Hytech

Arkadeb Ghosal

Overview

- Introduction
- Hybrid automaton
- What do we want?
- A closer look
- Demos
- References

Overview

- Introduction
- Hybrid automaton
- What do we want?
- A closer look
- Demos
- References

What is Hytech?

- A model checker for Hybrid systems
- A tool for automated analysis of embedded systems
- Procedure for checking linear CTL requirements of linear hybrid automata has been implemented in tool Hytech

Hytech Contributors

- Thomas Henzinger
- Rajeev Alur
- Pei-Hsin Ho
- Howard Wong-Toi
- Peter Kopke
- Jorg Preubig
- Benjamin Horowitz
- Rupak Majumdar

Overview

- Introduction
- Hybrid automaton
- What do we want?
- A closer look
- Demos
- References

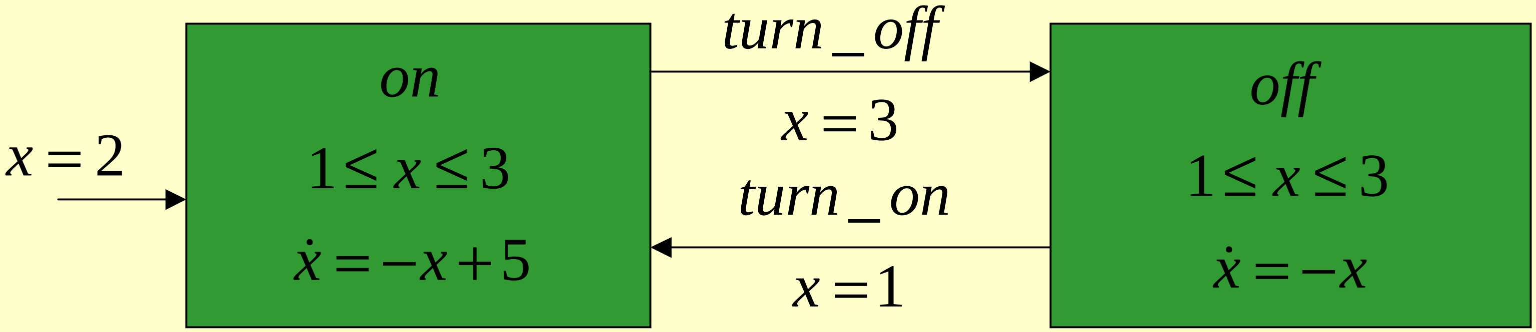
Examples of Hybrid systems

- manufacturing controllers
- automotive and flight controllers
- medical equipment
- micro-electromechanical systems
- robots
- mission critical applications

Hybrid Automaton

- A hybrid automaton $A = (X, V, flow, inv, init, E, jump, \Sigma, syn)$
  - Variables
  - Control Modes
  - Flow conditions
  - Invariant conditions
  - Initial conditions
  - Control switches
  - Jump Conditions
  - Events

Thermostat automaton



Flow and jumps

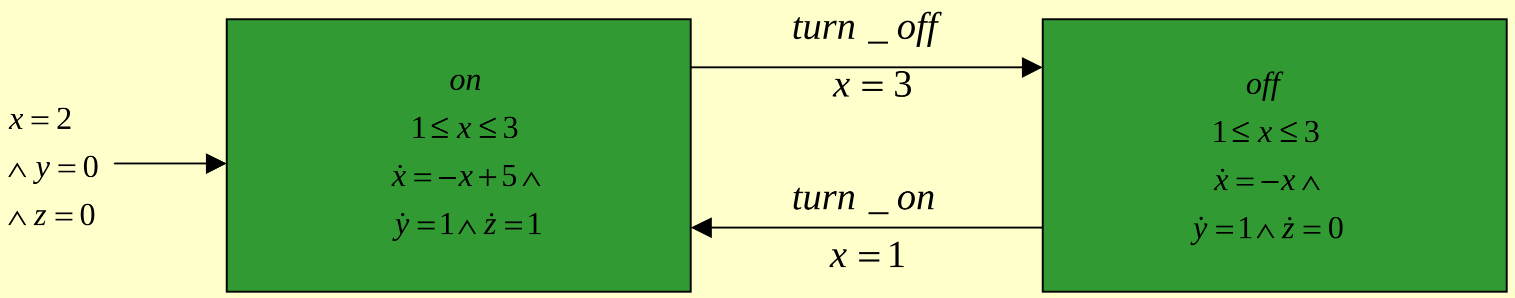
- states
  - the state (on,1.5) is admissible while the state (on, .5) is not
- jumps
  - thermostat automaton has two jumps ((on,3), (off,3)) and ((off,1),(on,1))
- flows
  - ((off,3),(off,2)) and ((off,3), (off,2.5)) are flows of thermostat automaton
- trajectories
  - a finite sequence of admissible states
  - first state is an initial state and each pair of consecutive states in the sequence is either a jump or flow

Overview

- Introduction
- Hybrid automaton
- What do we want?
- A closer look
- Demos
- References

Safety requirements

- what is a safety requirement?
  - it asserts that nothing bad will happen
  - often specified by describing the "unsafe" values
- *A satisfies the safety requirement* specified by *unsafe* if the state assertion *unsafe* is false for all reachable states of *A*



Thermostat automaton augmented for safety verification

Computing reachable states

- Given a state assertion *unsafe* we try to compute another state assertion *reach* which is true for reachable states of the automaton
  - for a state assertion φ, *Post(*φ) is a state assertion that is true for the jump and flow successors of the φ-states
- Success of computation of reach depends on
  - *Post(*φ) can be calculated reasonably efficiently for a restricted class of hybrid automata called **linear hybrid automata**
  - Iterative computation of reach must converge within a finite number of *Post* applications and this can be guaranteed for certain restricted class of linear hybrid automata such as class of **timed automata**

Linear Hybrid Automata

- hybrid automaton *A* is *linear hybrid automaton* if it satisfies
  - Linearity : for every control mode, the flow condition, the invariant condition, and the initial condition are convex linear predicates and for every control switch jump condition is a convex linear predicate
  - flow independence : for every control mode, the flow condition is a predicate over the variables in $\dot{X}$ only and not in X
    - quite limiting but it allows
      - clocks
      - stopwatches
      - clocks with bounded drift

Linear Hybrid Automata

- Theorem:
  - If *A* is a linear hybrid automaton and φ is a linear state assertion for *A*, then *Post*(φ) can be computed and the result again is again a linear state assertion for *A*
    - every flow curve can be replaced by a straight line between the two endpoints
- This theorem enables
  - automatic analysis
  - safety verification
  - temporal model checking

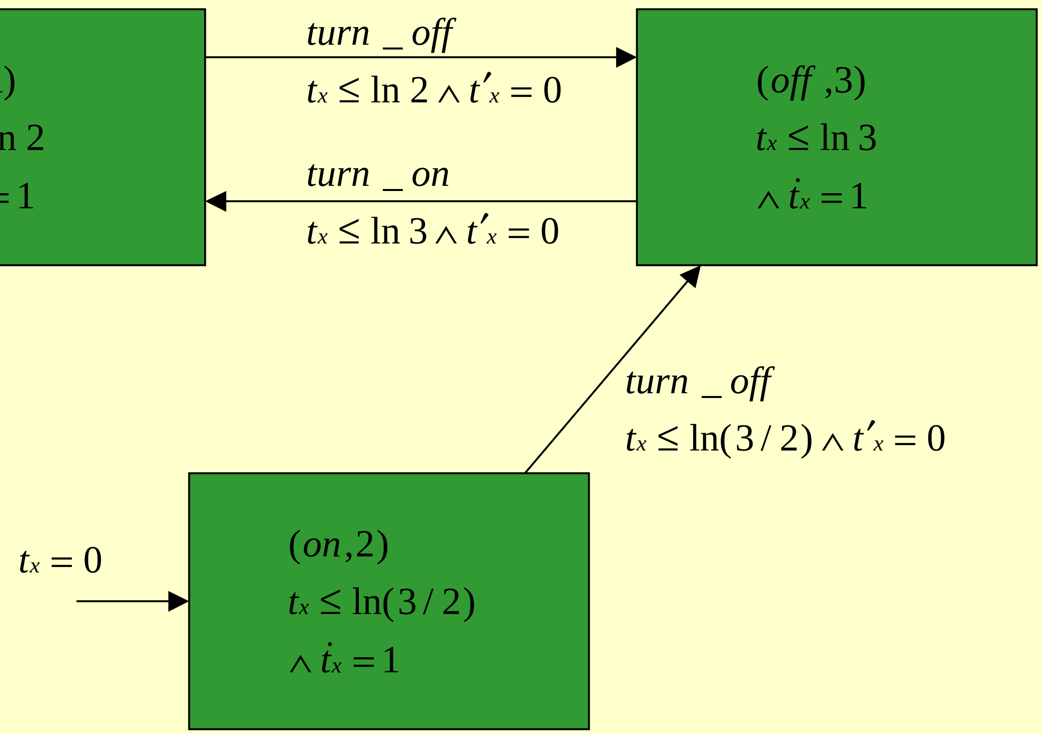
Overview

- Introduction
- Hybrid automaton
- What do we want?
- A closer look
- Demos
- References

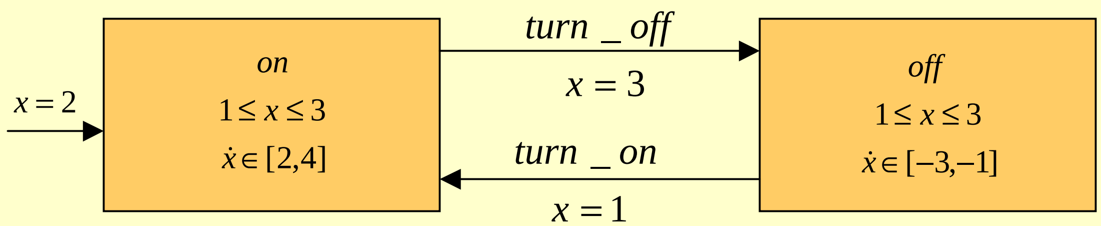
Non-linear to linear hybrid automata

- Clock Translation
- Linear phase-portrait approximation

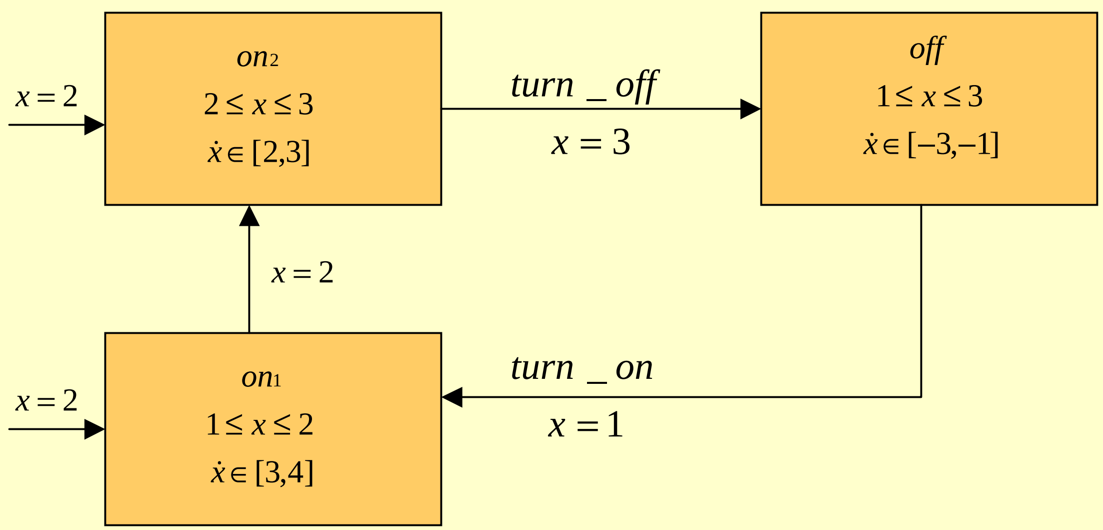
Clock translation



Linear phase-portrait approx.



Linear phase portrait approx. of thermostat automaton

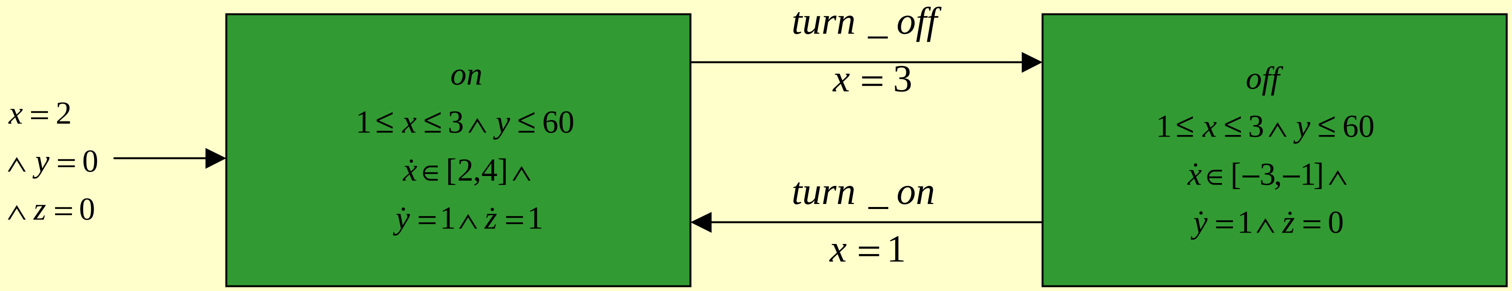


Tighter Linear phase portrait approx. of thermostat automaton

Safety Verification

Property to be verified:

The heater is active for less than 2/3 of the first hour of operation

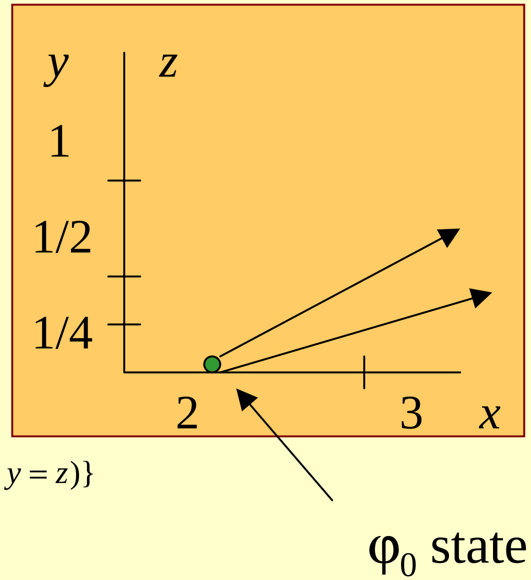


Unsafe state:

$$y=60\wedge z\ge 2y/3$$

Safety verification

Initial state $\varphi_0 = init = \{(on, x = 2 \wedge y = 0 \wedge z = 0), (off, false)\}$



Jump successor: none

Flow successor $\varphi_1 = Post(\varphi_0)$

$$= \{(on, x \le 3 \wedge 2z + 2 \le x \le 4z + 2 \wedge y = z), (off, false)\}$$

$\varphi_2 = Post(\varphi_1)$ Jump successor $\{(on, false), (off, x = 3 \wedge \frac{1}{4} \le z \le \frac{1}{2} \wedge y = z)\}$

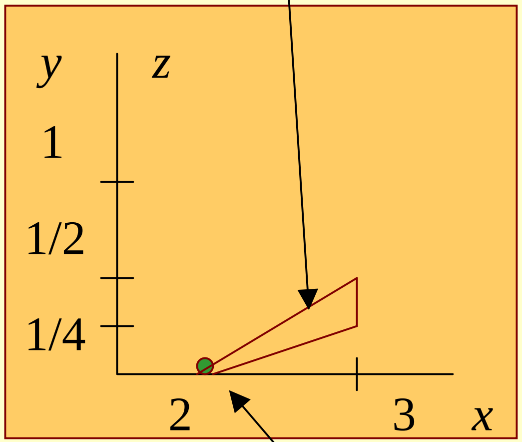
Flow successor : closed

$$\varphi_2 = Post(\varphi_1) = \{(on, x \le 3 \wedge 2z + 2 \le x \le 4z + 2 \wedge y = z), (off, x = 3 \wedge \frac{1}{4} \le z \le \frac{1}{2} \wedge y = z)\}$$

$$\varphi_3 = Post(\varphi_2) = \{(on, x \le 3 \wedge 2z + 2 \le x \le 4z + 2 \wedge y = z), (off, 1 \le x \le 3 \wedge z + \frac{2}{3} \le y \le z + 2 \wedge 2z \le x \le 4z)\}$$

$$\varphi_3 = Post(\varphi_2) = \{(on, x \le 3 \wedge 2z + 2 \le x \le 4z + 2 \wedge y = z) \vee (x = 1 \wedge \frac{1}{4} \le z \le \frac{1}{2} \wedge z + \frac{2}{3} \le y \le z + 2)), (off, 1 \le x \le 3 \wedge z + \frac{2}{3} \le y \le z + 2 \wedge 2z \le x \le 4z)\}$$

Safety verification



Initial state $\varphi_0 = init = \{(on, x = 2 \wedge y = 0 \wedge z = 0), (off, false)\}$

Jump successor: none

Flow successor $\varphi_1 = Post(\varphi_0)$

$$= \{(on, x \le 3 \wedge 2z + 2 \le x \le 4z + 2 \wedge y = z), (off, false)\}$$

$\varphi_2 = Post(\varphi_1)$ Jump successor $\{(on, false), (off, x = 3 \wedge \frac{1}{4} \le z \le \frac{1}{2} \wedge y = z)\}$

Flow successor : closed

$$\varphi_2 = Post(\varphi_1) = \{(on, x \le 3 \wedge 2z + 2 \le x \le 4z + 2 \wedge y = z), (off, x = 3 \wedge \frac{1}{4} \le z \le \frac{1}{2} \wedge y = z)\}$$

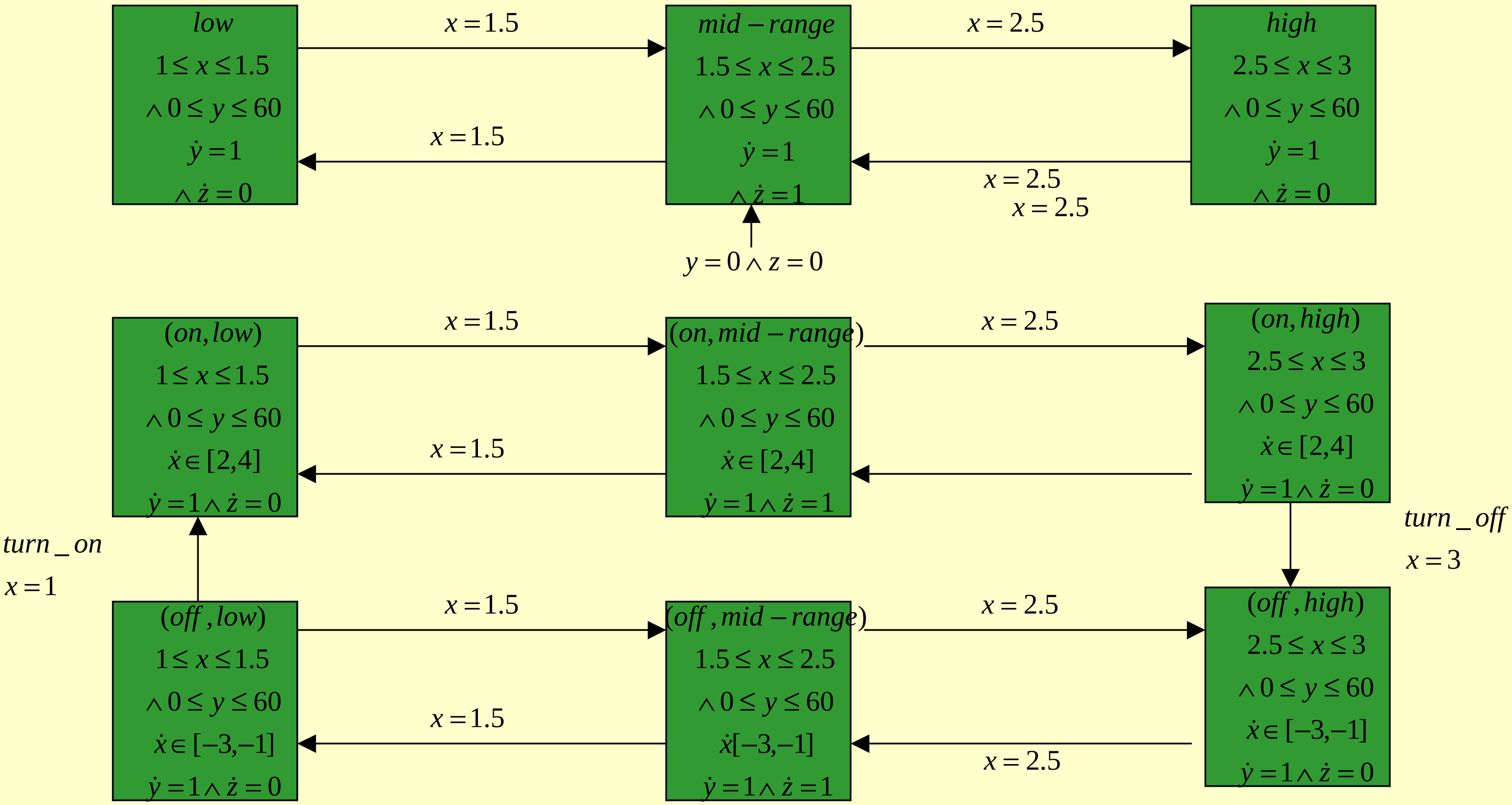
$$\varphi_3 = Post(\varphi_2) = \{(on, x \le 3 \wedge 2z + 2 \le x \le 4z + 2 \wedge y = z), (off, 1 \le x \le 3 \wedge z + \frac{2}{3} \le y \le z + 2 \wedge 2z \le x \le 4z)\}$$

$$\varphi_3 = Post(\varphi_2) = \{(on, x \le 3 \wedge 2z + 2 \le x \le 4z + 2 \wedge y = z) \vee (x = 1 \wedge \frac{1}{4} \le z \le \frac{1}{2} \wedge z + \frac{2}{3} \le y \le z + 2)), (off, 1 \le x \le 3 \wedge z + \frac{2}{3} \le y \le z + 2 \wedge 2z \le x \le 4z)\}$$

Some related issues

- Monitors
  - safety requirements cannot always be specified by state assertions
  - sometimes it is convenient to build a separate automaton, called a monitor
    - it enters an unsafe state precisely when the original system violates a requirement
    - it observes the original system without changing its behavior
    - reachability analysis is then performed on the parallel composition of the system with the monitor

Monitors and Parallel Composition



Some related issues (cont.)

- Parametric analysis
  - High level system often use design parameters
    - symbolic constants with unknown fixed values
    - parameters are not assigned values until the implementation phase of design
- goal
  - to determine necessary and sufficient constraints on the parameters under which safety violations cannot occur

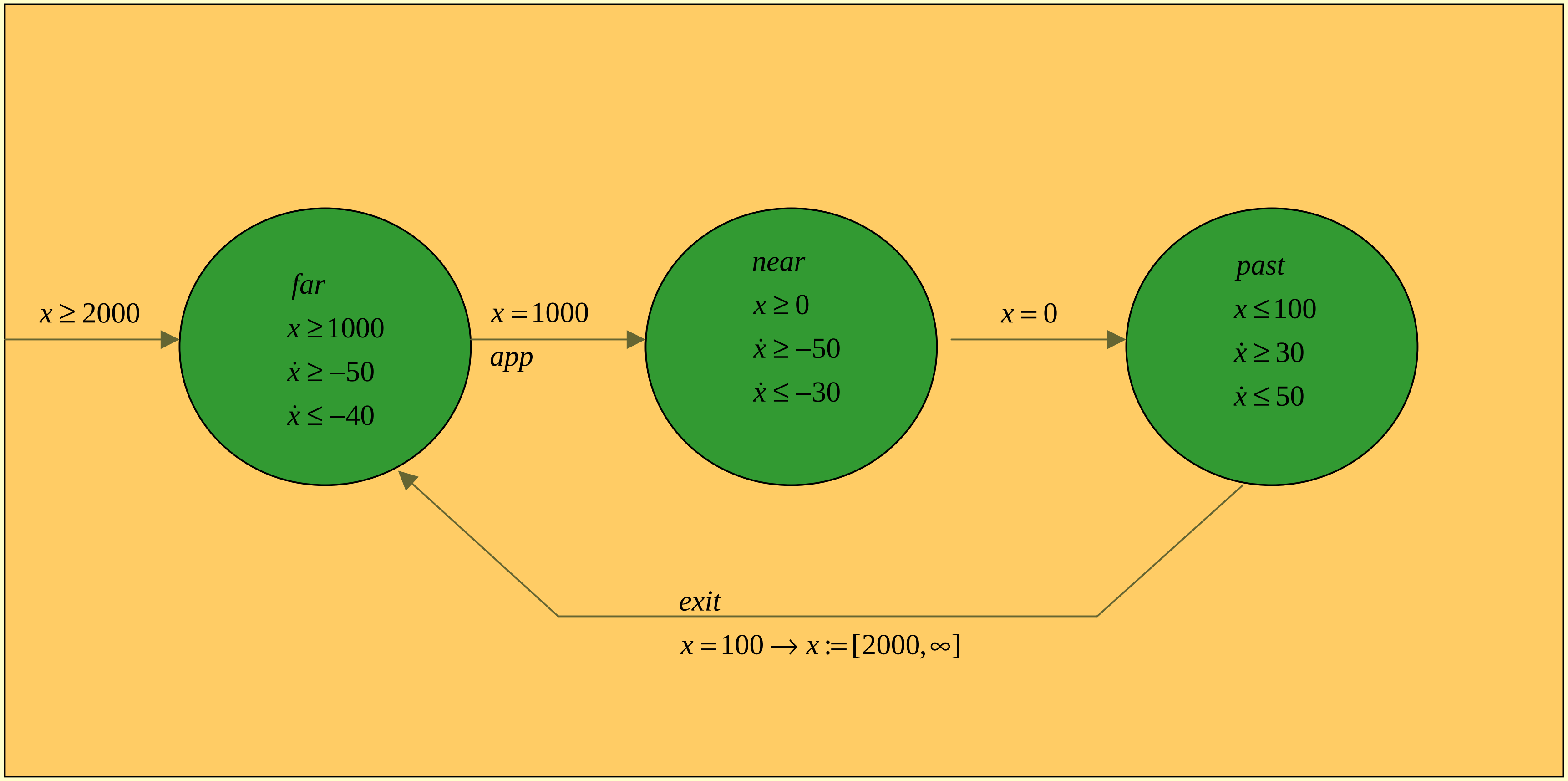
Overview

- Introduction
- Hybrid automaton
- What do we want?
- A closer look
- Demos
- References

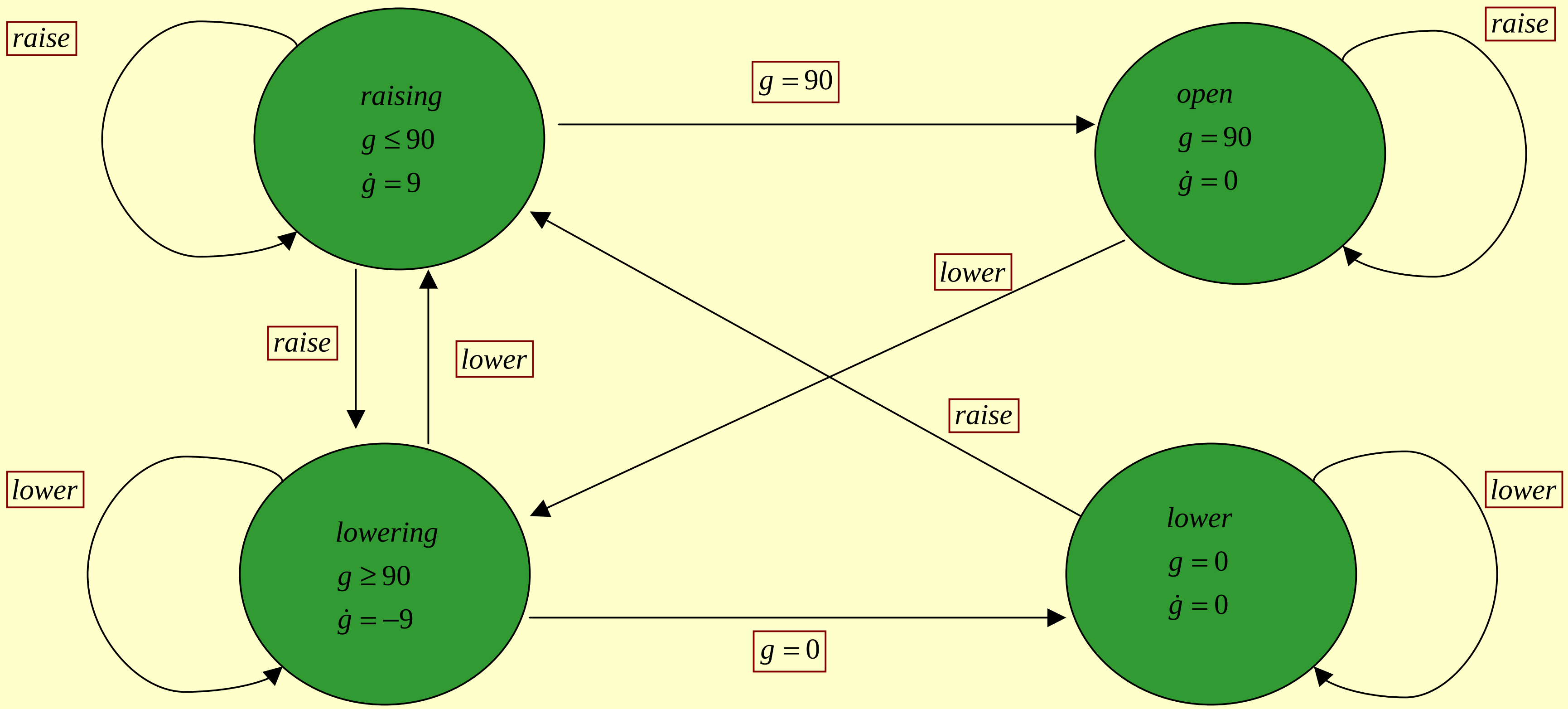
Examples

- A gas burner
- Trajectories of a billiard ball
- Temperature of a reactor core
- Fischer’s timing based mutual exclusion protocol
- Train-gate controller
- Corbett’s distributed control system
- Audio-control protocol

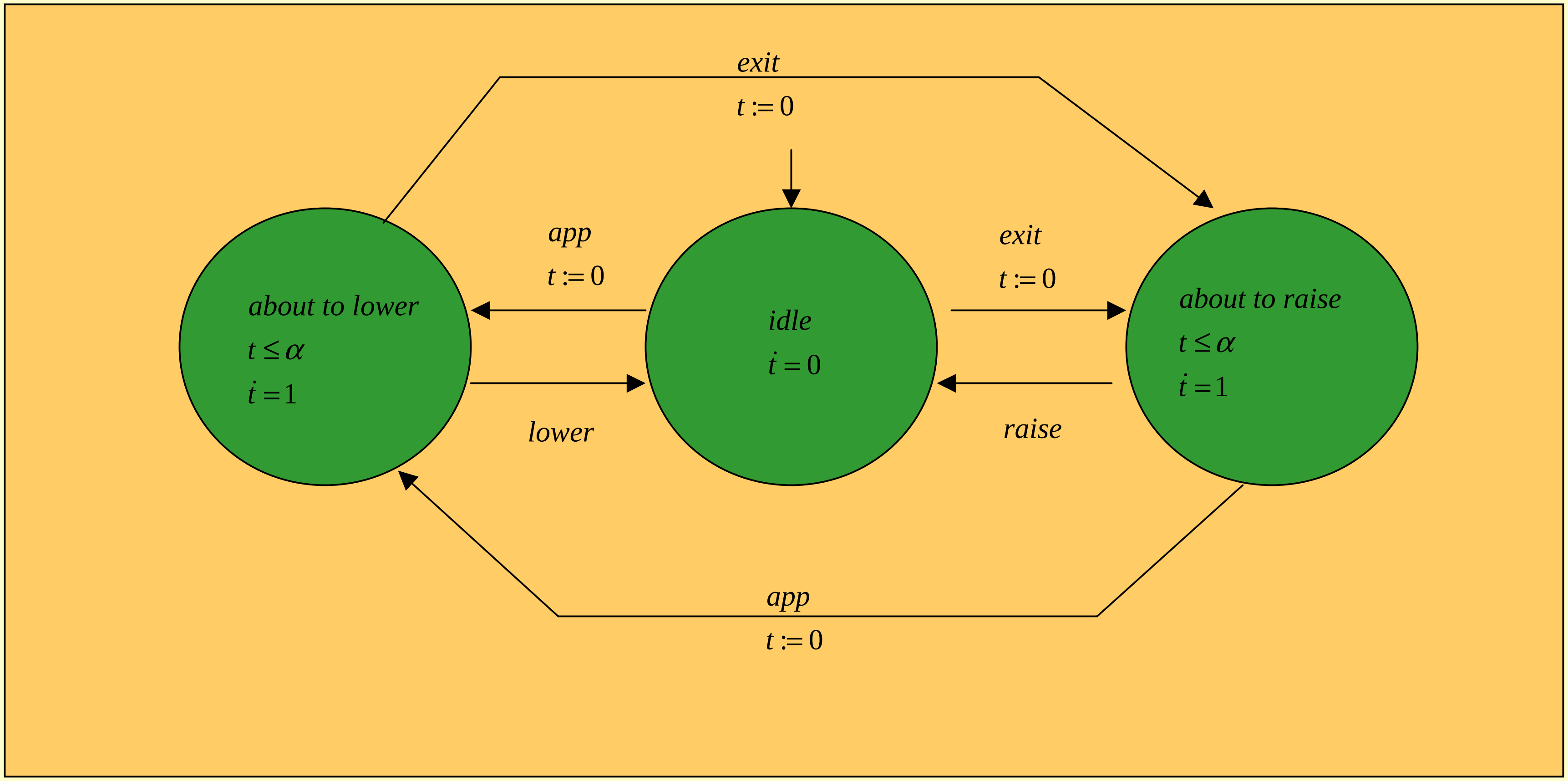
Train automaton



Gate Automaton



Controller automaton

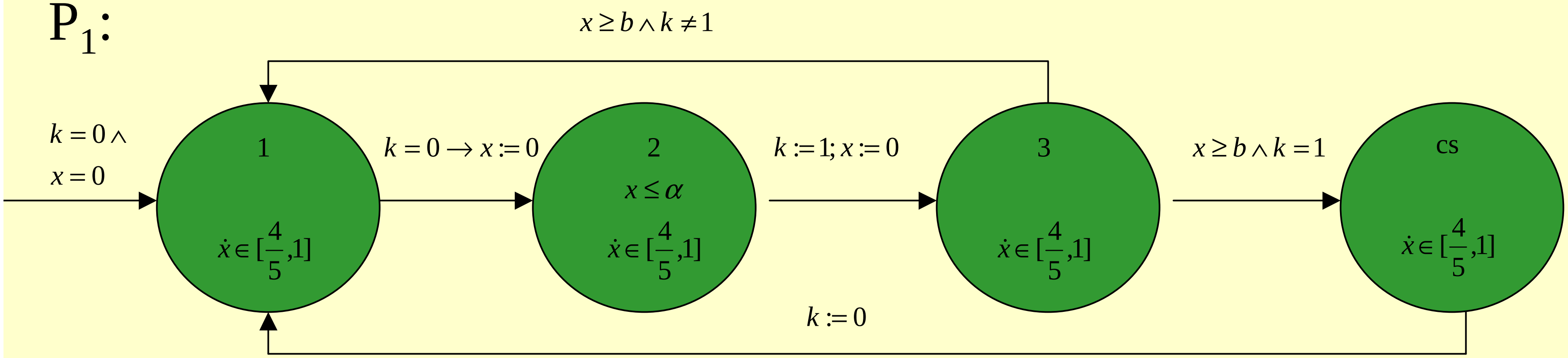


Mutual Exclusion Protocol

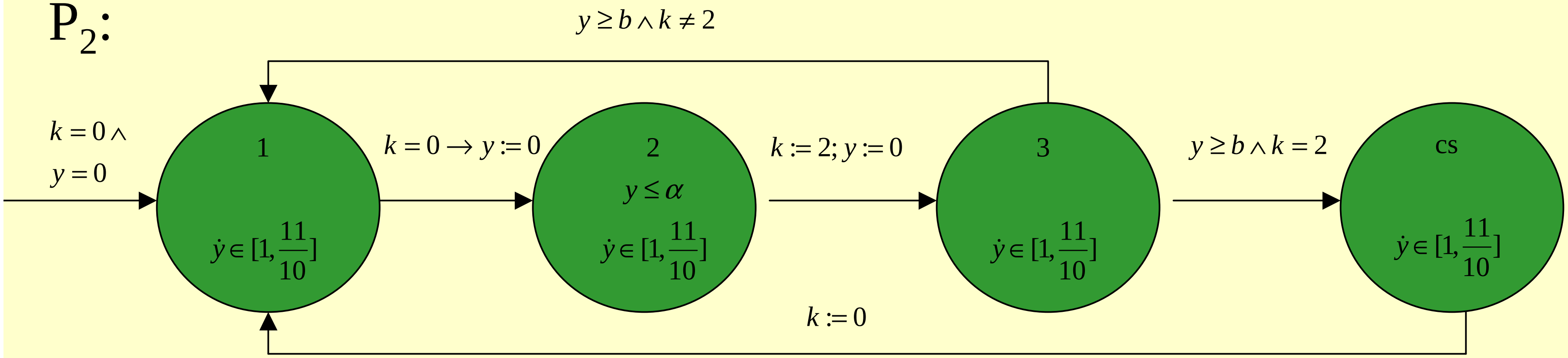
```
repeat
    repeat
        await k = 0 ;k  = c; delay b
    until k = c;
    Critical section
    k := 0;
forever
```

Mutual Exclusion Protocol

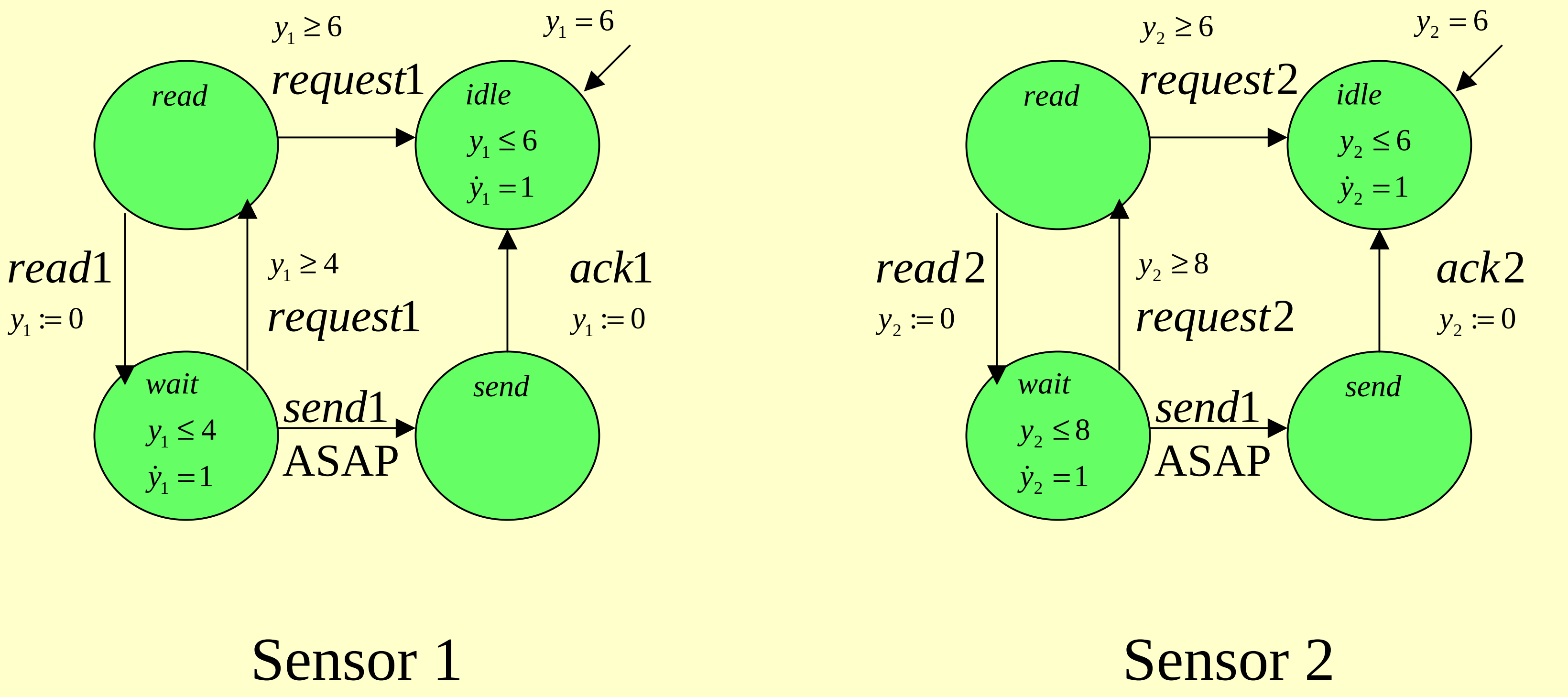
P_1:



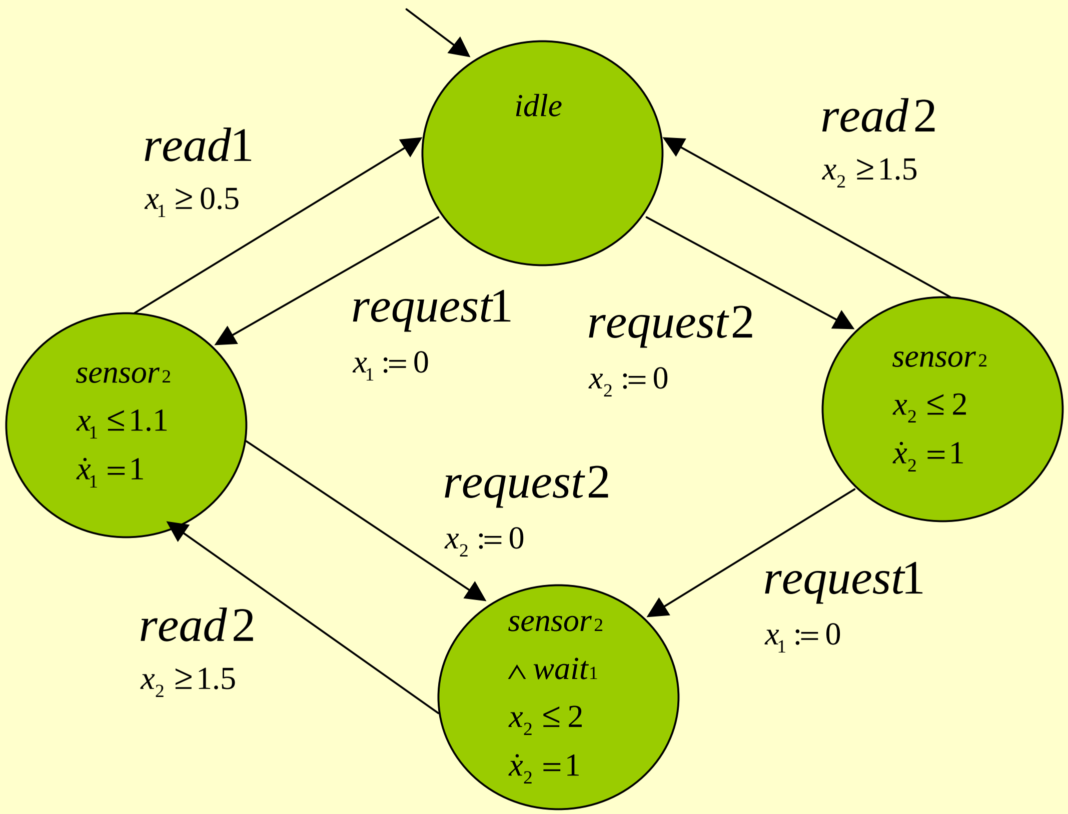
P_2:



Corbett's Distributed Controller

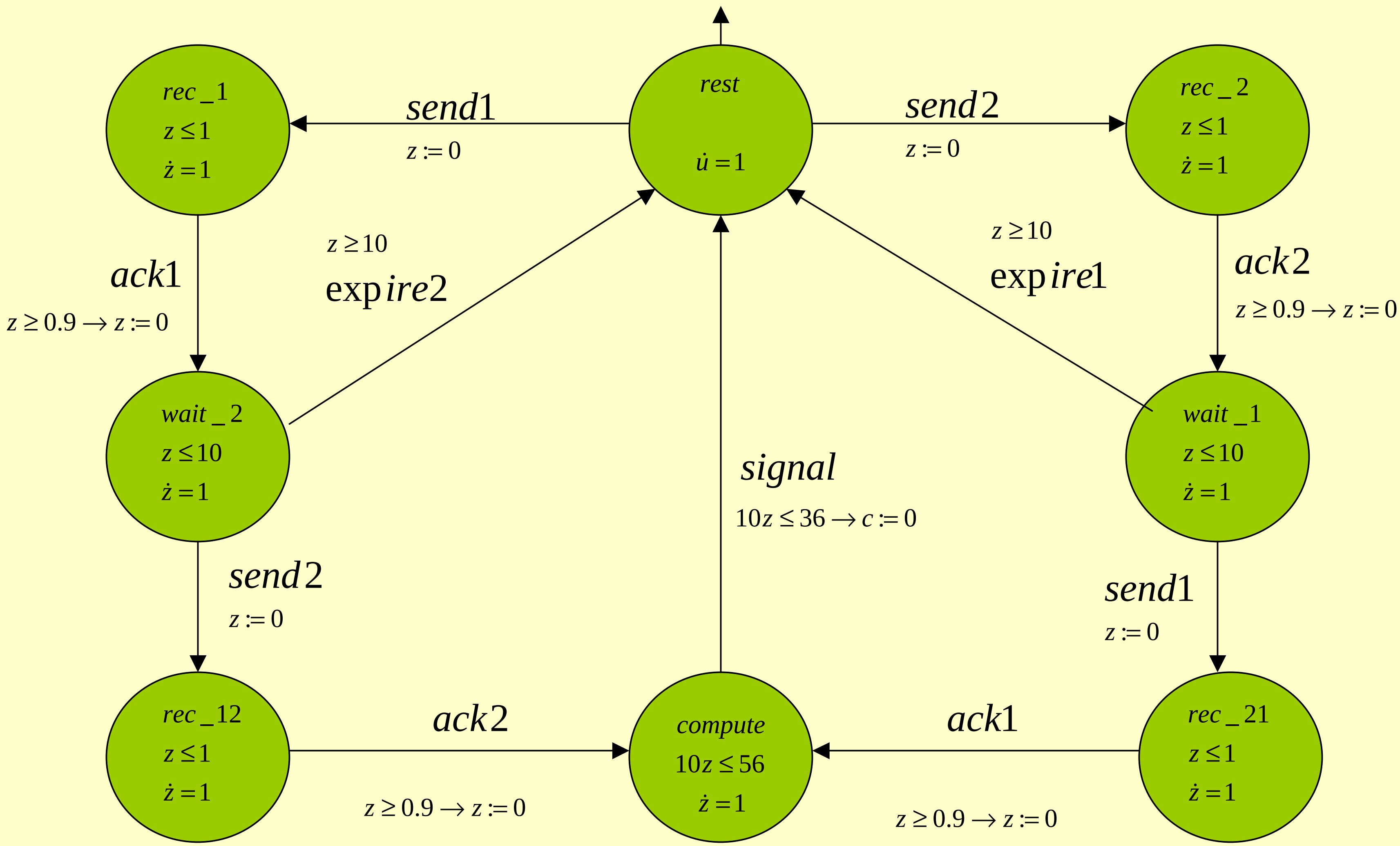


Corbett's Distributed Controller



Scheduler

Corbett's Distributed Controller



Controller

Overview

- Introduction
- Hybrid automaton
- What do we want?
- A closer look
- Demos
- References

References

www-cad.eecs.berkeley.edu/~tah/Hytech

References

- paper presented:
  - Hytech: A Model Checker for Hybrid Systems
- timed automaton
  - A theory of timed automata
- rectangular hybrid automaton
- bisimulation
  - The theory of hybrid automaton
- Integrator computation tree logic(ICTL)
  - Automatic Symbolic verification of Embedded Systems
- examples and brief overview
  - A user guide to Hytech
- talk on hytech
  - http://robotics.eecs.berkeley.edu/~koo/EE291E/Sp02/ (lec Apr 2 and 4)
- a nice example
  - A computational Framework for the verification and synthesis of Force-guided robotic assembly strategies (HSCC 2002)