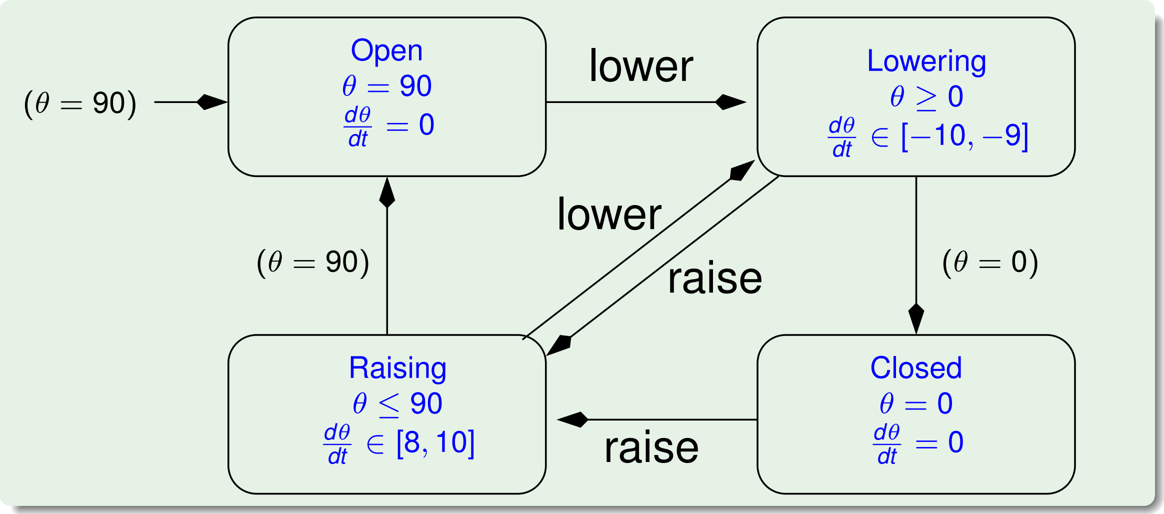
(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X\rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X\rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X\rangle \xrightarrow{a} \langle l', X'\rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X'\rangle$ satisfies the jump condition of e (i.e., $\langle X, X'\rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X\rangle \xrightarrow{f} \langle l, X'\rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t)\rangle \;:\; [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

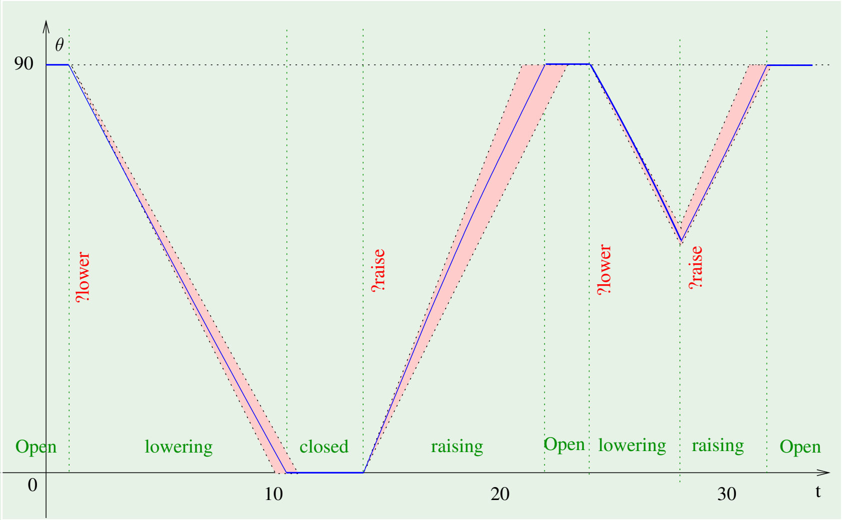
(Finite) Executions of Hybrid Automata

- State: pair $\langle l, X\rangle$ such that $X \in Inv_l(X)$
- Initialization: $\langle l, X\rangle$ such that $X \in Init_l(X)$
- Two types of state updates (transitions)
  - Discrete switches: $\langle l, X\rangle \xrightarrow{a} \langle l', X'\rangle$
    if there there is an a-labeled edge e from l to l' s.t.
    - X, X' satisfy $Inv_l(X)$ and $Inv_{l'}(X)$ respectively
    - X satisfies the guard of e (i.e. $g(X) \geq 0$) and
    - $\langle X, X'\rangle$ satisfies the jump condition of e (i.e., $\langle X, X'\rangle \in J(X, X')$)
  - Continuous flows: $\langle l, X\rangle \xrightarrow{f} \langle l, X'\rangle$
    $f(t) \stackrel{\text{def}}{=} \langle f_1(t), ..., f_k(t)\rangle \ : \ [0, \delta] \longmapsto \mathbb{R}^k$ is a continuous function s.t.
    - $f(0) = X$
    - $f(\delta) = X'$
    - for every $t \in [0, \delta]$, $f(t) \in Inv_l(X)$
    - for every $t \in [0, \delta]$, $\frac{df(t)}{dt} \in flow_l(X)$

Example: Gate for a railroad controller



Example: Gate for a railroad controller



Outline

1. Motivations
2. Timed systems: Modeling and Semantics
   - Timed automata
   - Semantics
   - Combination
3. Symbolic Reachability for Timed Systems
   - Making the state space finite
   - Region automata
   - Zone automata
4. Hybrid Systems: Modeling and Semantics
   - Hybrid automata
5. Symbolic Reachability for Hybrid Systems
   - Multi-Rate and Rectangular Hybrid Automata
   - Linear Hybrid Automata
6. Exercises

General Symbolic-Reachability Schema

```
F = R = I(X)
while (F ≠ ∅) do
    if (R intersects F) then
        return True
    else
        if (Image(F) ⊆ R) then
            return False
        else
            R_old = R
            R = R ∪ Image(F)
            F = R \ R_old
        end if
    end if
end while
```

- I: initial; F: Final; R: Reachable; Image(F): successors of F
- need a data type to represent state sets (regions)
- Termination may or may not be guaranteed

Symbolic Representations

- Necessary operations on Regions
  - Union
  - Intersection
  - Negation
  - Projection
  - Renaming
  - Equality/containment test
  - Emptiness test
- Different choices for different classes of problems
  - BDDs for Boolean variables in hardware verification
  - DBMs in Timed automata
  - Polyhedra in Linear Hybrid Automata
  - ...

Symbolic Representations

- Necessary operations on Regions
  - Union
  - Intersection
  - Negation
  - Projection
  - Renaming
  - Equality/containment test
  - Emptiness test
- Different choices for different classes of problems
  - BDDs for Boolean variables in hardware verification
  - DBMs in Timed automata
  - Polyhedra in Linear Hybrid Automata
  - ...

Reachability for Hybrid Systems

- Same algorithm works in principle
- Problem: What is a suitable representation of regions?
  - Region: subset of $\mathbb{R}^k$
  - Main problem: handling continuous dynamics
- Precise solutions available for restricted continuous dynamics
  - Timed automata
  - Multi-rate & Rectangular Hybrid Automata (reduced to Timed aut.)
  - Linear Hybrid Automata
- Even for linear systems, over-approximations of reachable set needed

Reachability for Hybrid Systems

- Same algorithm works in principle
- Problem: What is a suitable representation of regions?
  - Region: subset of $\mathbb{R}^k$
  - Main problem: handling continuous dynamics
- Precise solutions available for restricted continuous dynamics
  - Timed automata
  - Multi-rate & Rectangular Hybrid Automata (reduced to Timed aut.)
  - Linear Hybrid Automata
- Even for linear systems, over-approximations of reachable set needed

Reachability for Hybrid Systems

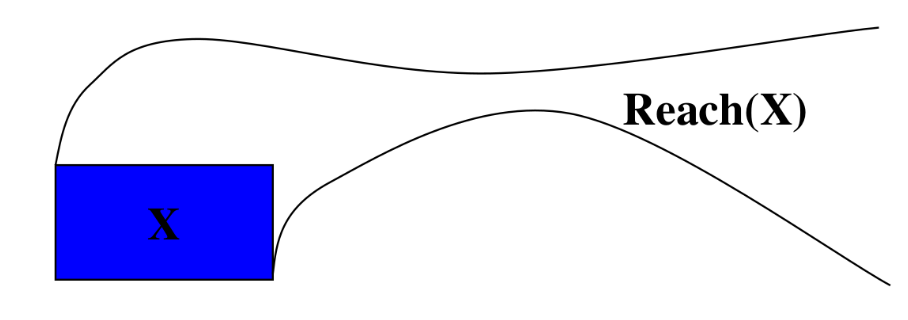
- Same algorithm works in principle
- Problem: What is a suitable representation of regions?
  - Region: subset of $\mathbb{R}^k$
  - Main problem: handling continuous dynamics
- Precise solutions available for restricted continuous dynamics
  - Timed automata
  - Multi-rate & Rectangular Hybrid Automata (reduced to Timed aut.)
  - Linear Hybrid Automata
- Even for linear systems, over-approximations of reachable set needed

Reachability for Hybrid Systems

- Same algorithm works in principle
- Problem: What is a suitable representation of regions?
  - Region: subset of $\mathbb{R}^k$
  - Main problem: handling continuous dynamics
- Precise solutions available for restricted continuous dynamics
  - Timed automata
  - Multi-rate & Rectangular Hybrid Automata (reduced to Timed aut.)
  - Linear Hybrid Automata
- Even for linear systems, over-approximations of reachable set needed

Reachability Analysis for Dynamical Systems

- Goal: Given an initial region, compute whether a bad state can be reached
- Key step: compute Reach(X) for a given set X under $\frac{dX}{dt} = f(X)$



Notation: (hereafter we often use "dX" or "$\dot{X}$" as a shortcut of "$\frac{dX}{dt}$"

Outline

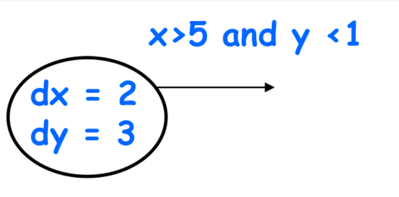
Simple Hybrid Automata: Multi-Rate and Rectangular

Two simple forms of Hybrid Automata

- Multi-Rate Automata
- Rectangular Automata
- Idea: can be reduced to Timed Automata
- Typically used as over-approximations of complex hybrid automata

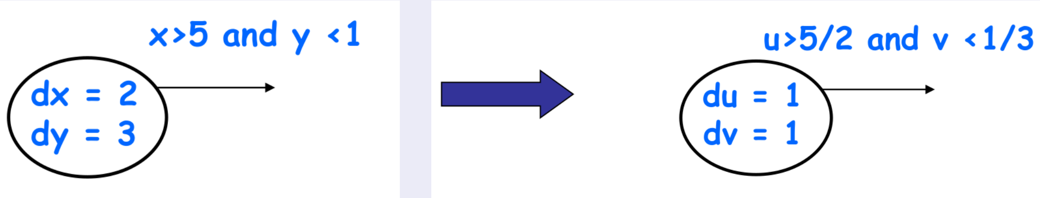
Multi-rate Automata

- Modest extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} = const$
  - Guards and invariants: $x < const$, $x > const$
  - Resets: $x := const$
- Simple translation to timed automata by shifting and scaling:
  - if $x_i := d_i$ then rename it with a fresh var v_i s.t. $v_i + d_i = x_i$
  - if $\frac{dx_i}{dt} = c_i$, then rename it with a fresh var u_i s.t. $c_i \cdot u_i = x_i$
  - shift & rescale constants in constraints accordingly



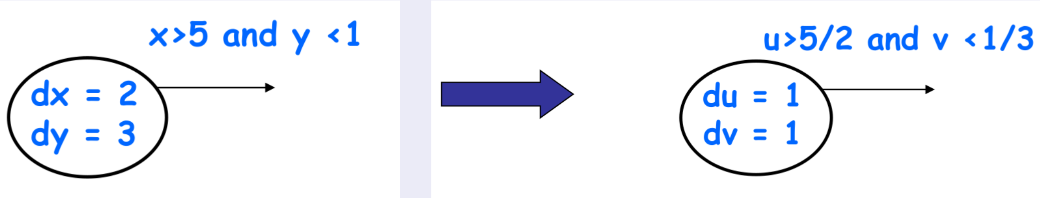
Multi-rate Automata

- Modest extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} = const$
  - Guards and invariants: $x < const$, $x > const$
  - Resets: $x := const$
- Simple translation to timed automata by shifting and scaling:
  - if $x_i := d_i$ then rename it with a fresh var v_i s.t. $v_i + d_i = x_i$
  - if $\frac{dx_i}{dt} = c_i$, then rename it with a fresh var u_i s.t. $c_i \cdot u_i = x_i$
  - shift & rescale constants in constraints accordingly



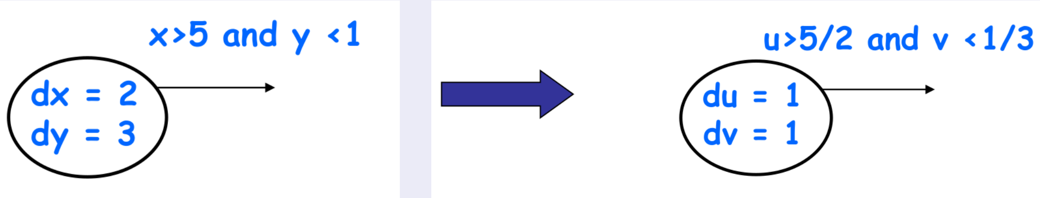
Multi-rate Automata

- Modest extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} = const$
  - Guards and invariants: $x < const$, $x > const$
  - Resets: $x := const$
- Simple translation to timed automata by shifting and scaling:
  - if $x_i := d_i$ then rename it with a fresh var v_i s.t. $v_i + d_i = x_i$
  - if $\frac{dx_i}{dt} = c_i$, then rename it with a fresh var u_i s.t. $c_i \cdot u_i = x_i$
  - shift & rescale constants in constraints accordingly



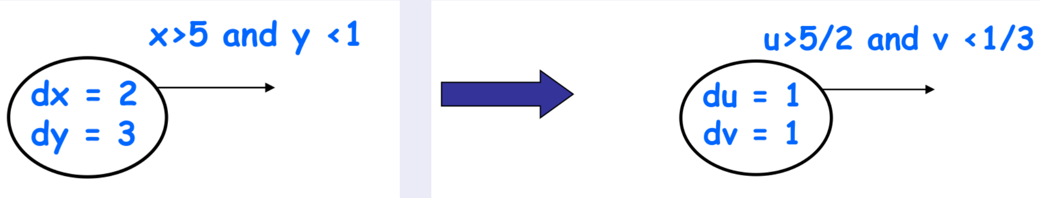
Multi-rate Automata

- Modest extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} = const$
  - Guards and invariants: $x < const$, $x > const$
  - Resets: $x := const$
- Simple translation to timed automata by shifting and scaling:
  - if $x_i := d_i$ then rename it with a fresh var v_i s.t. $v_i + d_i = x_i$
  - if $\frac{dx_i}{dt} = c_i$, then rename it with a fresh var u_i s.t. $c_i \cdot u_i = x_i$
  - shift & rescale constants in constraints accordingly



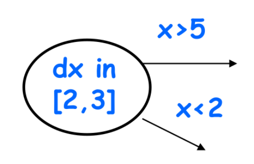
Multi-rate Automata

- Modest extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} = const$
  - Guards and invariants: $x < const$, $x > const$
  - Resets: $x := const$
- Simple translation to timed automata by shifting and scaling:
  - if $x_i := d_i$ then rename it with a fresh var v_i s.t. $v_i + d_i = x_i$
  - if $\frac{dx_i}{dt} = c_i$, then rename it with a fresh var u_i s.t. $c_i \cdot u_i = x_i$
  - shift & rescale constants in constraints accordingly



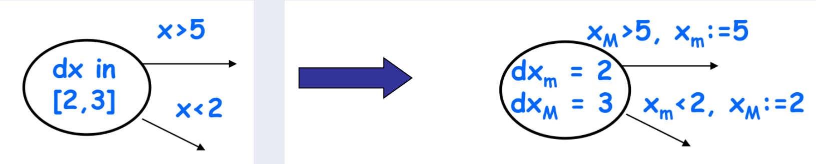
Rectangular Automata (simplified)

- More interesting extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} \in [const1, const2]$ ($\dot{x} \in [const1, const2]$)
  - Guards and invariants: $x < const$, $x > const$
  - Jumps: $x := const$
- Translation to multi-rate automata (hints). For each x:
  - Introduce x_M, x_m describing the greatest/least possible x values
  - flow: substitute $\dot{x} < c_u$ with $\dot{x}_M = c_u$ and $\dot{x} > c_l$ with $\dot{x}_m = c_l$
  - invariants: substitute $Inv_l(x)$ with $Inv_l(x_M), Inv_l(x_m)$
  - guards: substitute $x > c$ with $x_M > c$, add jump $x_m := c$ (if none)
    substitute $x < c$ with $x_m < c$, add jump $x_M := c$ (if none)
  - jump: if $x := c$, then both $x_M := c$ and $x_m := c$



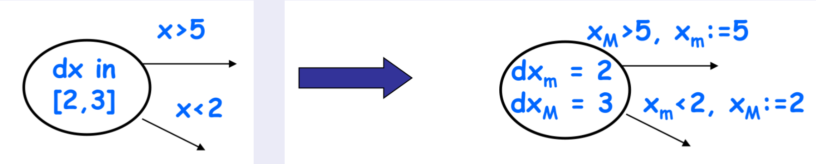
Rectangular Automata (simplified)

- More interesting extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} \in [const1, const2]$ ($\dot{x} \in [const1, const2]$)
  - Guards and invariants: $x < const$, $x > const$
  - Jumps: $x := const$
- Translation to multi-rate automata (hints). For each x:
  - Introduce x_M, x_m describing the greatest/least possible x values
  - flow: substitute $\dot{x} < c_u$ with $\dot{x}_M = c_u$ and $\dot{x} > c_l$ with $\dot{x}_m = c_l$
  - invariants: substitute $Inv_l(x)$ with $Inv_l(x_M), Inv_l(x_m)$
  - guards: substitute $x > c$ with $x_M > c$, add jump $x_m := c$ (if none)
    guards: substitute $x < c$ with $x_m < c$, add jump $x_M := c$ (if none)
  - jump: if $x := c$, then both $x_M := c$ and $x_m := c$



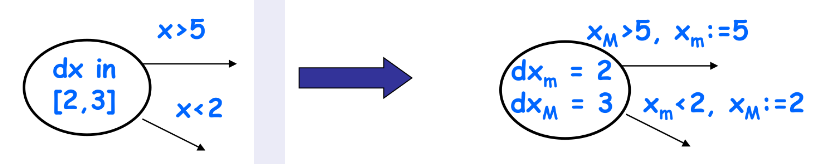
Rectangular Automata (simplified)

- More interesting extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} \in [const1, const2]$ ($\dot{x} \in [const1, const2]$)
  - Guards and invariants: $x < const$, $x > const$
  - Jumps: $x := const$
- Translation to multi-rate automata (hints). For each x:
  - Introduce x_M, x_m describing the greatest/least possible x values
  - flow: substitute $\dot{x} < c_u$ with $\dot{x}_M = c_u$ and $\dot{x} > c_l$ with $\dot{x}_m = c_l$
  - invariants: substitute $Inv_l(x)$ with $Inv_l(x_M), Inv_l(x_m)$
  - guards: substitute $x > c$ with $x_M > c$, add jump $x_m := c$ (if none)
    guards: substitute $x < c$ with $x_m < c$, add jump $x_M := c$ (if none)
  - jump: if $x := c$, then both $x_M := c$ and $x_m := c$



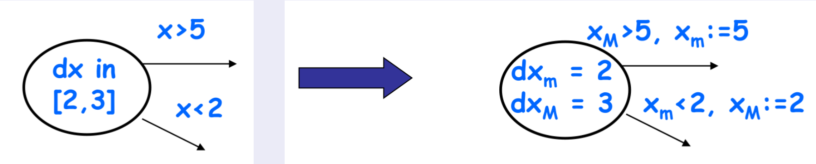
Rectangular Automata (simplified)

- More interesting extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} \in [const1, const2]$ ($\dot{x} \in [const1, const2]$)
  - Guards and invariants: $x < const$, $x > const$
  - Jumps: $x := const$
- Translation to multi-rate automata (hints). For each x:
  - Introduce x_M, x_m describing the greatest/least possible x values
  - flow: substitute $\dot{x} < c_u$ with $\dot{x}_M = c_u$ and $\dot{x} > c_l$ with $\dot{x}_m = c_l$
  - invariants: substitute $Inv_l(x)$ with $Inv_l(x_M), Inv_l(x_m)$
  - guards: substitute $x > c$ with $x_M > c$, add jump $x_m := c$ (if none)
    guards: substitute $x < c$ with $x_m < c$, add jump $x_M := c$ (if none)
  - jump: if $x := c$, then both $x_M := c$ and $x_m := c$



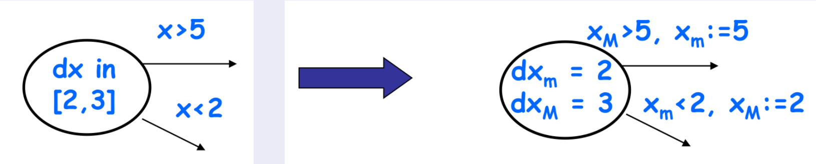
Rectangular Automata (simplified)

- More interesting extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} \in [const1, const2]$ ($\dot{x} \in [const1, const2]$)
  - Guards and invariants: $x < const$, $x > const$
  - Jumps: $x := const$
- Translation to multi-rate automata (hints). For each x:
  - Introduce x_M, x_m describing the greatest/least possible x values
  - flow: substitute $\dot{x} < c_u$ with $\dot{x}_M = c_u$ and $\dot{x} > c_l$ with $\dot{x}_m = c_l$
  - invariants: substitute $Inv_l(x)$ with $Inv_l(x_M), Inv_l(x_m)$
  - guards: substitute $x > c$ with $x_M > c$, add jump $x_m := c$ (if none)
    guards: substitute $x < c$ with $x_m < c$, add jump $x_M := c$ (if none)
  - jump: if $x := c$, then both $x_M := c$ and $x_m := c$



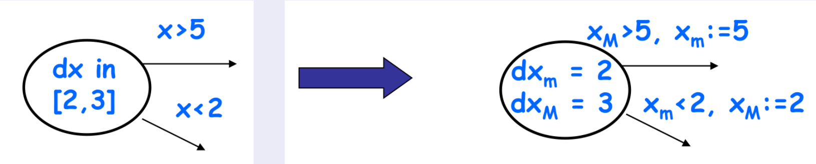
Rectangular Automata (simplified)

- More interesting extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} \in [const1, const2]$ ($\dot{x} \in [const1, const2]$)
  - Guards and invariants: $x < const$, $x > const$
  - Jumps: $x := const$
- Translation to multi-rate automata (hints). For each x:
  - Introduce x_M, x_m describing the greatest/least possible x values
  - flow: substitute $\dot{x} < c_u$ with $\dot{x}_M = c_u$ and $\dot{x} > c_l$ with $\dot{x}_m = c_l$
  - invariants: substitute $Inv_l(x)$ with $Inv_l(x_M), Inv_l(x_m)$
  - guards: substitute $x > c$ with $x_M > c$, add jump $x_m := c$ (if none)
    guards: substitute $x < c$ with $x_m < c$, add jump $x_M := c$ (if none)
  - jump: if $x := c$, then both $x_M := c$ and $x_m := c$



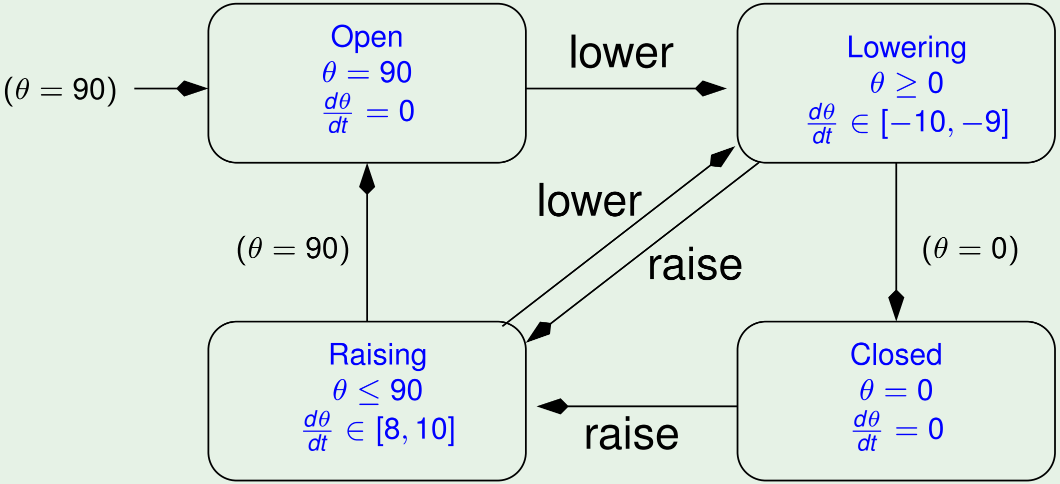
Rectangular Automata (simplified)

- More interesting extension of timed automata
  - Dynamics of the form $\frac{dX}{dt} \in [const1, const2]$ ($\dot{x} \in [const1, const2]$)
  - Guards and invariants: $x < const$, $x > const$
  - Jumps: $x := const$
- Translation to multi-rate automata (hints). For each x:
  - Introduce x_M, x_m describing the greatest/least possible x values
  - flow: substitute $\dot{x} < c_u$ with $\dot{x}_M = c_u$ and $\dot{x} > c_l$ with $\dot{x}_m = c_l$
  - invariants: substitute $Inv_l(x)$ with $Inv_l(x_M), Inv_l(x_m)$
  - guards: substitute $x > c$ with $x_M > c$, add jump $x_m := c$ (if none)
    guards: substitute $x < c$ with $x_m < c$, add jump $x_M := c$ (if none)
  - jump: if $x := c$, then both $x_M := c$ and $x_m := c$



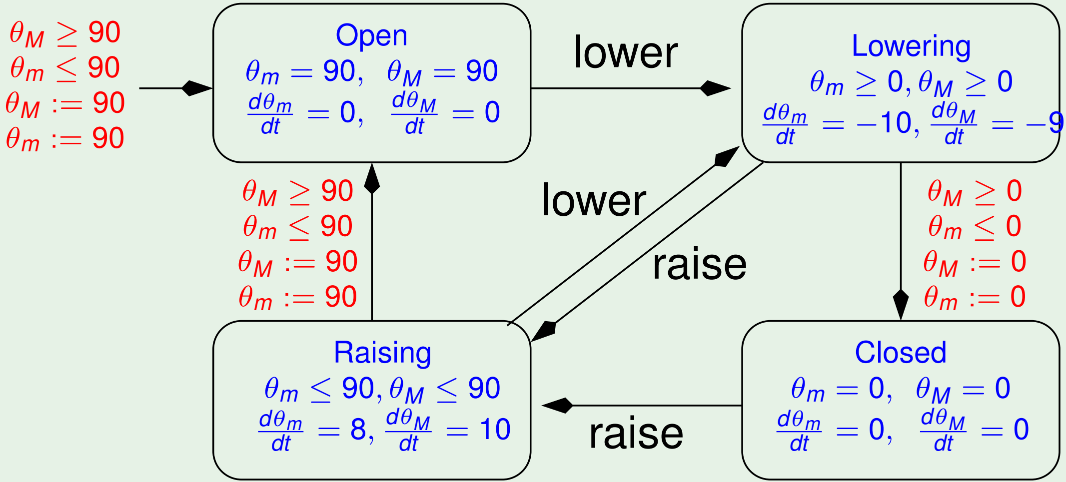
Example: Gate for a railroad controller

Rectangular Automaton



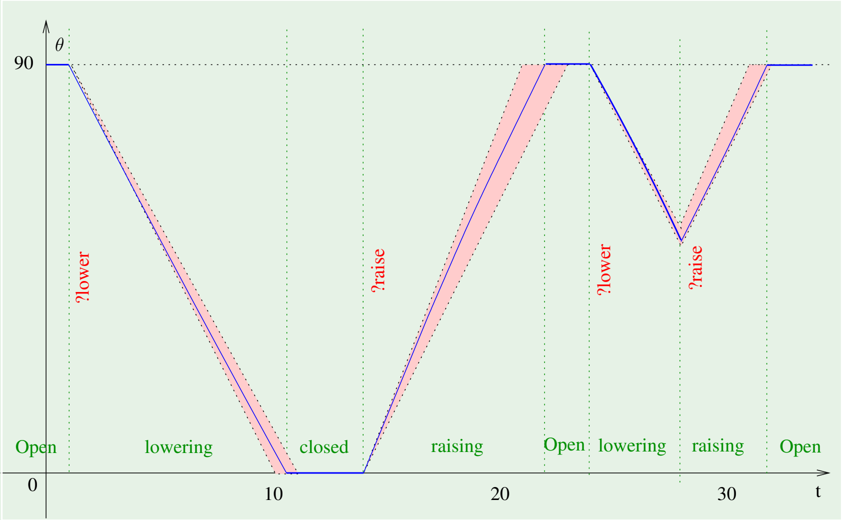
Example: Gate for a railroad controller

Multi-rate Automaton



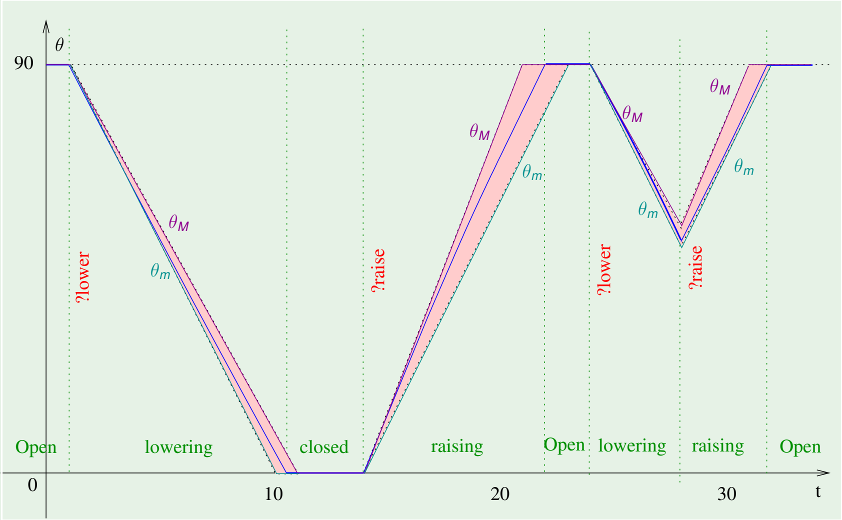
Example: Gate for a railroad controller

Rectangular automaton



Example: Gate for a railroad controller

Multi-rate automaton



Outline

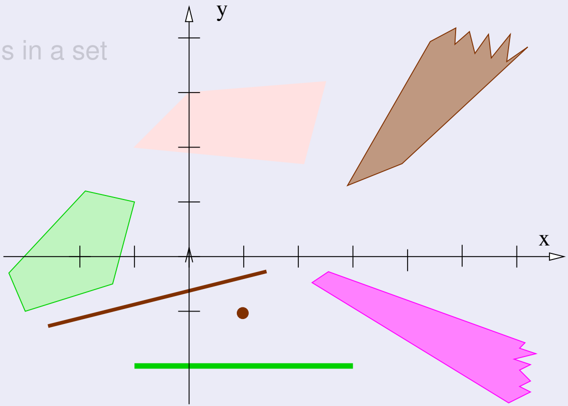
Linear Hybrid Automata

- Polyhedron φ: set/conjunction of linear inequalities on X in the form $(A \cdot X \geq B)$, s.t. $A \in \mathbb{R}^m \times \mathbb{R}^k$ and $B \in \mathbb{R}^m$ for some m.
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible values for all variables in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: polyhedron

(generalization of zone automata)



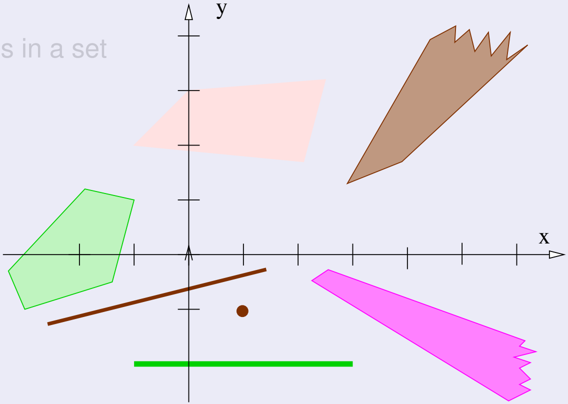
Linear Hybrid Automata

- Polyhedron φ: set/conjunction of linear inequalities on X in the form $(A \cdot X \geq B)$, s.t. $A \in \mathbb{R}^m \times \mathbb{R}^k$ and $B \in \mathbb{R}^m$ for some m.
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible values for all variables in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: polyhedron

(generalization of zone automata)



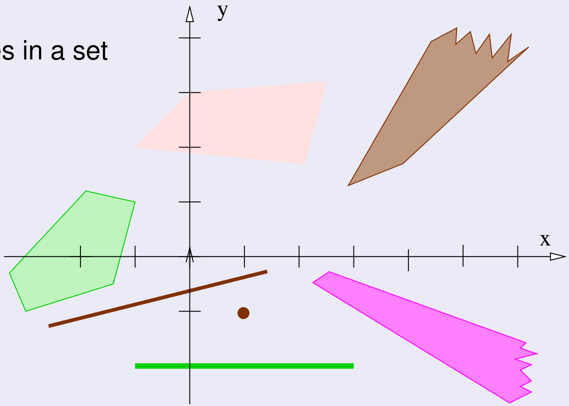
Linear Hybrid Automata

- Polyhedron φ: set/conjunction of linear inequalities on X in the form $(A \cdot X \geq B)$, s.t. $A \in \mathbb{R}^m \times \mathbb{R}^k$ and $B \in \mathbb{R}^m$ for some m.
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible values for all variables in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: polyhedron

(generalization of zone automata)



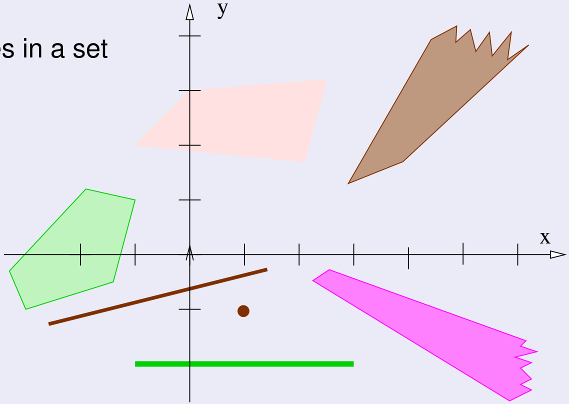
Linear Hybrid Automata

- Polyhedron φ: set/conjunction of linear inequalities on X in the form $(A \cdot X \geq B)$, s.t. $A \in \mathbb{R}^m \times \mathbb{R}^k$ and $B \in \mathbb{R}^m$ for some m.
- φ is a convex set in the k-dimensional euclidean space
  - possibly unbounded

$\Longrightarrow$ Contains all possible values for all variables in a set

- Symbolic state: $\langle l, \varphi \rangle$
  - l: location
  - φ: polyhedron

(generalization of zone automata)



Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E\rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi\rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives

Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E \rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi \rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives

Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E\rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi\rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region ($A \cdot X \geq B$): polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives
Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E\rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi\rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives

Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E\rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi\rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives
Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E\rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi\rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives
Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E \rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi \rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation “Jump” $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives

Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E \rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi \rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives
Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E \rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi \rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives

Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E \rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi \rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives
Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

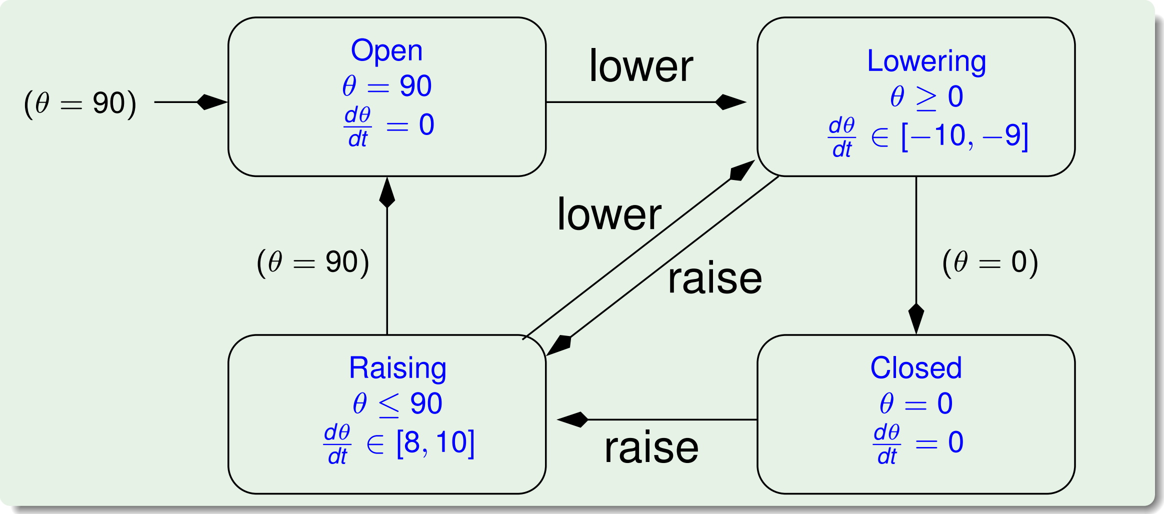
Linear Hybrid Automata $A = \langle L, L^0, X, \Sigma, \Phi(X), E\rangle$

- State space: $L \times \mathbb{R}^k$,
  - state: $\langle l, \psi\rangle$ s.t. $l \in L$ and $\psi \in \mathbb{R}^k$
  - polyhedron ψ: subset of $\mathbb{R}^k$ in the form $A \cdot X \geq B$
- For each edge e from location l to location l'
  - Guard: region $(A \cdot X \geq B)$: polyhedron on X
  - Update relation "Jump" $J(X, X')$: $X' := T \cdot X + B$, $T \in \mathbb{R}^k \times \mathbb{R}^k$, $B \in \mathbb{R}$
  - Synchronization label $a \in \Sigma$ (communication information)
- For each location l:
  - Initial states: region $Init_l(X)$: polyhedron on X
  - Invariant: region $Inv_l(X)$: polyhedron on X
  - Continuous dynamics $flow_l(X)$: polyhedron on $\frac{dX}{dt}$

Continuous Dynamics

Time-invariant, state-independent dynamics specified by a convex polyhedron constraining first derivatives
Es: $\frac{dx}{dt} \geq 3$, $\frac{dx}{dt} = \frac{dy}{dt}$, $2.1\frac{dx}{dt} - 3.5\frac{dy}{dt} + 1.7\frac{dz}{dt} \geq 3.1$, ...

Example: Gate for a railroad controller



Reachability Computation: Key Steps

- Compute "discrete" successors of $\langle l, \psi \rangle$
- Compute "continuous" successor of $\langle l, \psi \rangle$
- Check if ψ intersects with "bad" region
- Check if newly-found ψ is covered by already-visited polyhedra $\psi_1, \ldots, \psi_n$ (expensive!)

Reachability Computation: Key Steps

- Compute "discrete" successors of $\langle l, \psi \rangle$
- Compute "continuous" successor of $\langle l, \psi \rangle$
- Check if ψ intersects with "bad" region
- Check if newly-found ψ is covered by already-visited polyhedra $\psi_1, ..., \psi_n$ (expensive!)

Reachability Computation: Key Steps

- Compute “discrete” successors of $\langle I, \psi \rangle$
- Compute “continuous” successor of $\langle I, \psi \rangle$
- Check if ψ intersects with “bad” region
- Check if newly-found ψ is covered by already-visited polyhedra $\psi_1, \ldots, \psi_n$ (expensive!)

Reachability Computation: Key Steps

- Compute "discrete" successors of $\langle I, \psi \rangle$
- Compute "continuous" successor of $\langle I, \psi \rangle$
- Check if ψ intersects with "bad" region
- Check if newly-found ψ is covered by already-visited polyhedra $\psi_1, ..., \psi_n$ (expensive!)

Computing Discrete Successors of $\langle l, \psi \rangle$

- Intersect ψ with the guard ϕ
  $\Longrightarrow$ result is a polyhedron
- Apply linear transformation of J to the result
  $\Longrightarrow$ result is a polyhedron
- Intersect with the invariant of target location l'
  $\Longrightarrow$ result is a polyhedron

Computing Discrete Successors of $\langle l, \psi \rangle$

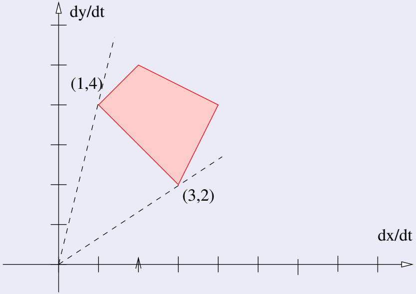
- Intersect ψ with the guard ϕ
  $\Longrightarrow$ result is a polyhedron
- Apply linear transformation of J to the result
  $\Longrightarrow$ result is a polyhedron
- Intersect with the invariant of target location l'
  $\Longrightarrow$ result is a polyhedron

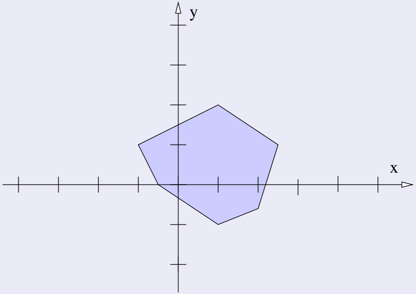
Computing Discrete Successors of $\langle l, \psi \rangle$

- Intersect ψ with the guard ϕ
  $\Longrightarrow$ result is a polyhedron
- Apply linear transformation of J to the result
  $\Longrightarrow$ result is a polyhedron
- Intersect with the invariant of target location l'
  $\Longrightarrow$ result is a polyhedron

Computing Time Successor

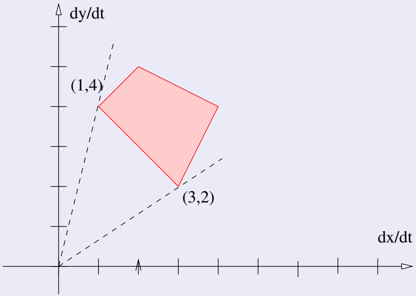
- Consider maximum and minimum rates between derivatives (external vertices in the flow polyhedron)
- Apply these extremal rates for computing the projection to infinity (to be intersected with invariant)
  - Hint: $\frac{dy}{dx} = \frac{\frac{dy}{dt}}{\frac{dx}{dt}}$, s.t. $max_{x,y} \frac{dy}{dx} = max_{x,y} \frac{\frac{dy}{dt}}{\frac{dx}{dt}}$ and $min_{x,y} \frac{dy}{dx} = min_{x,y} \frac{\frac{dy}{dt}}{\frac{dx}{dt}}$

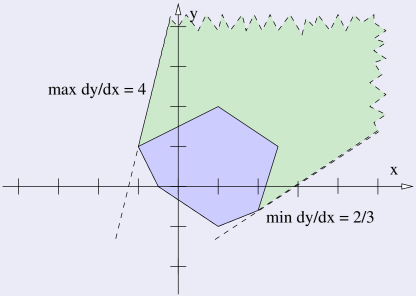




Computing Time Successor

- Consider maximum and minimum rates between derivatives (external vertices in the flow polyhedron)
- Apply these extremal rates for computing the projection to infinity (to be intersected with invariant)
  - Hint: $\frac{dy}{dx} = \frac{\frac{dy}{dt}}{\frac{dx}{dt}}$, s.t. $max_{x,y}\frac{dy}{dx} = max_{x,y}\frac{\frac{dy}{dt}}{\frac{dx}{dt}}$ and $min_{x,y}\frac{dy}{dx} = min_{x,y}\frac{\frac{dy}{dt}}{\frac{dx}{dt}}$





Linear Hybrid Automata: Symbolic Transitions

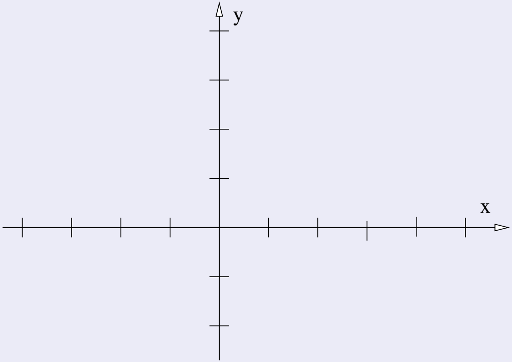
Definition: $succ(\varphi, e)$

- Let $e \stackrel{\text{def}}{=} \langle l, a, \psi, J, l' \rangle$, and ϕ, ϕ' the invariants in l, l'
- Then
$$succ(\varphi, e) \stackrel{\text{def}}{=} J(((\varphi \wedge \phi) \Uparrow \wedge \phi) \wedge \psi)$$
(φ immediately before entering the location)
$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi \Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$
(φ immediately after entering the location):
  - $\wedge$: standard conjunction/intersection
  - $\Uparrow$: continuous successor $\psi \Uparrow$
  - J: Jump transformation $J(X) \stackrel{\text{def}}{=} T \cdot X + B$
- note: φ is considered "immediately after entering l"

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

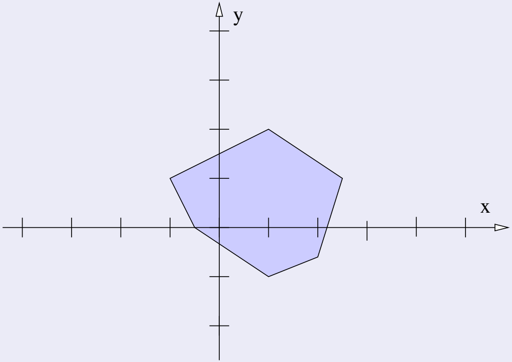


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

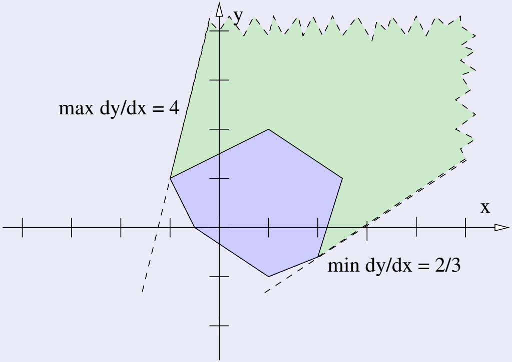


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

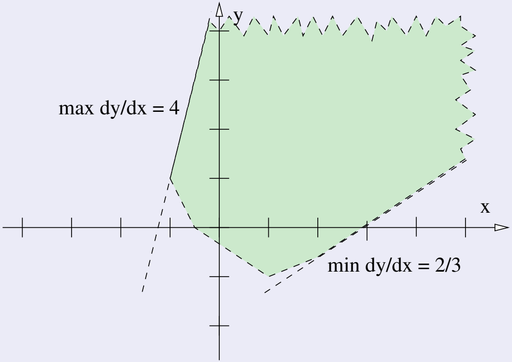


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

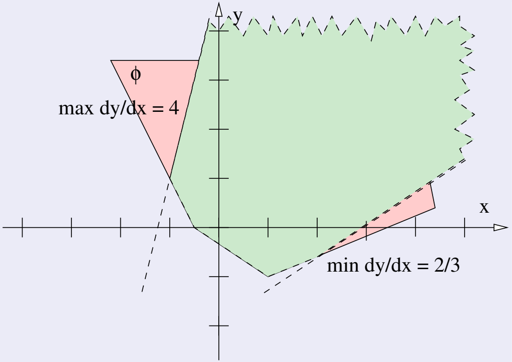


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

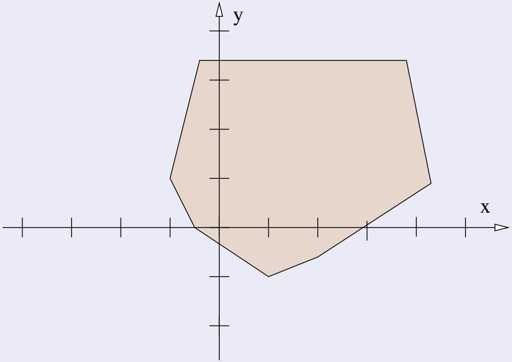


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \ \wedge\phi) \wedge \psi) \ \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

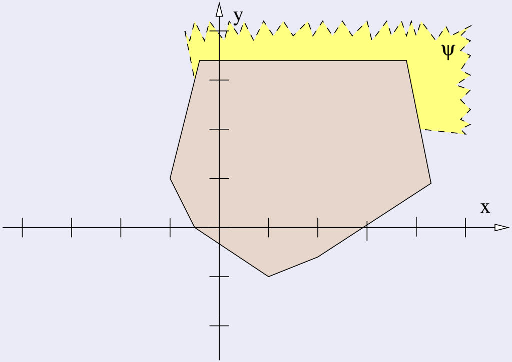


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

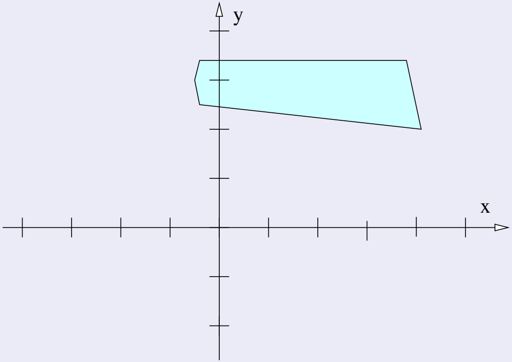


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

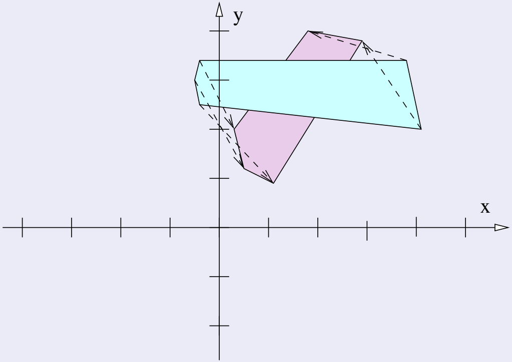


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

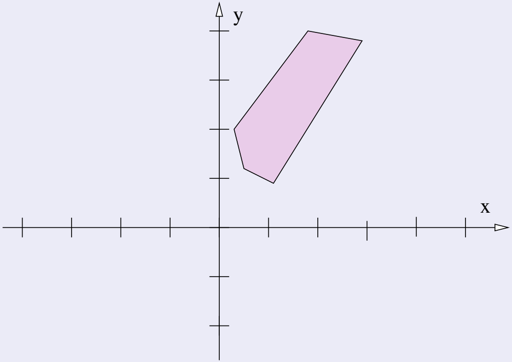


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

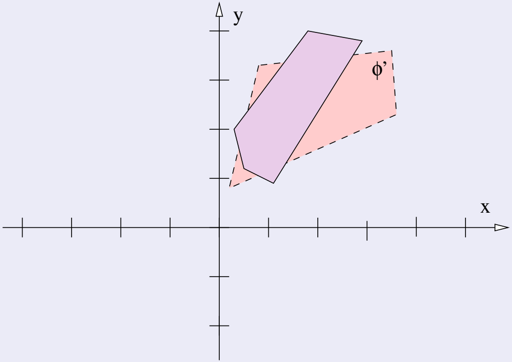


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

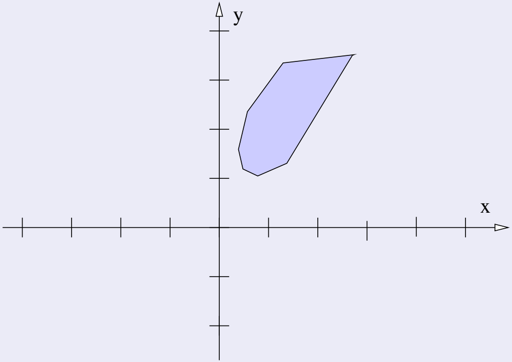


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow\ \wedge \phi) \wedge \psi)\ \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!

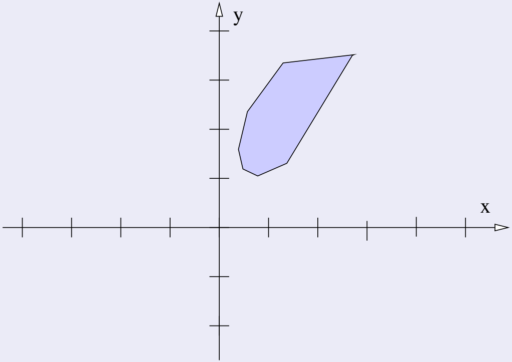


$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow\ \wedge \phi) \wedge \psi)\ \wedge \phi'$$

Linear Hybrid Automata: Symbolic Transitions (cont.)

- Initial zone: values allowed to enter location l
- Projection to infinity: ... after waiting unbounded time
- Intersection with invariant ϕ: ... waiting a legal amount of time
- Intersection with guard ψ: ... from which the switch can be shot
- Jump J: ..., after jump
- Intersection with invariant ϕ': ... values allowed to enter location l'

$\Longrightarrow$ Final!



$$succ(\varphi, e) \stackrel{\text{def}}{=} J((\varphi\Uparrow \wedge \phi) \wedge \psi) \wedge \phi'$$

Symbolic Reachability Analysis

function Reachable (A, F) // $A \stackrel{\text{def}}{=} \langle L, L^0, \Sigma, X, \Phi(X), E \rangle$, $F \stackrel{\text{def}}{=} \{\langle l_i, \phi_i \rangle\}_i$
    Reachable $= \emptyset$
    Frontier $= \{\langle l, Init_l(X) \rangle \mid l \in L^0\}$
    while (*Frontier* $\neq \emptyset$) **do**
        extract $\langle l, \varphi \rangle$ *from Frontier*
        if $((\varphi \wedge \phi) \neq \bot$ for some $\langle l, \phi \rangle \in F)$ **then**
            return True
        end if
        if $(\nexists \langle l, \varphi' \rangle \in$ *Reachable* $s.t.\ \varphi \subseteq \varphi')$ **then**
            add $\langle l, \varphi \rangle$ *to Reachable*
            for $e \in outcoming(l)$ **do**
                add $succ(\varphi, e)$ *to Frontier*
            end for
        end if
    end while
    return False

$\Longrightarrow$ same schema as with zone automata

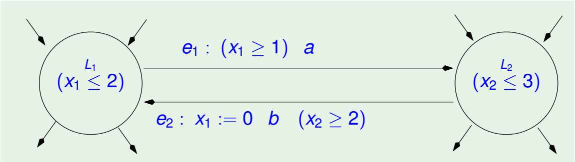
Summary: Linear Hybrid Automata

- Strategy implemented in HyTech
- Core computation: manipulation of polyhedra
- Bottlenecks
  - proliferation of polyhedra (unions)
  - computing with high-dimension polyhedra
- Many case studies

Outline

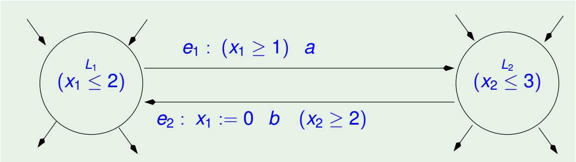
Ex: Execution of a Timed System

Consider only the following piece of a timed automaton A, x_1 and x_2 being clocks.



Ex: Execution of a Timed System

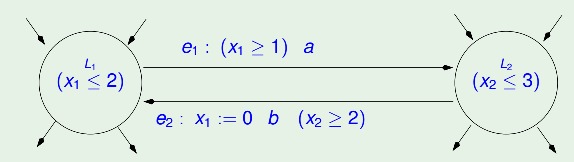
Consider only the following piece of a timed automaton A, x_1 and x_2 being clocks.



(a) In general, what is the minimum amount of time from an occurrence of event b and the subsequent occurrence of the event a?

Ex: Execution of a Timed System

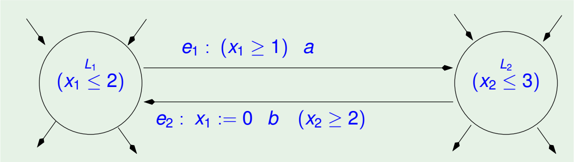
Consider <u>only</u> the following piece of a timed automaton A, x_1 and x_2 being clocks.



(*a*) In general, what is the minimum amount of time from an occurrence of event *b* and the subsequent occurrence of the event *a*? [Solution: 1 time unit.]

Ex: Execution of a Timed System

Consider only the following piece of a timed automaton A, x_1 and x_2 being clocks.

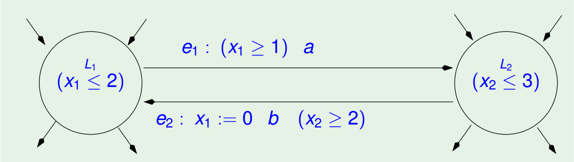


(*a*) In general, what is the minimum amount of time from an occurrence of event *b* and the subsequent occurrence of the event *a*? [Solution: 1 time unit.]

(*b*) Write a legal execution from state $\langle L_1, 0.0, 2.0\rangle$ to state $\langle L_1, 0.0, 3.0\rangle$.

Ex: Execution of a Timed System

Consider only the following piece of a timed automaton A, x_1 and x_2 being clocks.

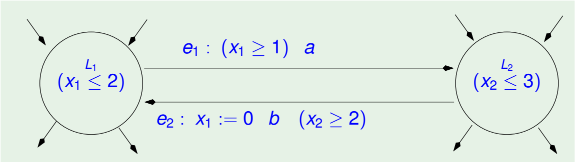


(a) In general, what is the minimum amount of time from an occurrence of event b and the subsequent occurrence of the event a? [Solution: 1 time unit.]

(b) Write a legal execution from state $\langle L_1, 0.0, 2.0\rangle$ to state $\langle L_1, 0.0, 3.0\rangle$. [Solution:
$\langle L_1, 0.0, 2.0\rangle \xrightarrow{1.0} \langle L_1, 1.0, 3.0\rangle \xrightarrow{a} \langle L_2, 1.0, 3.0\rangle \xrightarrow{0.0} \langle L_2, 1.0, 3.0\rangle \xrightarrow{b} \langle L_1, 0.0, 3.0\rangle$]

Ex: Execution of a Timed System

Consider only the following piece of a timed automaton A, x_1 and x_2 being clocks.



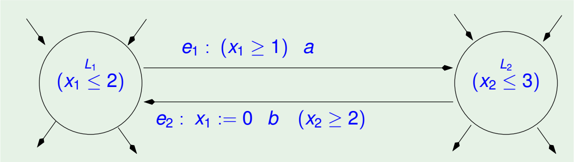
(*a*) In general, what is the minimum amount of time from an occurrence of event b and the subsequent occurrence of the event a? [Solution: 1 time unit.]

(*b*) Write a legal execution from state $\langle L_1, 0.0, 2.0\rangle$ to state $\langle L_1, 0.0, 3.0\rangle$. [Solution:
$\langle L_1, 0.0, 2.0\rangle \xrightarrow{1.0} \langle L_1, 1.0, 3.0\rangle \xrightarrow{a} \langle L_2, 1.0, 3.0\rangle \xrightarrow{0.0} \langle L_2, 1.0, 3.0\rangle \xrightarrow{b} \langle L_1, 0.0, 3.0\rangle$]

(*c*) Is it possible to have a legal execution in which switches e_2, e_1, e_2 are shot consecutively (possibly interleaved by time elapses), without being interleaved by other switches? If yes, write one such execution. If not, explain why.

Ex: Execution of a Timed System

Consider only the following piece of a timed automaton A, x_1 and x_2 being clocks.



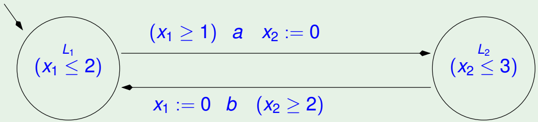
(a) In general, what is the minimum amount of time from an occurrence of event b and the subsequent occurrence of the event a? [Solution: 1 time unit.]

(b) Write a legal execution from state $\langle L_1, 0.0, 2.0\rangle$ to state $\langle L_1, 0.0, 3.0\rangle$. [Solution:
$\langle L_1, 0.0, 2.0\rangle \xrightarrow{1.0} \langle L_1, 1.0, 3.0\rangle \xrightarrow{a} \langle L_2, 1.0, 3.0\rangle \xrightarrow{0.0} \langle L_2, 1.0, 3.0\rangle \xrightarrow{b} \langle L_1, 0.0, 3.0\rangle$]

(c) Is it possible to have a legal execution in which switches e_2, e_1, e_2 are shot consecutively (possibly interleaved by time elapses), without being interleaved by other switches? If yes, write one such execution. If not, explain why. [Solution:
Yes: $\langle L_2, ..., 2.0\rangle \xrightarrow{b} \langle L_1, 0.0, 2.0\rangle \xrightarrow{1.0} \langle L_1, 1.0, 3.0\rangle \xrightarrow{a} \langle L_2, 1.0, 3.0\rangle \xrightarrow{0.0} \langle L_2, 1.0, 3.0\rangle \xrightarrow{b} \langle L_1, 0.0, 3.0\rangle$
Note: if the guard of e_2 were strictly greater than 2, this would not be possible.]

Ex: Timed Automata: Regions

Consider the following timed automaton A.



Considere the correponding Region automaton R(A). For each of the following pairs of states of A, say if the two states belong to the same region.

(a) $s_0 = (L_1, 2.5, 3.2)$, $s_1 = (L_1, 2.5, 3.7)$

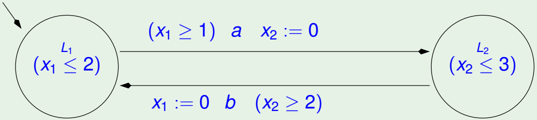
(b) $s_0 = (L_1, 1.5, 2.2)$, $s_1 = (L_1, 1.5, 2.7)$

(c) $s_0 = (L_2, 0.5, 1.4)$, $s_1 = (L_2, 0.5, 1.0)$

(d) $s_0 = (L_2, 1.7, 0.5)$, $s_1 = (L_2, 1.5, 0.1)$

Ex: Timed Automata: Regions

Consider the following timed automaton A.



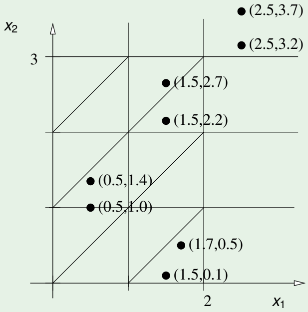
Considere the correponding Region automaton R(A). For each of the following pairs of states of A, say if the two states belong to the same region.

(a) $s_0 = (L_1, 2.5, 3.2)$, $s_1 = (L_1, 2.5, 3.7)$
[Solution: yes]

(b) $s_0 = (L_1, 1.5, 2.2)$, $s_1 = (L_1, 1.5, 2.7)$

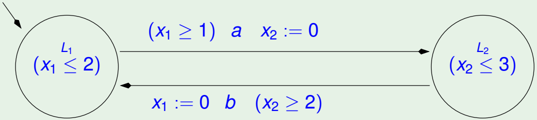
(c) $s_0 = (L_2, 0.5, 1.4)$, $s_1 = (L_2, 0.5, 1.0)$

(d) $s_0 = (L_2, 1.7, 0.5)$, $s_1 = (L_2, 1.5, 0.1)$



Ex: Timed Automata: Regions

Consider the following timed automaton A.



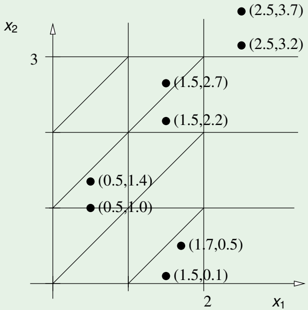
Considere the correponding Region automaton R(A). For each of the following pairs of states of A, say if the two states belong to the same region.

(a) $s_0 = (L_1, 2.5, 3.2)$, $s_1 = (L_1, 2.5, 3.7)$
[Solution: yes]

(b) $s_0 = (L_1, 1.5, 2.2)$, $s_1 = (L_1, 1.5, 2.7)$
[Solution: no]

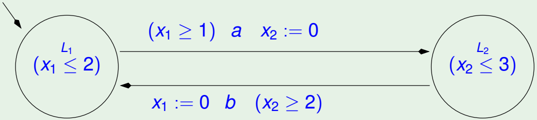
(c) $s_0 = (L_2, 0.5, 1.4)$, $s_1 = (L_2, 0.5, 1.0)$

(d) $s_0 = (L_2, 1.7, 0.5)$, $s_1 = (L_2, 1.5, 0.1)$



Ex: Timed Automata: Regions

Consider the following timed automaton A.



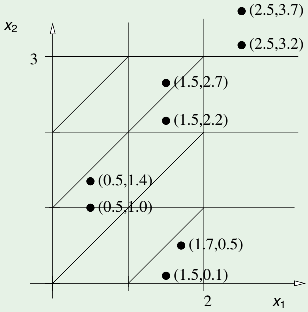
Considere the correponding Region automaton R(A). For each of the following pairs of states of A, say if the two states belong to the same region.

(a) $s_0 = (L_1, 2.5, 3.2)$, $s_1 = (L_1, 2.5, 3.7)$
[Solution: yes]

(b) $s_0 = (L_1, 1.5, 2.2)$, $s_1 = (L_1, 1.5, 2.7)$
[Solution: no]

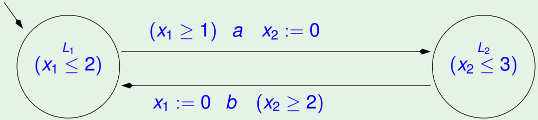
(c) $s_0 = (L_2, 0.5, 1.4)$, $s_1 = (L_2, 0.5, 1.0)$
[Solution: no]

(d) $s_0 = (L_2, 1.7, 0.5)$, $s_1 = (L_2, 1.5, 0.1)$



Ex: Timed Automata: Regions

Consider the following timed automaton A.



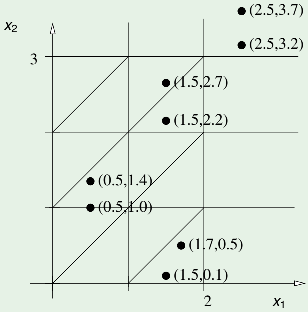
Considere the correponding Region automaton R(A). For each of the following pairs of states of A, say if the two states belong to the same region.

(a) $s_0 = (L_1, 2.5, 3.2)$, $s_1 = (L_1, 2.5, 3.7)$
[Solution: yes]

(b) $s_0 = (L_1, 1.5, 2.2)$, $s_1 = (L_1, 1.5, 2.7)$
[Solution: no]

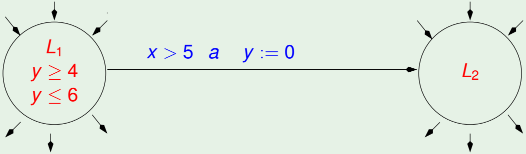
(c) $s_0 = (L_2, 0.5, 1.4)$, $s_1 = (L_2, 0.5, 1.0)$
[Solution: no]

(d) $s_0 = (L_2, 1.7, 0.5)$, $s_1 = (L_2, 1.5, 0.1)$
[Solution: yes]



Ex: Timed Automata: Zones

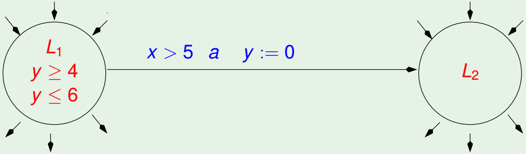
Consider the following switch e in a timed automaton, x and y being clocks:



and let $Z_1 \stackrel{\text{def}}{=} \langle L_1, \varphi \rangle$ s.t $\varphi \stackrel{\text{def}}{=} (x \geq 2) \wedge (x \leq 3) \wedge (y \geq 2) \wedge (y \leq 5) \wedge (y - x \leq 2)$. Compute $succ(Z_1, e)$, drawing the process on the cartesian space $\langle x, y \rangle$.

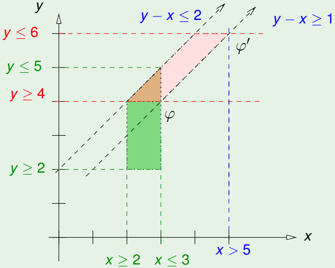
Ex: Timed Automata: Zones

Consider the following switch e in a timed automaton, x and y being clocks:



and let $Z_1 \stackrel{\text{def}}{=} \langle L_1, \varphi \rangle$ s.t $\varphi \stackrel{\text{def}}{=} (x \geq 2) \wedge (x \leq 3) \wedge (y \geq 2) \wedge (y \leq 5) \wedge (y - x \leq 2)$. Compute $succ(Z_1, e)$, drawing the process on the cartesian space $\langle x, y \rangle$.

[Solution: The solution is $succ(Z_1, e) = \langle Z_2, \bot \rangle$. In fact, the zone reached by waiting in L_1 has empty intersection with the guard, as displayed in figure:



]

Difference Bound Matrices

Consider the zone:

$\varphi \stackrel{\text{def}}{=} (x_1 \leq 3) \wedge (x_2 \leq 2) \wedge (x_3 \leq 5) \wedge$
$(x_1 - x_3 \leq 2) \wedge (x_2 - x_1 \leq -2) \wedge (x_3 - x_1 \leq 3) \wedge (x_3 - x_2 \leq 1)$

(a) Compute the corresponding DBM

(b) Compute the reduced DBM

Difference Bound Matrices

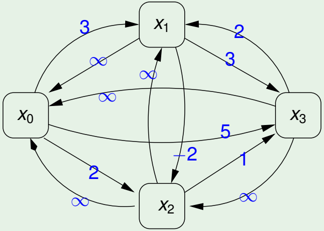
[Solution: $\varphi \stackrel{\text{def}}{=} (x_1 \leq 3) \land (x_2 \leq 2) \land (x_3 \leq 5) \land$
$(x_1 - x_3 \leq 2) \land (x_2 - x_1 \leq -2) \land (x_3 - x_1 \leq 3) \land (x_3 - x_2 \leq 1)$

]

[Solution: $\varphi \stackrel{\text{def}}{=} (x_1 \leq 3) \wedge (x_2 \leq 2) \wedge (x_3 \leq 5) \wedge$
$(x_1 - x_3 \leq 2) \wedge (x_2 - x_1 \leq -2) \wedge (x_3 - x_1 \leq 3) \wedge (x_3 - x_2 \leq 1)$

Initial DBM:

	x_0	x_1	x_2	x_3
x_0	$\langle\infty, \leq\rangle$	$\langle\infty, \leq\rangle$	$\langle\infty, \leq\rangle$	$\langle\infty, \leq\rangle$
x_1	$\langle 3, \leq\rangle$	$\langle\infty, \leq\rangle$	$\langle\infty, \leq\rangle$	$\langle 2, \leq\rangle$
x_2	$\langle 2, \leq\rangle$	$\langle -2, \leq\rangle$	$\langle\infty, \leq\rangle$	$\langle\infty, \leq\rangle$
x_3	$\langle 5, \leq\rangle$	$\langle 3, \leq\rangle$	$\langle 1, \leq\rangle$	$\langle\infty, \leq\rangle$



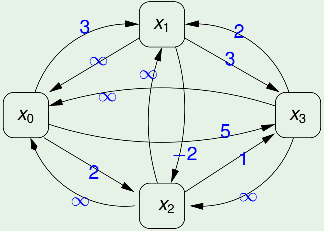
]

Difference Bound Matrices

[Solution: $\varphi \stackrel{\text{def}}{=} (x_1 \le 3) \wedge (x_2 \le 2) \wedge (x_3 \le 5) \wedge$
$(x_1 - x_3 \le 2) \wedge (x_2 - x_1 \le -2) \wedge (x_3 - x_1 \le 3) \wedge (x_3 - x_2 \le 1)$

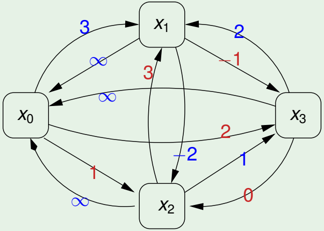
Initial DBM:

	x_0	x_1	x_2	x_3
x_0	$\langle\infty, \le\rangle$	$\langle\infty, \le\rangle$	$\langle\infty, \le\rangle$	$\langle\infty, \le\rangle$
x_1	$\langle 3, \le\rangle$	$\langle\infty, \le\rangle$	$\langle\infty, \le\rangle$	$\langle 2, \le\rangle$
x_2	$\langle 2, \le\rangle$	$\langle -2, \le\rangle$	$\langle\infty, \le\rangle$	$\langle\infty, \le\rangle$
x_3	$\langle 5, \le\rangle$	$\langle 3, \le\rangle$	$\langle 1, \le\rangle$	$\langle\infty, \le\rangle$



Reduced DBM:

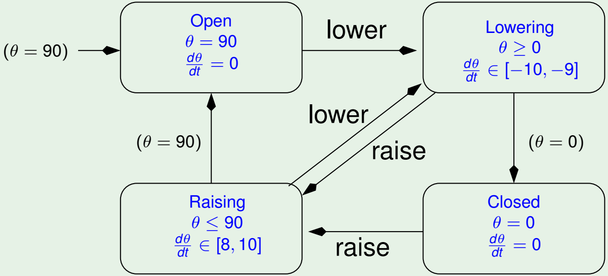
	x_0	x_1	x_2	x_3
x_0	$\langle 0, \le\rangle$	$\langle\infty, \le\rangle$	$\langle\infty, \le\rangle$	$\langle\infty, \le\rangle$
x_1	$\langle 3, \le\rangle$	$\langle 0, \le\rangle$	$\langle 3, \le\rangle$	$\langle 2, \le\rangle$
x_2	$\langle 1, \le\rangle$	$\langle -2, \le\rangle$	$\langle 0, \le\rangle$	$\langle 0, \le\rangle$
x_3	$\langle 2, \le\rangle$	$\langle -1, \le\rangle$	$\langle 1, \le\rangle$	$\langle 0, \le\rangle$



]

Hybrid Automata

A railway-crossing gate, whose dynamics is represented by the hybrid automaton in the figure, receives from a controller two possible input signals {lower,raise}. (θ, in degrees, represents the angle between the bar and the ground.)
When the gate is open the controller receives a signal "incoming" when a train is incoming, it waits a fixed amount of time Δt, then it sends the gate the lower order.
It is known that an incoming train takes an amount of time within the interval [70,100] time units to get from the remote sensor to the gate.
Compute the *maximum* amount of time Δt which guarantees that the train does not reach the gate before the bar is completely lowered, and briefly explain why.



Hybrid Automata

[Solution: Δt is 60 time units. In fact, the maximum value of Δt the controller can afford waiting is given by the minimum time the train may take to reach the gate (70), minus the maximum time taken by the bar to lower, that is, the time taken to lower the angle from 90 to 0 at the lowest absolute speed (90/|-9|). Overall, we have thus $\Delta t = 70 - 90/(|-9|) = 60$.]