Timed automata

Introduction

- Automata allow action sequences, loops, and choices to be specified
- But they do not allow the quantified time constraints of real-time systems to be modeled
  - Example 1: if the mouse is clicked twice within less than 50 ms then this is a double click
  - Example 2: the barrier of the railroad crossing lowers between 1 min and 45 s before the train arrival
- In this lecture: timed extension of automata to model this type of constraints

Modeling time elapsing (1/2)

2 representations are possible:

- Discrete time: measured at regular intervals (clock ticks as in electronic circuits)

  $\Rightarrow$ the time domain is a discrete set (e.g., $\mathbb{N}$)
  $\Rightarrow$ verification techniques are based on discrete maths

- Dense time: measured with (quasi) illimited precision

  $\Rightarrow$ the time domain is a dense set (e.g., $\mathbb{Q}^+$ or $\mathbb{R}^+$)
  $\Rightarrow$ verification techniques are based on continuous maths

Modeling time elapsing (2/2)

One may want to measure:

- Relative time: the time that elapses between two successive actions
  Example: 5 to 8 seconds elapse between pressing the door opening button and actual door opening
- Absolute time: the occurrence dates of actions
  Example: the door opened at 22 minutes (after the system start)
- The duration of actions
  Example: the door opening lasted 3 seconds

Timed transition system (1/2)

- Conceptual model / low level / infinite to introduce the notion of time and define the semantics of higher-level models
- Continuous-time model
  - Discrete-time can still be modeled ($\mathbb{N} \subseteq \mathbb{Q}^+ \subseteq \mathbb{R}^+$)
  - Enables methods without equivalent in discrete maths
- Relative-time model
  - Absolute time can be obtained by simulation (sum)
  - A lasting action can be modeled as two instantaneous begin/end actions and a relative-time constraints
    Example: 3 s elapse between begin_open and end_open

Timed transition system (2/2)

- Extension of the STE model
- A TTS is a 6-tuple $(S, A, \Delta, T, \Theta, s_0)$ such that
  - S is a set of states
  - A is a set of discrete (instantaneous) actions, whose elements are written $a, a_0, a_1, \ldots, a'$
  - $T \subseteq S \times A \times S$ is a set of discrete transitions
  - s_0 is the initial state
  - Δ is a dense time domain ($\mathbb{Q}$ or $\mathbb{R}$), whose elements are written $t, t_0, t_1, \ldots, t', \ldots$
  - $\Theta \subseteq S \times \Delta \times S$ is a set of timed transitions, representing time elapsing

LTS

Linear properties of time

Θ must satisfy two fundamental properties:

- Time determinism: time elapsing alone cannot lead to distinct states

$$\text{if } s \xrightarrow{t} s_1 \text{ and } s \xrightarrow{t} s_2 \text{ then } s_1 = s_2$$

- Time additivity: delays can be added

$$\text{if } s_1 \xrightarrow{t_1} s_2 \text{ and } s_2 \xrightarrow{t_2} s_3 \text{ then } s_1 \xrightarrow{t_1+t_2} s_3$$

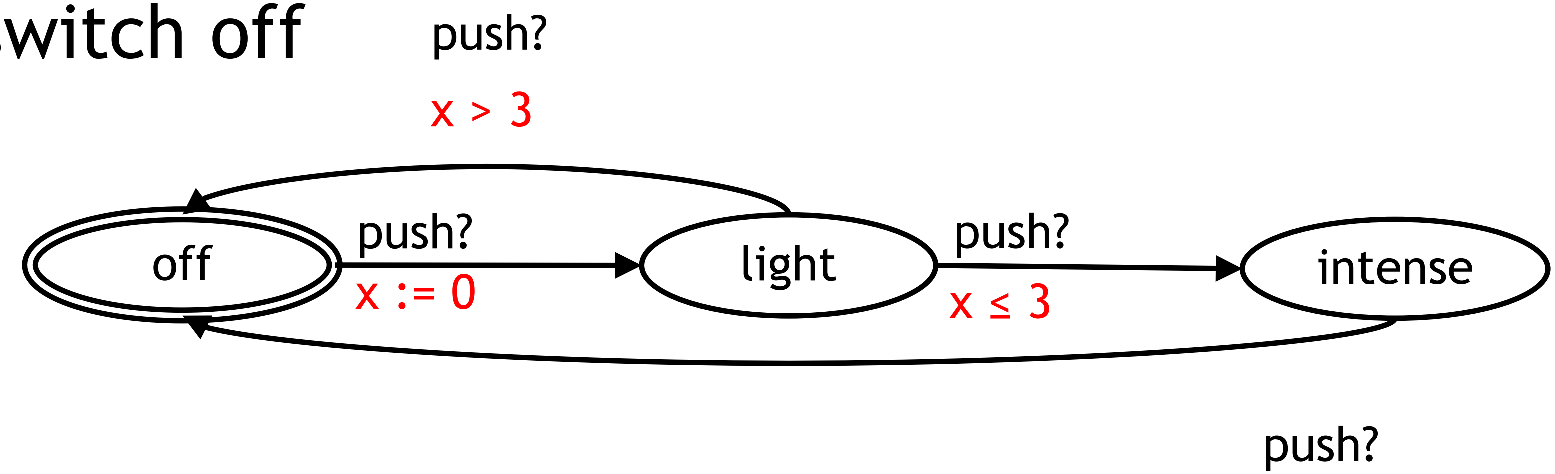
Θ is thus generally an infinite and even *non-enumerable set*

Timed automata (TA)

- It is not possible to describe a system directly as a TTS: a symbolic, finite representation is needed
- A popular formalism: timed automata (TA) [Alur-Dill-90]
  - Extension of CA
  - Add the notion of clocks: variables whose values evolve continuously in states, all at the same speed, and can be tested or reset
  - Software support: Uppaal www.uppaal.org

Example of timed automaton: light switch

- If the button is pushed twice then, depending on time, the light may either get more intense or switch off



- Conditions and resets on clock x describe the time constraints
- State of the TTS = state of the TA + clock values

Clock conditions

- Boolean conditions that depend on clock values
- Two types of conditions:
  - Timed guards: associated to a transition, must be true for the transition to be fireable
  - Timed invariant: associated to a state, must be true whenever the system is in that state
- Syntax :

  Ψ ::= x **op** c | x - x' **op** c | $\Psi \wedge \Psi$ | $\Psi \vee \Psi$ | $\neg\Psi$

  where x, x' are clocks

  c is an integer constant (time units TU)

  op $\in \{ <, \leq, >, \geq, = \}$

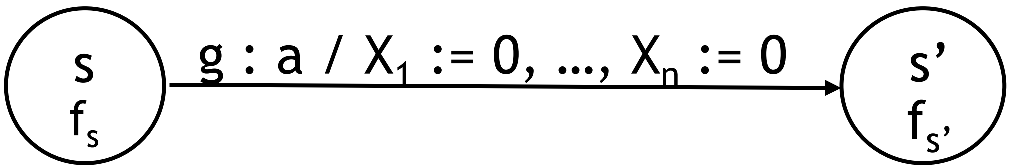
Timed automaton: formal definition

A TA is a 6-tuple (S, X, A, T, Inv, s_0) such that:

- S is a finite set of control states, s_0 initial state
- X is a finite set of clocks
- A is a finite set of actions
- $T \subseteq S \times A \times \Psi \times 2^X \times S$ is a finite set of transitions of the form (s, a, g, r, s') such that:
  - (s, a, s') is the same as in an LTS
  - $g \in \Psi$ is a timed guard such that vars (g) $\subseteq X$
  - $r \subseteq X$ is a set of clocks to be reset after the transition has been fired
- $Inv : S \to \Psi$ is a mapping that associates to each state s an invariant f_s such that vars (f_s) $\subseteq X$

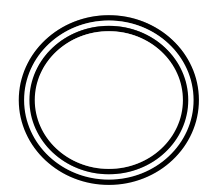
Graphical representation

- Transition (s, a, g, r, s') with f_s = Inv(s), $f_{s'}$ = Inv(s') and r = {X_1, …, X_n} is represented as follows:

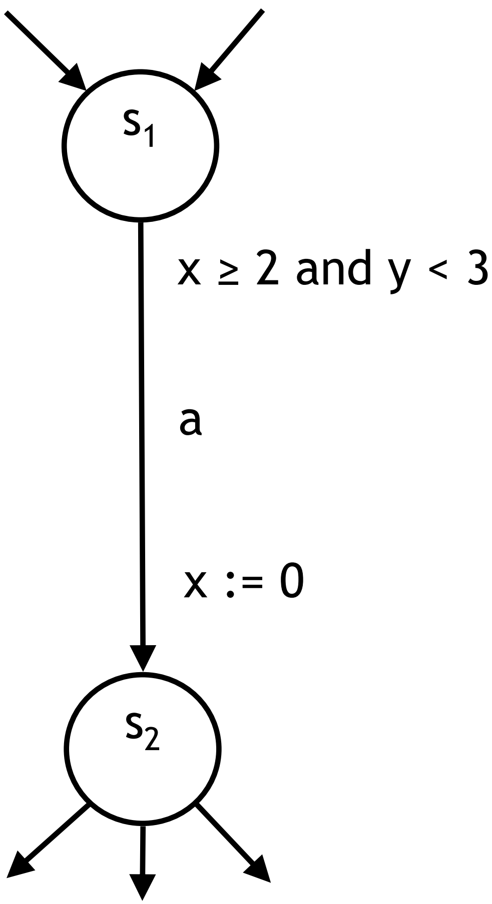


- a may be absent (transition without label)
- If f_s, $f_{s'}$ or g are absent: condition always true
- / and : are optional if clear from context

- initial state represented by a double line

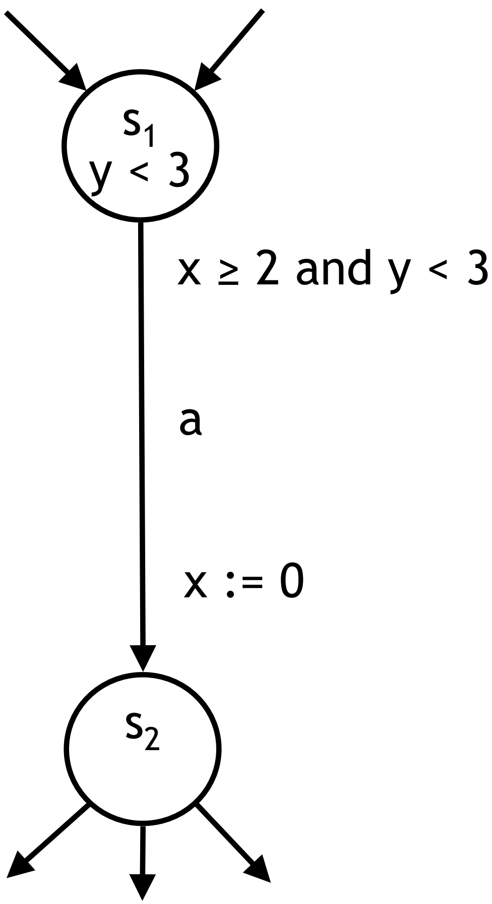


Example: Transition with guard



- No invariant: the system can stay indefinitely in s_1
- The transition can be fired only if the clocks x and y verify
  $x >= 2$ and $y < 3$
- After the transition, clock x is reset

Example: transition with guard and state with invariant

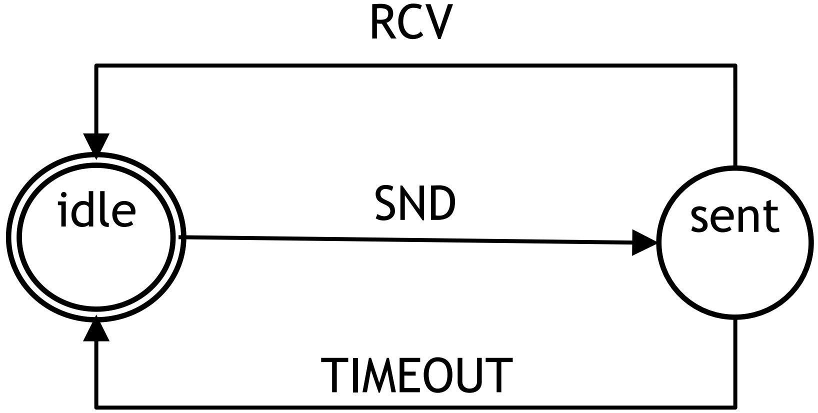


- The system can stay in s_1 only while $y < 3$
- The remainder is as in the previous slide

Exercise: communication medium with timeout

Complete the following CA to make a TA such that:

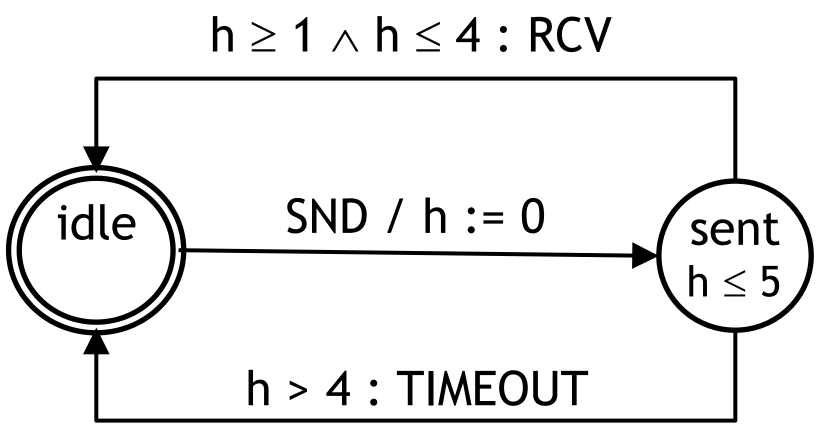
- Action RCV can occur between 1 and 4 TU after action SND
- If action RCV has not occurred after 4 TU, then action TIMEOUT occurs within 1 TU



Solution

Complete the following CA to make a TA such that:

- Action RCV can occur between 1 and 4 TU after action SND
- If action RCV has not occurred after 4 TU, then action TIMEOUT occurs within 1 TU

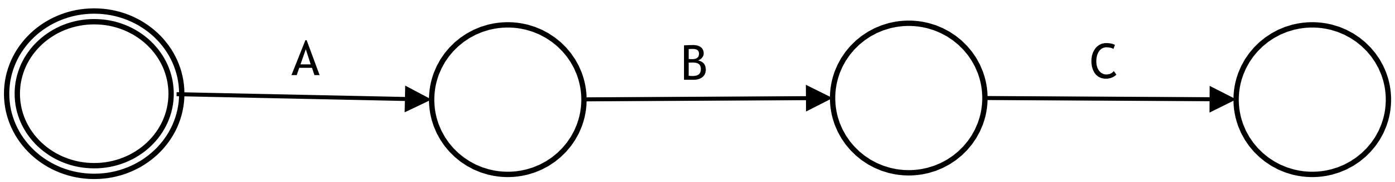


Exercise

Complete the following CA to make a TA such that:

- Action B occurs between 2 and 4 TU after action A
- Action C occurs at least 4 TU after action A and at least 1 TU after action B

Hint: use two clocks

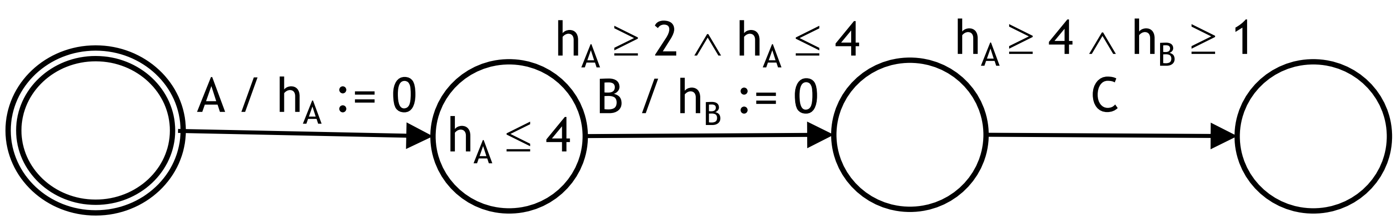


Solution

Complete the following CA to make a TA such that:

- Action B occurs between 2 and 4 TU after action A
- Action C occurs at least 4 TU after action A and at least 1 TU after action B

Hint: use two clocks



TTS semantics of TA (1/4)

- Clock valuation $v: X \rightarrow \Re^+$
  - Total function mapping each clock to a real value
  - The initial valuation v_0 is defined by $(\forall x \in X)\ v_0\ (x) = 0$
  - $v+t$ defines the valuation v' st $(\forall x \in X)\ v'\ (x) = v\ (x) + t$
  - reset (v, r) defines the valuation v' st $(\forall x \in r)\ v'\ (x) = 0$ and $(\forall x \in X \setminus r)\ v'\ (x) = v\ (x)$
- sat (v, Ψ): valuation v satisfies constraint Ψ

$$\left.\begin{array}{ll} \text{sat}(v, x\ \textbf{op}\ c) & \text{iff}\ \ v(x)\ \textbf{op}\ c \\ \text{sat}(v, x - x'\ \textbf{op}\ c) & \text{iff}\ \ v(x) - v(x')\ \textbf{op}\ c \end{array}\right\}\ \textbf{op} \in \{<, \leq, =\}$$

$$\begin{array}{ll} \text{sat}(v, \Psi \wedge \Psi') & \text{iff}\ \ \text{sat}(v, \Psi) \wedge \text{sat}(v, \Psi') \\ \text{sat}(v, \neg\Psi) & \text{iff}\ \neg\text{sat}(v, \Psi) \end{array}$$

TTS semantics of TA (2/4)

The semantics of the TA (S, X, A, T, Inv, s_0) is defined by the TTS (S', A, $\Re^+$, T', Θ, s_0'), where:

- Each TTS state (also called configuration) is made of a TA control state and a valuation: $S' = S \times (X \to \Re^+)$
- The TTS initial state (or configuration) is made of the TA initial state and the initial valuation: $s_0' = (s_0, v_0)$
- …

TTS semantics of TA (3/4)

The semantics of the TA (S, X, A, T, Inv, s_0) is defined by the TTS (S', A, $\Re^+$, T', Θ, s_0'), where:

- …
- Discrete transitions: T' is the set of transitions of the form (s, v) –a-> (s', v') such that:
  - There exists a TA transition: (s, a, g, r, s') $\in$ T
  - The guard g is satisfied: sat (v, g) is true
  - The clocks in r are reset: v' = reset (v, r)
  - The invariant is satisfied in the target state: sat (v', Inv (s')) is true
- …

TTS semantics of TA (4/4)

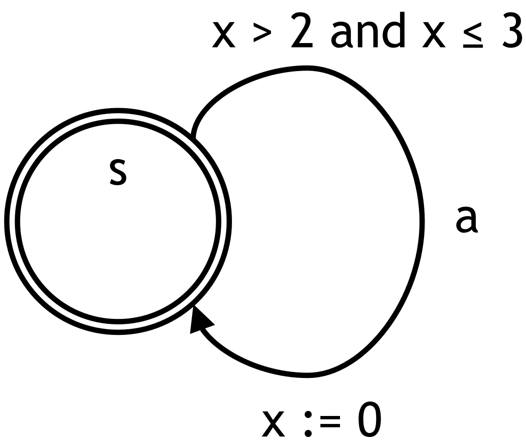
The semantics of the TA (S, X, A, T, Inv, s_0) is defined by the TTS $(S', A, \Re^+, T', \Theta, s_0')$, where:

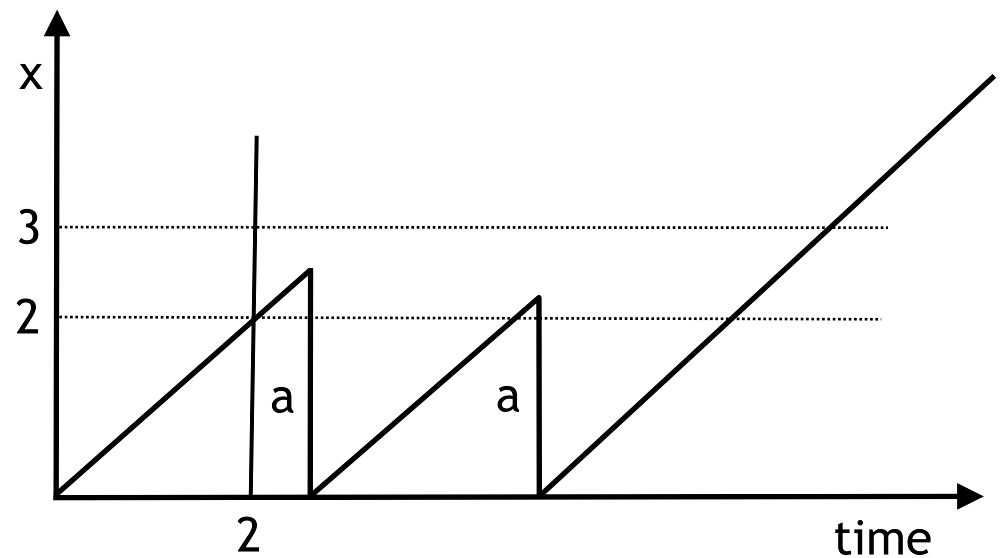
- …
- Timed transitions: Θ is the set of transitions of the form (s, v) –t-> (s, v+t) such that the invariant of s is satisfied in every intermediate configuration:

  $(\forall\ 0 \leq t' \leq t)$ sat (v+t', Inv (s))

This TTS satisfies the linear properties of time

Example without invariant

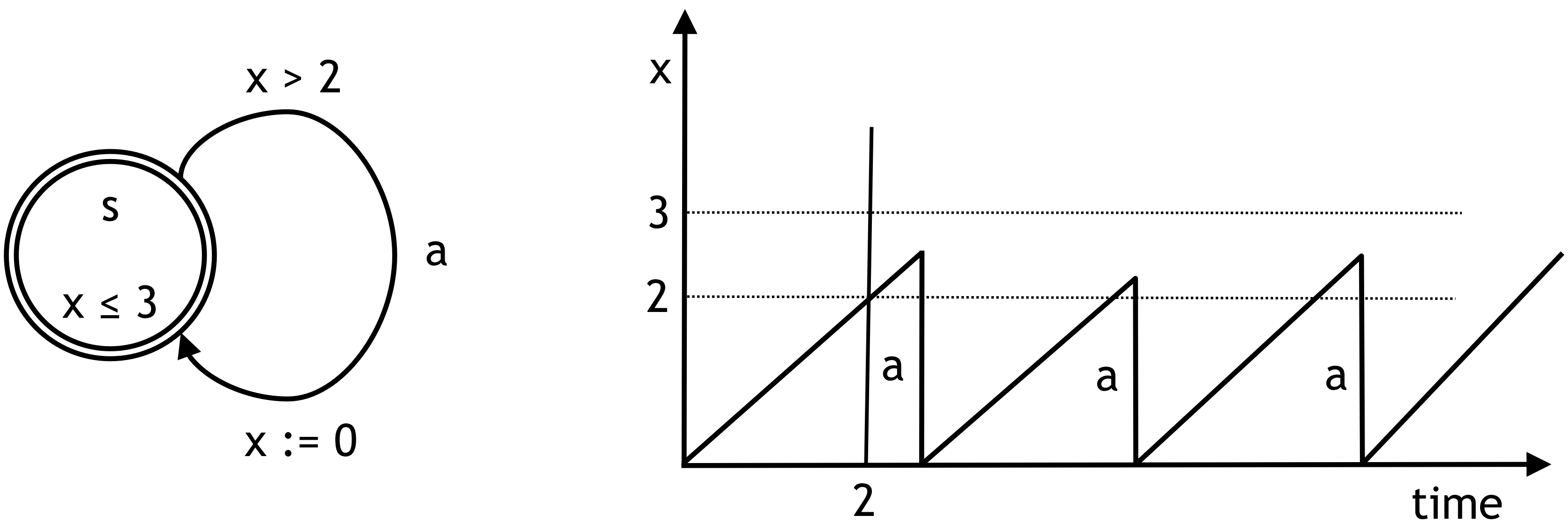




(s, x = 0) -2.5-> (s, x = 2.5) –a-> (s, x = 0)
-2.21-> (s, x = 2.21) –a-> (s, x = 0)
-10.678-> (s, x = 10.678) …

The discrete transition cannot be fired anymore

Example with invariant



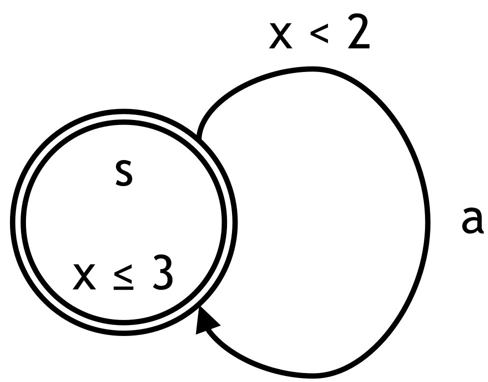
(s, x = 0) -2.5-> (s, x = 2.5) –a-> (s, x = 0)
-2.21-> (s, x = 2.21) –a-> (s, x = 0)
-2.52-> (s, x = 2.52) …

At most 3 TU elapse at each timed transition

Pathological case: timelock

- The use of invariants can lead to timelock

  Example:



  after 3 TU, time cannot elapse anymore because x is never reset

- This behaviour is the result of a modeling error
- It can be detected by verification

Critical path and Zeno effect

There are infinite paths where time does not elapse

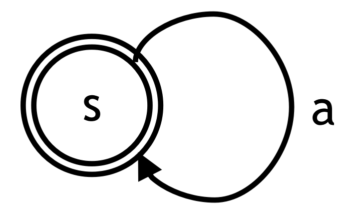
- Critical path: infinity of actions in zero time

  $(s, \emptyset) \xrightarrow{a} (s, \emptyset) \xrightarrow{a} (s, \emptyset) \xrightarrow{a} \ldots$

- Zeno path: infinity of actions in bounded time (even if every discrete sub-sequence is finite)

  $(s, \emptyset) \xrightarrow{a} (s, \emptyset) \xrightarrow{1/2} (s, \emptyset) \xrightarrow{a} (s, \emptyset) \xrightarrow{1/4} (s, \emptyset) \xrightarrow{a} \ldots$

- Many correct TTS have these types of paths: these paradoxes are accepted



- Good property to be shown: time progress

  ($\exists t \in \mathbb{R}^{>0}, n \in \mathbb{N}^{>0}$) st in every configuration, at least one path of length $\leq n$ exists on which $\geq t$ time units elapse

Parallel composition of TA (1/2)

- As CA, TA can also be composed in parallel
- Fundamental principle: time elapses at the same speed in all the parallel components
- Let $M_1 = (S_1, X_1, A_1, T_1, Inv_1, s_{01})$ and
  $M_2 = (S_2, X_2, A_2, T_2, Inv_2, s_{02})$
  two TA such that $X_1 \cap X_2 = \varnothing$
- The parallel composition of M_1 and M_2 synchronized by $L \subseteq A_1 \cap A_2$ is written $M_1 \otimes_L M_2$

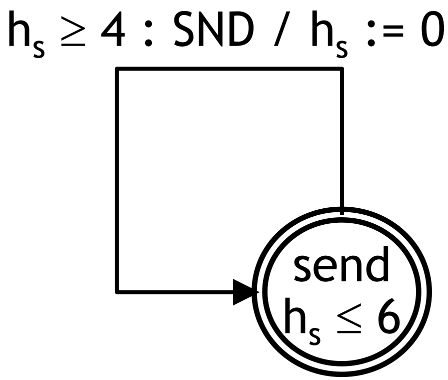
Parallel composition of TA (2/2)

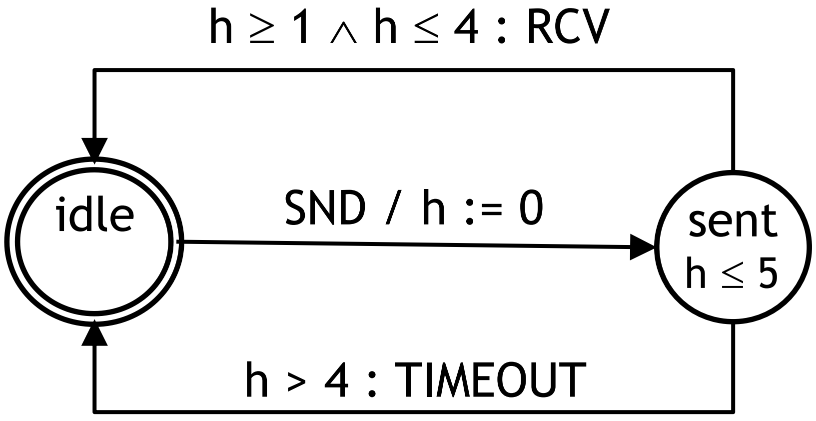
$M_1 \otimes_L M_2 = (S_1 \times S_2, X_1 \cup X_2, A_1 \cup A_2, T, Inv, (s_{01}, s_{02}))$

- Every parallel state invariant is the conjunction of the state invariants of the TA states:
  $(\forall (s_1, s_2) \in S_1 \times S_2)\ Inv(s_1, s_2) = Inv_1(s_1) \wedge Inv_2(s_2)$
- T is the set of transitions of the form $((s_1, s_2), a, g, r, (s_1', s_2'))$ such that, either:
  - $a \in L$, $(s_1, a, g_1, r_1, s_1') \in T_1$, $(s_2, a, g_2, r_2, q_2') \in T_2$, $g = g_1 \wedge g_2$, and $r = r_1 \cup r_2$ or
  - $a \in A_1 \setminus L$, $(s_1, a, g, r, s_1') \in T_1$, and $s_2' = s_2$ or
  - $a \in A_2 \setminus L$, $(s_2, a, g, r, s_2') \in T_2$, and $s_1' = s_1$

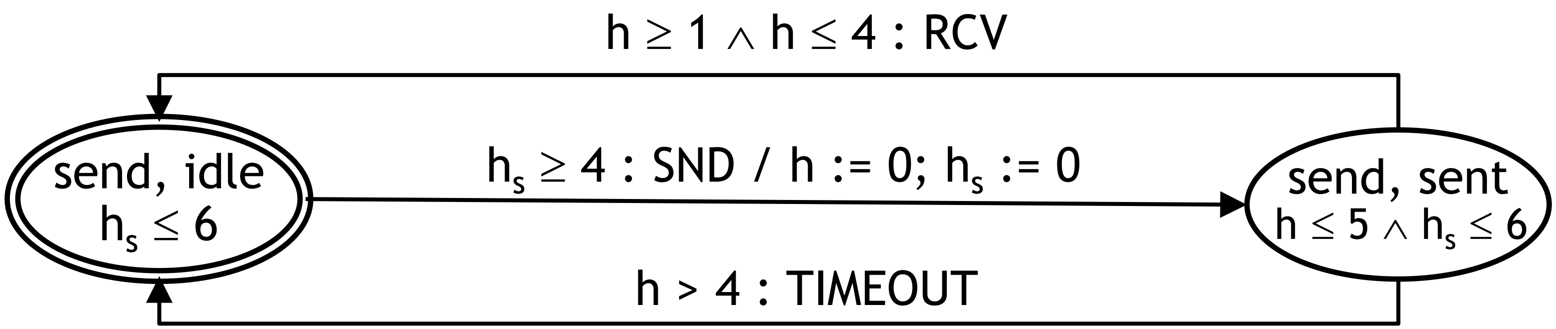
Exercise

Draw the parallel composition with synchronisation on SND of the following TA:

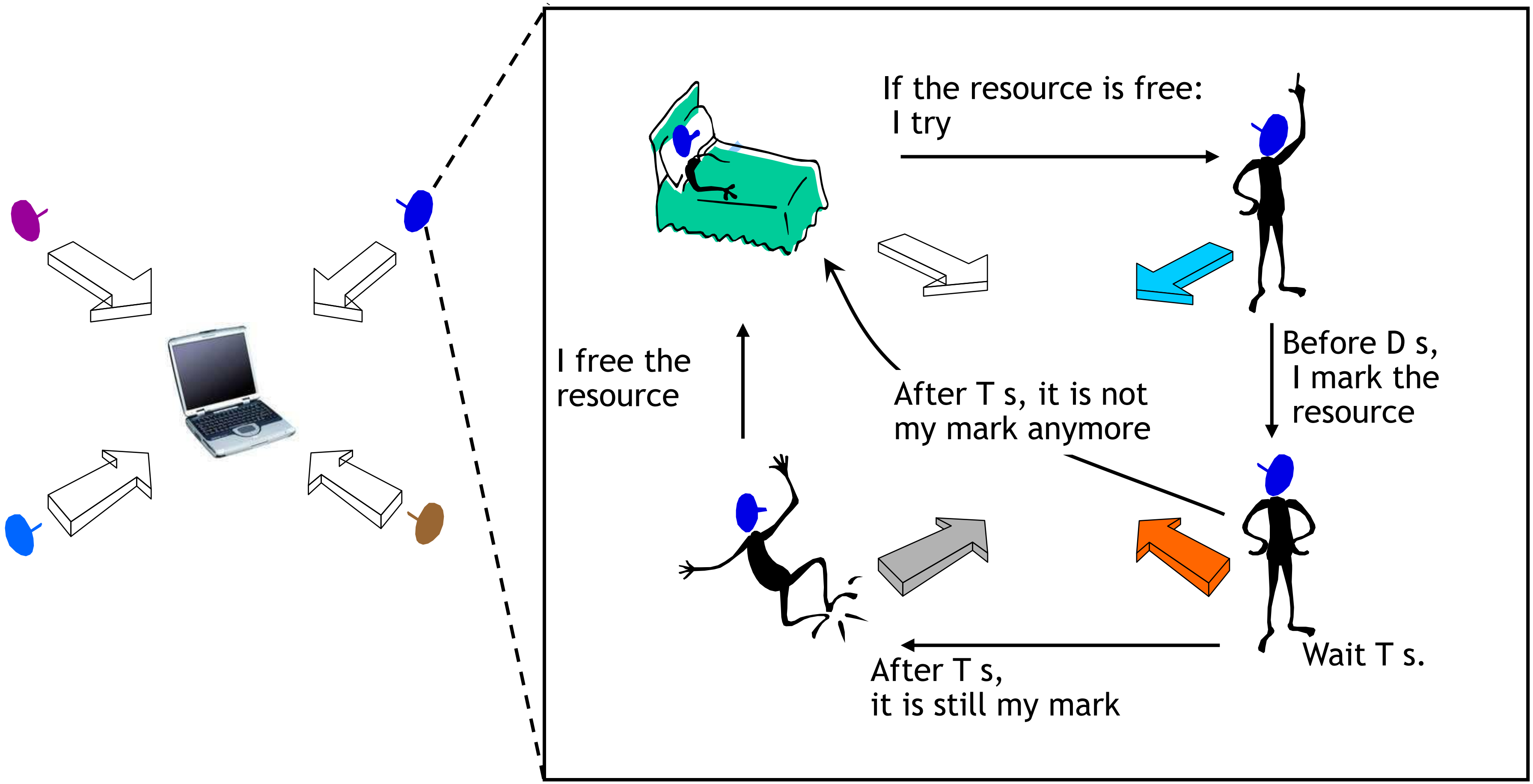




Solution


$h \geq 1 \wedge h \leq 4$: RCV
send, idle
$h_s \leq 6$
$h_s \geq 4$: SND / $h := 0; h_s := 0$
send, sent
$h \leq 5 \wedge h_s \leq 6$
$h > 4$: TIMEOUT

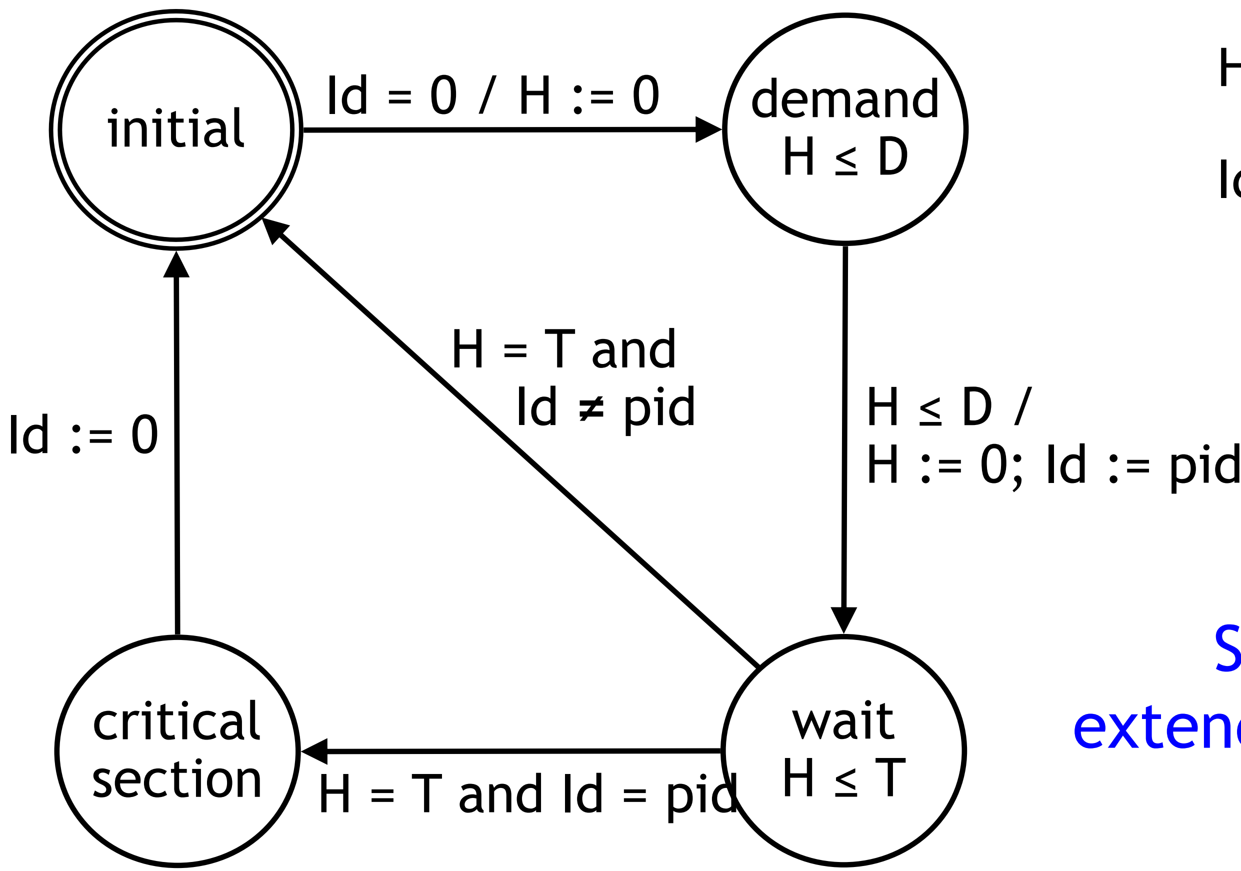
Example: the Fischer algorithm



Goal: guarantee the mutual exclusion of accesses to the resource

Fischer: Timed automata (1/2)

Automaton for process number « pid »

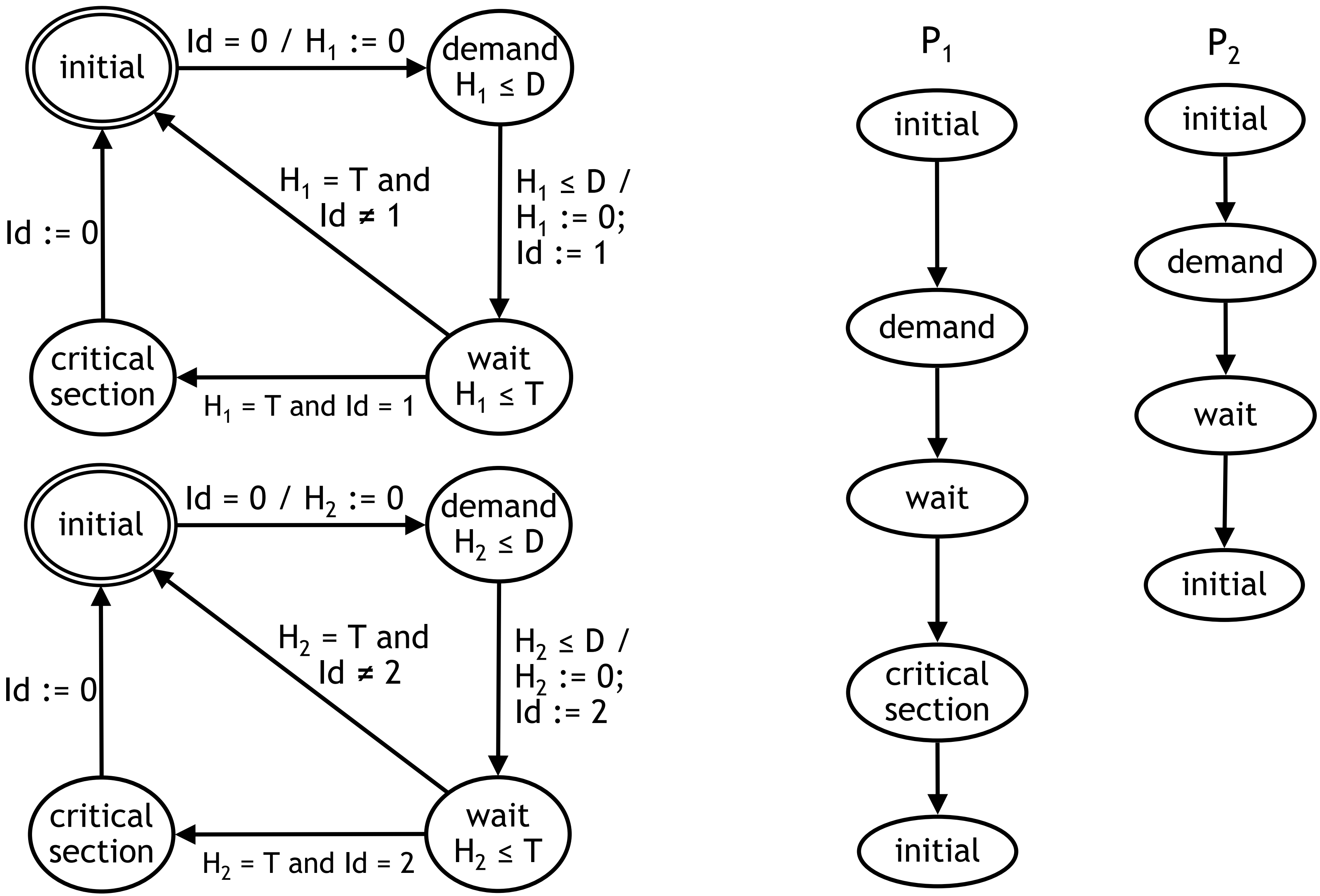


H: clock (local)

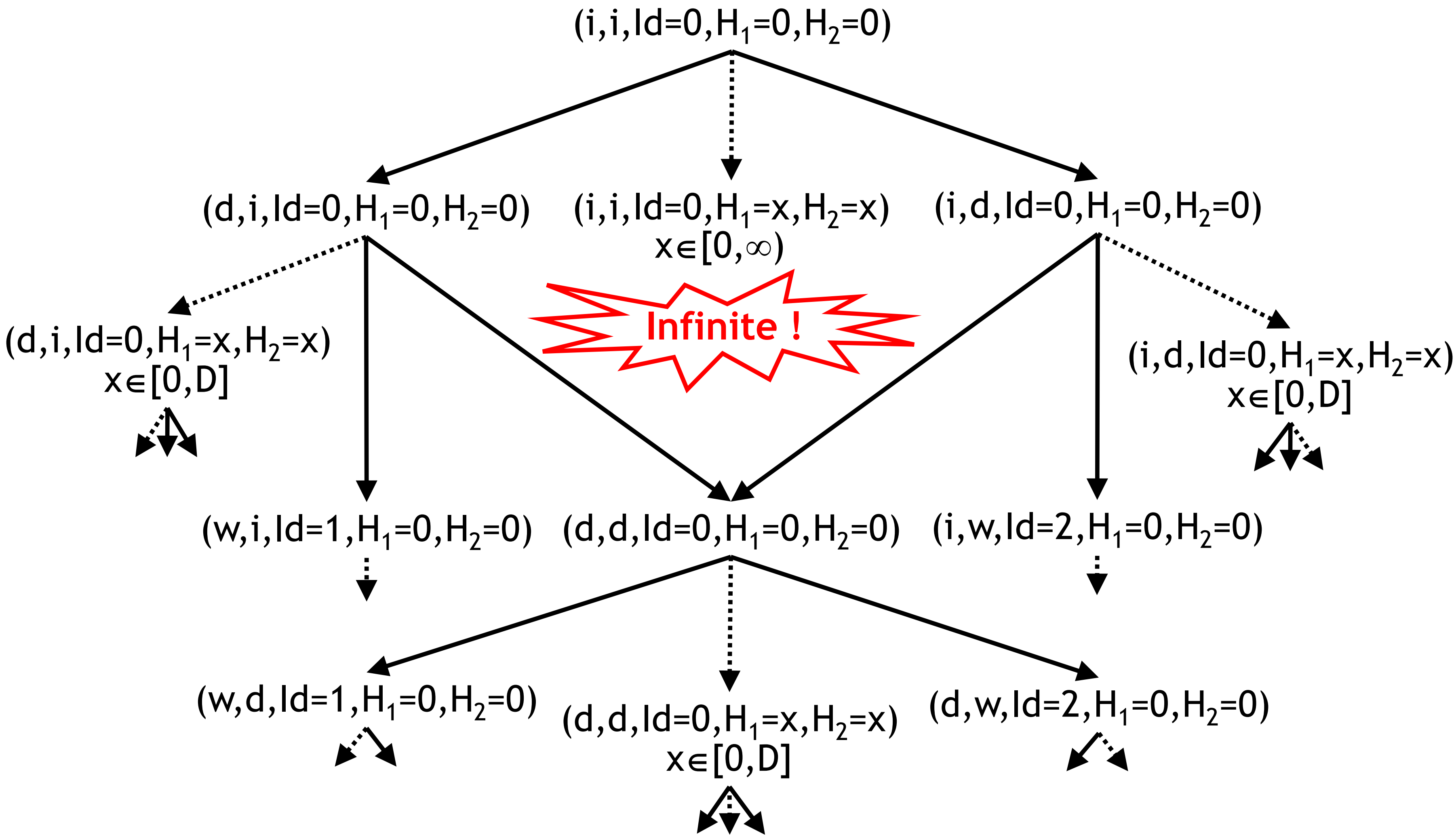
Id: shared variable
enum { 0, 1, 2 }

Symbolic TA =
extended with variables

Fischer: Timed automata (2/2)



Fischer: Building the state space



Time elapses at the same speed in all processes

Verification methods

- TA have infinitely many states (not enumerable)
- Necessity to work on finite abstractions
- Example: Zones
  - Symbolic representation of sets of TTS states by a control state, linear constraints on clocks, and variable values
  - Example : ((i, d), [H2 $\leq$ D, H2 – H1 $\leq$ 0, Id = 0])
  - Symbolic transitions:

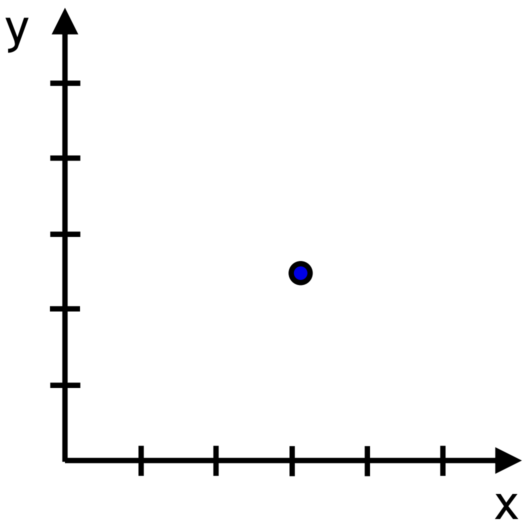
    ((i, d), [H2 $\leq$ D, H2 – H1 $\leq$ 0, Id = 0])

    $\rightarrow$ ((i, w), [H2 = 0, Id = 2])

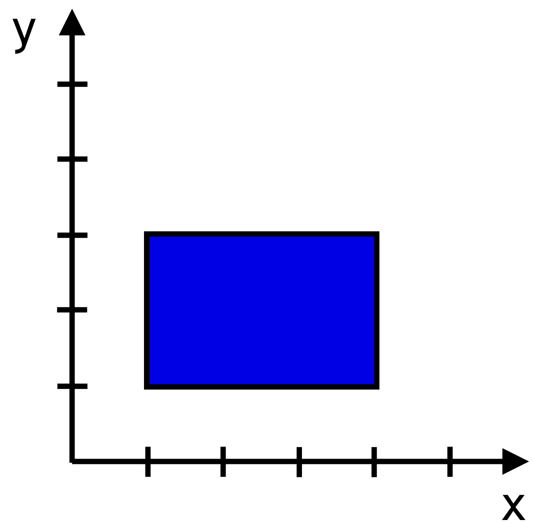
    $\rightarrow$ ((i, w), [H2 – H1 $\leq$ 0, Id = 2]) $\rightarrow$ etc.

Zones: from infinite to finite

configuration
(m, $x=3.12$, $y=2.5$)

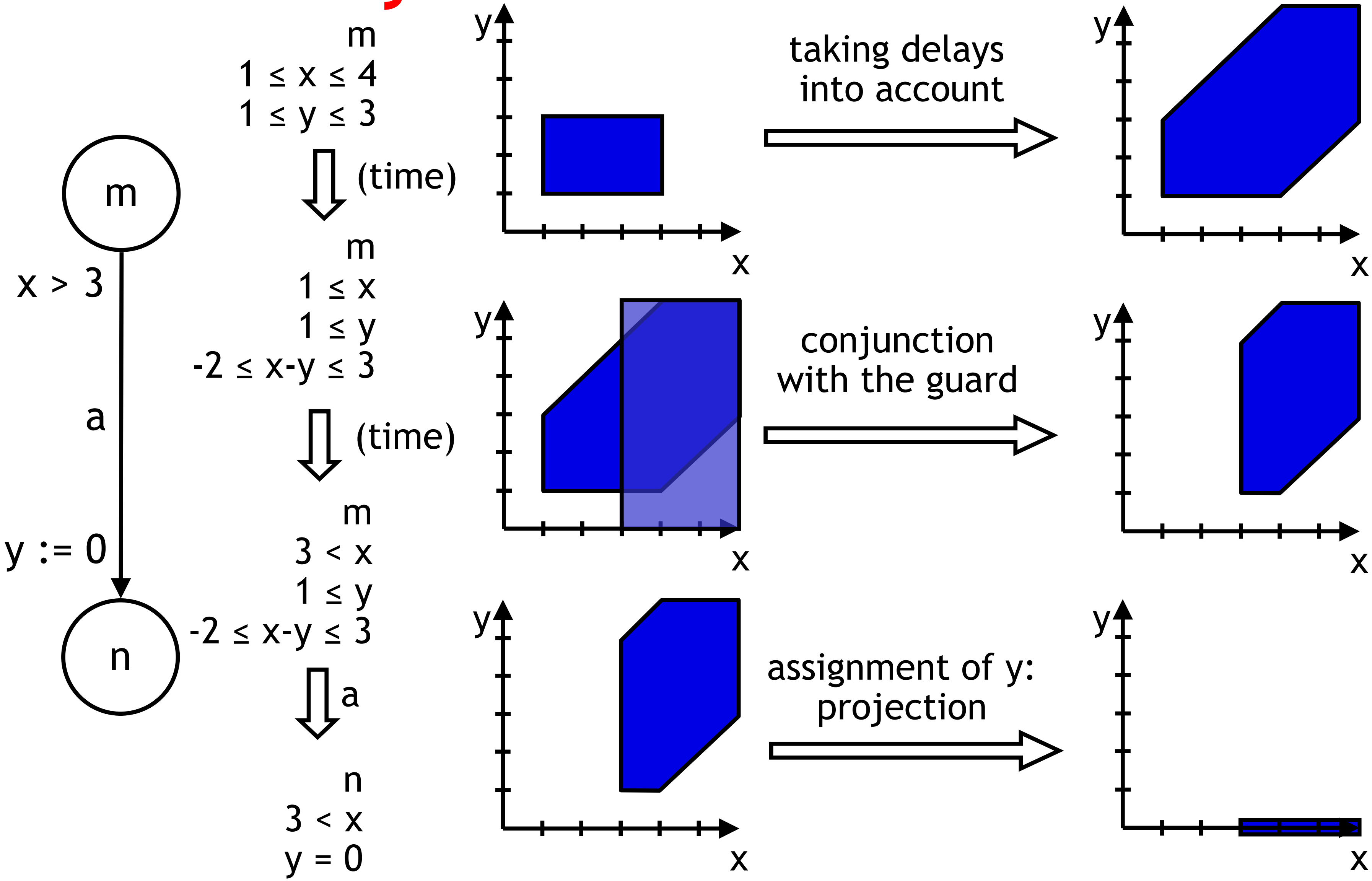


symbolic configuration
(m, $1 \leq x \leq 4$, $1 \leq y \leq 3$)

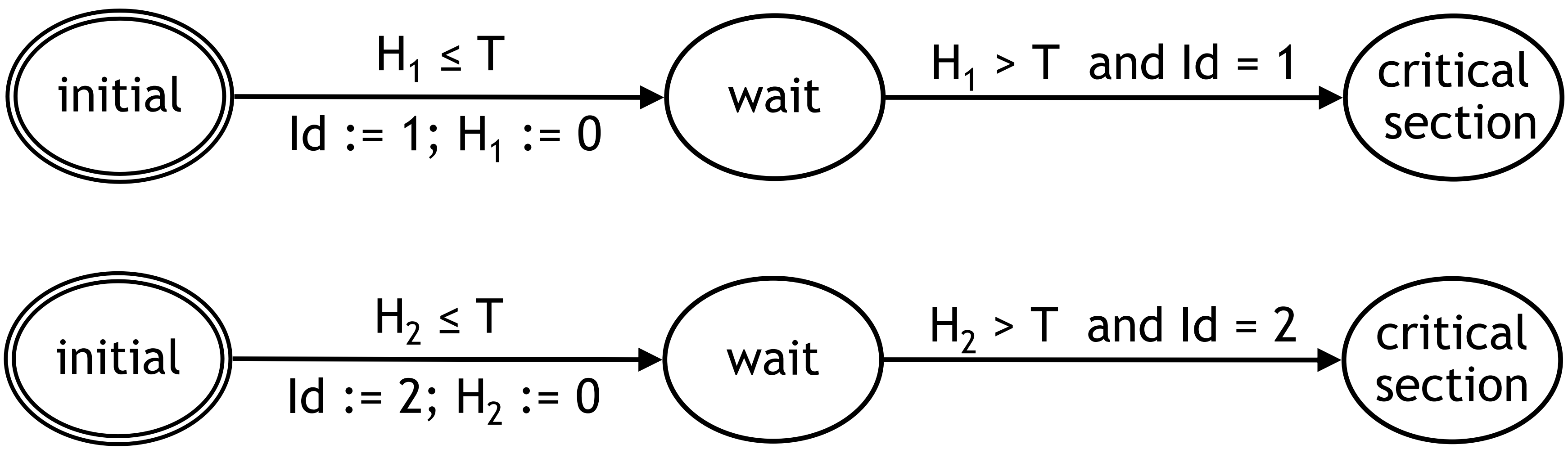


Zone: conjunction of constraints of the form
« x-y op c » and « x op c »

Symbolic transitions

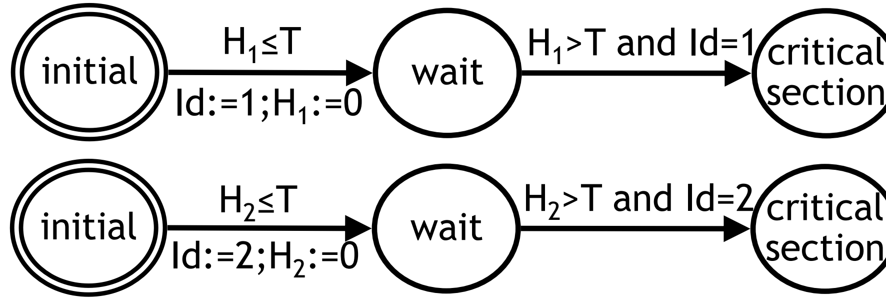


Example on a simplistic Fischer algorithm (without the demand state)

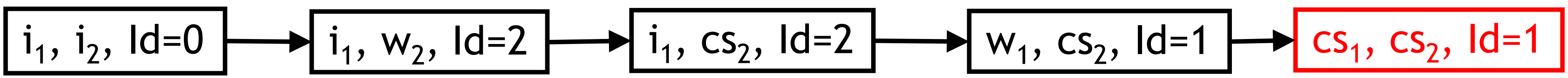


Initially: $Id = 0$

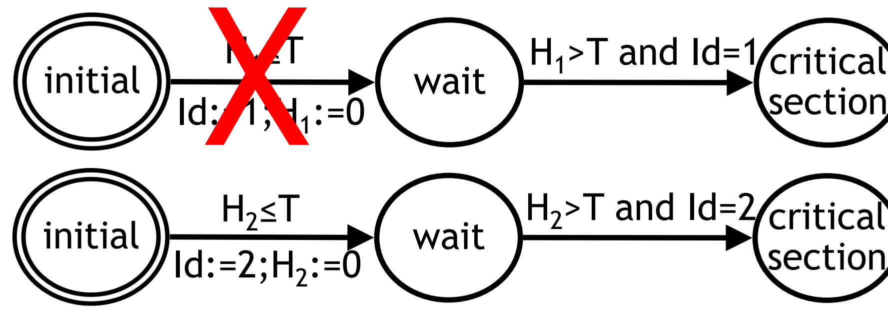
Simplistic Fischer



Without time constraints



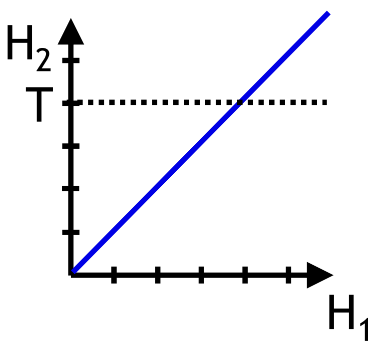
Simplistic Fischer: zones



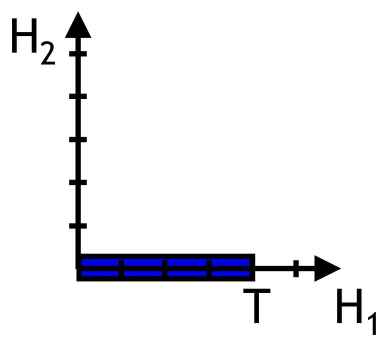
Taking into account time constraints

$\boxed{i_1, i_2, Id=0} \rightarrow \boxed{i_1, w_2, Id=2} \rightarrow \boxed{i_1, cs_2, Id=2} \cancel{\rightarrow} \boxed{w_1, cs_2, Id=1} \rightarrow \boxed{cs_1, cs_2, Id=1}$

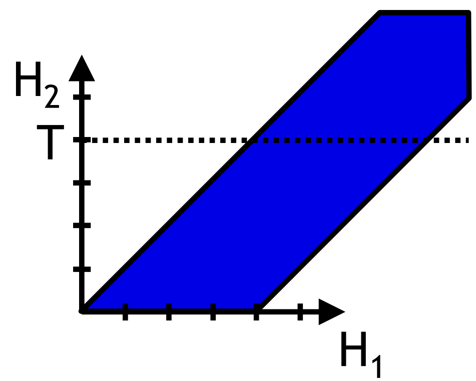
Mandatorily: $H_1 > T$ This transition requires $H_1 \le T$



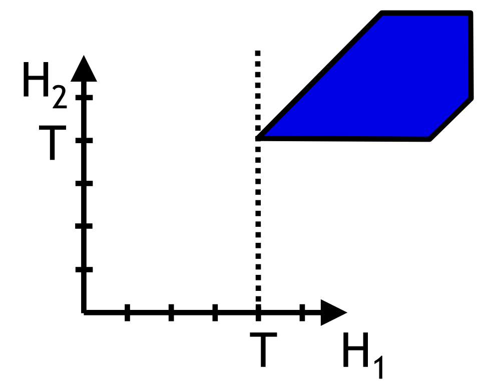
Time elapsing in $(i_1, i_2, Id=0)$



Conjunction with the guard and reset when entering $(i_1, w_2, Id=2)$



Time elapsing in $(i_1, w_2, Id=2)$



Conjunction with the guard and reset when entering $(i_1, cs_2, Id=2) \Rightarrow H_1 > T$

Other symbolic data structures

- Regions [Alur Dill]
- Numerical Decision Diagrams [Maler et al.]
- Clock Difference Diagrams [UPPAAL/CAV99]
- Difference Decision Diagrams [Møller, Lichtenberg]
- Polyedra [HyTech]
- Difference Bounded Matrices [UPPAAL]
- …

TD Uppaal

- Uppaal (http://www.uppaal.org) :
  - free for academics
  - Windows, Linux, and MacOS X versions
- subject of the next session
  in computer lab
- goal: manipulate timed automata