Formale Modellierung
Vorlesung 13 vom 14.07.2014: Hybride Systeme

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Formale Modellierung mit der UML und OCL
  - Lineare Temporale Logik
  - Temporale Logik und Modellprüfung
  - Hybride Systeme
  - Zusammenfassung, Rückblick, Ausblick

What are Hybrid Systems?

What are Hybrid Systems?

How are they modeled?

- Finite Automata
- Discrete Automata
- Timed Automata
- Multi-Phase Automata
- Rectangular Automata
- Affine Automata

- What are Hybrid Systems?
- How are they modeled?
  - Finite Automata
  - Discrete Automata
  - Timed Automata
  - Multi-Phase Automata
  - Rectangular Automata
  - Affine Automata
- How are properties specified?
  - Temporal Logic
  - CTL as a Branching Temporal Logic
  - ICTL - Integrator CTL

- What are Hybrid Systems?
- How are they modeled?
  - Finite Automata
  - Discrete Automata
  - Timed Automata
  - Multi-Phase Automata
  - Rectangular Automata
  - Affine Automata
- How are properties specified?
  - Temporal Logic
  - CTL as a Branching Temporal Logic
  - ICTL - Integrator CTL
- How are safety properties verified?
  - Forward Reachability
  - Backward Reachability
  - Location Elimination

- What are Hybrid Systems?
- How are they modeled?
  - Finite Automata
  - Discrete Automata
  - Timed Automata
  - Multi-Phase Automata
  - Rectangular Automata
  - Affine Automata
- How are properties specified?
  - Temporal Logic
  - CTL as a Branching Temporal Logic
  - ICTL - Integrator CTL
- How are safety properties verified?
  - Forward Reachability
  - Backward Reachability
  - Location Elimination
- Approximations for Affine Automata

---

* Thanks to Andreas Nonnengart for the slides

What are Hybrid Systems?

How are they modeled?

- Finite Automata
- Discrete Automata
- Timed Automata
- Multi-Phase Automata
- Rectangular Automata
- Affine Automata

How are properties specified?

- Temporal Logic
- CTL as a Branching Temporal Logic
- ICTL - Integrator CTL

How are safety properties verified?

- Forward Reachability
- Backward Reachability
- Location Elimination

Approximations for Affine Automata

What are Hybrid Systems?

Alur, Henzinger et al

A hybrid system is a digital real-time system that is embedded in an analog environment. It interacts with the physical world through sensors and actuators.

Wikipedia

A hybrid system is a system that exhibits both continuous and discrete dynamic behavior – a system that can both flow (described by differential equations) and jump (described by a difference equation).

What are Hybrid Systems?

How are they modeled?

- Finite Automata
- Discrete Automata
- Timed Automata
- Multi-Phase Automata
- Rectangular Automata
- Affine Automata

How are properties specified?

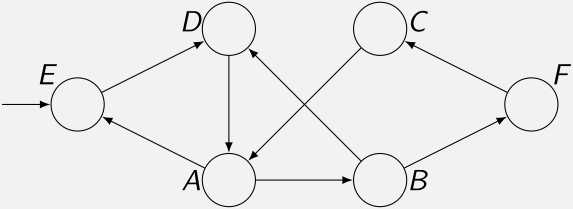
- Temporal Logic
- CTL as a Branching Temporal Logic
- ICTL - Integrator CTL

How are safety properties verified?

- Forward Reachability
- Backward Reachability
- Location Elimination

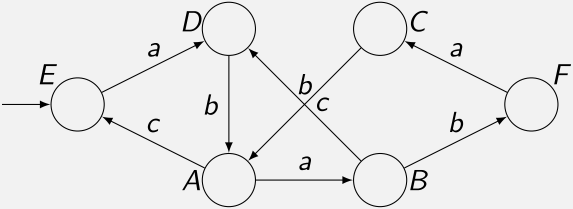
Approximations for Affine Automata

Finite Automata



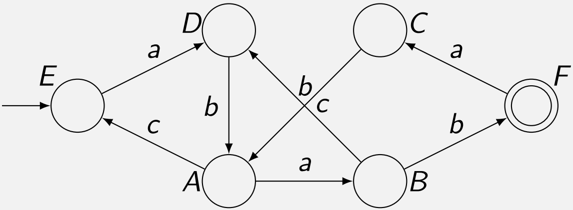
- There are vertices (states, locations) and edges (transitions)

Finite Automata



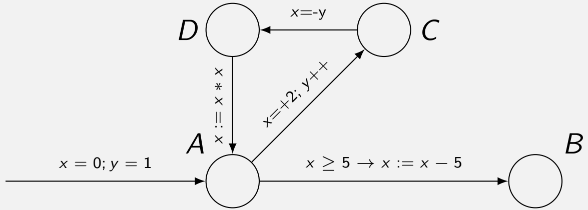
- There are vertices (states, locations) and edges (transitions)
- and maybe some input alphabet

Finite Automata



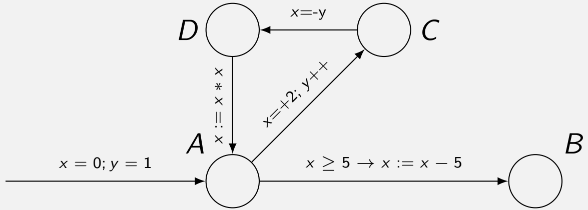
- There are vertices (states, locations) and edges (transitions)
- and maybe some input alphabet
- and maybe some "accepting" state

Discrete Automata



- there are variables involved, and they can be manipulated
- transitions may be guarded

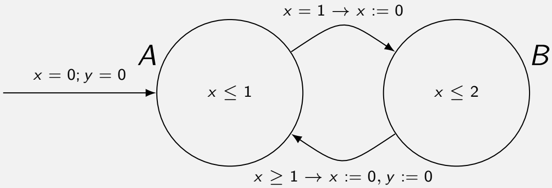
Discrete Automata



- there are variables involved, and they can be manipulated
- transitions may be guarded
- in general not finite state

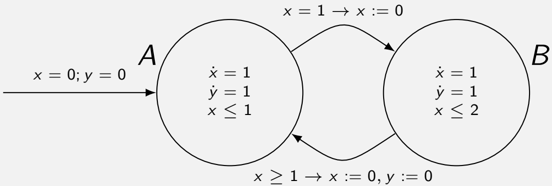
Timed Automata

- additional *clock variables*
- they continuously increase their value in locations
- all of them behave identically
- only operation: reset to 0



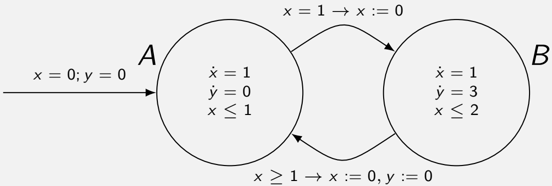
Timed Automata

- additional *clock variables*
- they continuously increase their value in locations
- all of them behave identically
- only operation: reset to 0



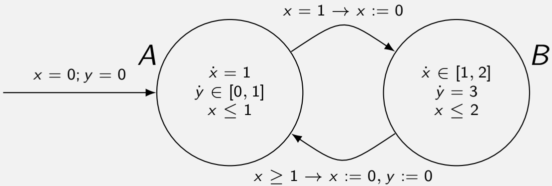
Multi-Phase Automata

- additional variables with a fixed rate, not only clocks
- they increase their value according to the rate
- thus not all of them behave identically
- arbitrary operations

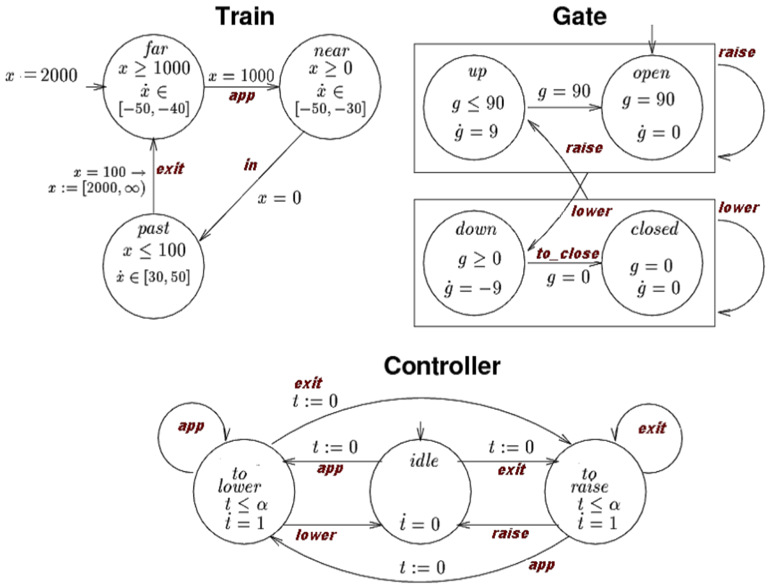


Rectangular Automata

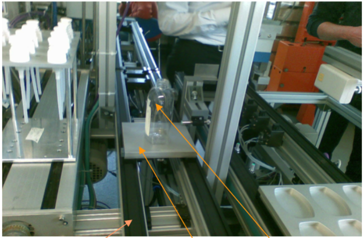
- additional variables with a *bounded* rate
- they increase their value according to these bounds
- they represent arbitrary functions wrt/ bounds
- arbitrary operations



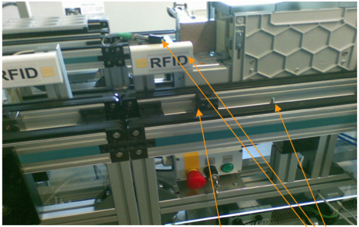
Railroad Gate Controller



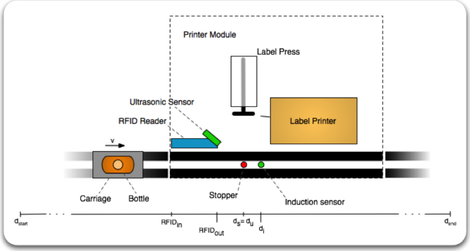
Smart Factory



transportation belt, carriage, bottle

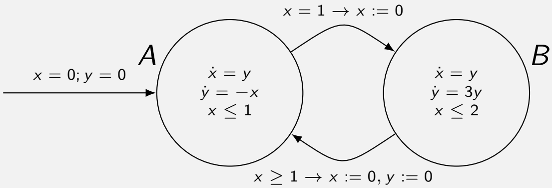


Labeling Section with stoppers and sensors



Affine Automata

- additional variables with arbitrary rate
- the rate may be in terms of the (other) variables
- they represent in general non-linear functions
- arbitrary operations



What are Hybrid Systems?

How are they modeled?
- Finite Automata
- Discrete Automata
- Timed Automata
- Multi-Phase Automata
- Rectangular Automata
- Affine Automata

How are properties specified?
- Temporal Logic
- CTL as a Branching Temporal Logic
- ICTL - Integrator CTL

How are safety properties verified?
- Forward Reachability
- Backward Reachability
- Location Elimination

Approximations for Affine Automata

Temporal Logic - operators $\Box$ and $\Diamond$

Linear Temporal Logic

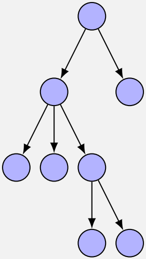
Interpret $\Box$ as *Always, Henceforth, from now on*
Interpret $\Diamond$ as *Eventually, Unavoidable*

Branching Temporal Logic

Interpret $\Box$ as *Always, Henceforth, from now on*
Interpret $\Diamond$ as *Eventually in a possible future*

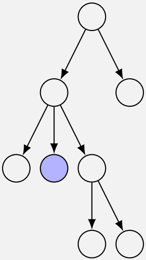
Computation Tree Logic Illustrated

$\forall\Box$ *for each path - always*



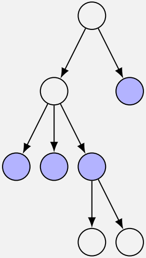
Computation Tree Logic Illustrated

$\exists\Diamond$ *for some path - eventually*



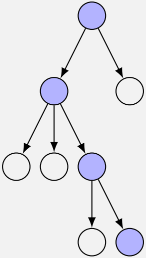
Computation Tree Logic Illustrated

$\forall\Diamond$ *for each path - eventually*



Computation Tree Logic Illustrated

$\exists\Box$ *for some path - always*



Timed (Integrator) CTL

- add clock variables
- these may be used in formulas
- restrict these clocks to certain locations (stopwatches)

$$z.\exists\Diamond\ \{A \wedge z \leq 5\}$$
$$c^{\{N,M\}}.\forall\Box\ \{P \rightarrow c \geq 12\}$$

What are Hybrid Systems?

How are they modeled?

- Finite Automata
- Discrete Automata
- Timed Automata
- Multi-Phase Automata
- Rectangular Automata
- Affine Automata

How are properties specified?

- Temporal Logic
- CTL as a Branching Temporal Logic
- ICTL - Integrator CTL

How are safety properties verified?

- Forward Reachability
- Backward Reachability
- Location Elimination

Approximations for Affine Automata

Safety Properties

A safety property is of the form

$$\forall\Box\Phi$$

where Φ is a classical logic formula (with arithmetics)
We call a state s safe if $\Phi(s)$ is true

It has to be shown that all reachable states are safe (forward reachability)

or, equivalently,

It has to be shown that no unsafe state is reachable (backward reachability)

Forward Reachability

The Operator $post(S)$

Given a set S of states

$$post(S) = \{s \mid \exists s' \in S : s' \mapsto_{\delta} \mapsto_{tr} s\}$$

Fixpoint Iteration

Start with S a the initial states
repeat until $post(S) \subseteq S$: $S := S \cup post(S)$

Finally

Check whether $\Phi(S)$ holds

Backward Reachability

The Operator *pre*(S)

Given a set S of states

$$pre(S) = \{s \mid \exists s' \in S : s \mapsto_{tr} \mapsto_{\delta} s'\}$$

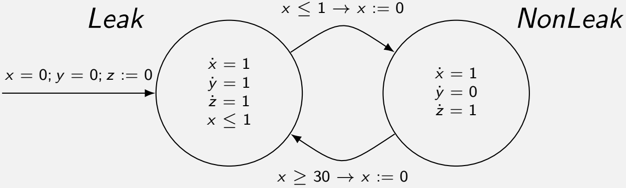
Fixpoint Iteration

Start with $S = \{s \mid \neg\Phi(s)\}$
repeat until $pre(S) \subseteq S$: $S := S \cup pre(S)$

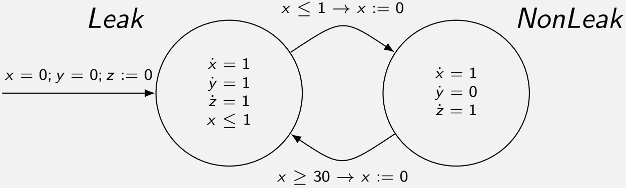
Finally

Check whether the initial state is contained in S

Example: Leaking Gas Burner



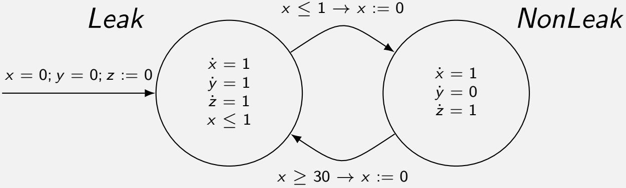
Example: Leaking Gas Burner



Safety Property

$$\forall\Box\ z \geq 60 \rightarrow 20 * y \leq z$$

Example: Leaking Gas Burner

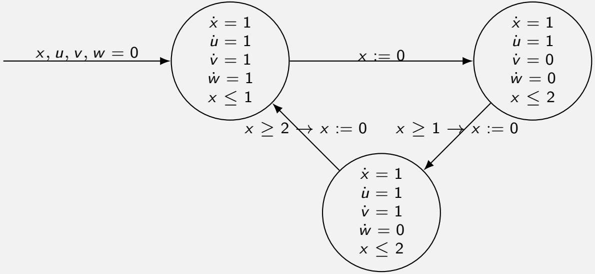


Safety Property

$$\forall\Box\; z \geq 60 \to 20 * y \leq z$$

$$\begin{aligned} I &= \{Leak(0,0,0)\} \\ post(I) &= \{Leak(x,y,z) \mid 0 \leq x \leq 1, y = x, z = x\} \\ &\qquad \cup \{NonLeak(0,y,z) \mid 0 \leq y \leq 1, z = y\} \end{aligned}$$

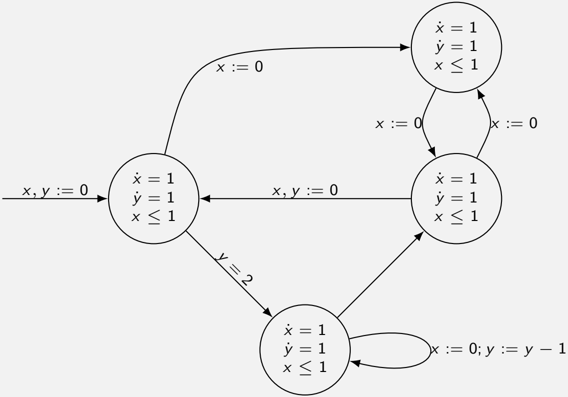
Problem: Long Loops



Property (many iterations)

$\forall\Box\ (u \geq 154 \to 5.9 * w \leq u + v)$

Another Problem: Termination

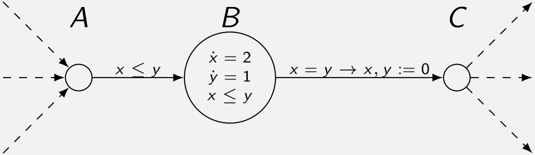


Location Elimination

General Idea

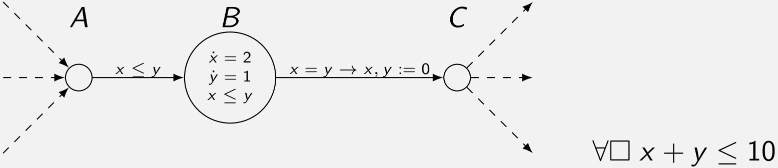
- Compute the responsibility for a location once and for all
- thereby compute a definition for this location
- insert this definition into the automaton
- delete the location (and all the transitions to and fro)

Elimination Example



$\forall\Box\ x + y \leq 10$

Elimination Example



Reachability Theory for B

$$A(x, y) \to x \leq y \to B(x, y)$$
$$B(x, y) \to x \leq y$$
$$B(x, y) \to x + y \leq 10$$
$$B(x, y) \to \forall\delta\ 0 \leq \delta \wedge x' = x + 2\delta \wedge y' = y + \delta \wedge x' \leq y' \to B(x', y')$$
$$B(x, y) \to x = y \to C(0, 0)$$

Elimination Approach

Reachability Theory simplified

$A(x, y) \to x \leq y \to B(x, y)$
$B(x, y) \to x \leq y$
$B(x, y) \to x + y \leq 10$
$B(x, y) \to x \leq x' \wedge x + 2 * y' = x' + 2 * y \wedge x' \leq y' \to B(x', y')$
$B(x, y) \to x = y \to C(0, 0)$

Elimination Approach

Reachability Theory simplified

$A(x, y) \to x \leq y \to B(x, y)$
$B(x, y) \to x \leq y$
$B(x, y) \to x + y \leq 10$
$B(x, y) \to x \leq x' \wedge x + 2 * y' = x' + 2 * y \wedge x' \leq y' \to B(x', y')$
$B(x, y) \to x = y \to C(0, 0)$

Fixpoint Computation (Definition for B)

$B(x, y) \to x \leq y \to C(0, 0)$
$B(x, y) \to x \leq y \to 2 * y \leq x + 5$

Elimination Approach

Reachability Theory simplified

$A(x, y) \to x \le y \to B(x, y)$
$B(x, y) \to x \le y$
$B(x, y) \to x + y \le 10$
$B(x, y) \to x \le x' \wedge x + 2 * y' = x' + 2 * y \wedge x' \le y' \to B(x', y')$
$B(x, y) \to x = y \to C(0, 0)$

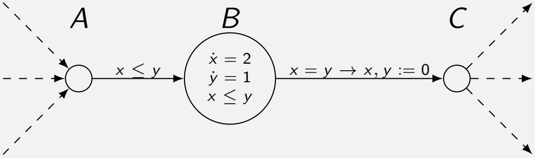
Fixpoint Computation (Definition for B)

$B(x, y) \to x \le y \to C(0, 0)$
$B(x, y) \to x \le y \to 2 * y \le x + 5$

Insertion (in A)

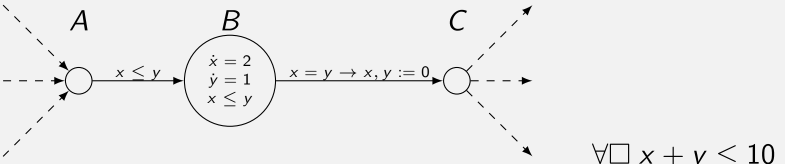
$A(x, y) \to x \le y \to C(0, 0)$
$A(x, y) \to x \le y \to 2 * y \le x + 5$

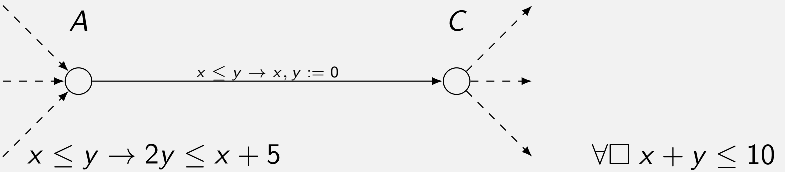
Elimination Result



$\forall\Box\ x + y \leq 10$

Elimination Result





Elimination Approach

Advantages

- with each elimination the verification problem decreases
- no need for multiple turns through the automaton
- in a sense mixes (and generalizes) standard reachability approaches

What are Hybrid Systems?

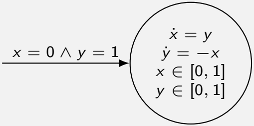
How are they modeled?
- Finite Automata
- Discrete Automata
- Timed Automata
- Multi-Phase Automata
- Rectangular Automata
- Affine Automata

How are properties specified?
- Temporal Logic
- CTL as a Branching Temporal Logic
- ICTL - Integrator CTL

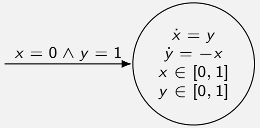
How are safety properties verified?
- Forward Reachability
- Backward Reachability
- Location Elimination

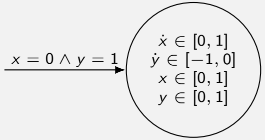
Approximations for Affine Automata

Approximation of Affine Behavior

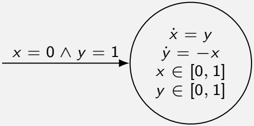


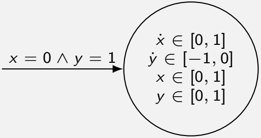
Approximation of Affine Behavior

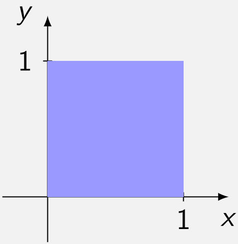




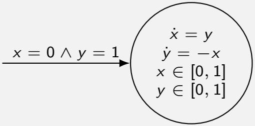
Approximation of Affine Behavior



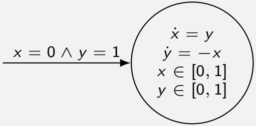


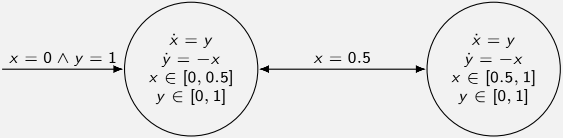


Location Splitting

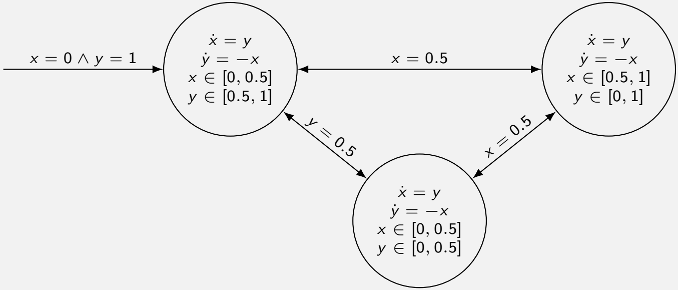


Location Splitting

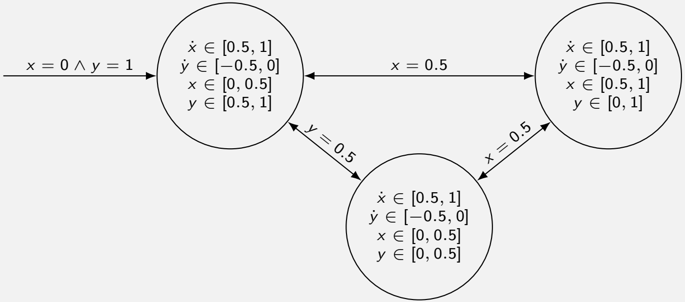




One More Splitting



One More Splitting



Eliminating A

Positive A-clauses

$x = 0 \wedge y = 1 \rightarrow A(x, y)$	initial state
$B(x, y) \rightarrow x = 0.5 \wedge y \in [0.5, 1] \rightarrow A(x, y)$	from B to A
$C(x, y) \rightarrow y = 0.5 \wedge x \in [0, 0.5] \rightarrow A(x, y)$	from C to A
$A(x, y) \rightarrow y' \leq y \wedge x' \in [0, 0.5] \wedge y' \in [0.5, 1] \wedge x + y \leq x' + y' \rightarrow A(x', y')$	continuous change

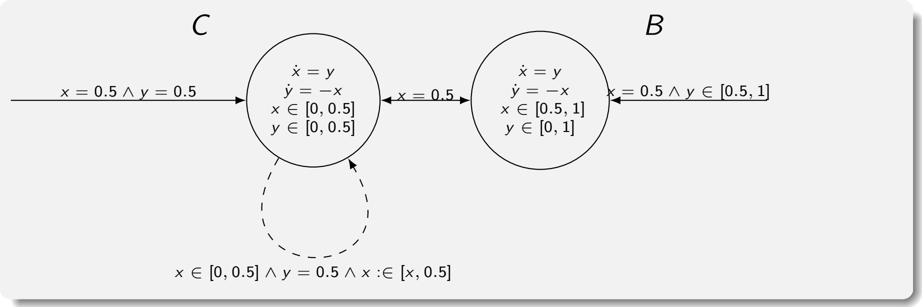
Fixpoint Computation and Definition of A

$x \in [0, 0.5] \wedge y \in [0.5, 1] \wedge 1 \leq x + y \rightarrow A(x, y)$
$C(x, y) \rightarrow y = 0.5 \wedge y' = 0.5 \wedge x \in [0, 0.5] \wedge x \leq x' \wedge x' \in [0, 0.5] \rightarrow A(x', y')$

Insertion of A's Definition

$x = 0.5 \wedge y \in [0.5, 1] \rightarrow B(x, y)$
$x = 0.5 \wedge y = 0.5 \rightarrow C(x, y)$
$C(x, y) \rightarrow x \in [0, 0.5] \wedge y = 0.5 \wedge x' \in [x, 0.5] \wedge y' = y \rightarrow C(x', y')$

After Eliminating A



Eliminating C

Positive C-clauses

$x = 0.5 \wedge y = 0.5 \rightarrow C(x, y)$
$B(x, y) \rightarrow x = 0.5 \wedge y \in [0, 0.5] \rightarrow C(x, y)$
$C(x, y) \rightarrow x \leq x' \wedge y' \leq y \wedge x' \in [0, 0.5] \wedge y' \in [0, 0.5] \rightarrow C(x', y')$

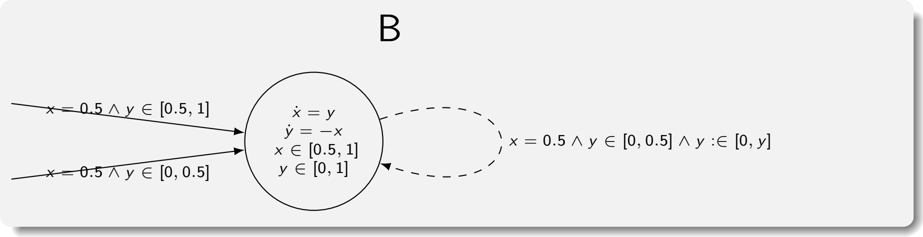
Fixpoint Computation and Definition of C

$x = 0.5 \wedge y \in [0, 0.5] \rightarrow C(x, y)$
$B(x, y) \rightarrow x = 0.5 \wedge y \in [0, 0.5] \wedge x' = 0.5 \wedge y' \in [0, y] \rightarrow C(x', y')$

Insertion of C's Definition

$x = 0.5 \wedge y \in [0, 0.5] \rightarrow B(x, y)$
$B(x, y) \rightarrow x = 0.5 \wedge y \in [0, 0.5] \wedge x' = 0.5 \wedge y' \in [0, y] \rightarrow B(x', y')$

After Eliminating C



Eliminating B

Positive B-clauses

$x = 0.5 \land y \in [0.5, 1] \rightarrow B(x, y)$
$x = 0.5 \land y \in [0, 0.5] \rightarrow B(x, y)$
$B(x, y) \rightarrow x \leq x' \land y' \leq y \land x' + 2y' \leq x + 2y \land x' \in [0.5, 1] \land y' \in [0, 1] \rightarrow B(x', y')$

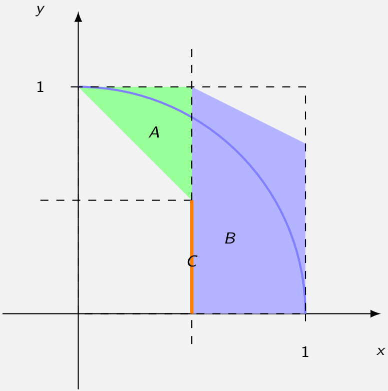
Fixpoint Computation and Definition of B

$x + 2y \leq 2.5 \land x \in [0.5, 1] \land y \in [0, 1] \rightarrow B(x, y)$

Final Insertion and Result

$x \in [0, 0.5] \land y \in [0.5, 1] \land 1 \leq x + y \rightarrow A(x, y)$
$x + 2y \leq 2.5 \land x \in [0.5, 1] \land y \in [0, 1] \rightarrow B(x, y)$
$x = 0.5 \land y \in [0, 0.5] \rightarrow C(x, y)$

After Eliminating All



Summary

- Modelling of systems with continuous state changes requires different techniques
- Inspired by state machines, but with continuous behaviour in states expressed by first derivatives
- Different aspects
  - Timed Automata
  - Multi-Phase Automata
  - Rectangular Automata
  - Affine Automata
- Properties formulated using CTL;
- Verification approaches beyond forward/bachward reachability analysis