Controller Synthesis for Linear Hybrid Systems

Part 1: Introduction

Formal Methods for Cyber-Physical Systems
Verona, September 12-16, 2017

Marco Faella
Università di Napoli “Federico II”

Course Summary

- Introduction
- Models of hybrid systems
- Verification
  - the reachability problem
- Controller synthesis
  - safety control
  - reachability control
- Hands-on laboratory

Lesson Summary

- Introduction
- Models of hybrid systems
- Verification – the reachability problem

Introduction

- What are hybrid systems?
- Why study hybrid systems?
- What models for hybrid systems?

What are Hybrid Systems?

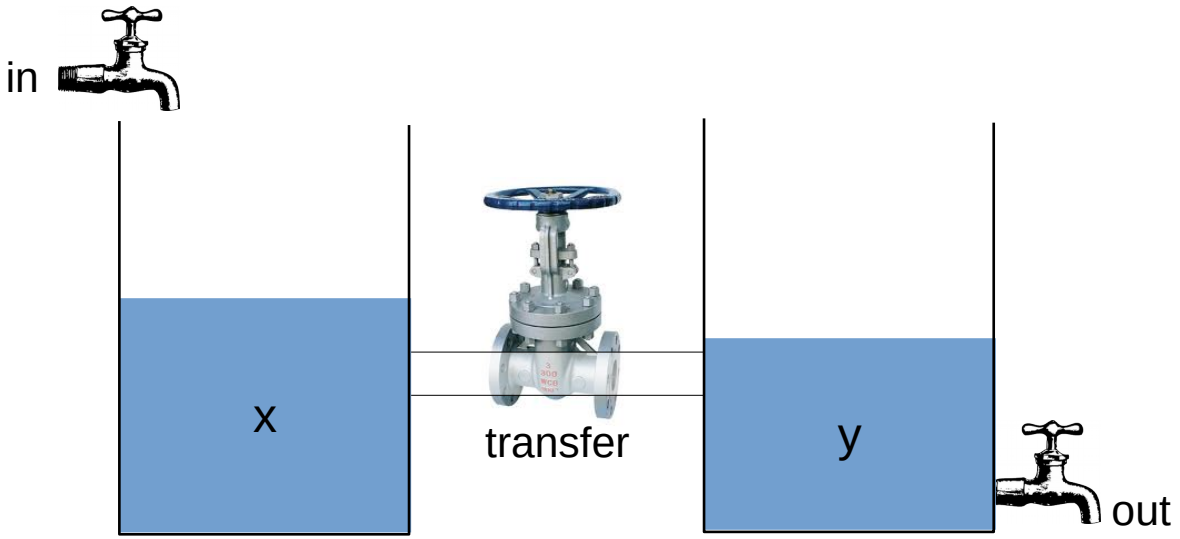
Dynamic systems with discrete and continuous components

- Dynamic: evolving in time
- “discrete transitions” as in a finite automaton
- “continuous evolution” as in traditional dynamic systems

Why study Hybrid Systems?

1) Represent interaction between **digital computing devices** (e.g., controllers) and physical actuators and sensors
   - plants
   - vehicles (automotive, aerial, etc.)

Example: Two Tanks



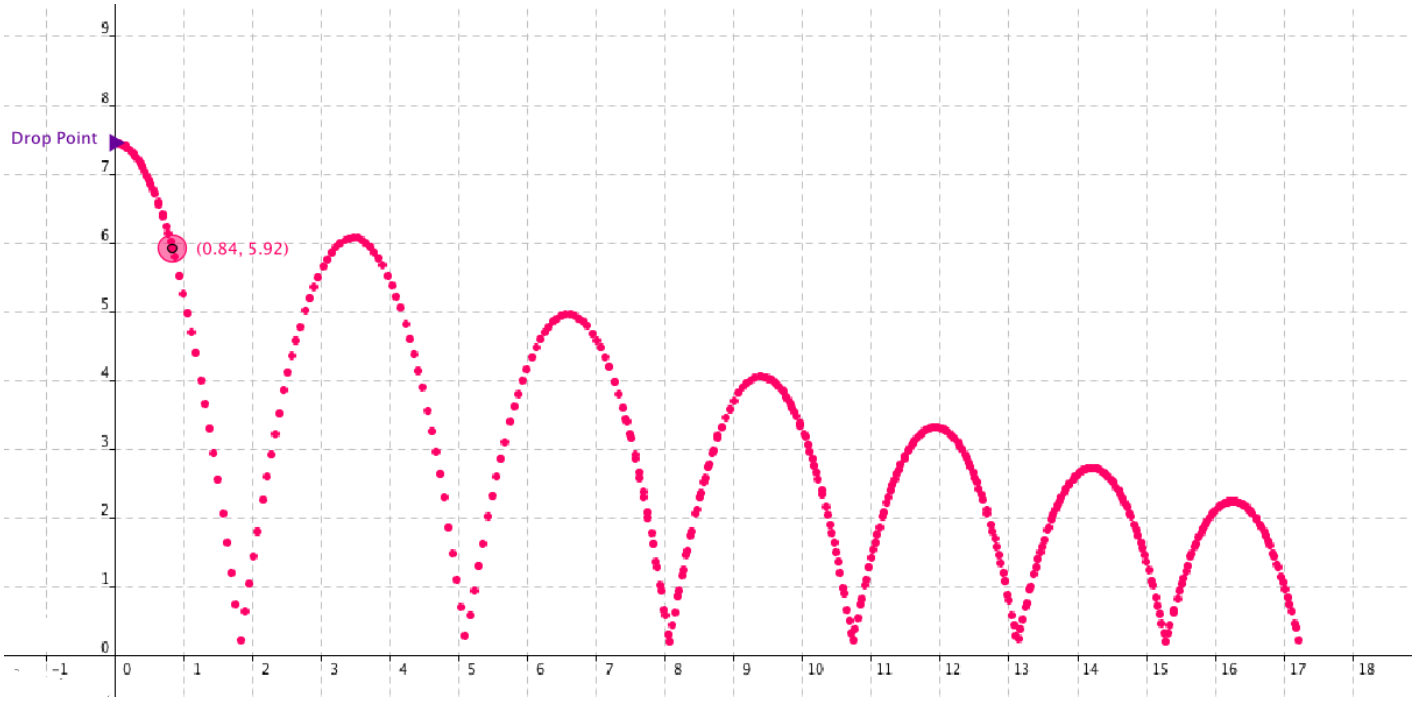
- 2 continuous variables (water levels x, y)
- 3 discrete (on/off) valves
- Status of discrete components influences continuous behavior
  - 8 different dynamics

Why study Hybrid Systems?

1) Represent interaction between **digital computing devices** (e.g., controllers) and physical actuators and sensors
   - plants
   - vehicles (automotive, aerial, etc.)

2) Represent **inherently discontinuous** physical phenomena
   - collisions
   - (bio)chemical processes

Example: Bouncing Ball



- 2 continuous variables: ball position and velocity
- discontinuity (a.k.a. *jump*) on velocity when position=0

What models for Hybrid Systems?

- In control engineering:
  - Switched systems
  - Piecewise affine systems
- In computer science:
  - **Hybrid Automata**
  - generalize Timed Automata by allowing *more general dynamics* for continuous variables

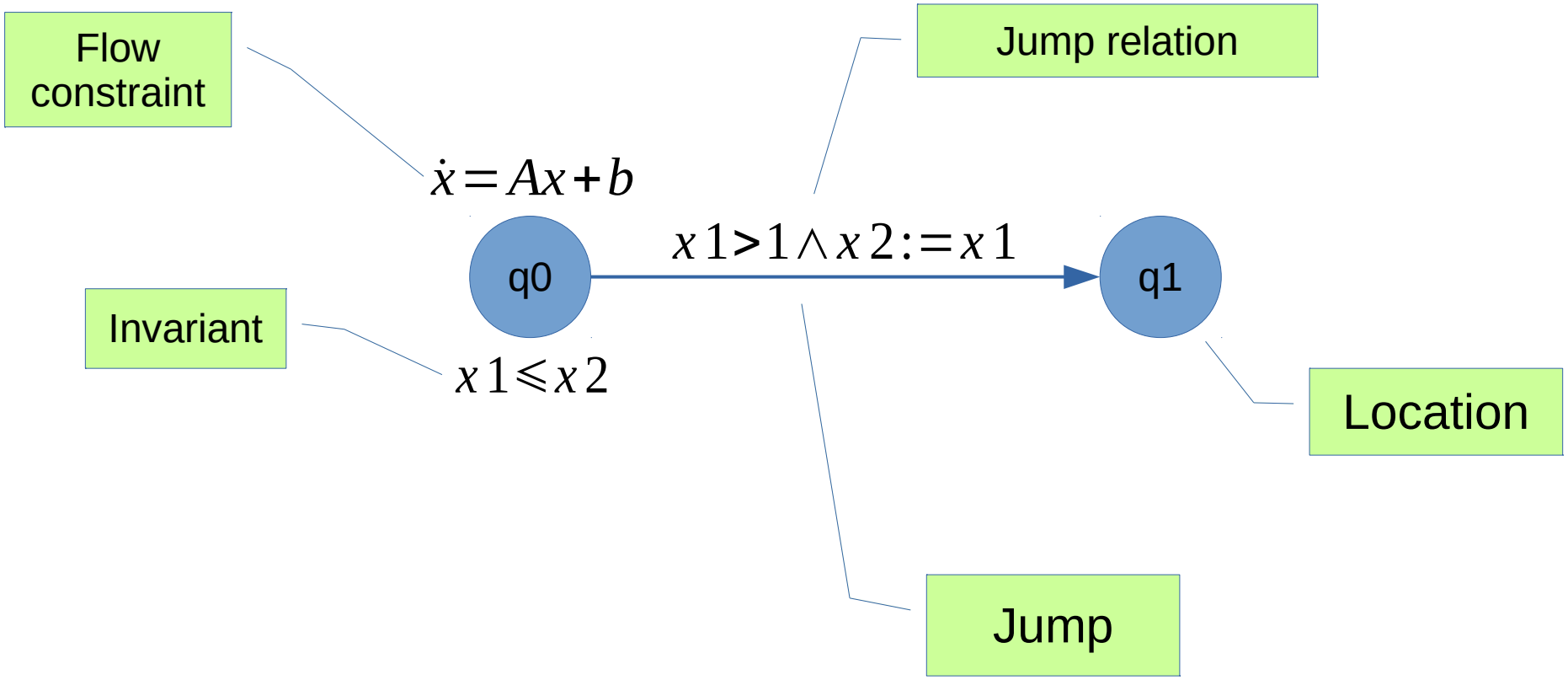
Hybrid Automata

- Similar to timed automata:
  - Locations (a.k.a. *modes*), transitions (a.k.a. *jumps*), invariants
- Specific:
  - **Continuous variables** instead of clocks
  - Variables may have **different dynamics** (differential equations) in different locations
  - Jump relation instead of guards and resets

Notation

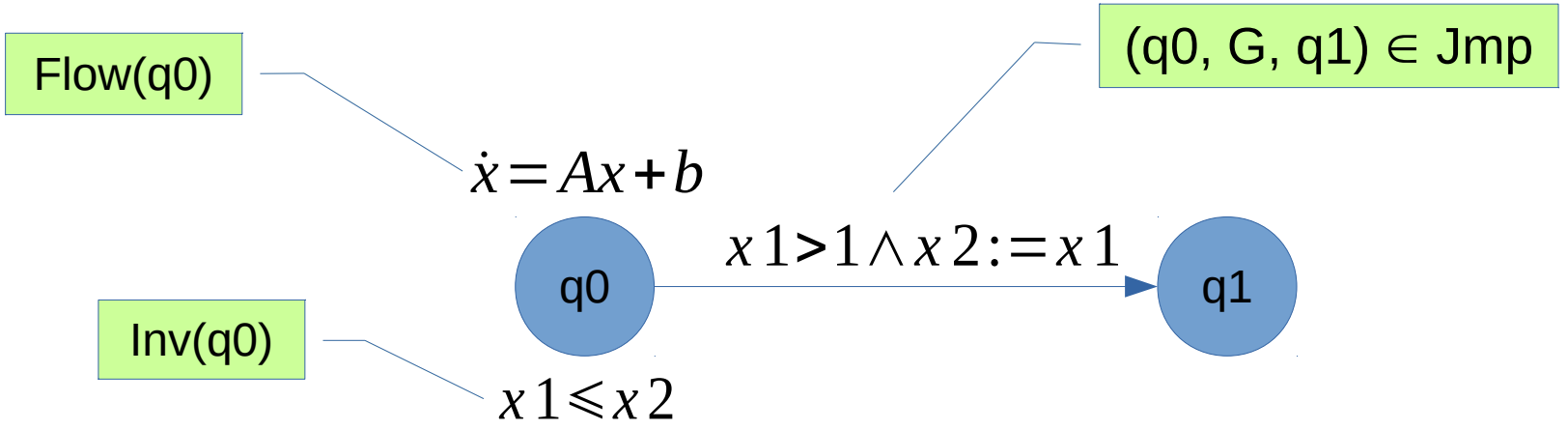
- $\mathbb{R}$ — real numbers
- $x = (x1, x2, \ldots, x\_n) \in \mathbb{R}^n$ — state vector
- $\dot{x} = (\dot{x}1, \dot{x}2, \ldots, \dot{x}\_n)$ — vector of first derivatives
- $x' = (x1', x2', \ldots, x\_n')$ — vector of state in *next* location
- $\wp(S)$ — powerset of S

Hybrid Automata



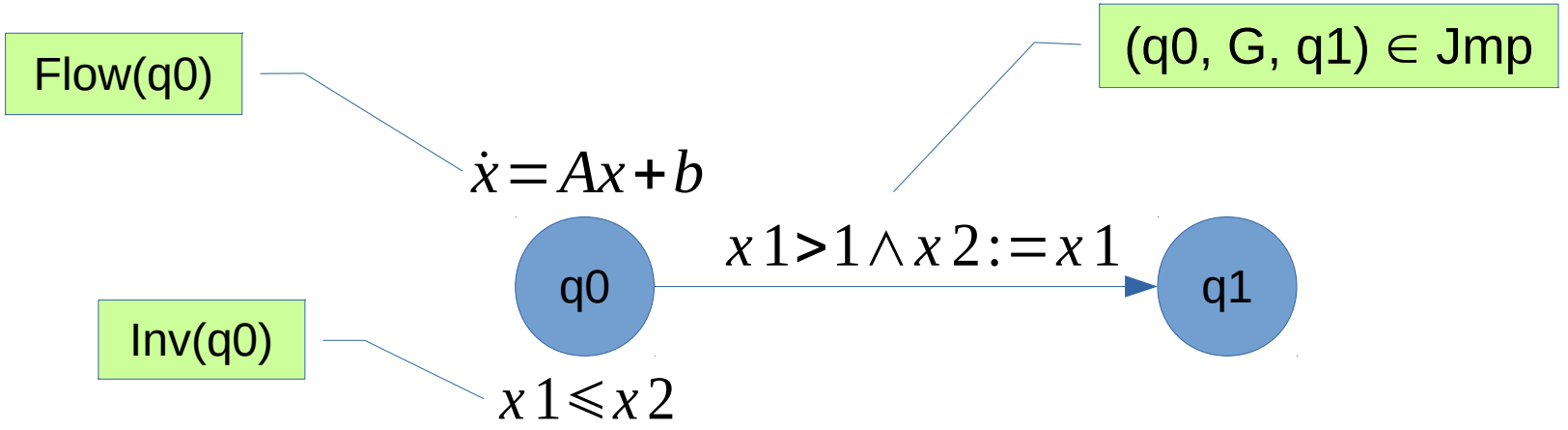
Hybrid Automata

- Finite set of locations — **Loc**
- Finite set of jumps — **Jmp** $\subseteq$ Loc $\times$ $\wp(\mathbb{R}^{2n})$ $\times$ Loc
- Invariants — **Inv** : Loc $\rightarrow$ $\wp(\mathbb{R}^{n})$
- Dynamics — **Flow** : Loc $\rightarrow$ Class of diff. eq.



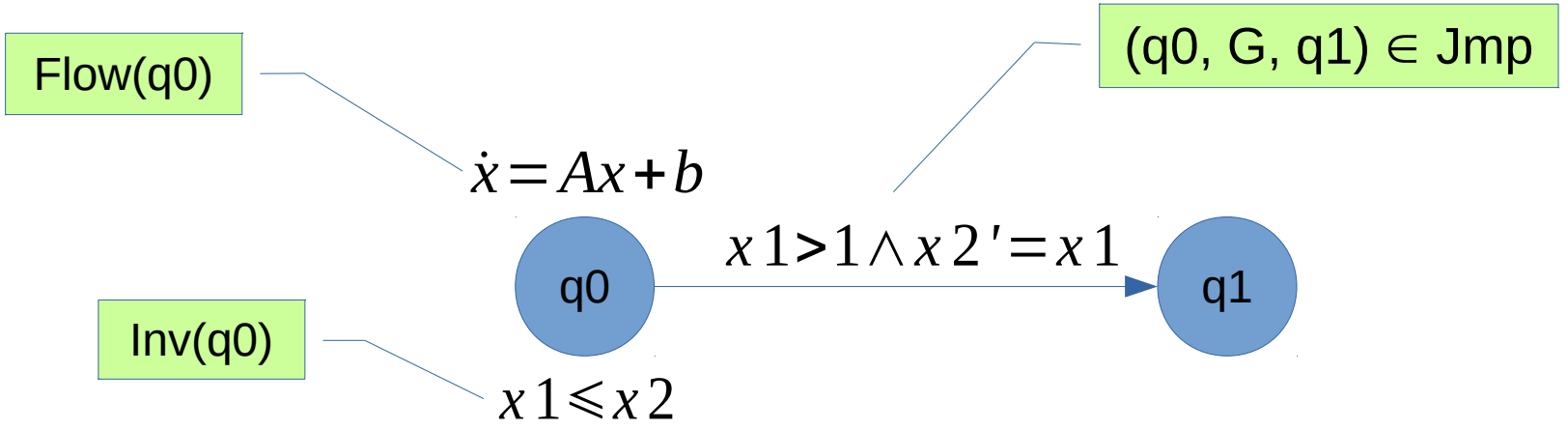
Hybrid Automata

- Finite set of locations — **Loc**
- Finite set of jumps — **Jmp** $\subseteq$ Loc $\times$ $\wp(\mathbb{R}^{2n})$ $\times$ Loc
- Invariants — **Inv** : Loc $\rightarrow$ $\wp(\mathbb{R}^{n})$
- Dynamics — **Flow** : Loc $\rightarrow$ Class of diff. eq.

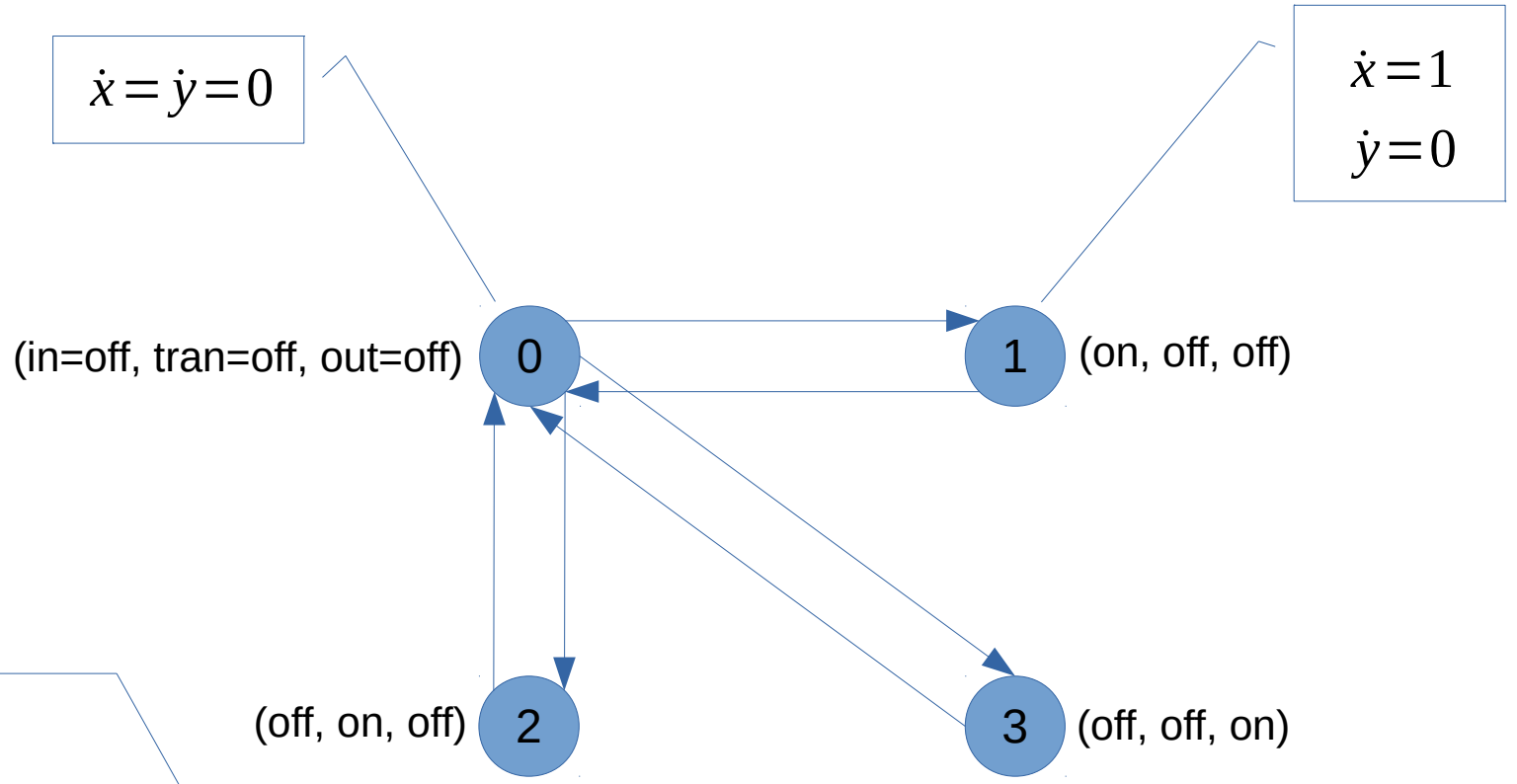


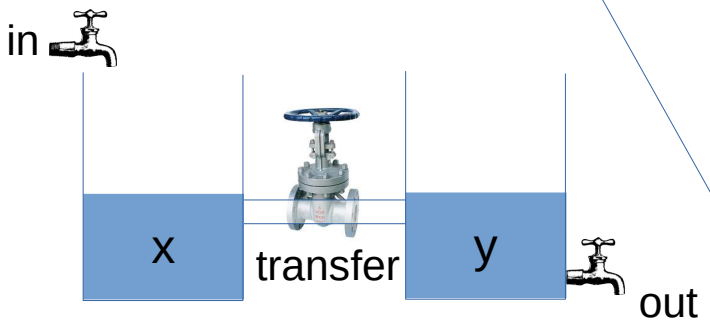
Hybrid Automata

- Finite set of locations — **Loc**
- Finite set of jumps — **Jmp** $\subseteq$ Loc $\times$ $\wp(\mathbb{R}^{2n})$ $\times$ Loc
- Invariants — **Inv** : Loc $\rightarrow$ $\wp(\mathbb{R}^{n})$
- Dynamics — **Flow** : Loc $\rightarrow$ Class of diff. eq.



Automaton Fragment for Two Tanks





Jumps

Note: a **configuration** or **state** of the automaton is a pair (q, x), where $q \in Loc$ and $x \in \mathbb{R}^n$

the state space is $Loc \times \mathbb{R}^n$

A jump (q, G, q') is **enabled** in

$$\{ (q, x) \mid \text{there exists } x' \text{ s.t. } (x, x') \text{ in } G \}$$

and non-deterministically leads to a state (q', x') s.t. (x, x') in G

Runs

- Semantics based on runs
- Sequences of alternating **timed steps** and **discrete steps**
- Timed step:
  - a continuous-time **trajectory** in $\mathbb{R}^n$, which satisfies the invariant and the diff.eq. of the current location
  - location remains fixed
- Discrete step:
  - an **instantaneous change of location**
  - variables change according to jump relation

Trajectories

For $T \geq 0$, a **T-trajectory** from a state (q, x) is a function

$$f: [0, T] \rightarrow \mathbb{R}^n$$

such that:

- f is continuous
- f is differentiable, *except for a finite set of time points*
- $\dot{f}$ stays in Flow(q), in all points where f is differentiable
  - similarly to a Carathéodory solution to an ODE
- f starts from x: $f(0) = x$
- f stays in the invariant: $f(t)$ in Inv(q) for all t in [0, T]

Trajectories

Why admitting a finite set of non-differentiable points?

(i.e., a finite number of *sharp turns*)

For computational convenience

Moreover, it only makes a difference in *non-robust* scenarios

For more info: [Benerecetti & F., HSCC 2013]

Timed Steps

A **timed step** is:

$$(q, x) \xrightarrow{f,T} (q, y)$$

where f is a T-trajectory from (q, x) and y = f(T)

Discrete Steps

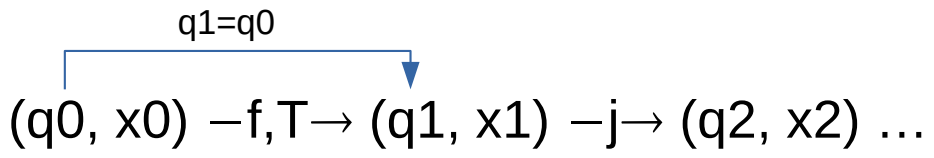
A discrete **step** is:

$$(q, x) \xrightarrow{j} (q', x')$$

where $j = (q, G, q') \in Jmp$ and $(x, x') \in G$

Runs

A run is then:



A run can be an infinite sequence, or it can terminate with a special *infinite-duration timed step*:

$$(q\_n, x\_n) -f,\infty\rightarrow$$

Zenoness

A run may take an **infinite** number of discrete transitions
in a **finite** amount of time

Such runs are *physically meaningless*

They should be discarded

Zenoness in Two Tanks

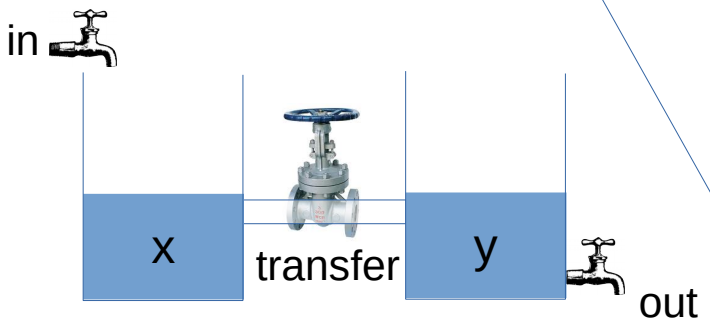
system could go back and forth between loc 0 and 1

$\dot{x} = \dot{y} = 0$

$\dot{x} = 1$
$\dot{y} = 0$

(in=off, tran=off, out=off) 0 1 (on, off, off)

(off, on, off) 2 3 (off, off, on)



Avoiding Zenoness

We assume that the system is **strongly non-Zeno**

i.e., there's a lower bound to the time needed to perform any loop of discrete transitions.

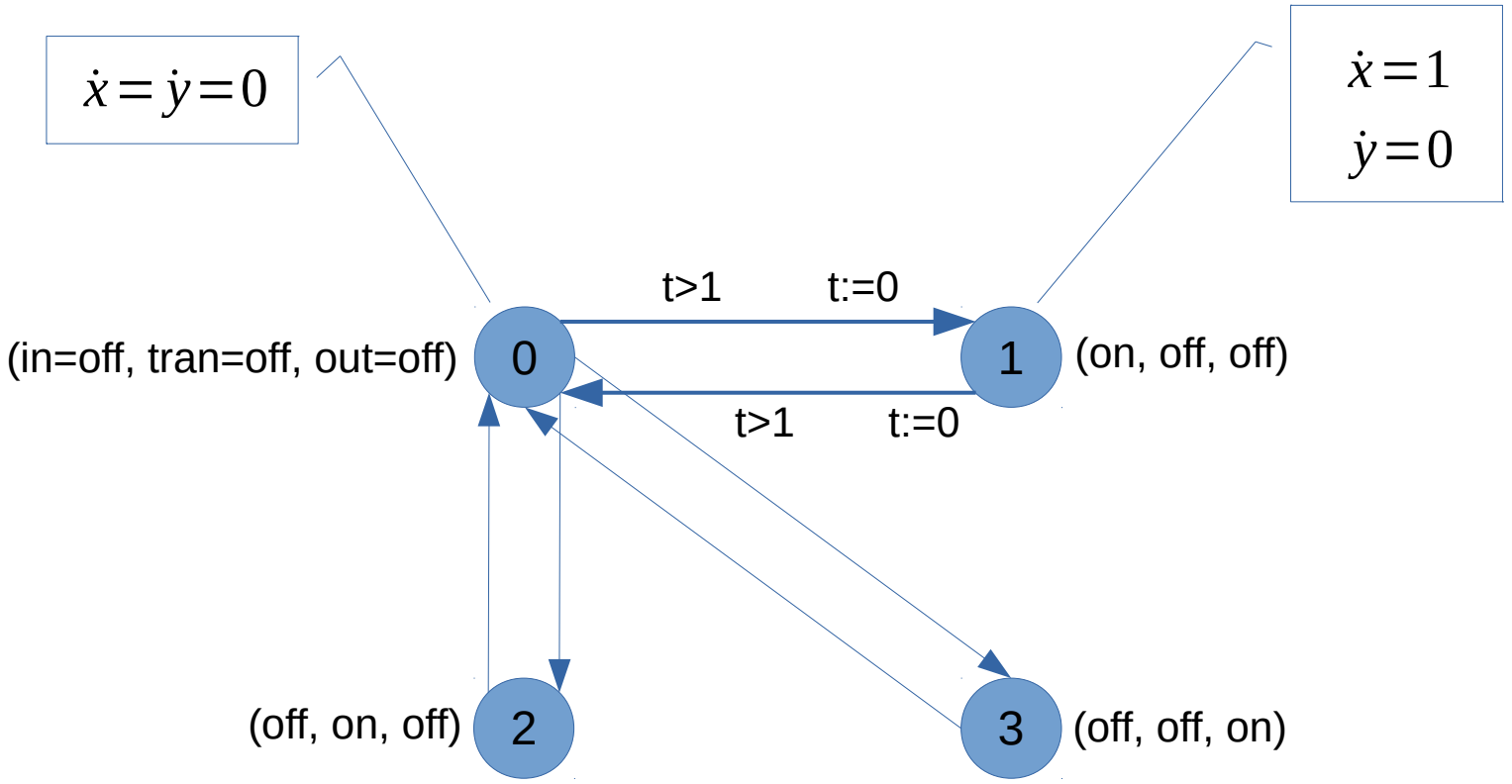
This is a simple syntactic approach to Zenoness.

More sophisticated (semantic) approaches are possible.

one such approach in [de Alfaro et al., CONCUR 03]

Strongly non-Zeno Fragment for Two Tanks

introduce an extra variable t (clock);
in all locations: $\dot{t} = 1$



Polyhedra

- **Convex polyhedron**: conjunction of finitely many linear inequalities
  - strict (<) and non-strict (≤)
- **Polyhedron**: union of finitely many convex polyhedra
- Examples:
  - a rectangle is a convex polyhedron
  - the interior of a rectangle is a convex polyhedron
  - an L-shaped polygon is a (non-convex) polyhedron
  - a disk is not a polyhedron

more on this later...

What kinds of differential equations?

1. Affine Hybrid Automaton: *linear* diff. eq.

2. Linear (sic!) Hybrid Automaton: *polyhedral* diff. inclusions

3. Rectangular Hybrid Automaton: *rectangular* diff. inclusions

What kinds of differential equations?

1. Affine Hybrid Automaton: *linear* diff. eq.

- $\dot{x} = A\,x + B\,d$

- d = vector of *disturbance* variables → non-determinism
- d ∈ D, set of admissible disturbances
- Example: $\left\{ \begin{array}{c} \dot{x}_1 = x_2 \\ \dot{x}_2 = -a\,x_1 \end{array} \right\}$ (a pendulum)

- Classical model from dynamic systems
- Very expressive

What kinds of differential equations?

2. Linear (sic!) Hybrid Automaton: *polyhedral* diff. inclusions

- $\dot{x} \in F$
- where *F* is a convex polyhedron
- i.e., *a set of linear constraints on the derivatives*

- special case of affine, when:
  - A=0 (no dependency on the current state)
  - B=I (identity matrix)
  - D=F (allowed disturbances)
- in other words, the environment picks directly the state derivative

What kinds of differential equations?

3. Rectangular Hybrid Automaton: rectangular diff. inclusions

- $\dot{x} \in F$
- special case of Linear, but *F* is a *hyper-rectangle* (Cartesian product of intervals)

Decision Problem

- **(point-to-point) Reachability**: given two states s, t, is there a run from s to t

- Application: **safety verification**
- Check whether a (specific) error state is reachable from a (specific) initial state

Practical Problems

In practice, we seek to solve the following:

- **Forward reachability problem (FRP)**: given a set of *initial states*, compute the set of states that are reachable from them
- **Backward reachability problem (BRP)**: given a set of *error states*, compute the set of states that can reach them

These sets must be *effective*

Effective Sets of States

- How do you represent a subset of $\mathbb{R}^n$?
- You choose an *encoding*

- However, you cannot hope to represent *all* subsets of $\mathbb{R}^n$
- You choose a *countable* class of subsets

Effective Sets of States

- Countable classes of subsets of $\mathbb{R}$:
  - intervals with rational endpoints
  - finite sets thereof
- Countable classes of subsets of $\mathbb{R}^n$:
  - hyper-rectangles with rational endpoints
  - convex polyhedra with rational coefficients
  - finite sets thereof
  - ...

Decision Problem

- In practice, we seek to solve the following:
  - **Forward reachability problem (FRP)**: given an effective set of *initial states*, compute the set of states that are reachable from them
  - **Backward reachability problem (BRP)**: given an effective set of *error states*, compute the set of states that can reach them
- Point-to-point reachability can be reduced to FRP and also to BRP
- BRP and FRP are inter-reducible provided the model supports inversion of time (most models do)

Algorithms

- How can the reachability problem be solved?

- Two approaches:
  - Finite bisimulation quotient (a.k.a. indirect approach)
  - On-the-fly exploration of the state-space (direct approach)

The Indirect Approach

- Discretize the automaton
- Prove existence of a **bisimulation** relation with finite index
  - similar to region construction for Timed Automata
- If such bisimulation is computable, the reachability problem (and much more) is decidable

The Indirect Approach

- Bisimulation is defined w.r.t. a finite set of actions or *observables*

- So, assume a finite partition **Obs** of the state space
  - each element of Obs represents the set of states sharing the same observable
  - e.g., locations may be observable
- We will in fact solve reachability w.r.t. Obs
  - reachability between observables

A Decidable Class

Initialized rectangular hybrid automata (IRHAs)

RHAs where: if the dynamics of a variable *x* changes when taking a transition, then *x* is re-initialized to a constant value.

Theorem [Henzinger et al.] The reachability problem for IRHAs is decidable.

More precisely, each IRHA is **bisimilar** to a finite-state system, which can be effectively computed.

Undecidable Extensions

- All main features of IRHAs are *necessary* for decidability

- The following are **undecidable**:
  - Standard RHAs
  - Initialized, but *triangular* dynamics
  - LHAs, etc.

- Proof idea: reduction from halting problem of 2-counter machines; value k of counter is encoded as value 2^{1-k} of a variable.

Another Decidable Class

Order-minimal hybrid systems

- Rich dynamics (even affine)
- Restricted transitions
  - All variables are re-initialized for each transition
  - Discrete behavior can be decoupled from continuous one

O-minimal Structures

Model-theoretic definition

insert constants, functions, and relations here

A totally ordered structure (M,<, ...) is **o-minimal** if any definable set is a finite union of points and open intervals.

here, definable means definable by a first-order formula with parameters

I.e., in an o-minimal structure the definable sets are the *simplest possible*: they are already definable in (M,<).

Examples of O-minimal Structures

1. The field of reals: $(\mathbb{R}, <, 0, 1, +, \bullet)$

This structure also admits effective quantifier elimination [Tarski,51]

2. The field of reals with the exponential function: $(\mathbb{R}, <, 0, 1, +, \bullet, \exp)$

It is not known if this structure admits effective quantifier elimination

see Tarski's exponential function problem

3. The field of reals with the exponential function and a finite number of analytic functions restricted to the unit hyper-cube:

$$(\mathbb{R}, <, 0, 1, +, \bullet, \exp, h_1, h_2, \ldots, h_n)$$

sine is an analytic function

O-minimal HAs

Consider a hybrid automaton whose dynamics is deterministic

i.e., for each state (q, x) there is a unique trajectory $\text{traj}_{q,x}$ from it.

Call *dynamics of q* the function $\mathbf{dyn_q} : (x, t) \rightarrow \text{traj}_{q,x}(t)$

Definition. The automaton is **o-minimal** if there exists an o-minimal structure M such that:

- for each location q the function dyn_q belongs to M (in the sense that its *graph* belongs to M)
- all invariants, jump relations, and observables belong to M
- **all variables are reset to constant values** after each discrete transition

Existence of a Finite-state Abstraction

Theorem. Each o-minimal HA is bisimilar to a finite-state system.

Decidability

The existence of a bisimilar finite-state system **does not guarantee decidability!**

Indeed, the finite-state system may not be computable.

To obtain decidability, we need decidability of the existential first-order theory of the underlying o-minimal structure.

References for O-minimality

1) L. van de Dries, Tame topology and o-minimal structures (book)

comprehensive treatment of o-minimality

2) M.J. Edmundo, An introduction to o-minimal structures

concise introduction to o-minimality

3) T. Brihaye & C. Michaux, On the expressiveness and decidability of o-minimal hybrid systems

key paper for o-minimal hybrid systems

4) P. Tabuada, **Verification and control of hybrid systems** (book)

concise introduction to o-minimal hybrid systems (and much more!)

The Direct Approach

- Decidability requires:
  - very simple dynamics (rectangular), or
  - very simple transitions (o-minimal)
- For the others: on-the-fly exploration of the state space
- Two sub-approaches:
  - Surrender exactness ⇨ approximate algorithms
  - Surrender termination ⇨ exact semi-algorithms

Approximate Algorithms

Consider the Forward Reachability Problem

Since the objective is safety verification, **over-approximate** the set of reachable states

- if the error is not reachable under the approximation, it is not reachable in the original system
- *better safe than sorry*

Approximate Algorithms

Abstractions used:

- polyhedra [HyTech, etc.]
- griddy polyhedra [Asarin et al., Proc. IEEE]
- zonotopes [Girard, HSCC 05]
- support functions [Frehse et al., SpaceEx]
- flowpipe sampling [Frehse et al., HSCC 13]
- ellipsoids [Kurzhanski & Varaiya, HSCC 00]

Exact Semi-algorithms

- An algorithm that may or may not terminate
- When it terminates, it provides the exact answer

- It may be stopped after a deadline
- It will provide the exact answer up to a **fixed number of discrete steps**
  - bounded horizon reachability