Modeling and Analysis of Hybrid Systems

Some decidability and undecidability results

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen University

SS 2015

Subclasses of hybrid automata for which reachability is decidable:

- Timed automata
- Initialized stopwatch automata
- Initialized singular automata
- Initialized rectangular automata
- Timed automata with difference constraints $x - y \sim c$
- Simple multirate timed systems

Subclasses of hybrid automata for which reachability is undecidable:

- Discrete automata
- Uninitialized stopwatch automata
- Uninitialized singular automata
- Uninitialized rectangular automata
- 2-rate timed systems

Decidability: Timed automata with difference constraints

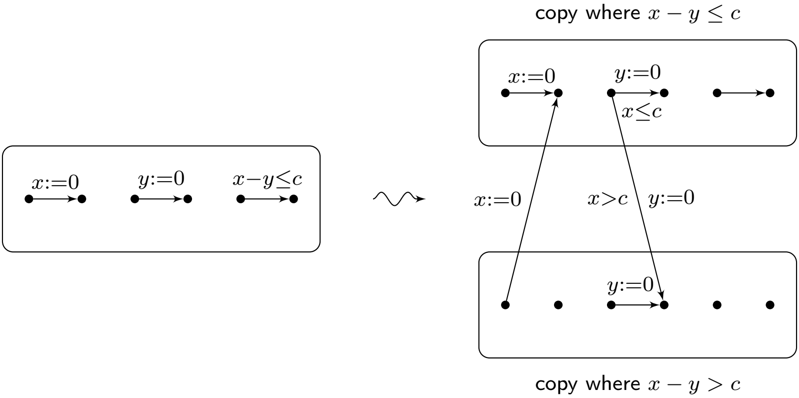
Difference constraint:

$x - y \sim c$ with x, y being clocks and c a non-negative integer

Decidability: Timed automata with difference constraints

Difference constraint:

$x - y \sim c$ with x, y being clocks and c a non-negative integer



A state is reachable in the original system iff it is reachable in one of the copies.

Multirate timed systems

- A skewed clock is a variable x with $\dot{x} = c$ in all locations for some $c \in \mathbb{Z}$.
- Multirate timed systems have
  - skewed clocks as variables,
  - resets to 0,
  - clock constraints $x \sim c$ and equality constraints $x = y$ in conditions and invariants.
- Simple multirate timed systems have no equality constraints.
- 2-rate timed systems are multirate timed systems with skewed clocks at two different rates.

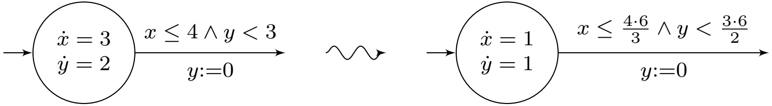
Decidability: Simple multirate timed systems

Decidability: Simple multirate timed systems

For each variable x let k_x denote its derivative and let k be the smallest common multiple of all non-zero derivatives. For each variable x with $k_x \neq 0$ we set its derivative to 1 and replace in all

- initial conditions,
- location invariants and
- transition guards

each clock constraint $x \sim c$ by $x \sim \frac{c \cdot k}{k_x}$.



Let $f : V \to V$ with $f(\nu)(x) = \nu(x)$ if $k_x = 0$ and $f(\nu)(x) = \frac{\nu(x) \cdot k}{k_x}$ otherwise. Then (l, ν) is reachable in the original system iff $(l, f(\nu))$ is reachable in the transformed system.

Proven undecidable: 2-counter machines

A 2-counter machine [Minsky (1961, 1967), Lambek (1961)] consists of

- 2 unsigned-integer-valued registers,
- a program counter, and
- a list of labelled sequential instructions:
  - increment a register and let the other register unchanged
  - decrement a register and let the other register unchanged
  - if a given register contains 0 then jump to a given instruction else continue in sequence; the register values remain unchanged

Proven undecidable: 2-counter machines

A 2-counter machine [Minsky (1961, 1967), Lambek (1961)] consists of

- 2 unsigned-integer-valued registers,
- a program counter, and
- a list of labelled sequential instructions:
  - increment a register and let the other register unchanged
  - decrement a register and let the other register unchanged
  - if a given register contains 0 then jump to a given instruction else continue in sequence; the register values remain unchanged

To encode the computations of a 2-counter machine by a 2-rate timed system we need to encode

- setting up the initial configuration,
- changing the program counter,
- testing a register for 0,
- letting a register unchanged,
- incrementing a register, and
- decrementing a register.

Undecidability: Uninitialized singular automata

Undecidability: 2-rate timed systems

Encoding the register values

Encoding the register values

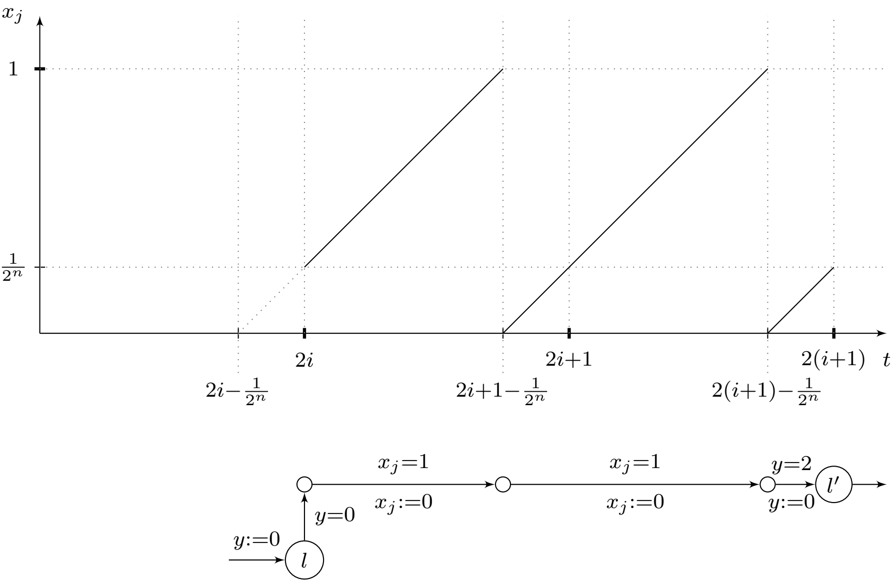
- We use two clocks x_1 and x_2 of rate 1 to encode the register values.

  The ith state of the 2-counter machine is encoded by the state of the 2-rate timed system at time $2i$.

  The value n of register i is encoded by the value $1/2^n$ of x_i.
- We use a clock y of rate 1 to measure the step length 1; it is reset to 0 whenever it reaches the value 1.
- We additionally use a clock z of rate 1,
  and a skewed clock z' of rate 2.

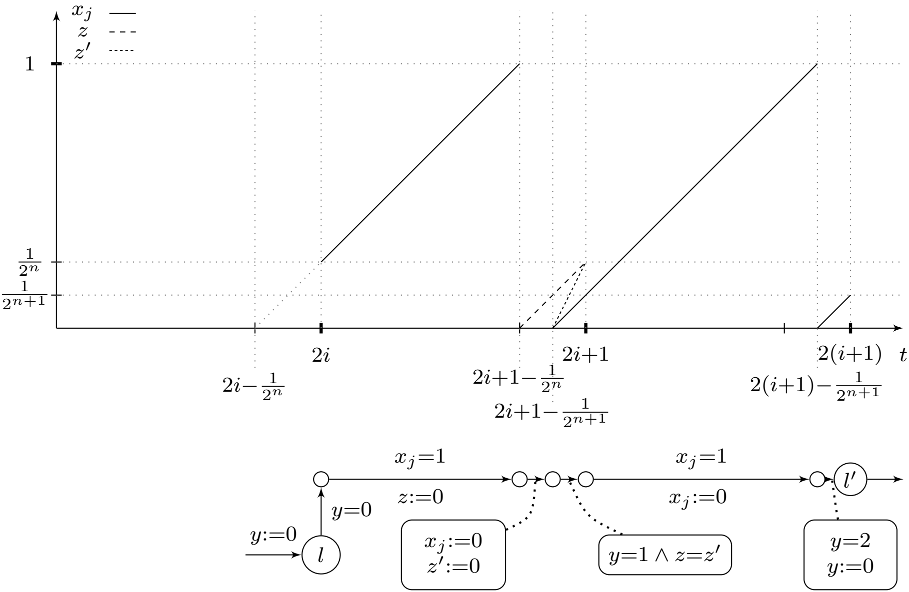
Letting a register unchanged

Letting a register unchanged



Incrementing a register

Incrementing a register



Decrementing a register

Decrementing a register