Verification, Performance Analysis and Controller Synthesis of Real Time Systems

using UPPAAL

Kim Guldstrand Larsen

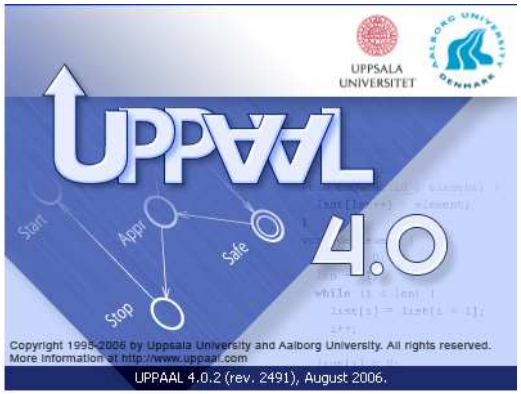
AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

UPPAAL Branches

- **Real Time Verification** — CLASSIC
- **Real Time Scheduling & Performance Evaluation** — CORA
- **Real Time Controller Synthesis** — TIGA
- **Real Time Testing** — TRON



Modeling Formalism
Theory
Alg.& Datastr.
Applications
Open Problems
DEMO's

UCb

Slides, Reading Material, Challenges,..

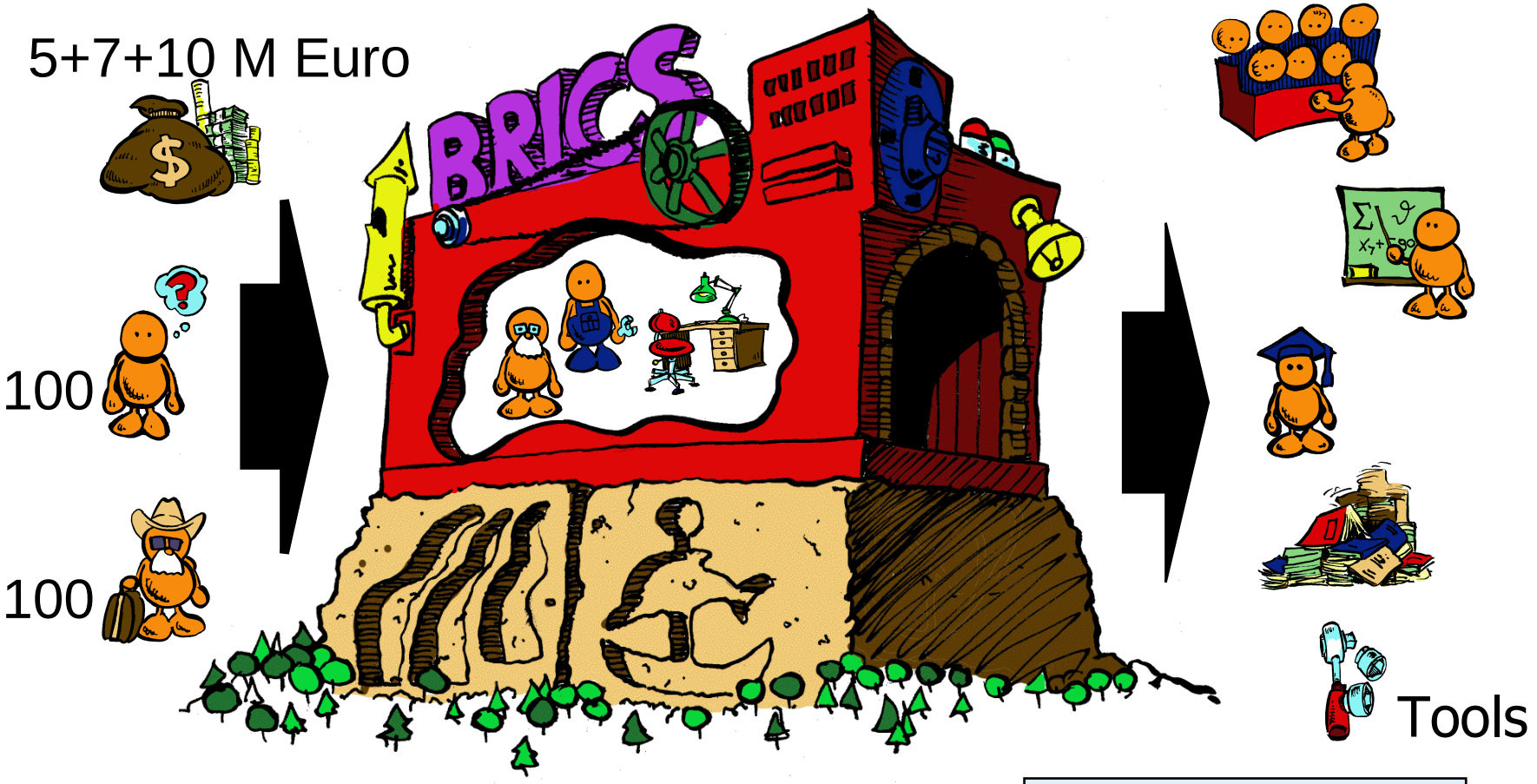
www.cs.aau.dk/~kgl/Marktoberdorf08

.../Material.html

UCb

BRICS Machine

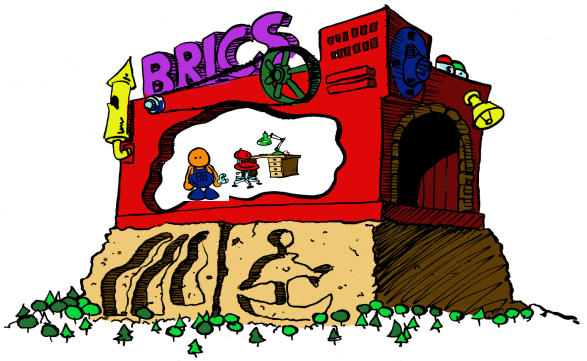
Basic Research in Computer Science, 1993-2006



Other revelvant projects
ARTIST, AMETIST

UCb

Tools and BRICS



1993-2006

Applications

visualSTATE **UPPAAL**

SPIN

PVS *HOL* *TLP* *ALF*

Logic

- Temporal Logic
- Modal Logic
- MSOL
- ⋮

Algorithmic

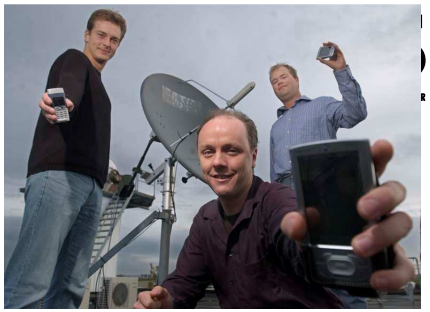
- (Timed) Automata Theory
- Graph Theory
- BDDs
- Polyhedra Manipulation
- ⋮

Semantics

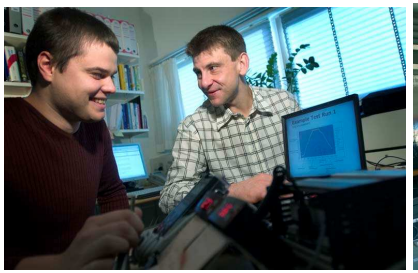
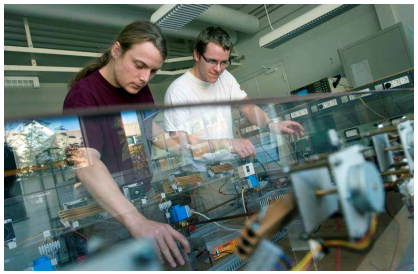
- Concurrency Theory
- Abstract Interpretation
- Compositionality
- Models for real-time & hybrid systems
- ⋮

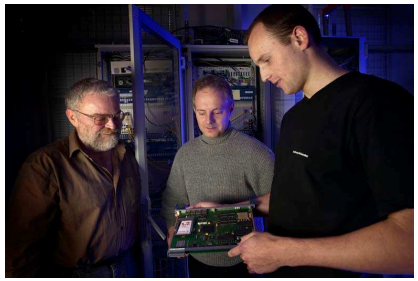
UCb

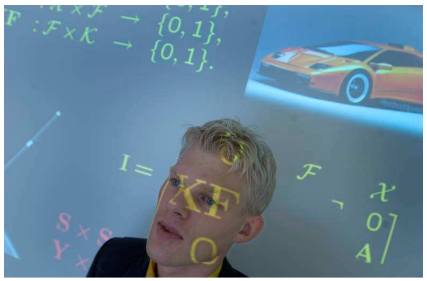
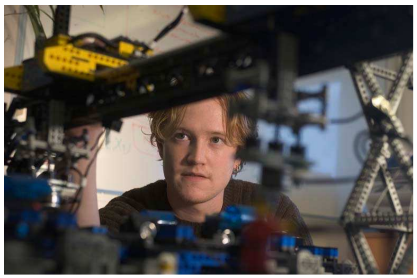
CISS *Center for Embedded Software Systems*



- National Competence Center 2002 sponsored by:
  - Minstry of Tech. & Res.
  - North Jutland
  - Aalborg City
  - Aalborg University



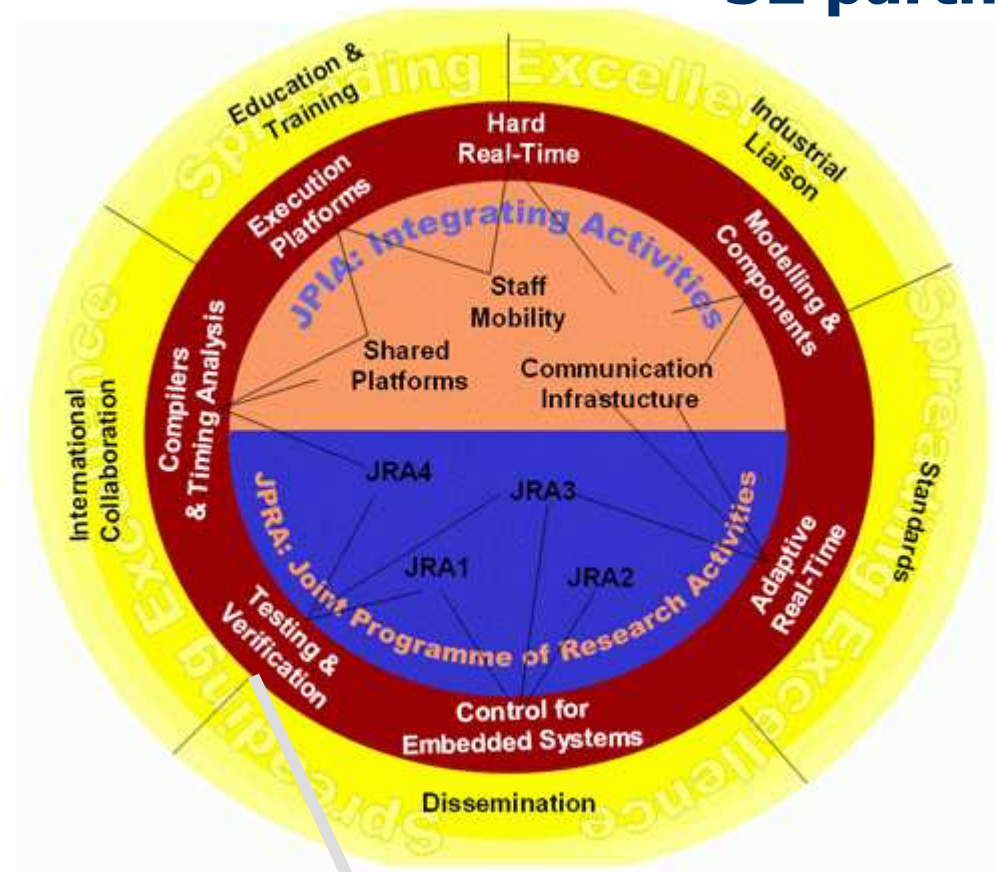




- **40** projects
- **20** CISS employees
- **25** CISS ass. Res.
- 20 industrial PhDs

UCb

European Network of Excellence

32 partners



ARTEMIS

Joseph Sifakis

Co-winner of Turing Award 2007

ARTIST Director

Testing & Verification
CISS coordinator

UCb

Why Verification and Testing

CSS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

- 30-40% of production time is currently spend on elaborate, ad-hoc testing:
  - Errors expensive and difficult to fix!
  - The potential of existing/improved testing methods and tools is enormous!
  - Time-to-market may be shortened considerable by verification and performance analyses of early designs!

UCb

Why Verification and Testing

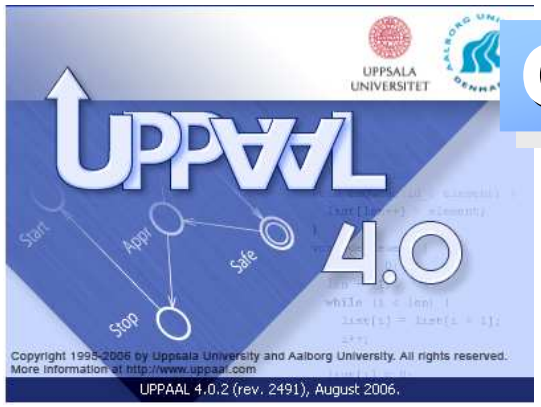
- **IMPORTANCE for EMBEDDED SYSTEMS**
  - Often safety critical
  - Often economical critical
  - Hard to patch

- **CHALLENGES for EMBEDDED SYSTEMS**
  - Correctness of embedded systems depend crucially on use of

    resources

    e.g. real-time, memory, bandwidth, energy.
  - Need for

    quantitative models



CLASSIC

Modelling, Specification, and Verification

using UPPAAL

Kim Guldstrand Larsen

AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Collaborators

@UPPsala

- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

@AALborg

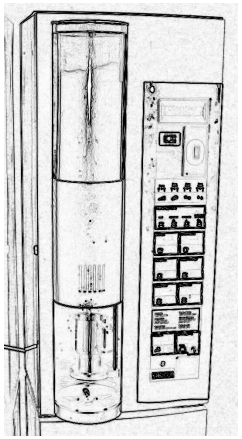
- Kim G Larsen
- Gerd Behrman
- Alexandre David
- Jacob I. Rasmussen
- Brian Nielsen
- Arne Skou
- Marius Mikucionis
- Thomas Chatain

@Elsewhere

- Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffioen, Ansgar Fehnker, Frits Vandraager, Theo Ruys, Pedro D'Argenio, J-P Katoen, Jan Tretmans, Judi Romijn, Ed Brinksma, Martijn Hendriks, Klaus Havelund, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Patricia Bouyer, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson...

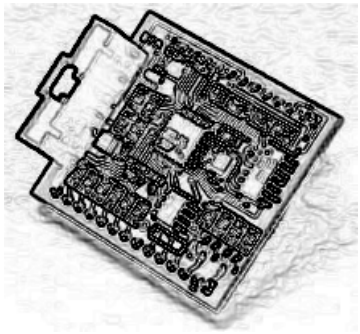
CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Real Time Systems



sensors

actuators



Plant
Continuous

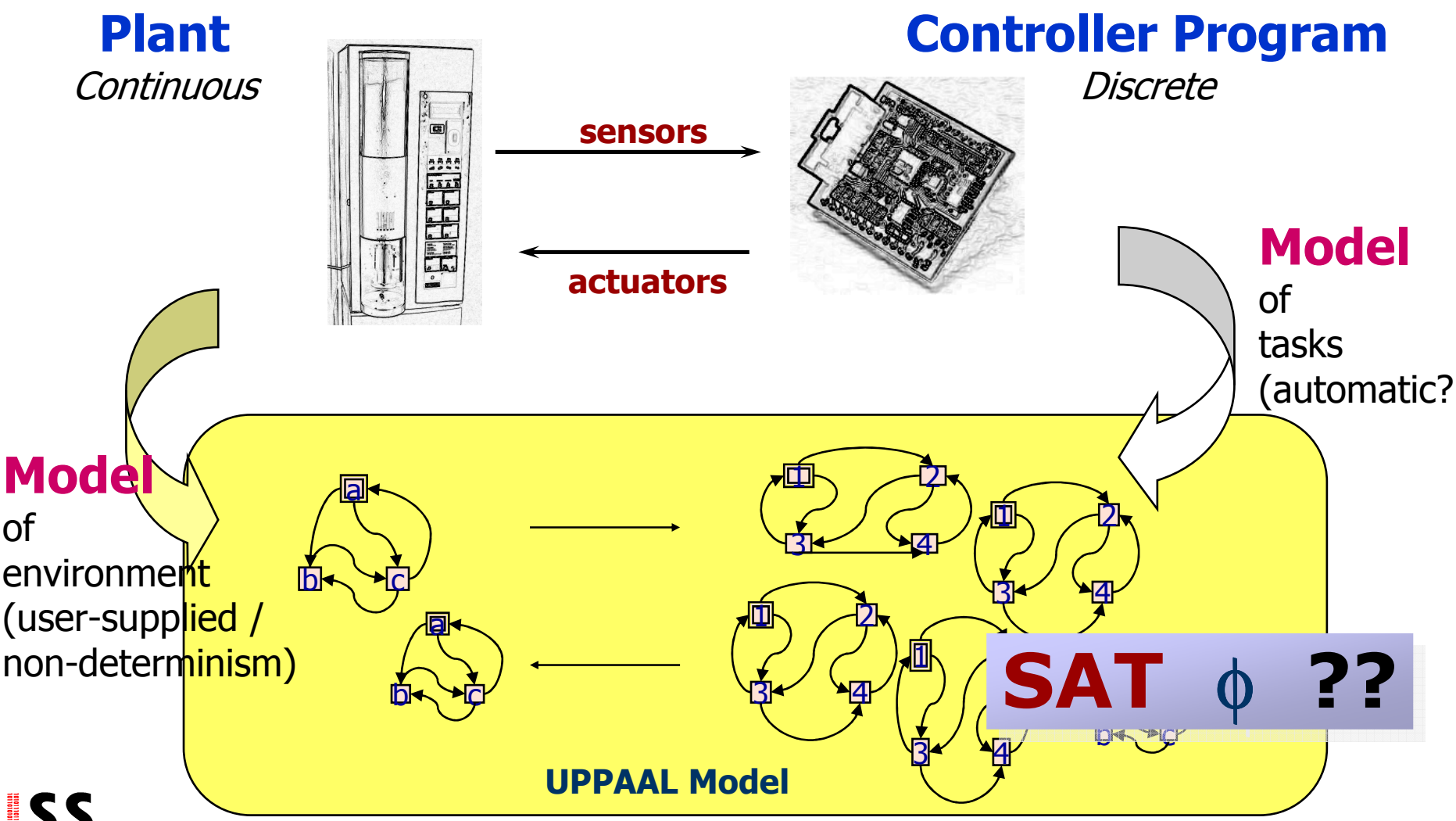
Controller Program
Discrete

Eg.: Realtime Protocols
Pump Control
Air Bags
Robots
Cruise Control
ABS
CD Players
Production Lines

Real Time System
A system where correctness not only depends on the logical order of events but also on their **timing**!!

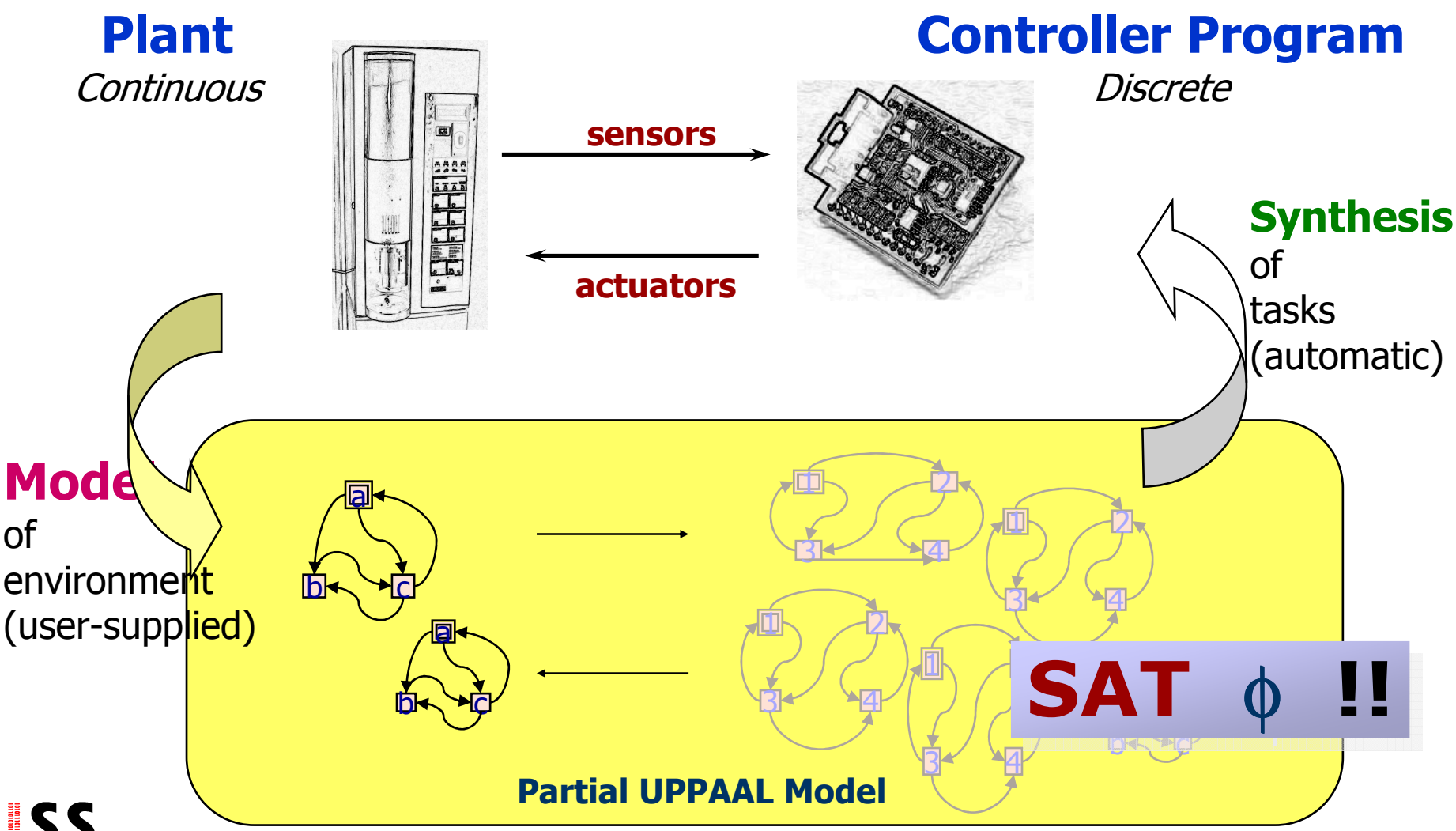
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Real Time Model Checking



CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Real Time Control Synthesis



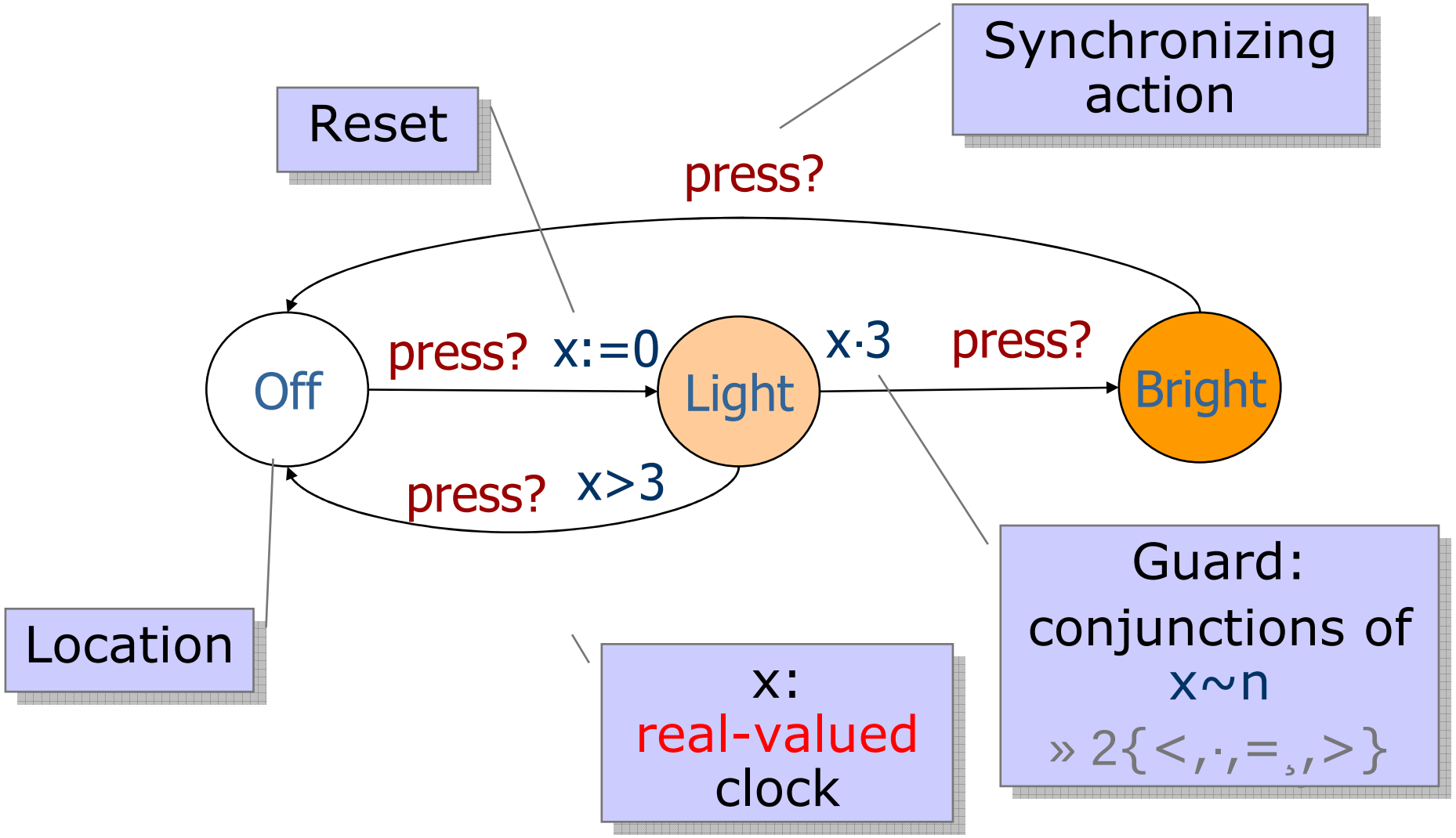
CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Timed Automata

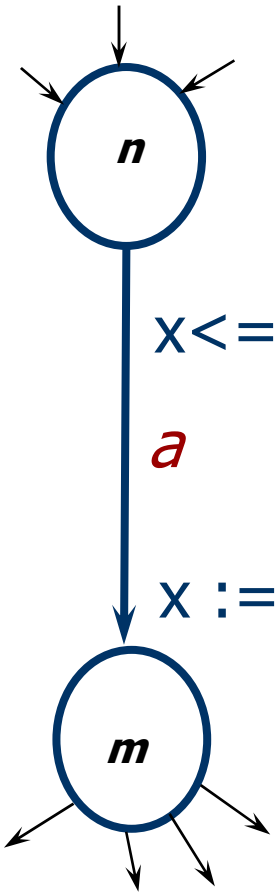
Alur & Dill 1989

BRICS
Basic Research
in Computer Science

Timed Automata



Timed Automata *semantics*



State

(***location*** , ***x***=v , ***y***=u) where v,u are in $\mathbf{R}$

Transitions

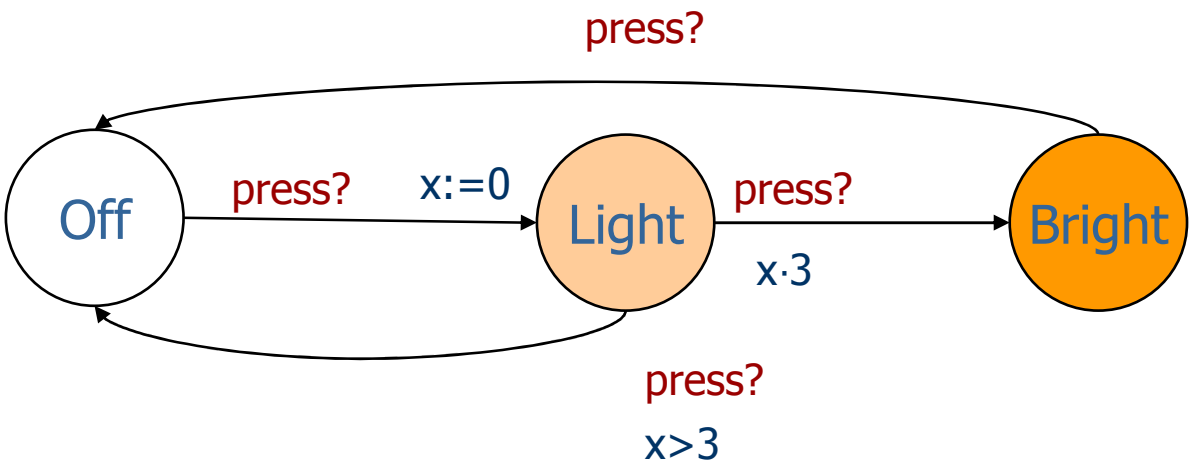
Discrete Trans

(***n*** , ***x***=2.4 , ***y***=3.1415) $\xrightarrow{a}$ (***m*** , ***x***=0 , ***y***=3.1415)

Delay Trans

(***n*** , ***x***=2.4 , ***y***=3.1415) $\xrightarrow{e(1.1)}$ (***n*** , ***x***=3.5 , ***y***=4.2415)

Timed Automata

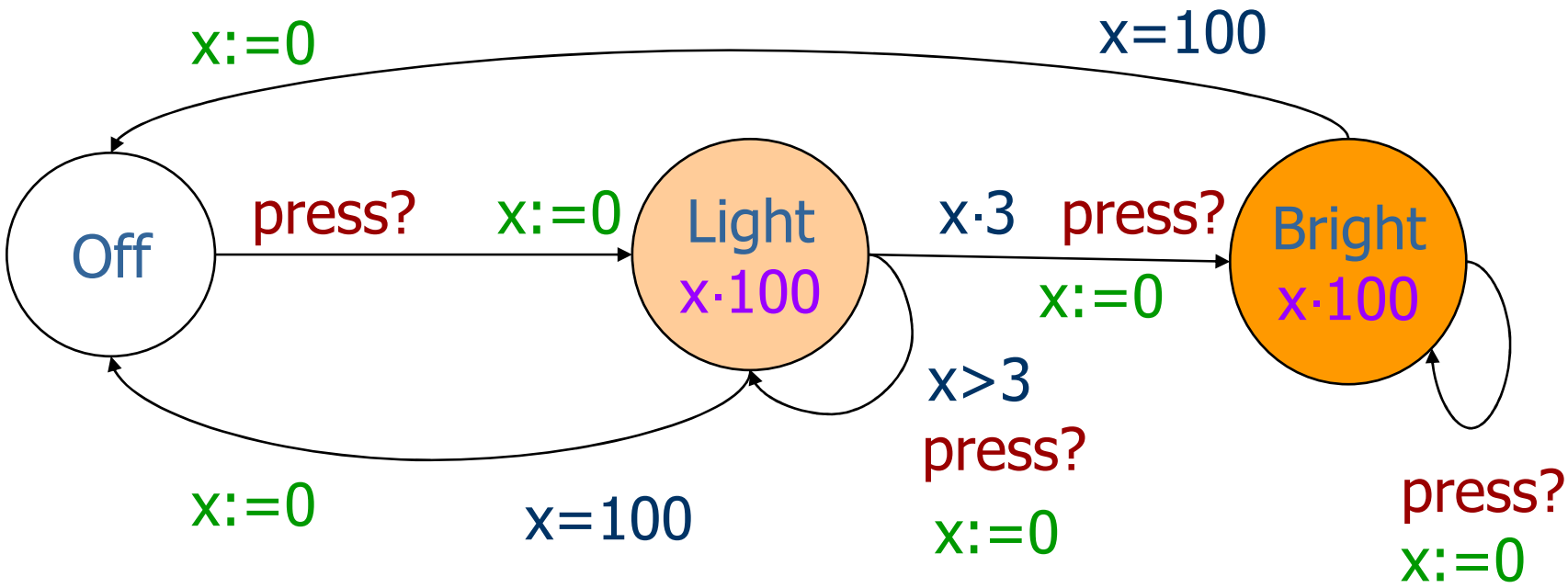


Transitions:

		(Off , x=0)
delay 4.32	→	(Off , x=4.32)
press?	→	(Light , x=0)
delay 2.51	→	(Light , x=2.51)
press?	→	(Bright , x=2.51)

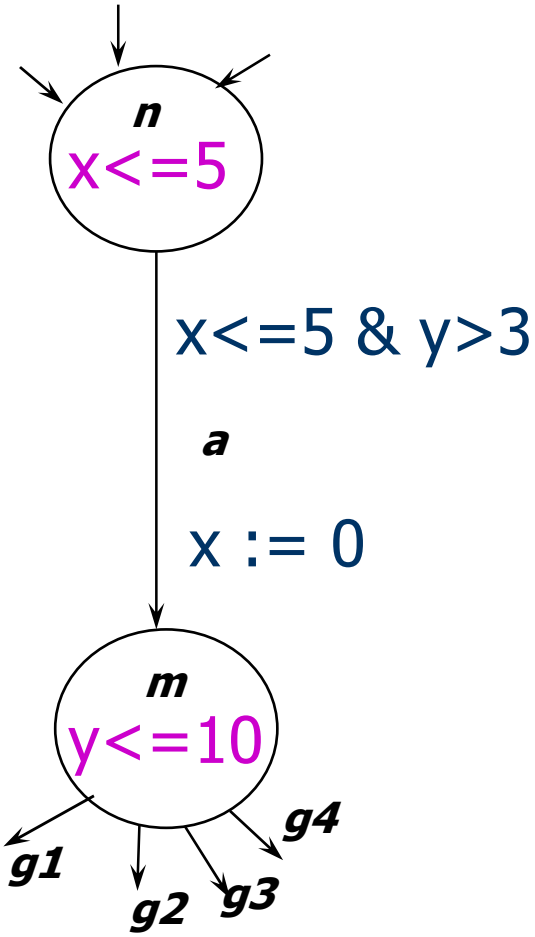
Intelligent Light Control

Using Invariants



Timed Automata *semantics*

Invariants



Transitions

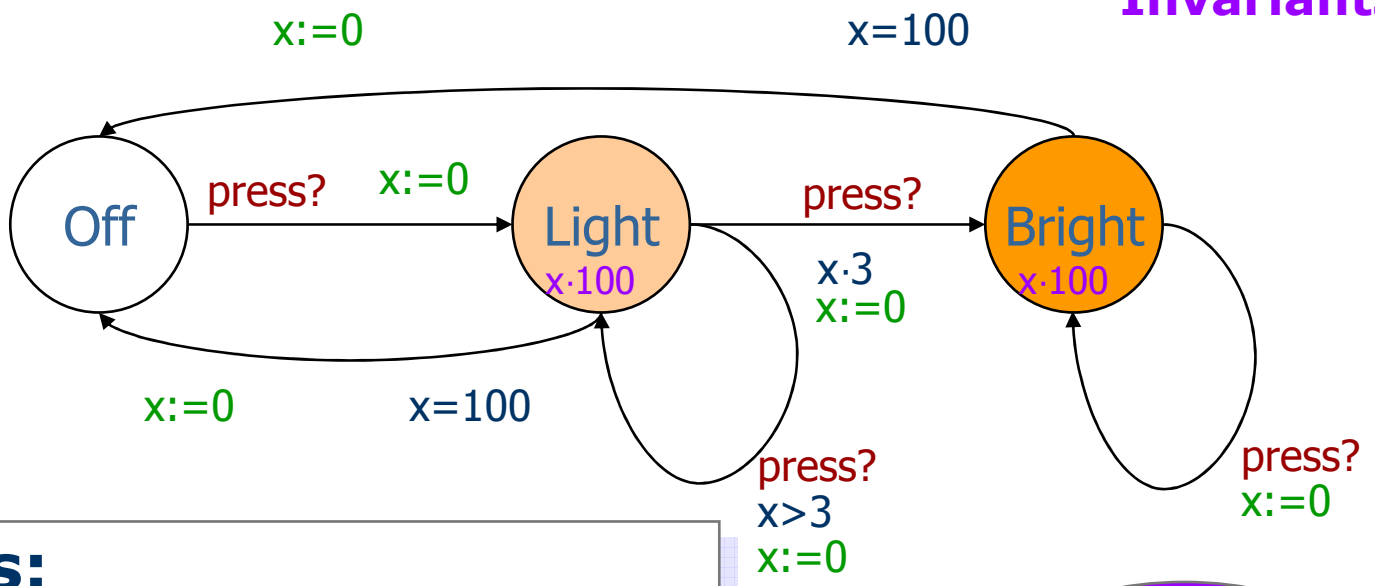
(***n*** , ***x***=2.4 , ***y***=3.1415) ~~$\xrightarrow{e(3.2)}$~~

(***n*** , ***x***=2.4 , ***y***=3.1415) $\xrightarrow{e(1.1)}$ (***n*** , ***x***=3.5 , ***y***=4.2415)

Invariants ensure progress !!

Intelligent Light Control

Invariants



Transitions:

	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 4.51	→ (Light , x=4.51)
press?	→ (Light , x=0)
delay 100	→ (Light , x=100)
τ	→ (Off , x=0)

Invariants ensures progress

Note:

(Light , x=0) delay 103 → X

UCb

Constraints

Definition

Let X be a set of clock variables. The set $\mathcal{B}(X)$ of *clock constraints* ϕ is given by the grammar:

$$\phi \ ::= \ x \leq c \mid c \leq x \mid x < c \mid c < x \mid \phi_1 \wedge \phi_2$$

where $c \in \mathbb{N}$ (or $\mathbb{Q}$).

Clock Valuations and Notation

Definition

The set of *clock valuations*, $\mathbb{R}^C$ is the set of functions $C \longrightarrow \mathbb{R}_{\geq 0}$ ranged over by $u, v, w, \ldots$.

Notation

Let $u \in \mathbb{R}^C$, $r \subseteq C$, $d \in \mathbb{R}_{\geq 0}$, and $g \in \mathcal{B}(X)$ then:

- $u + d \in \mathbb{R}^C$ is defined by $(u + d)(x) = u(x) + d$ for any clock x
- $u[r] \in \mathbb{R}^C$ is defined by $u[r](x) = 0$ when $x \in r$ and $u[r](x) = u(x)$ for $x \notin r$.
- $u \models g$ denotes that g is satisfied by u.

Timed Automata Formally

Timed Automata

Definition

A timed automaton A over clocks C and actions Act is a tuple (L, l_0, E, I), where:

- L is a finite set of locations
- $l_0 \in L$ is the initial location
- $E \subseteq L \times \mathcal{B}(X) \times Act \times \mathcal{P}(C) \times L$ is the set of edges
- $I : L \longrightarrow \mathcal{B}(X)$ assigns to each location an invariant

Semantics

Definition

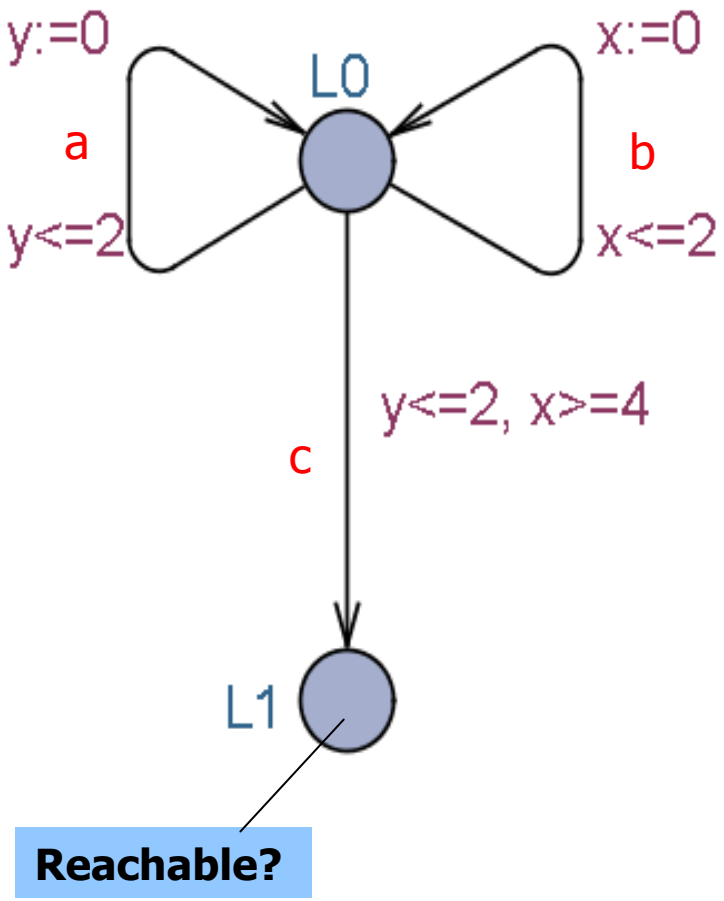
The semantics of a timed automaton A is a labelled transition system with state space $L \times \mathbb{R}^C$ with initial state $(l_0, u_0)^*$ and with the following transitions:

- $(l, u) \xrightarrow{\epsilon(d)} (l, u + d)$ iff $u \in I(l)$ and $u + d \in I(l)$,
- $(l, u) \xrightarrow{a} (l', u')$ iff there exists $(l, g, a, r, l') \in E$ such that
  - $u \models g$,
  - $u' = u[r]$, and
  - $u' \in I(l')$

*$u_0(x) = 0$ for all $x \in C$

Example

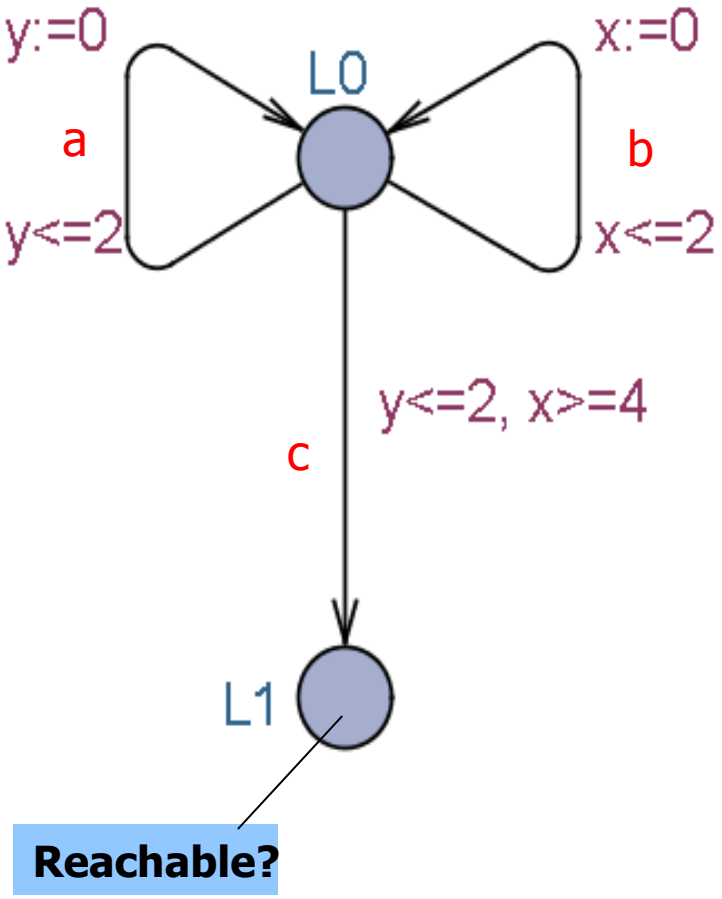
With two clocks

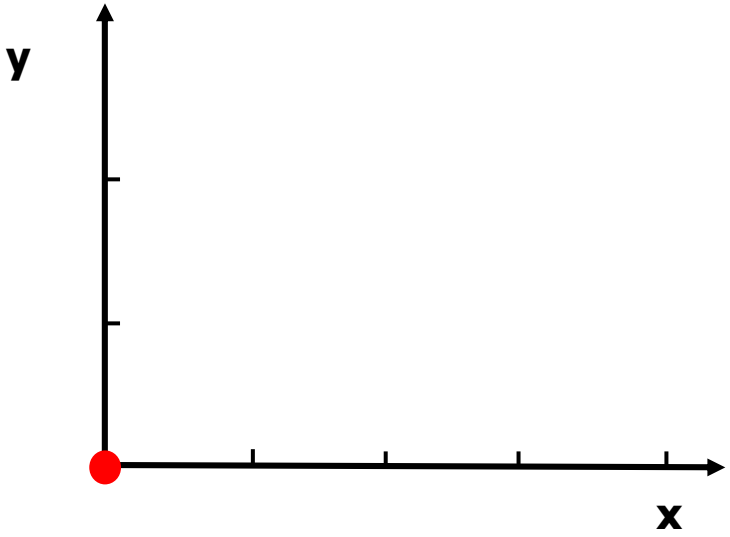


UCb

Example

With two clocks



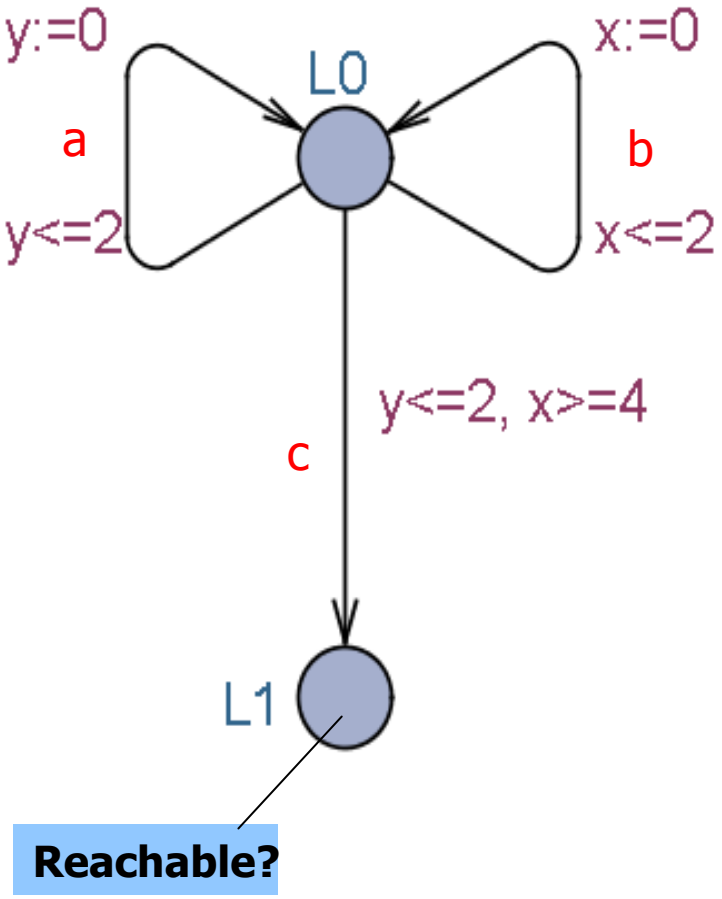


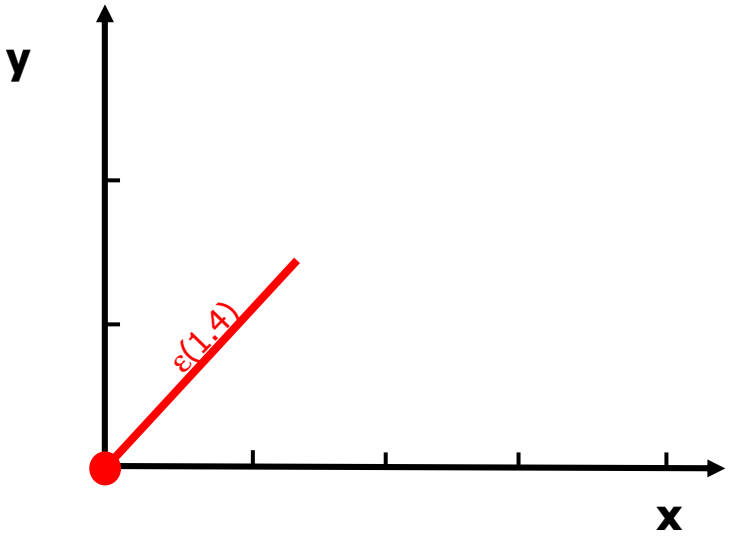
(L0,x=0,y=0)

UCb

Example

With two clocks





(L0,x=0,y=0)

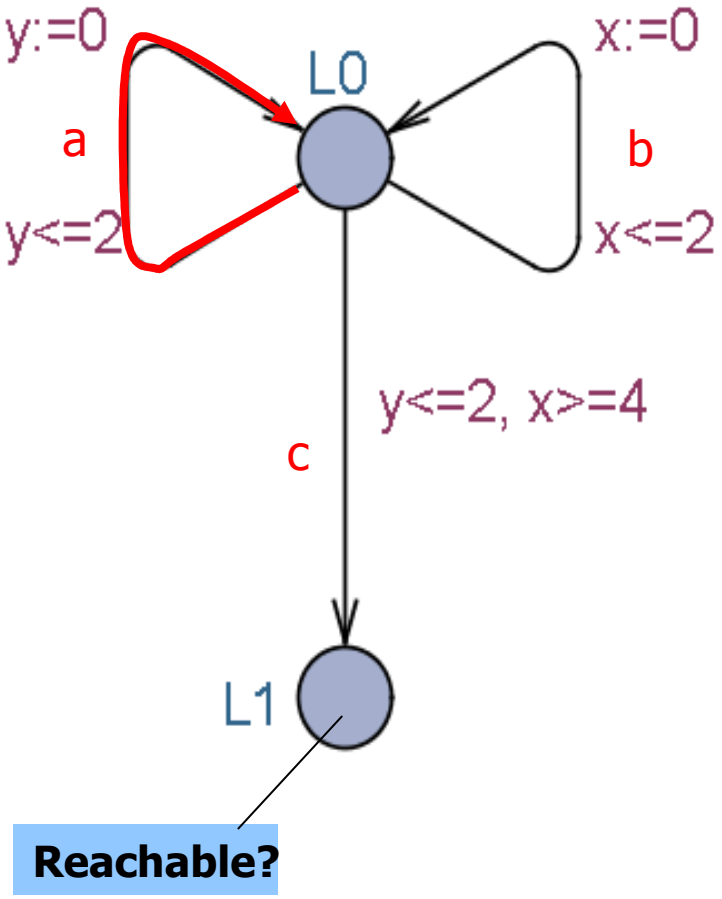
→ ε(1.4)

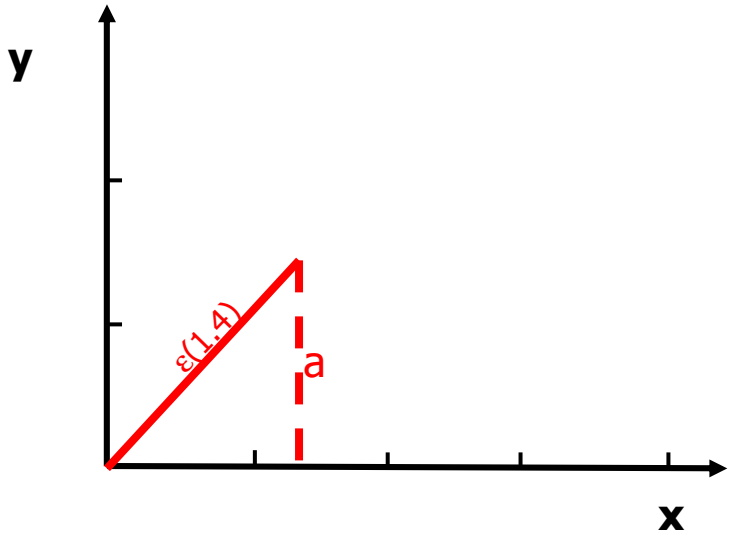
(L0,x=1.4,y=1.4)

UCb

Example

With two clocks



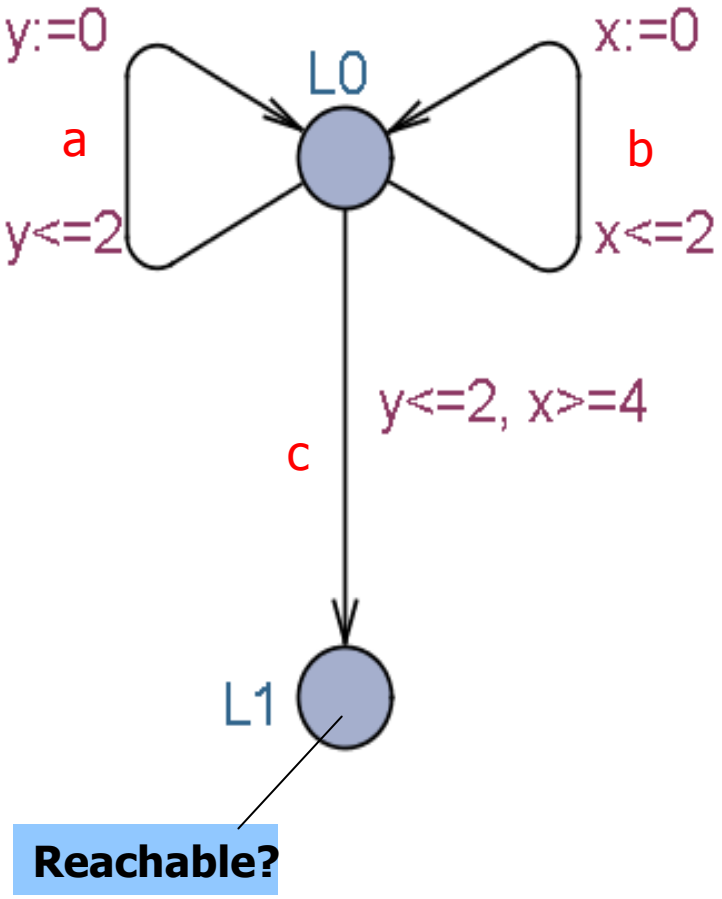


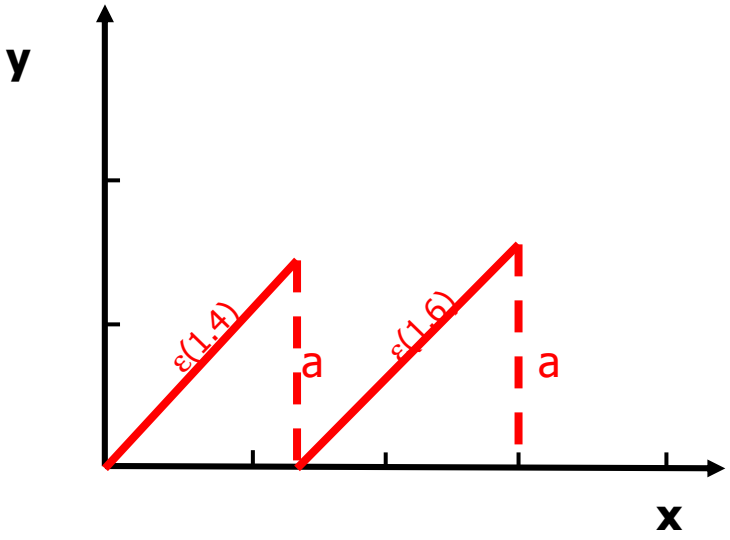
(L0,x=0,y=0)
→ε(1.4)
(L0,x=1.4,y=1.4)
→a
(L0,x=1.4,y=0)

UCb

Example

With two clocks

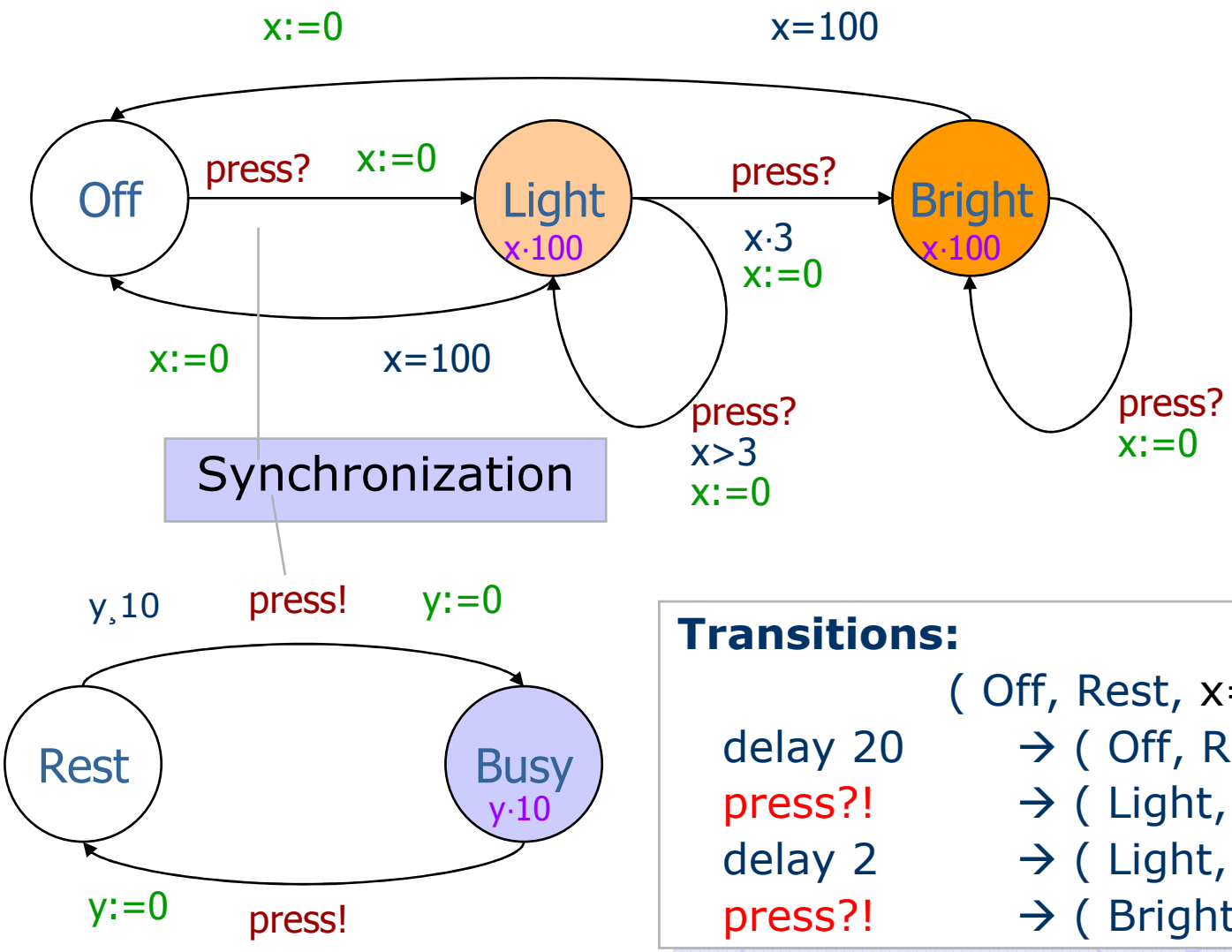




(L0,x=0,y=0)
→ε(1.4)
(L0,x=1.4,y=1.4)
→a
(L0,x=1.4,y=0)
→ε(1.6)
(L0,x=3.0,y=1.6)
→a
(L0,x=3.0,y=0)

UCb

Networks Light Controller & User



Transitions:

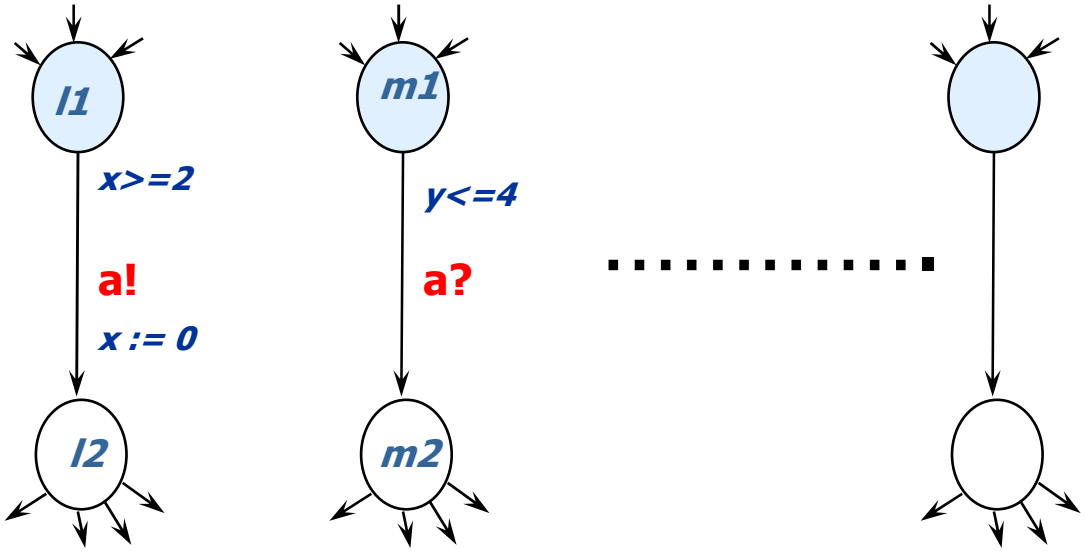
(Off, Rest, x=0, y=0)

delay 20	→ (Off, Rest, x=20, y=20)
press?!	→ (Light, Busy, x=0, y=0)
delay 2	→ (Light, Busy, x=2, y=2)
press?!	→ (Bright, Rest, x=0, y=0)

UCb

Networks of Timed Automata

(a'la CCS)



Two-way synchronization on *complementary* actions.

Closed Systems!

Example transitions

$(l1, m1, \ldots\ldots, x=2, y=3.5, \ldots\ldots) \xrightarrow{\tau} (l2, m2, \ldots\ldots, x=0, y=3.5, \ldots\ldots)$

0.2 ✗ $(l1, m1, \ldots\ldots, x=2.2, y=3.7, \ldots\ldots)$

If **a** URGENT CHANNEL

Network Semantics

$$T_1 \|_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \|_X s_0^2) \quad \text{where}$$

$$\frac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1' \|_X s_2} \qquad \frac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1 \|_X s_2'}$$

$$\frac{s_1 \xrightarrow{a!}_1 s_1' \qquad s_2 \xrightarrow{a?}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\tau} s_1' \|_X s_2'}$$

$$\frac{s_1 \xrightarrow{e(d)}_1 s_1' \qquad s_2 \xrightarrow{e(d)}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{e(d)} s_1' \|_X s_2'}$$

UCb

Network Semantics

(URGENT synchronization)

+ Urgent synchronization

$$T_1 \|_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \|_X s_0^2) \quad \text{where}$$

$$\frac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1' \|_X s_2} \qquad \frac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1 \|_X s_2'}$$

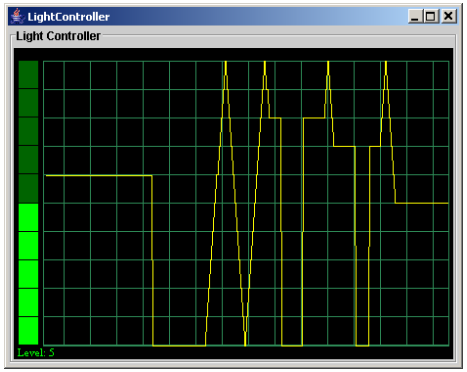
$$\frac{s_1 \xrightarrow{a!}_1 s_1' \qquad s_2 \xrightarrow{a?}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\tau} s_1' \|_X s_2'}$$

$\forall d' < d, \forall u \in \text{UAct}:$

$\neg (s_1 \xrightarrow{e(d')} \xrightarrow{u?} \wedge\ s_2 \xrightarrow{e(d')} \xrightarrow{u!})$

$$\frac{s_1 \xrightarrow{e(d)}_1 s_1' \qquad s_2 \xrightarrow{e(d)}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{e(d)} s_1' \|_X s_2'}$$

UCb



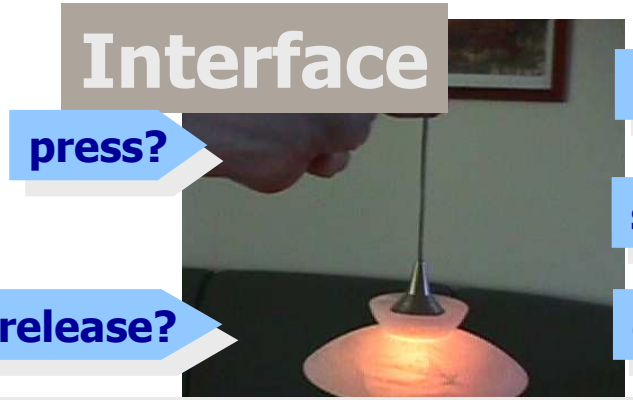
Light Control Interface

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Light Control Interface

press? d release? → touch! 0.5·d·1
press? 1 → starthold!
press? d release? → endhold! d >1



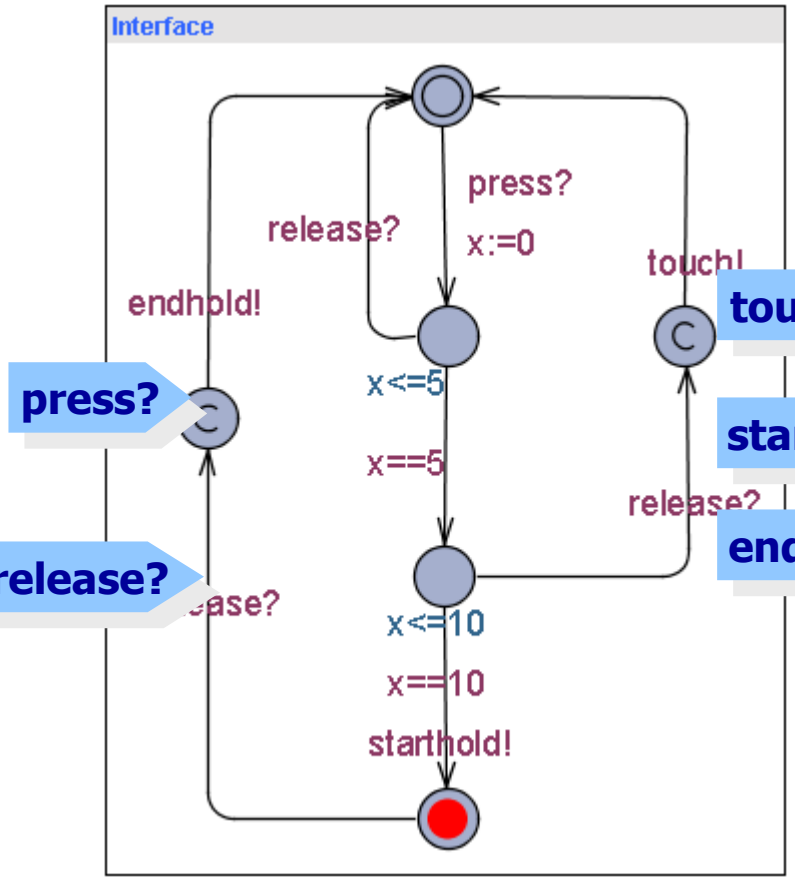
press? 0.2 release? … press? 0.7 release? … press? 1.0 2.4 release? ..

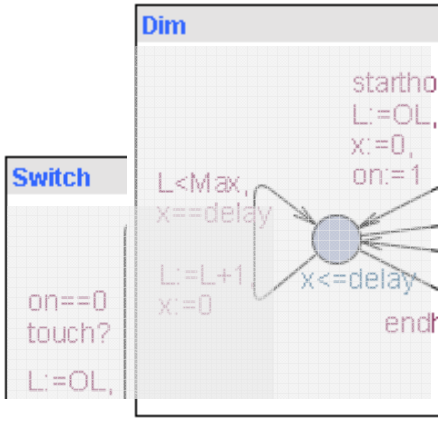
↓ ↓ ↓ ↓

Ø touch! starthold! endhold!

Light Control Interface

User





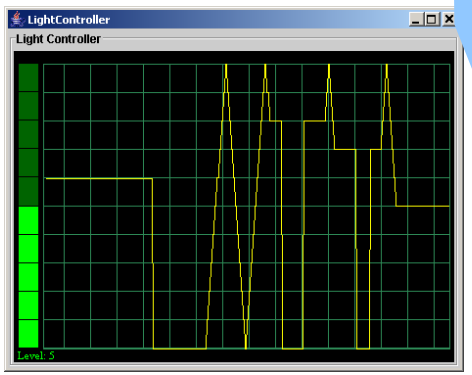
touch!

starthold!

endhold!

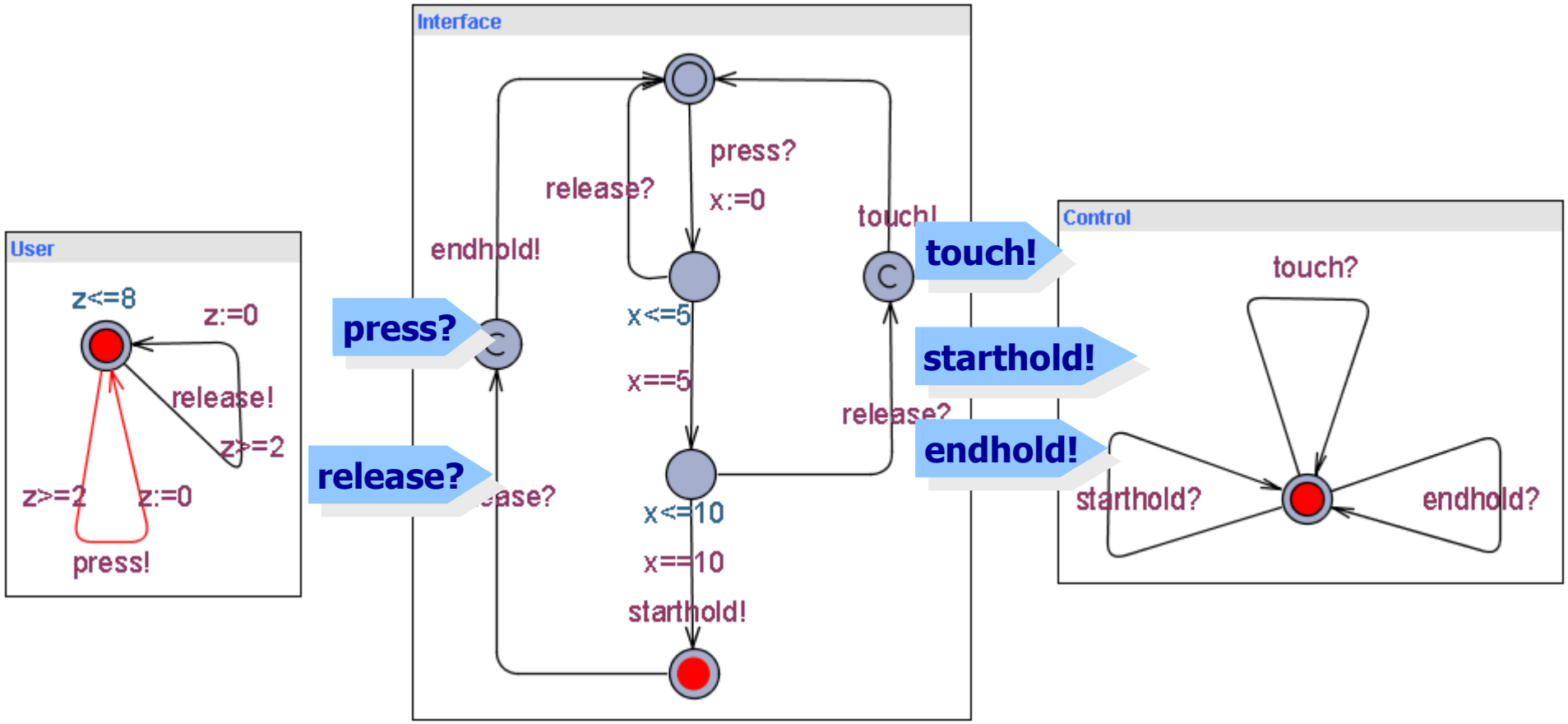
Control Program

L++/L--/L:=0



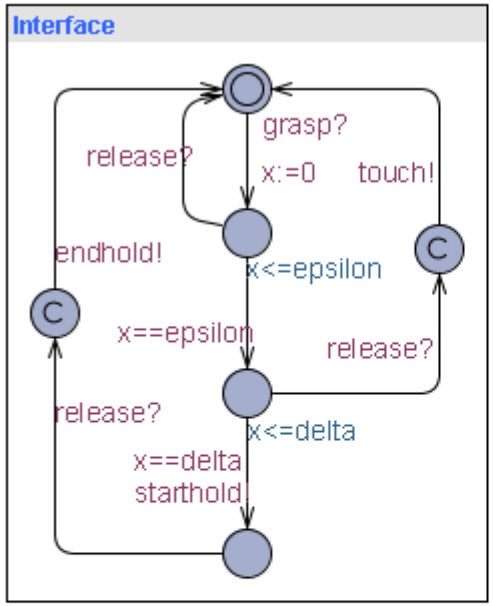
UCb

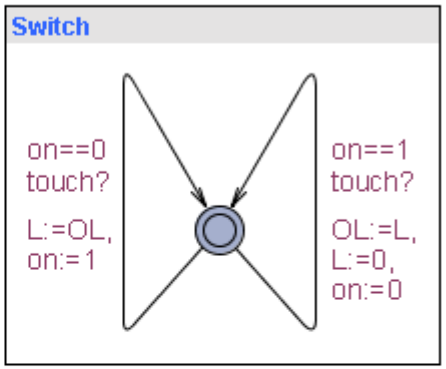
Light Control Network

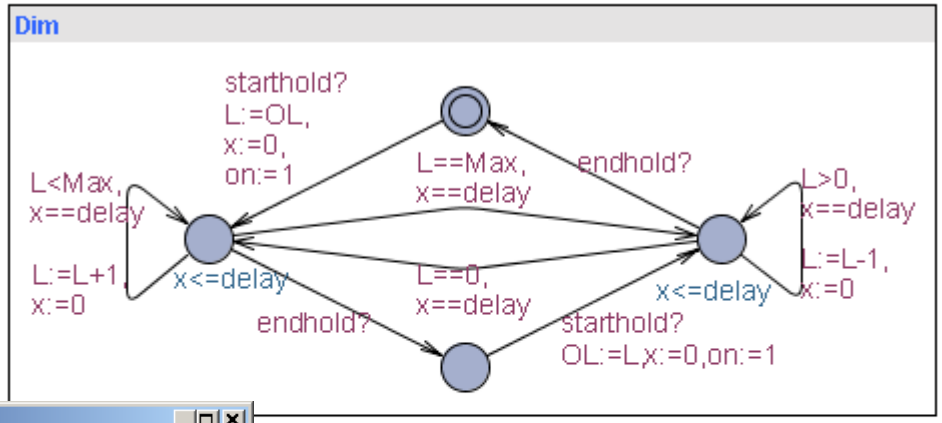


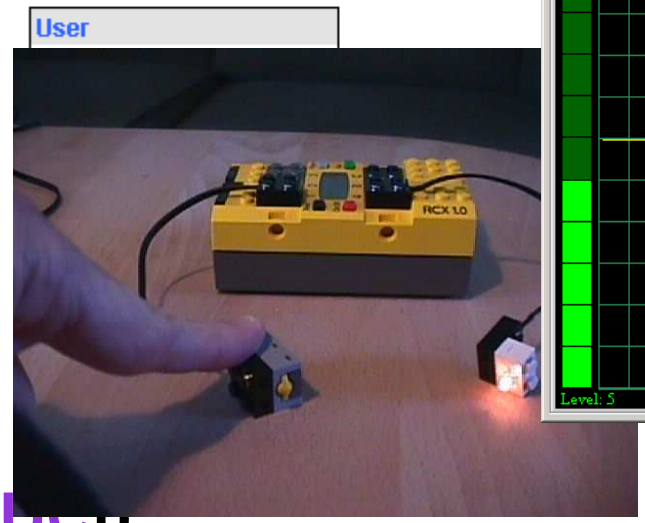
Validation

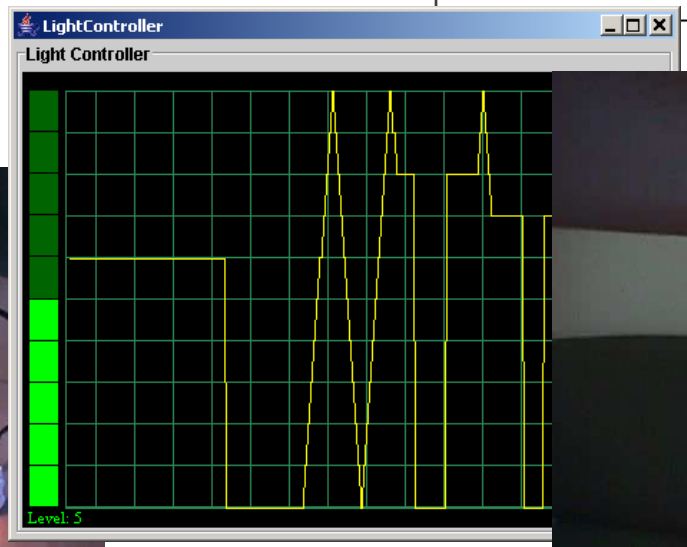
Light Controller

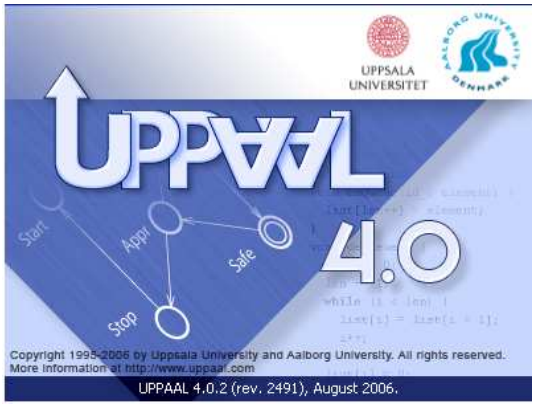












Timed Automata
Modeling
and
Decidability

Kim Guldstrand Larsen

AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Overview

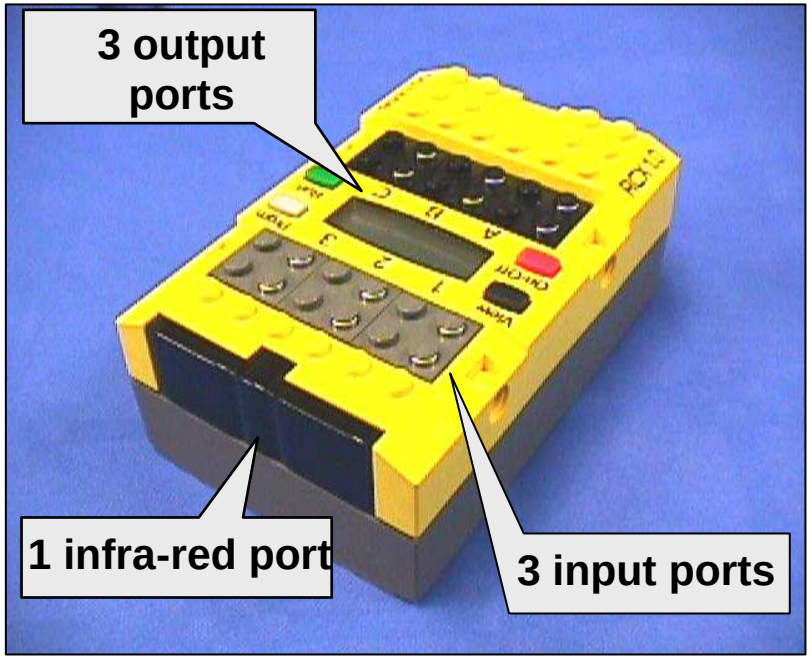
- BRICK Sorting
- Reachability Checking
  - Region Construction
- Bisimulation Checking
- Model Checking
- Trace Inclusion Checking

Brick Sorting

BRICS
Basic Research
in Computer Science

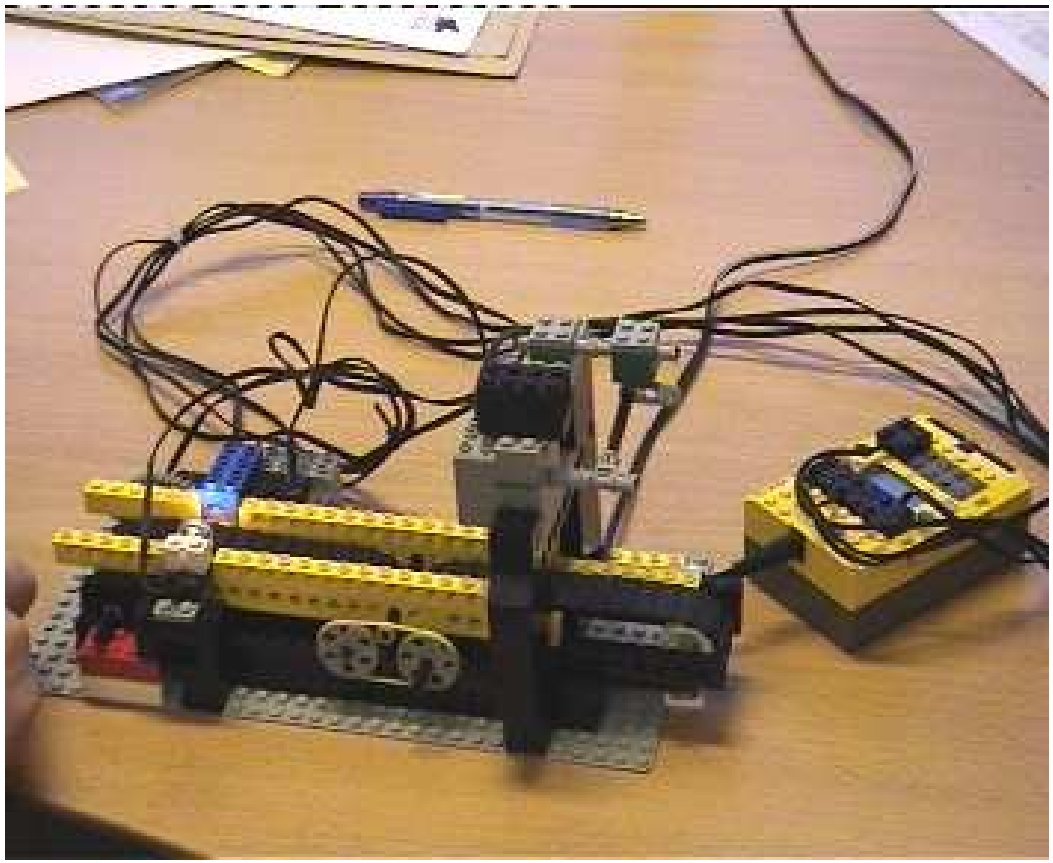
LEGO Mindstorms/RCX

- Sensors: temperature, light, rotation, pressure.
- Actuators: motors, lamps,
- Virtual machine:
  - 10 tasks, 4 timers, 16 integers.
- Several Programming Languages:
  - NotQuiteC, Mindstorm, Robotics, legOS, etc.



A Real Real Timed System

The Plant
Conveyor Belt
&
Bricks



Controller Program
LEGO MINDSTORM

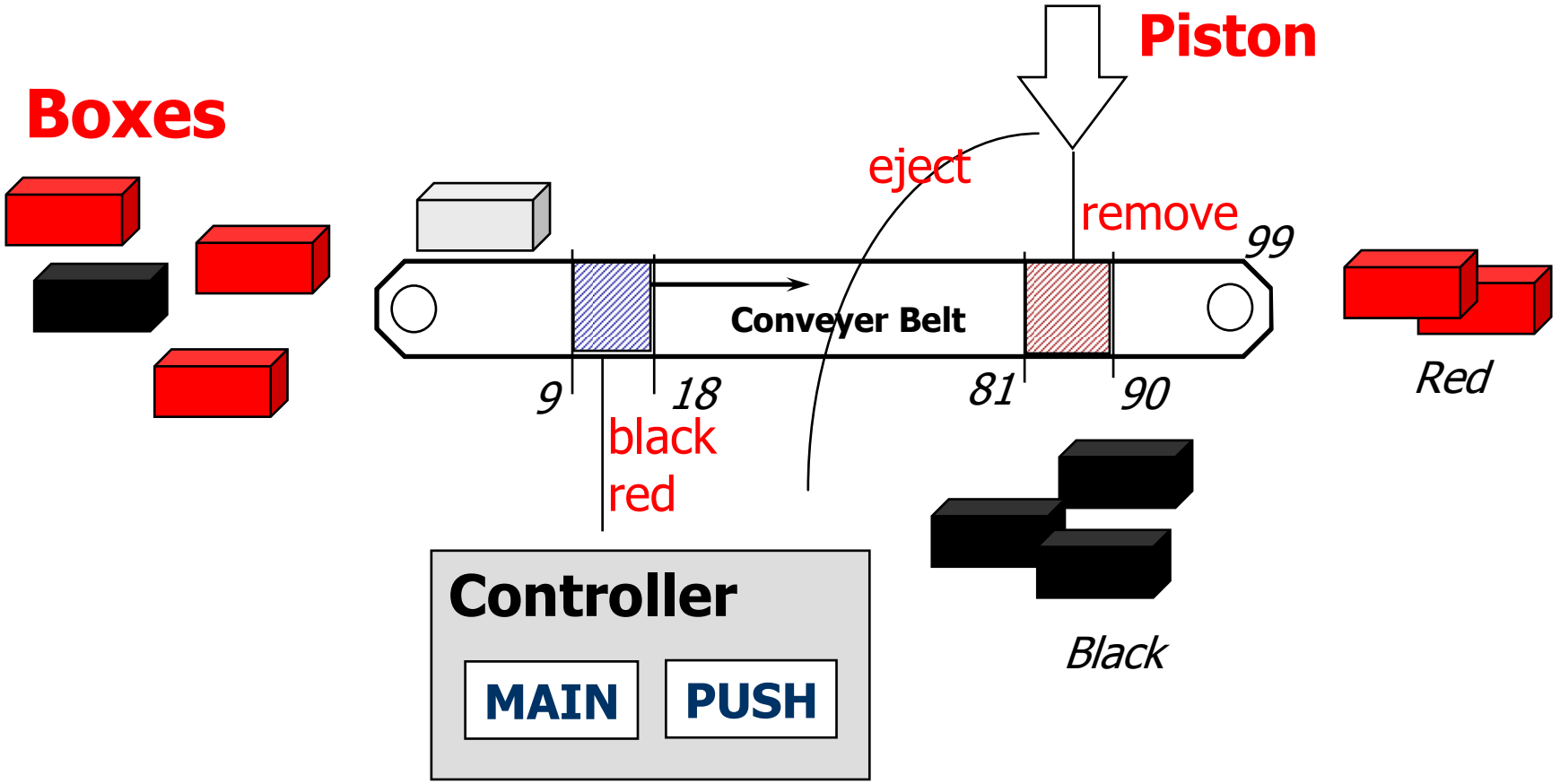
What is the CONTROL program doing?

UCb

First UPPAAL model

Sorting of Lego Boxes

Ken Tindell



NQC programs

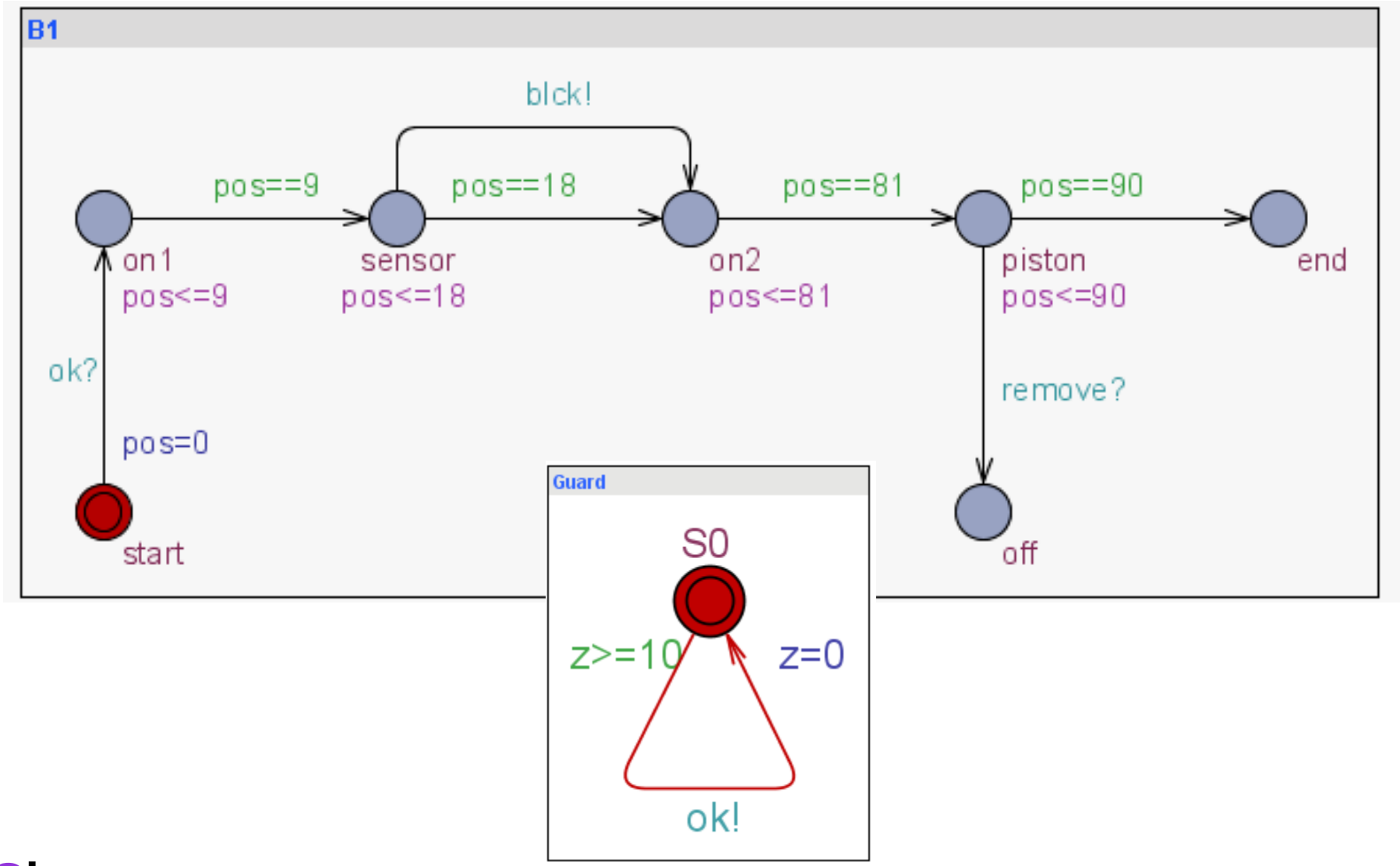
```
int active;
int DELAY;
int LIGHT_LEVEL;
```

```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 start PUSH;

 while(true){

wait(IN_1<=LIGHT_LEVEL);
   ClearTimer(1);
   active=1;
   PlaySound(1);

wait(IN_1>LIGHT_LEVEL);
  }
}
```

```
task PUSH{
  while(true){
    wait(Timer(1)>DELAY && active==1);
    active=0;
    Rev(OUT_C,1);
    Sleep(8);
    Fwd(OUT_C,1);
    Sleep(12);
    Off(OUT_C);
  }
}
```

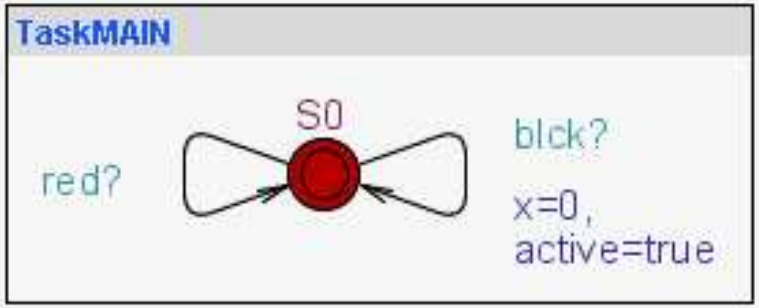
UCb

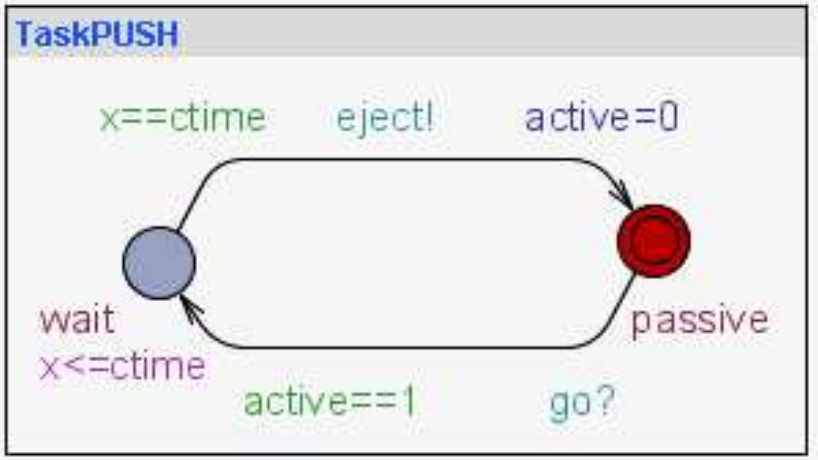
A Black Brick & The Guard

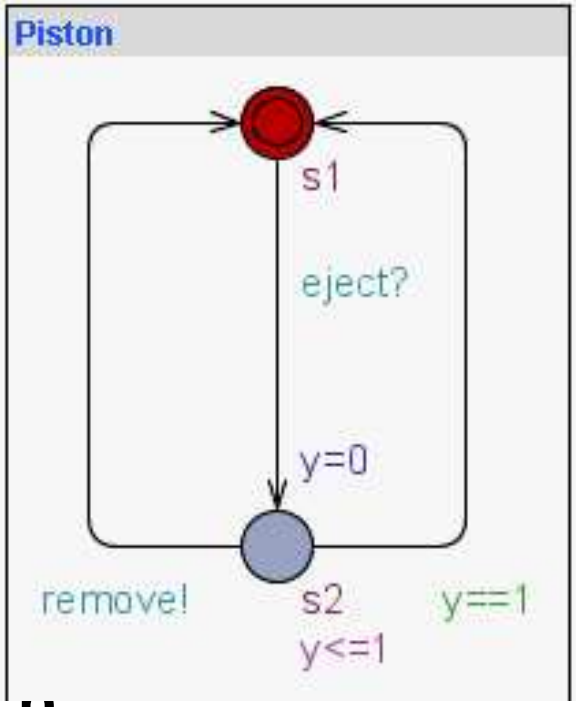


UCb

Control Tasks & Piston







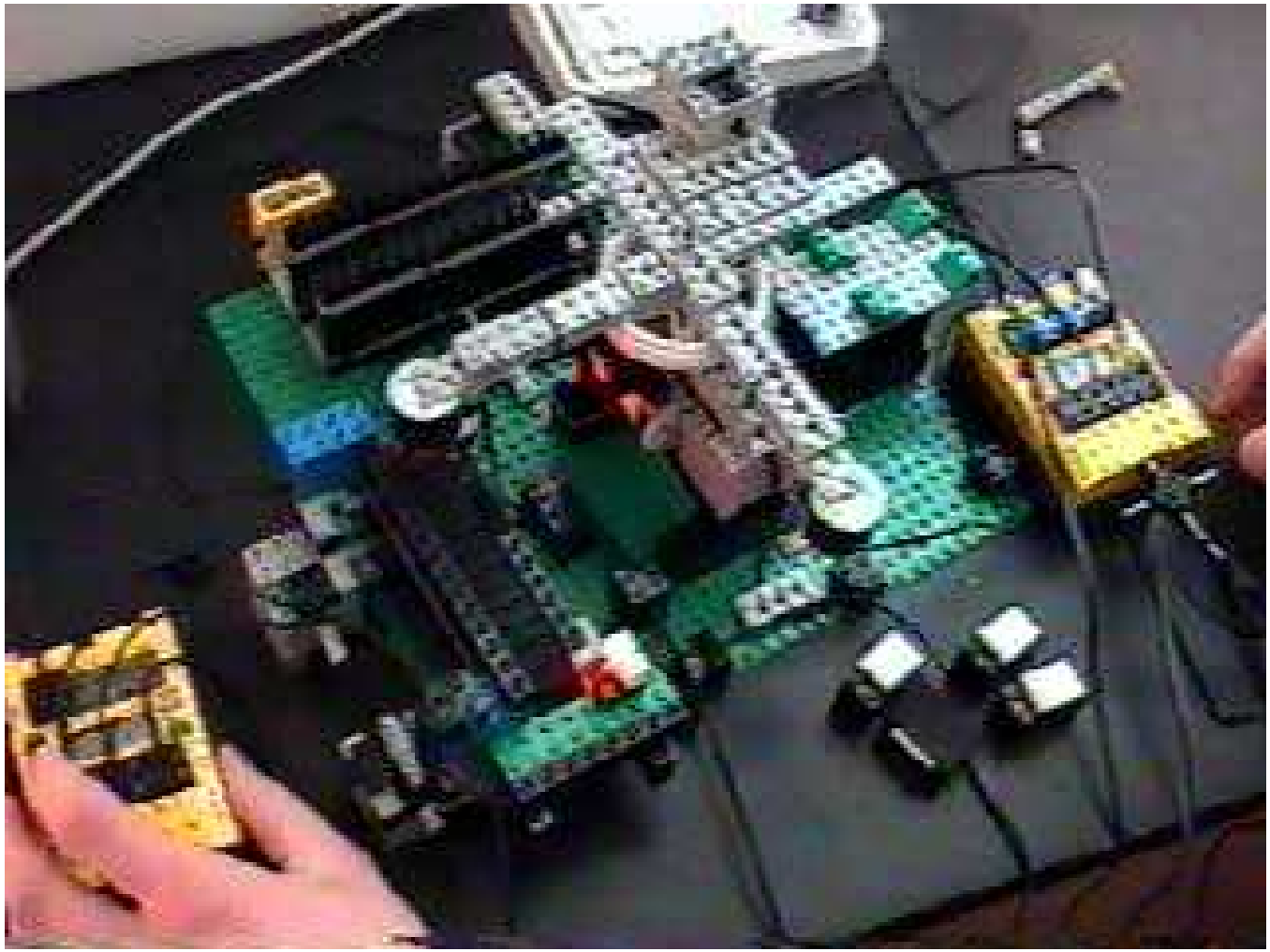
GLOBAL DECLARATIONS:

```
const int ctime = 75;

int[0,1] active;
clock x, time;

chan   eject, ok;
chan   blck, red, remove, go;
```

The Production Cell in LEGO

CSS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Course at DTU, Copenhagen

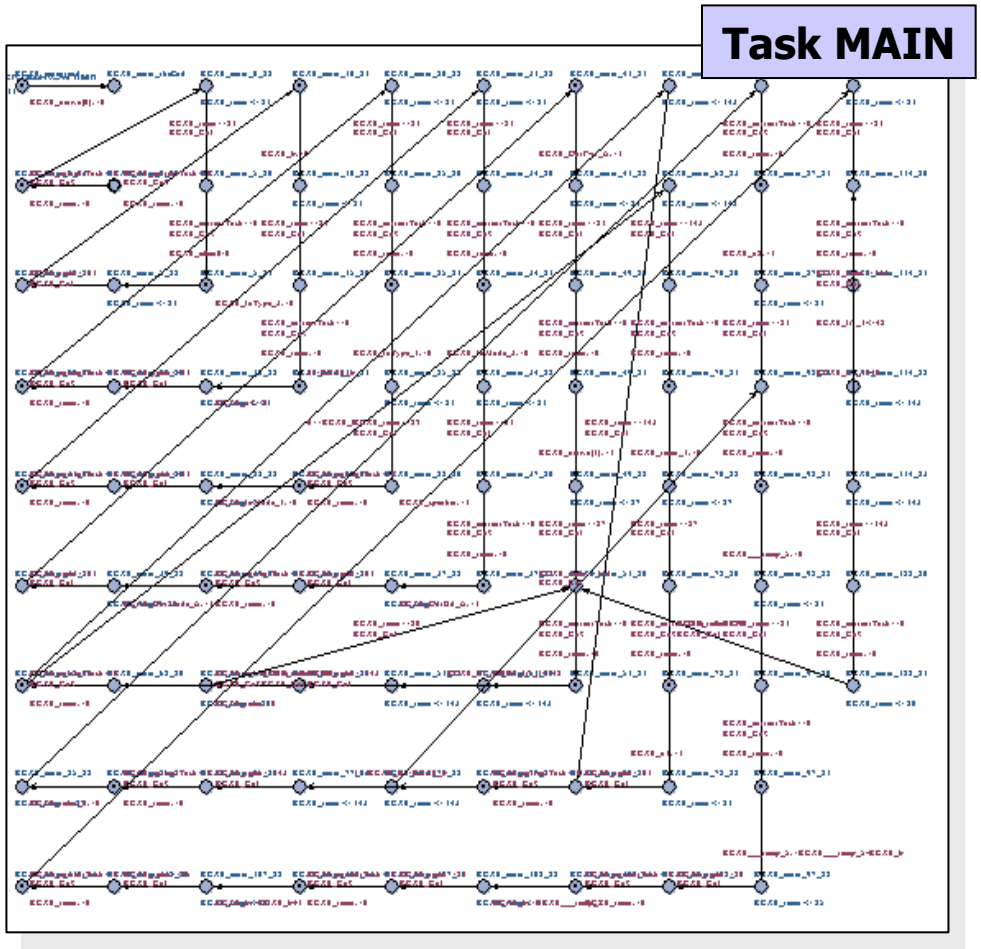


Rasmus Crüger Lund
Simon Tune Riemanni

UCb

From RCX to UPPAAL – and back

- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000 in Stockholm.



- From UPPAAL to RCX: Martijn Hendriks.

UCb

www.uppaal.com

RELATED SITES: TIMES | UPPAAL CORA | UPPAAL TRON

UPPAAL

Home

Home | About | Documentation | Download | Examples | Bugs

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University in Denmark.

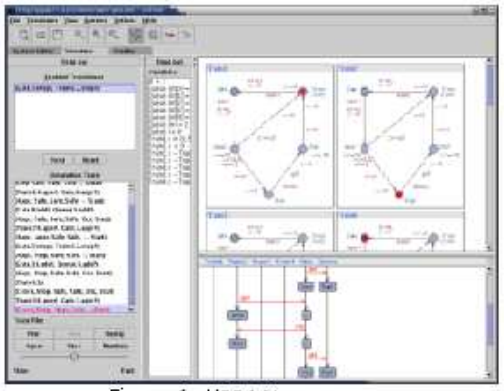


Figure 1: UPPAAL on screen.

Download

The current official release is UPPAAL 3.4.11 (Jun 23, 2005). A release of UPPAAL **3.6 alpha 3** (dec 20, 2005) is also available. For more information about UPPAAL version 3.4, we refer to this press release.

UPPSALA UNIVERSITET

AALBORG UNIVERSITY

License

The UPPAAL tool is **free** for non-profit applications. For information about commercial licenses, please email sales(at)uppaal(dot)com.

To find out more about UPPAAL, read this short introduction. Further information may be found at this web site in the pages About, Documentation, Download, and Examples.

Mailing Lists

UPPAAL has an open discussion forum group at Yahoo!Groups intended for users of the tool. To join or post to the forum, please refer to the information at the discussion forum page. Bugs should be reported using the bug tracking system. To email the development team directly, please use uppaal(at)list(dot)it(dot)uu(dot)se.

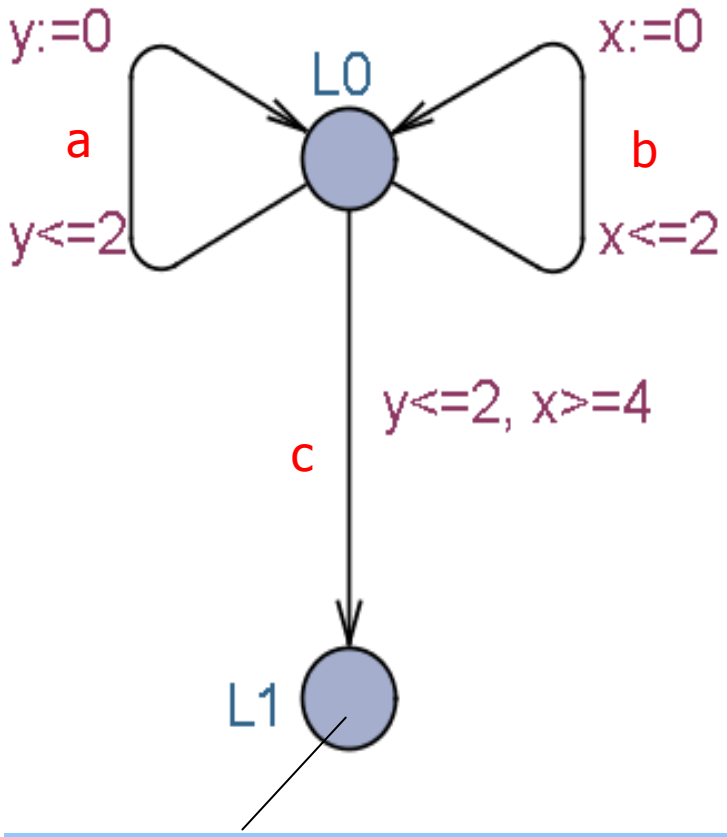
UCb

Decidability

The Region Construction

BRICS
Basic Research
in Computer Science

Reachability ?



OBSTACLE: Uncountably infinite state space

$$L \times \mathbb{R}^C$$

locations **clock-valuations**

Reachable from initial state (L0,x=0,y=0) ?

UCb

Constraints

Definition

Let X be a set of clock variables. The set $\mathcal{B}(X)$ of *clock constraints* ϕ is given by the grammar:

$$\phi \ ::= \ x \leq c \mid c \leq x \mid x < c \mid c < x \mid \phi_1 \wedge \phi_2$$

where $c \in \mathbb{N}$ (or $\mathbb{Q}$).

CSS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Clock Valuations and Notation

Definition

The set of *clock valuations*, $\mathbb{R}^C$ is the set of functions $C \longrightarrow \mathbb{R}_{\geq 0}$ ranged over by $u, v, w, \ldots$.

Notation

Let $u \in \mathbb{R}^C$, $r \subseteq C$, $d \in \mathbb{R}_{\geq 0}$, and $g \in \mathcal{B}(X)$ then:

- $u + d \in \mathbb{R}^C$ is defined by $(u + d)(x) = u(x) + d$ for any clock x
- $u[r] \in \mathbb{R}^C$ is defined by $u[r](x) = 0$ when $x \in r$ and $u[r](x) = u(x)$ for $x \notin r$.
- $u \models g$ denotes that g is satisfied by u.

UCb

CSS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Timed Automata

Definition

A timed automaton A over clocks C and actions Act is a tuple (L, l_0, E, I), where:

- L is a finite set of locations
- $l_0 \in L$ is the initial location
- $E \subseteq L \times \mathcal{B}(X) \times Act \times \mathcal{P}(C) \times L$ is the set of edges
- $I : L \longrightarrow \mathcal{B}(X)$ assigns to each location an invariant

Semantics

Definition

The semantics of a timed automaton A is a labelled transition system with state space $L \times \mathbb{R}^C$ with initial state $(l_0, u_0)^*$ and with the following transitions:

- $(l, u) \xrightarrow{\epsilon(d)} (l, u + d)$ iff $u \in I(l)$ and $u + d \in I(l)$,
- $(l, u) \xrightarrow{a} (l', u')$ iff there exists $(l, g, a, r, l') \in E$ such that
  - $u \models g$,
  - $u' = u[r]$, and
  - $u' \in I(l')$

*$u_0(x) = 0$ for all $x \in C$

Derived Relations and Reachability

$$(l,u) \xrightarrow{\delta} (l',u') \quad \text{iff} \quad \exists d > 0.\,(l,u) \xrightarrow{\epsilon(d)} (l',u').$$

$$(l,u) \xrightarrow{\alpha} (l',u') \quad \text{iff} \quad \exists a \in Act.\,(l,u) \xrightarrow{a} (l',u')$$

$$(l,u) \rightsquigarrow (l',u') \quad \text{iff} \quad (l,u)(\xrightarrow{\delta} \cup \xrightarrow{\alpha})^*(l',u')$$

Definition

The set of reachable locations, $Reach(A)$, of a timed automaton A is defined as:

$$l \in Reach(A) \equiv^{\triangle} \exists u.\,(l_0,u_0) \rightsquigarrow (l,u)$$

Time Abstracted Bisimulation

Definition

Let $G \subseteq L$ be a set of goal locations. An equivalence relation R on $L \times \mathbb{R}^C$ is a TAB wrt G if whenever $(l,u)R(n,v)$ the following holds:

1. $l \in G$ iff $n \in G$,
2. whenever $(l,u) \xrightarrow{\delta} (l',u')$ then $(n,v) \xrightarrow{\delta} (n',v')$ with $(l',u')R(n',v')$
3. whenever $(l,u) \xrightarrow{a} (l',u')$ then $(n,v) \xrightarrow{a} (n',v')$ with $(l',u')R(n',v')$

Stable Quotient

Definition

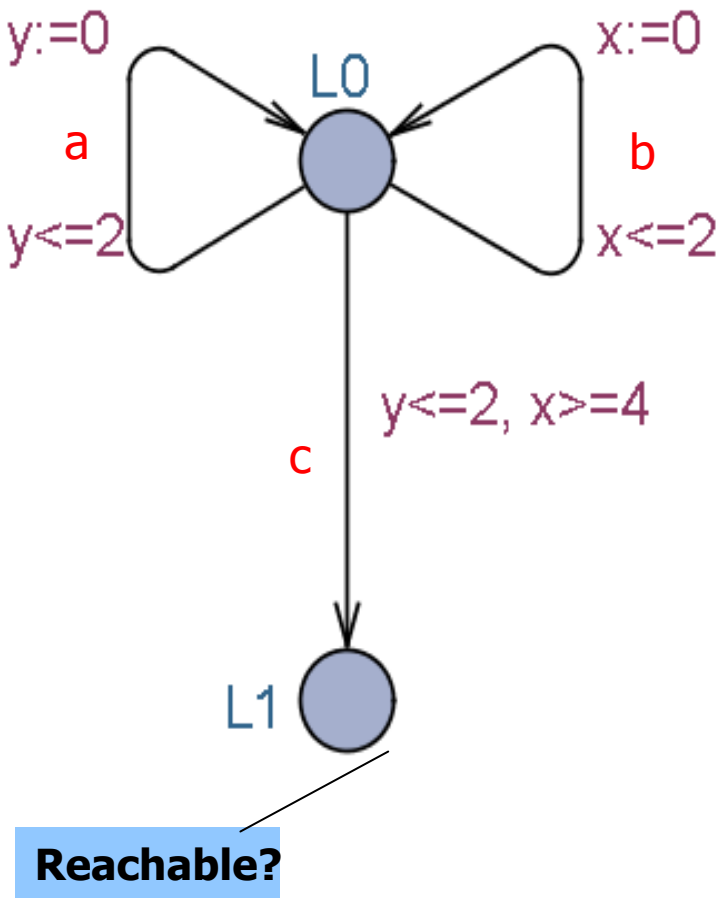
Let R be a TAB wrt G. The induced *quotient* has classes of R, $\pi \in (L \times \mathbb{R}^C/R)$, as states. For classes π, π' the transitions are

- $\pi \xrightarrow{\delta} \pi'$ iff $(l,u) \xrightarrow{\delta} (l',u')$ for some $(l,u) \in \pi$, $(l',u') \in \pi'$.
- $\pi \xrightarrow{a} \pi'$ iff $(l,u) \xrightarrow{a} (l',u')$ for some $(l,u) \in \pi$, $(l',u') \in \pi'$.

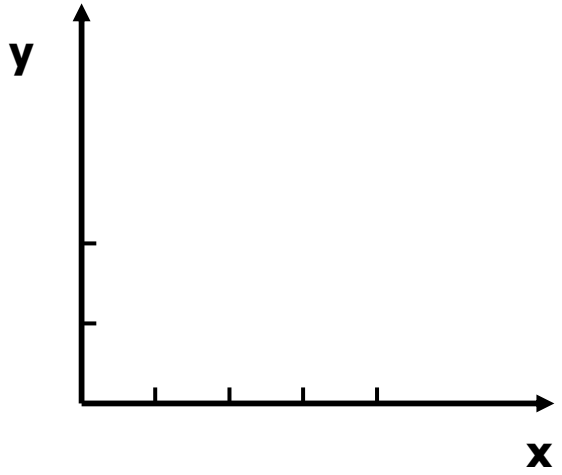
Theorem

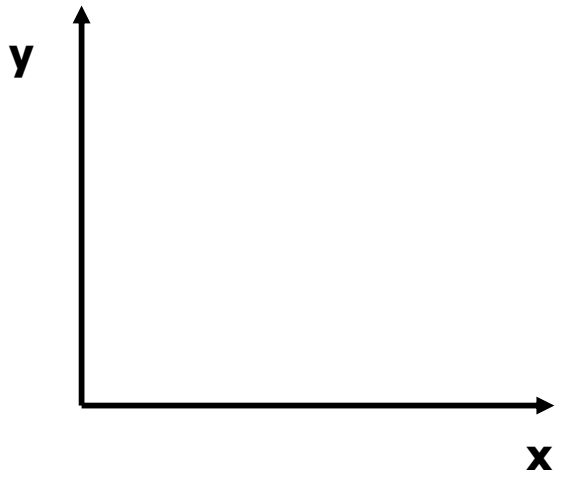
Let R be TAB wrt G. Then, a location from G is reachable iff there exists an equivalence class π of R such that π is reachable in the quotient and π contains a state whose location is in G.

Stable Quotient



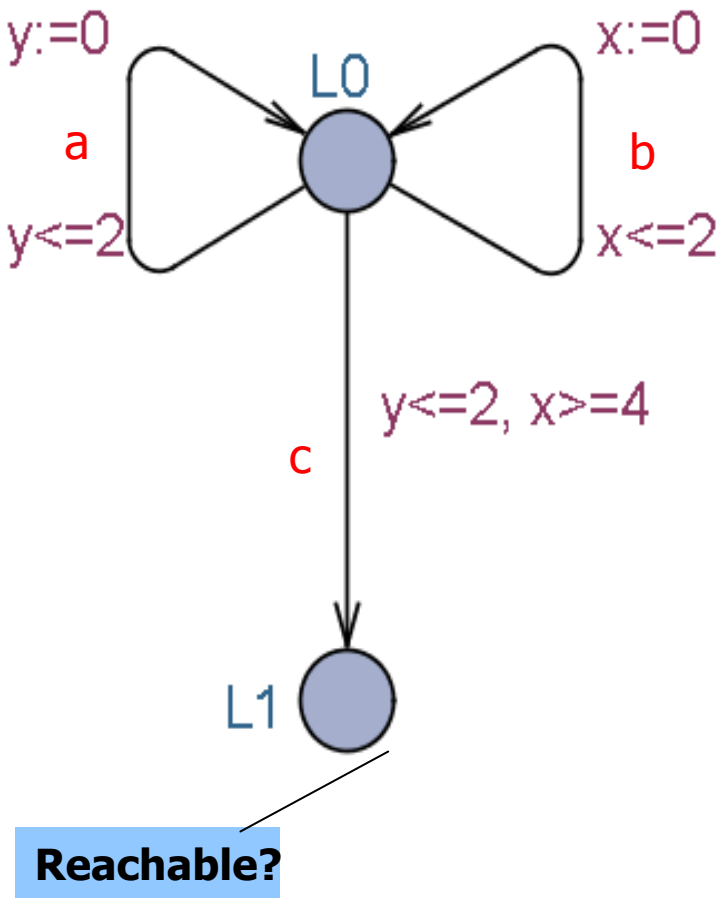
Partitioning



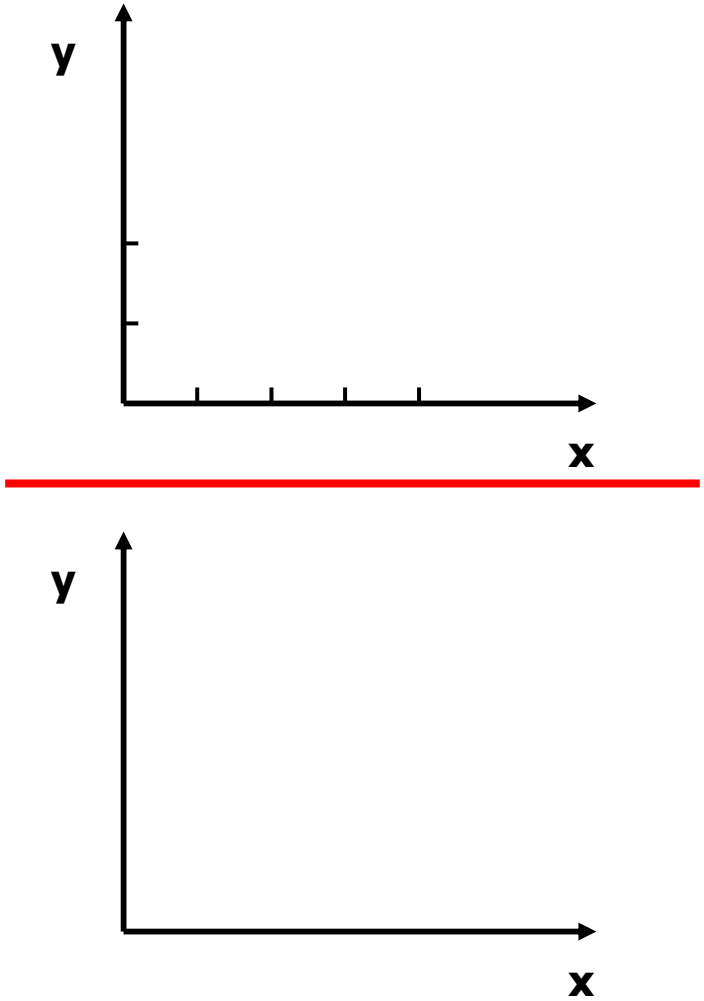


UCb

Stable Quotient

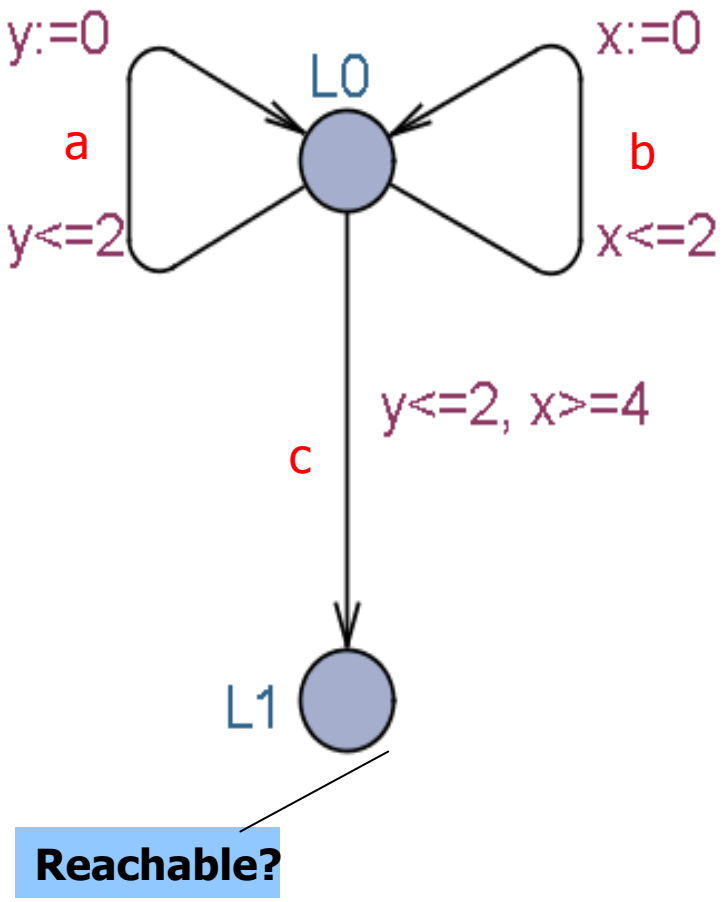


Partitioning

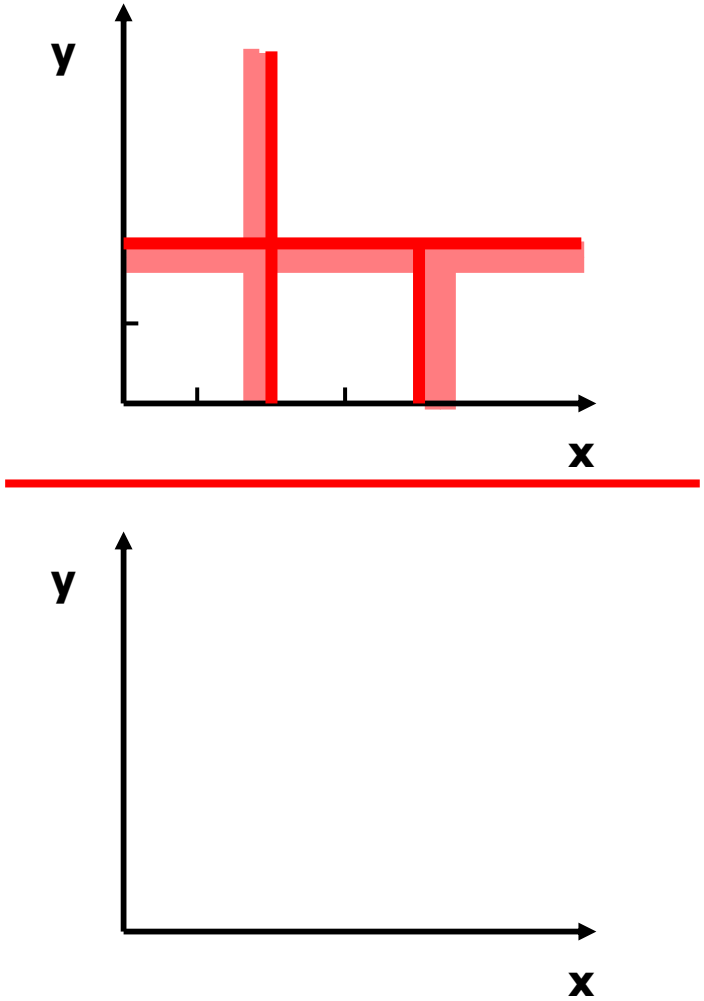


UCb

Stable Quotient

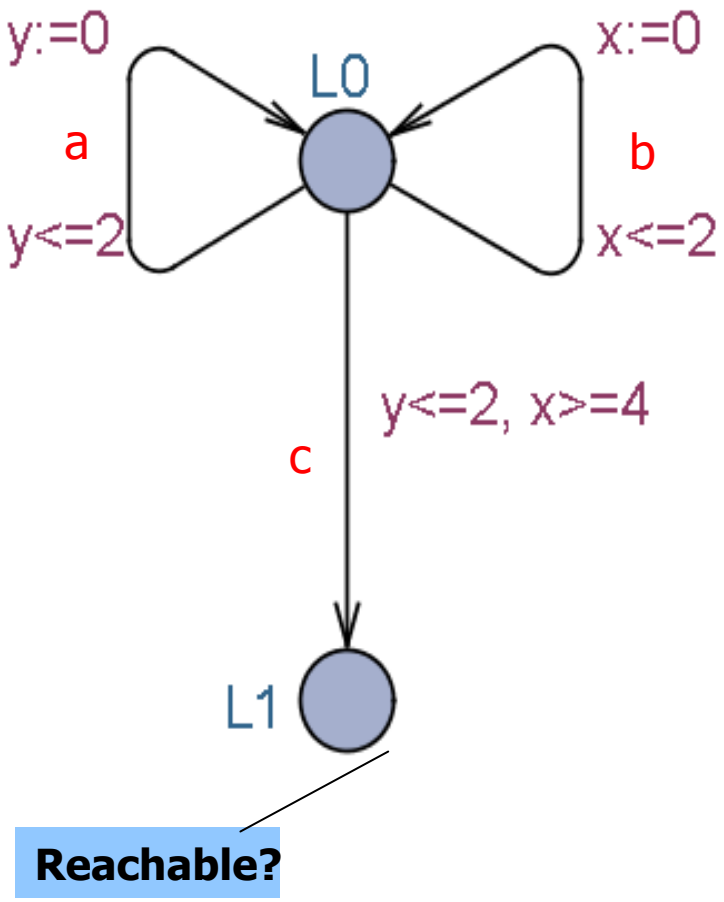


Partitioning

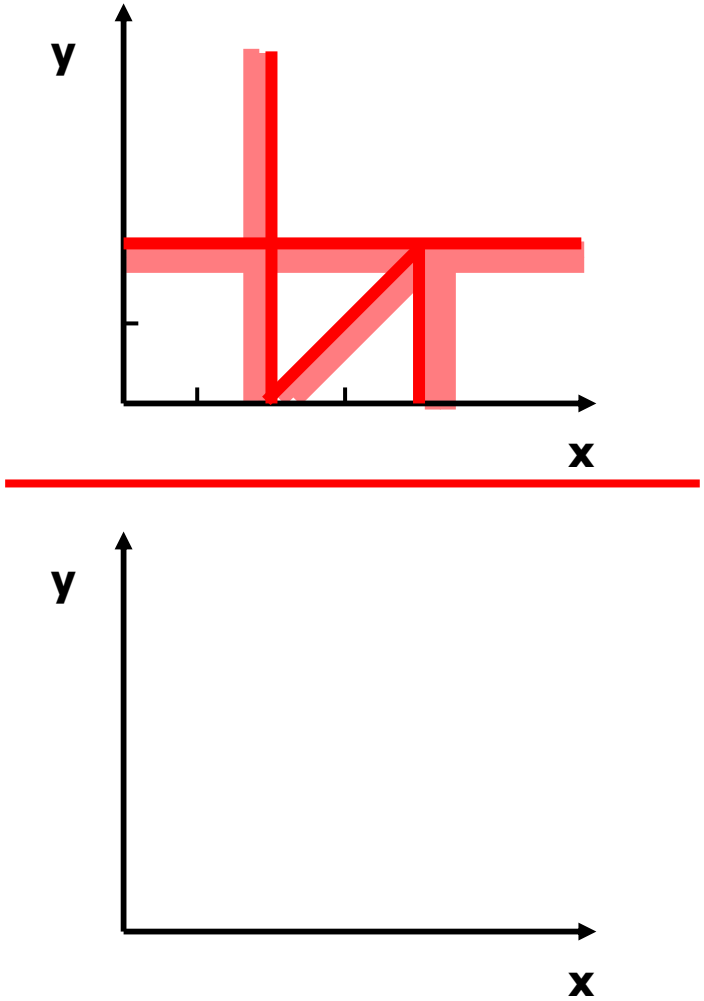


UCb

Stable Quotient

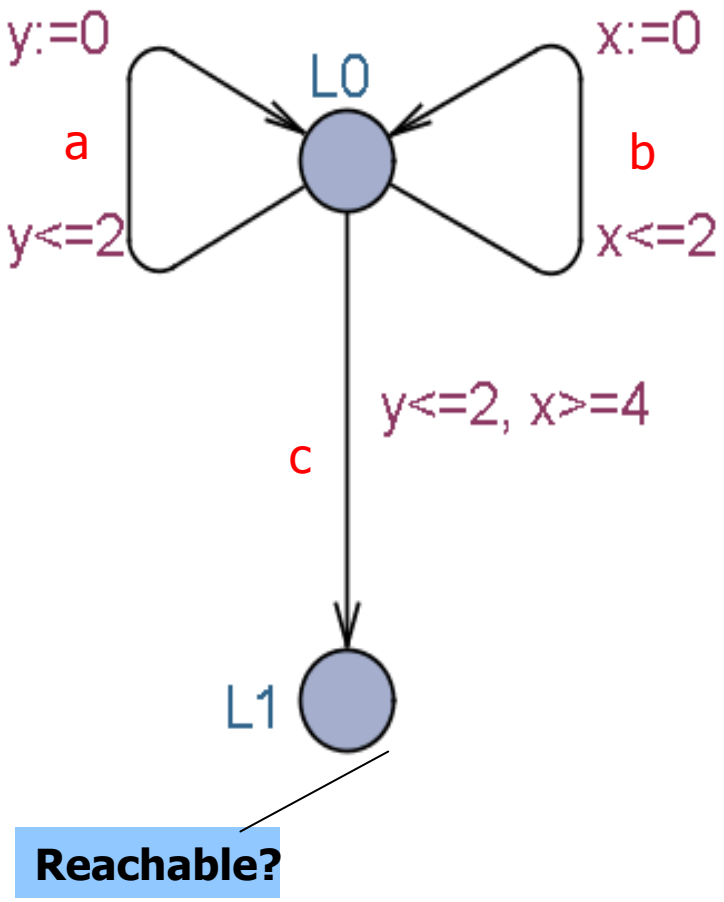


Partitioning

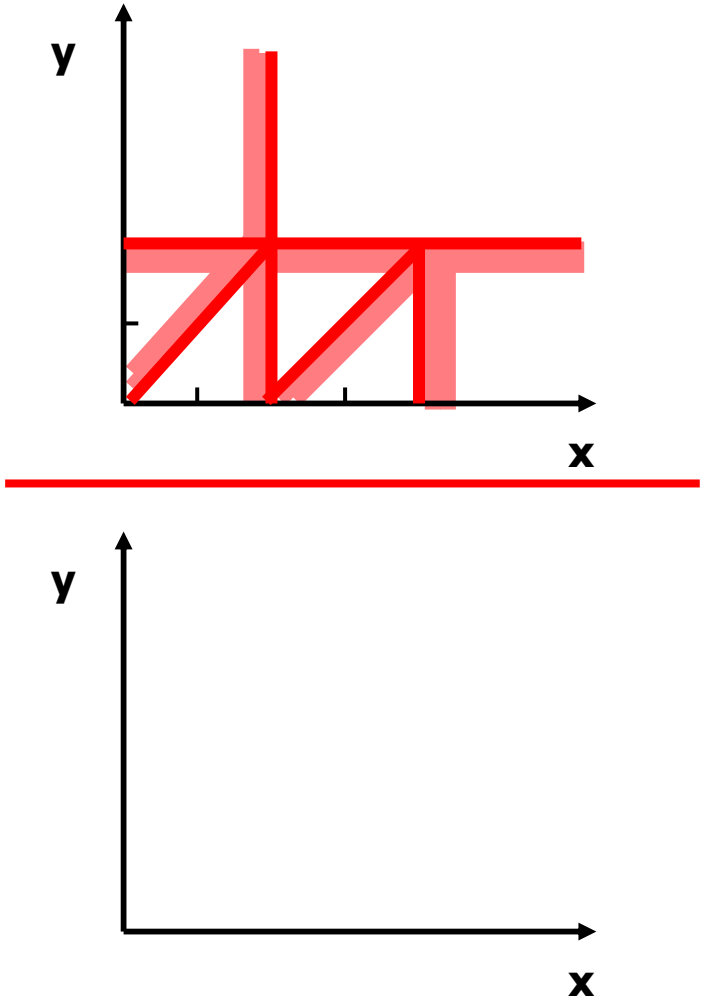


UCb

Stable Quotient

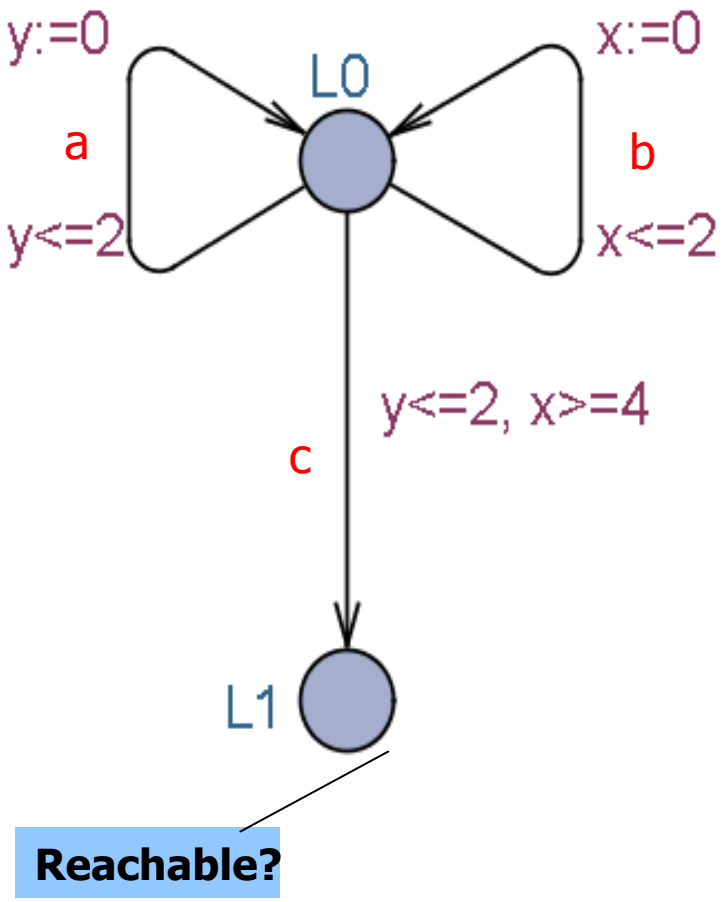


Partitioning

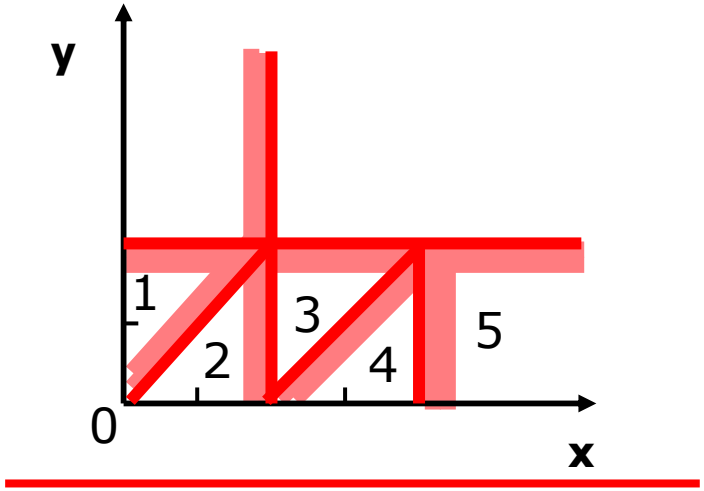


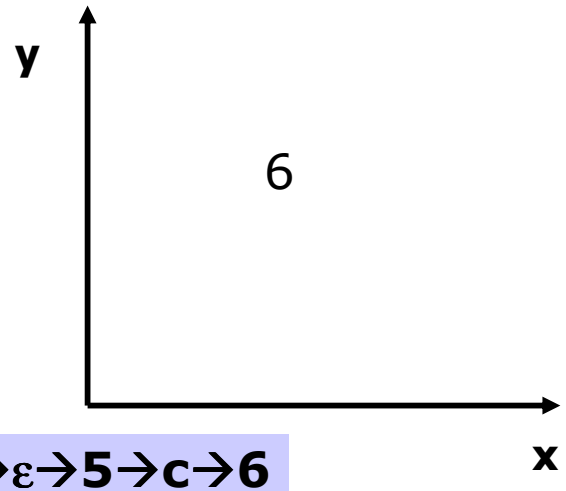
UCb

Stable Quotient



Partitioning





0 →ε→1→a→2→ε→3→a→4→ε→5→c→6

UCb

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

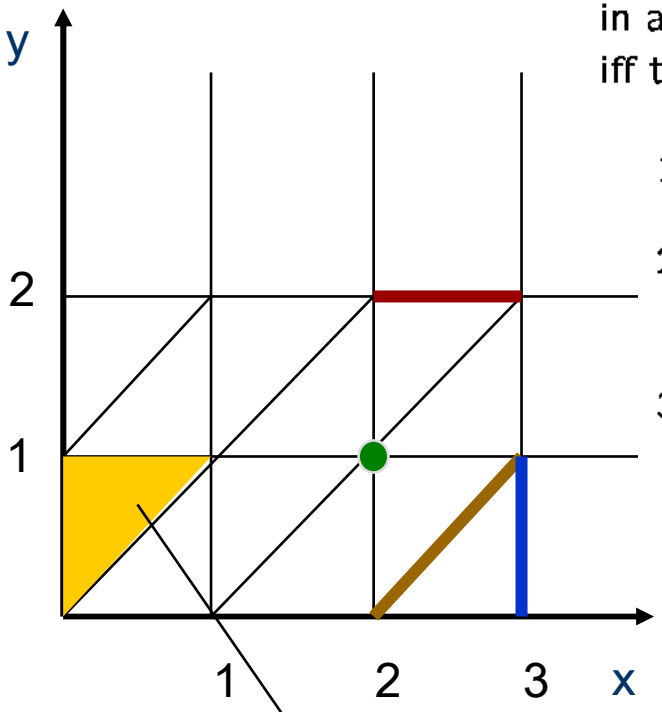
Region Equivalence

For each clock x let c_x be the largest integer with which x is compared in any guard or invariant of A. u and u' are *region equivalent*, $u \cong u'$ iff the following holds:

1. For all $x \in C$, either $\lfloor u(x) \rfloor = \lfloor u'(x) \rfloor$ or $u(x), u'(x) > c_x$;
2. For all $x, y \in C$ with $u(x) \leq c_x$ and $u(y) \leq c_y$, $fr(u(x)) \leq fr(u(y))$ iff $fr(u'(x)) \leq fr(u'(y))$;
3. For all $x \in C$ with $u(x) \leq c_x$, $fr(u(x)) = 0$ iff $fr(u'(x)) = 0$.

Regions

Finite Partitioning of State Space



For each clock x let c_x be the largest integer with which x is compared in any guard or invariant of A. u and u' are *region equivalent*, $u \cong u'$ iff the following holds:

1. For all $x \in C$, either $\lfloor u(x) \rfloor = \lfloor u'(x) \rfloor$ or $u(x), u'(x) > c_x$;
2. For all $x, y \in C$ with $u(x) \leq c_x$ and $u(y) \leq c_y$, $fr(u(x)) \leq fr(u(y))$ iff $fr(u'(x)) \leq fr(u'(y))$;
3. For all $x \in C$ with $u(x) \leq c_x$, $fr(u(x)) = 0$ iff $fr(u'(x)) = 0$.

An equivalence class (i.e. a *region*)
in fact there is only a *finite* number of regions!!

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Logical Characterization of Regions

Each region may be represented by specifying

1. for every clock x a constraint from

$$\{x = c \mid c = 0, 1, \ldots, c_x\} \cup \{c - 1 < x < c \mid c = 1, \ldots, c_x\} \cup \{x > c_x\}$$

2. for every pair of clocks x, y such that $c-1 < x < c$ and $d-1 < y < d$ appears in 1., whether $fr(x)$ is $<$, $=$ or $>$ than $fr(y)$.

Theorem

The number of regions is $n! \cdot 2^n \cdot \prod_{x \in C}(2c_x + 2)$.

Stability of Regions

Lemma

1. If $u \cong u'$ then $\forall d. \exists d'. u + d \cong u' + d'$;
2. If $u \cong u'$ then* $\forall g \in \mathcal{B}(X). u \models g \Leftrightarrow u' \models g$;
3. If $u \cong u'$ then $\forall r \subseteq C. u[r] \cong u'[r]$.

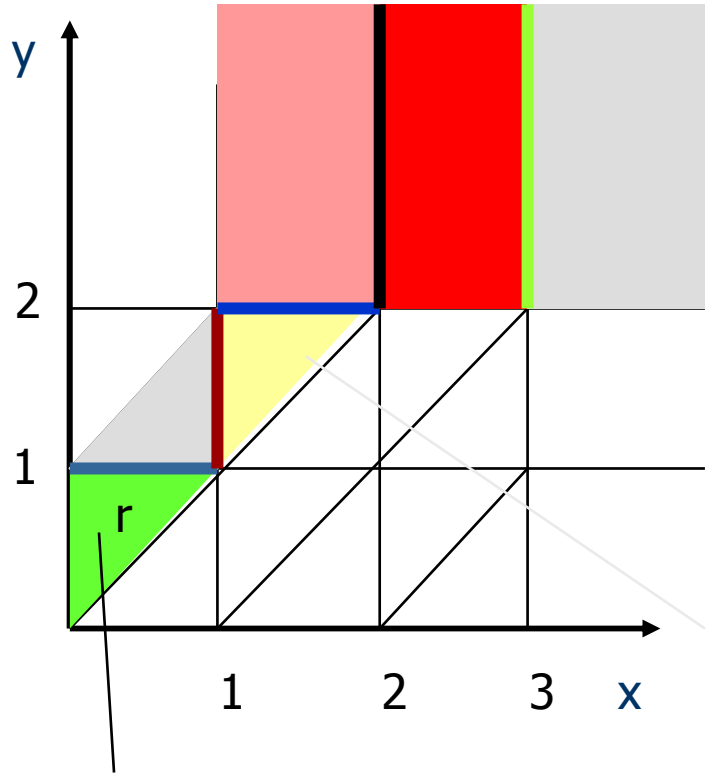
Theorem

Let $(l, u) \cong (l', u')$ iff $l = l'$ and $u \cong u'$. Then $\cong$ is a TAB with respect to any set of goal locations G.

*Here $\cong$ and g should agree on the maximal constants.

Regions

Successor Operation (wrt delay)

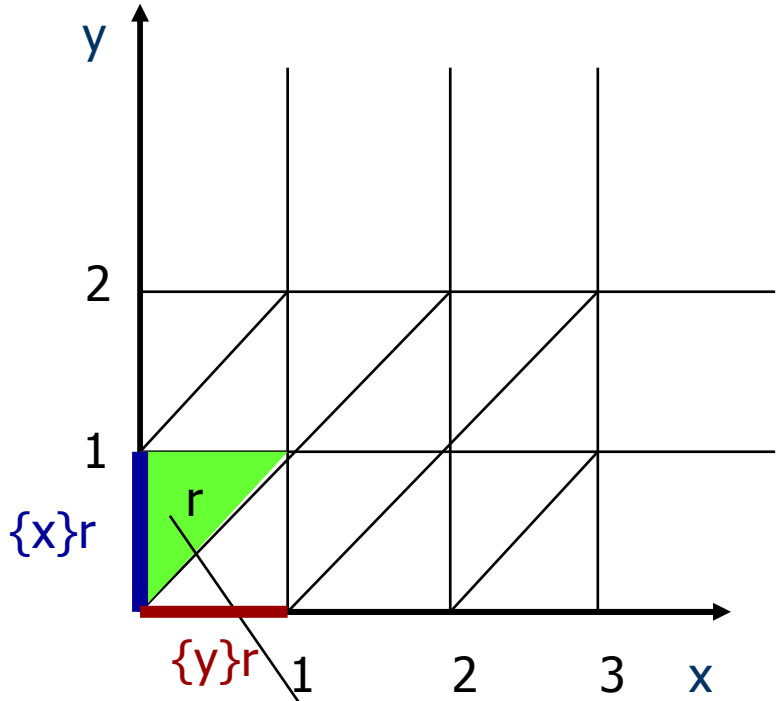


Successor regions, Succ(r)

An equivalence class (i.e. a *region*)

UCb

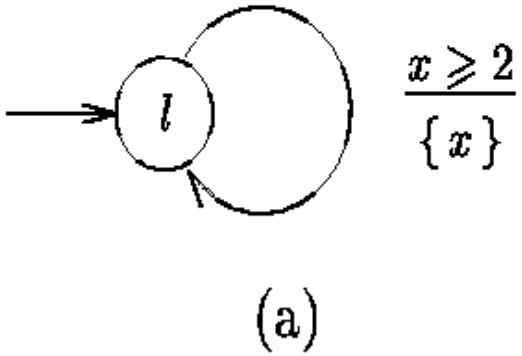
Regions
Reset Operation

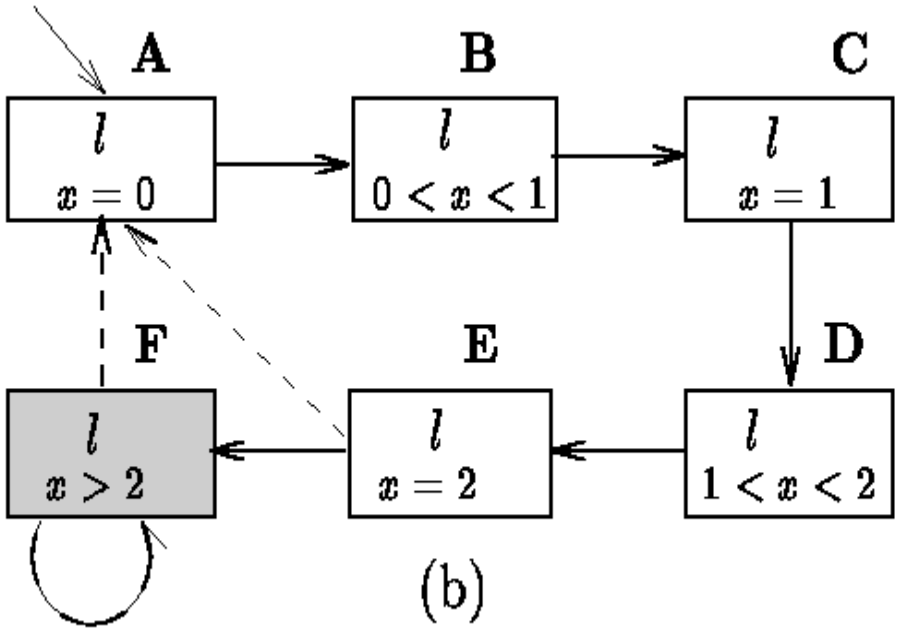


Reset regions

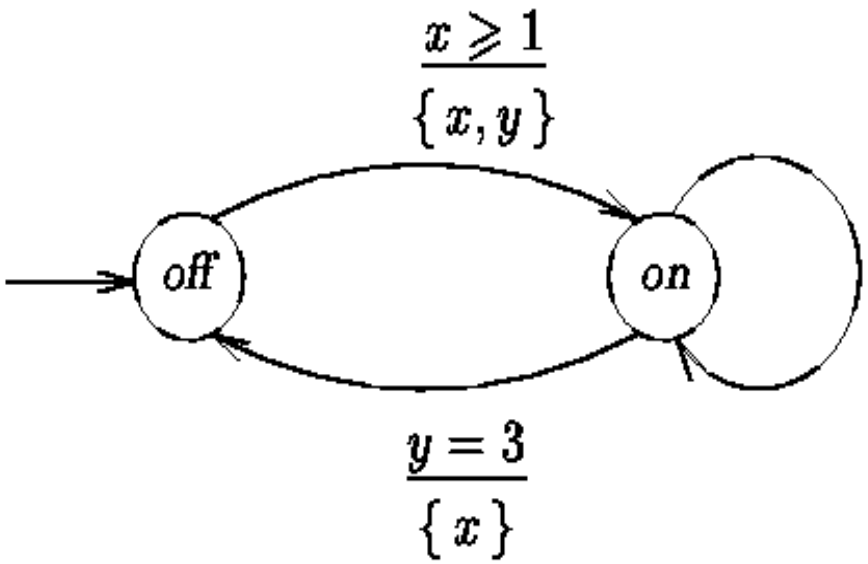
An equivalence class (i.e. a *region*) r

An Example Region Graph



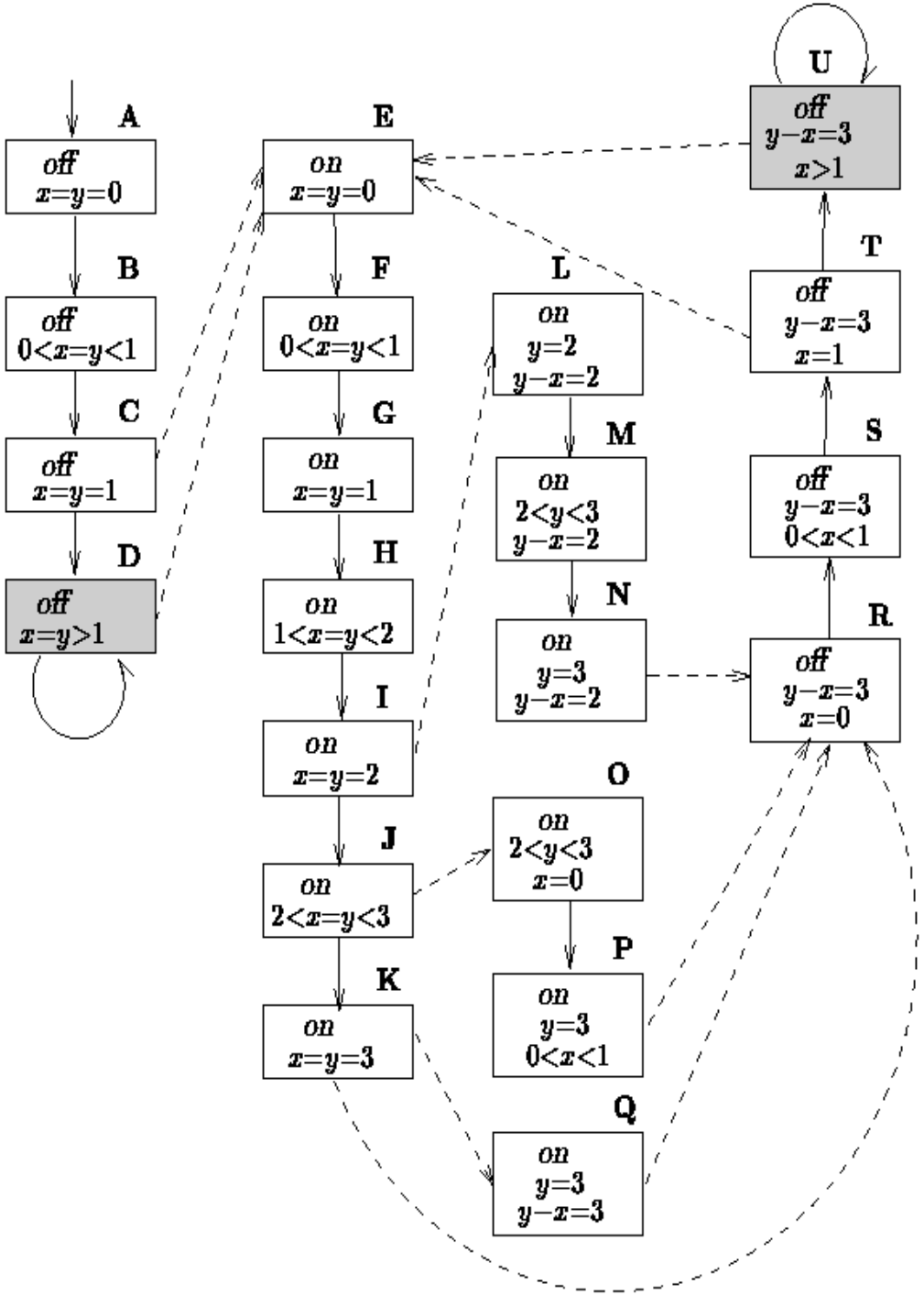


Modified light switch



$inv(off) = \text{true}$

$inv(on) = y \leqslant 3$



Reachable part of region graph

Decidability & Complexity

Theorem

The transition relations $\xrightarrow{\delta}$ and $\xrightarrow{a}$ between regions may be computed effectively using the triplet or the logical characterization of regions.

Theorem

Let A be a timed automaton with location l.
It is decidable whether $l \in Reach(A)$.

Theorem

Let A be a timed automaton with location l.
Deciding $l \in Reach(A)$ is PSPACE-complete.

Fundamental Results

- Reachability ☺
- Model-checking ☺
  - TCTL, L_{nu}, T_{mu},...
- Bisimulation, Simulation
  - Timed ☺ ; Untimed ☺
- Trace-inclusion
  - Timed ☹ ; Untimed ☺

UCb

Timed Bimulation

Wang'91, Cerans'92

BRICS
Basic Research
in Computer Science

Timed Bisimulation

Wang'91

R is a timed bisimulation if whenever sRt then the following holds:

i) $\left(s \xrightarrow{a} s'\right) \Rightarrow \left(\exists t'.t \xrightarrow{a} t' \wedge s'Rt'\right)$

ii) $\left(t \xrightarrow{a} t'\right) \Rightarrow \left(\exists s'.s \xrightarrow{a} s' \wedge s'Rt'\right)$

for all $a \in Act \cup Del$.

$Del = \{d : d \in R_{\geq 0}\}$

We write $s \approx t$ whenever sRt for some timed bisimulation R.

Timed Simulation

R is a timed simulation if whenever sRt then the following holds:

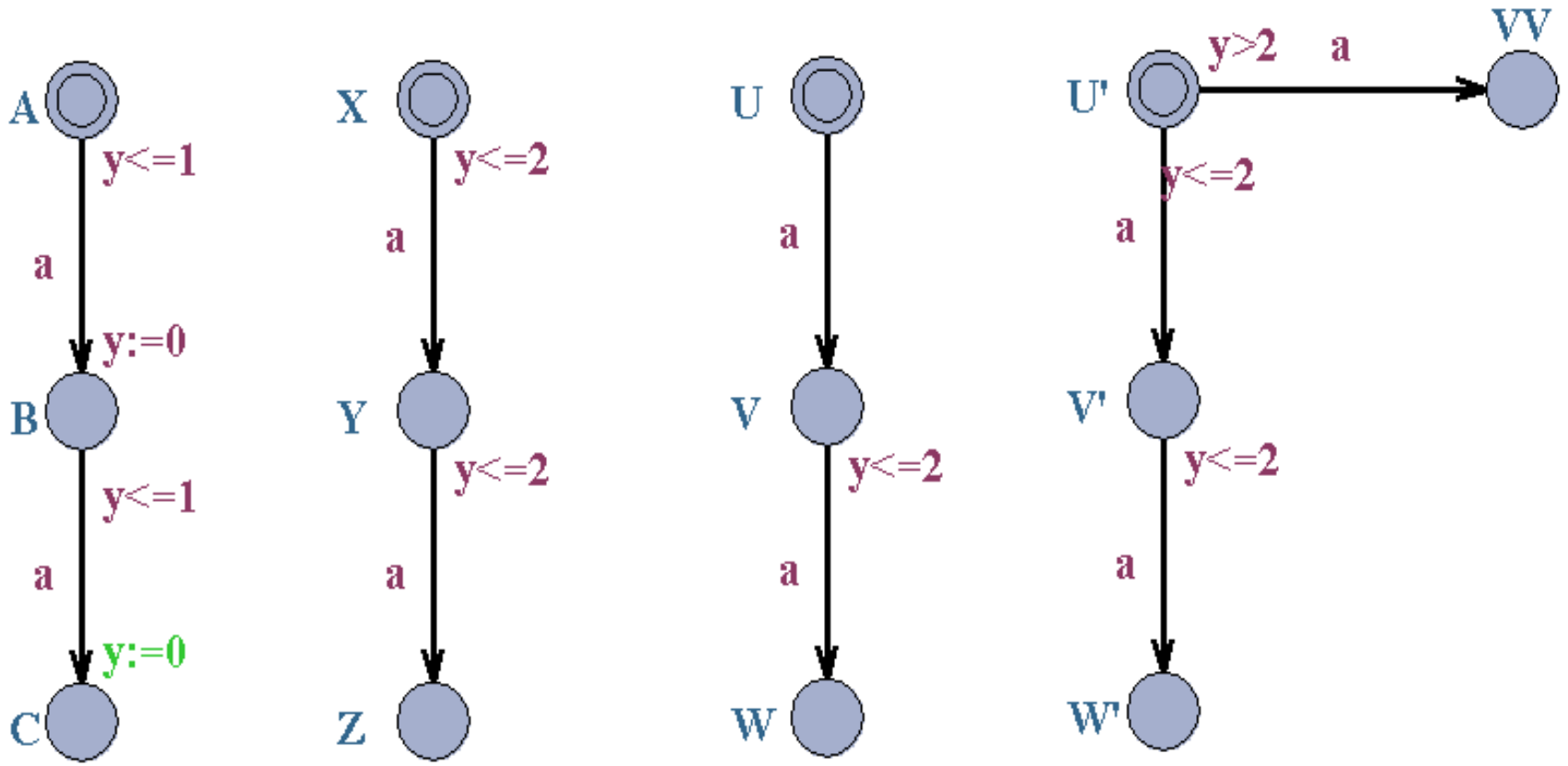
i) $\left(s \xrightarrow{a} s'\right) \Rightarrow \left(\exists t'.t \xrightarrow{a} t' \wedge s'Rt'\right)$

forall $a \in Act \cup Del$.

$$Del = \{d : d \in R_{\geq 0}\}$$

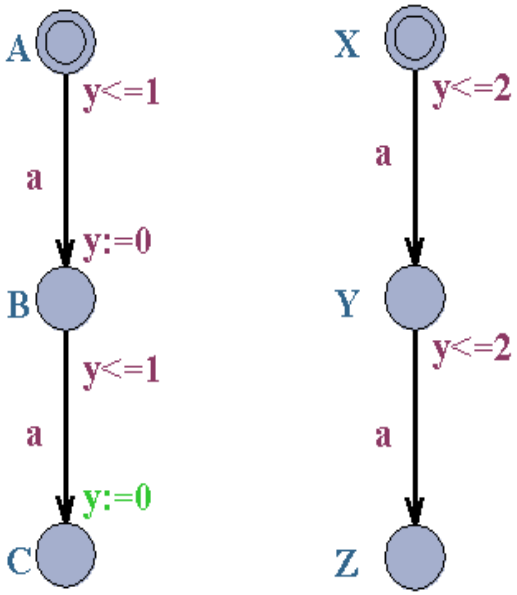
We write $s \prec t$ iff sRt for some timed simulation R.

Examples

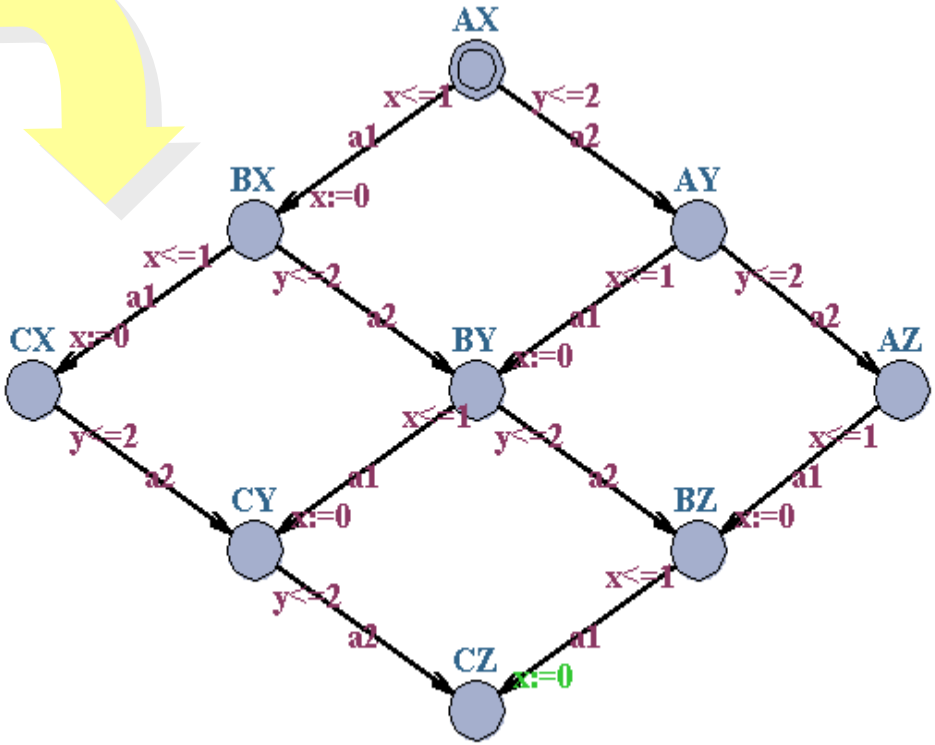


Towards Timed Bisimulation Algorithm

Cerans'92

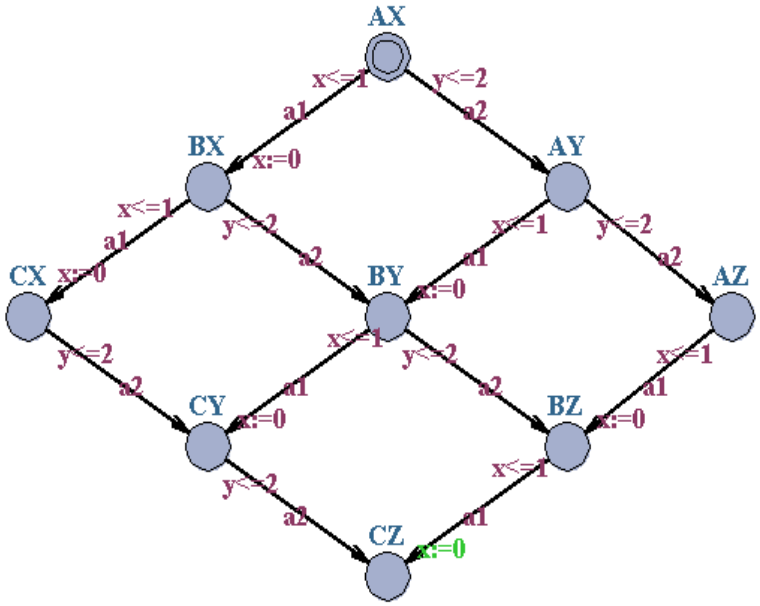


independent
"product-construction"



UCb

Towards Timed Bisimulation Algorithm



Definition

B is a timed product-bisimulation iff whenever $s \in B$ then the following holds:

i) if $s \xrightarrow{d} s'$ then $s' \in B$

ii) if $s \xrightarrow{a_1} s'$ then $s' \xrightarrow{a_2} s''$ s.t. $s'' \in B$

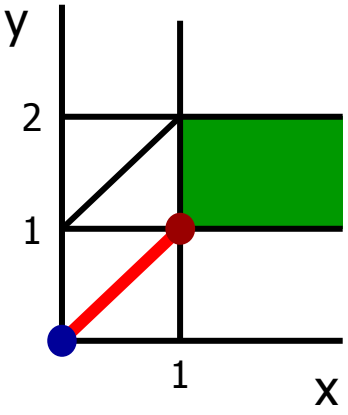
iii) if $s \xrightarrow{a_2} s'$ then $s' \xrightarrow{a_1} s''$ s.t. $s'' \in B$

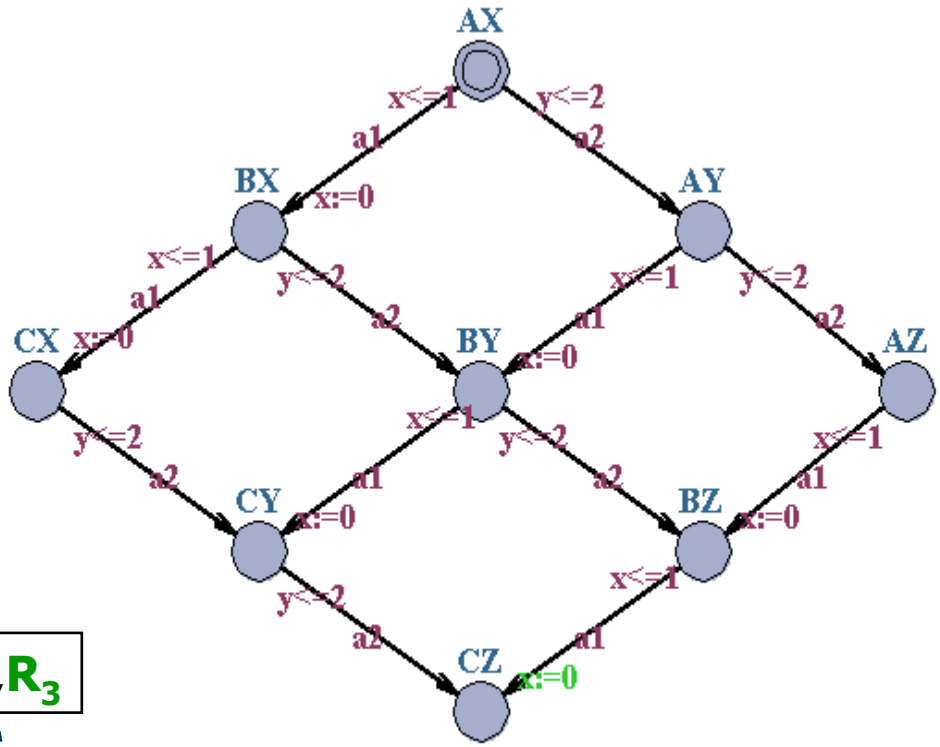
We write TB(s) whenever $s \in B$ for some timed product-bisimulation.

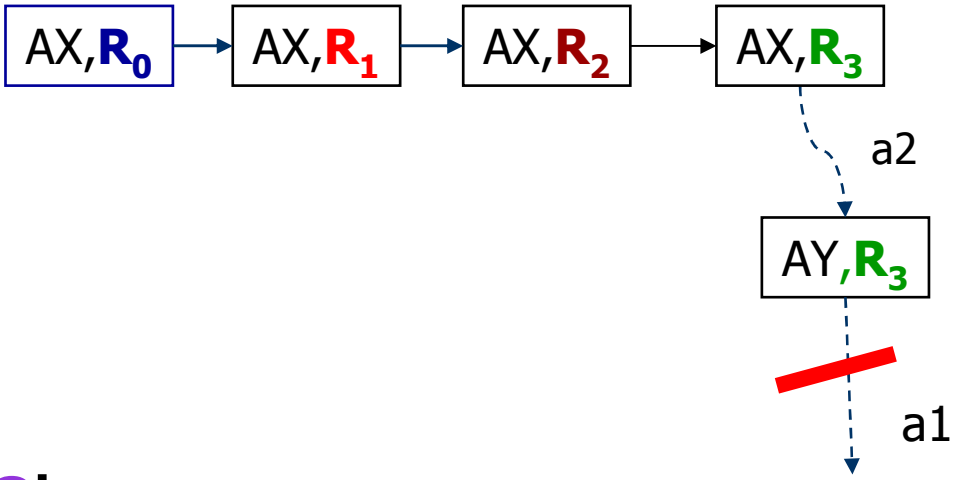
Theorem

$$TB(s) \Leftrightarrow (s_1 \approx s_2)$$

Timed Bisimulation Algorithm = Checking for TB-ness using Regions







UCb

Timed Trace Inclusion

Undecidability

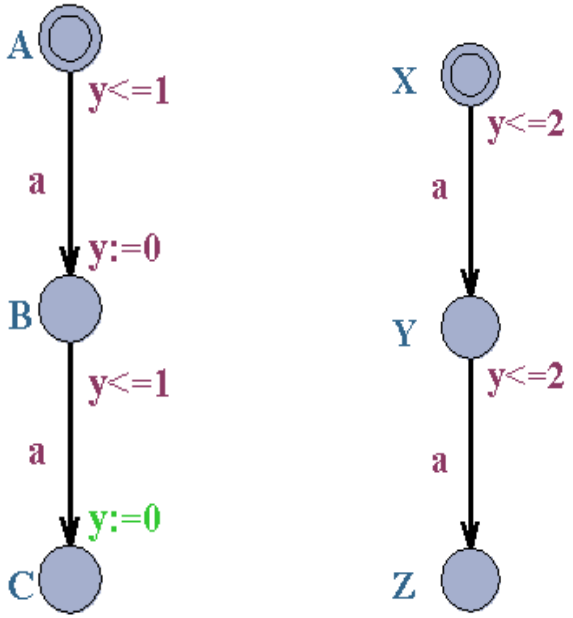
BRICS
Basic Research
in Computer Science

Timed Trace Languages

- **Timed trace**

  $(t_1,a_1),(t_2,a_2),\ ..,\ (t_k,a_k)$

  where a_i is an action and t_i 2 **R**, with t_i's non-decreasing.



$L(A)=\{\ (t_1,a)\ ,\ (t_2,a)\ :\ t_1 \cdot 1\ \text{Æ}\ t_1 \cdot t_2 \cdot t_1+1\ \}$

$L(X)=\{(t_1,a)\ ,\ (t_2,a)\ :\ t_1 \cdot t_2 \cdot 2\ \}$

Timed Trace Languages

- **Timed trace**

  $(t_1,a_1),(t_2,a_2),\ ..,\ (t_k,a_k)$

  where a_i is an action and t_i 2 **R**, with t_i's strictly increasing.

- **PROPOSITIONs**
  - Given a timed automaton A it is UNDECIDABLE whether the set of timed traces of A is the UNIVERSAL set.
  - Given two timed automata A and B it is UNDECIDABLE whether the set of timed traces of A is INCLUDED in the set of timed traces of B.

Two-counter Machine

- M=($\{b_0,b_1,..,b_k\}$, C, D)

  where b_i's are instructions
  C and D are counters ranging over **N**.
  Initially both C and D are 0.

- Instructions (i<k):
  - Increment b_j: C:=C+1 ; goto b_l
  - Decrement b_j: if C≠0 then C:=C-1; goto b_l else goto b_m
- b_k represents termination

Two-counter Machine

- M=($\{b_0,b_1,..,b_k\}$, C, D)

- Configuration of M:
  (b_i , c , d)
  where c and d are values of C and D.
- Computation of M is a "valid" sequence of configurations starting with $(b_0,0,0)$ and ending with $(b_k,\_,\_)$.

- **PROPOSITION**
  Deciding whether a two-counter machine has a (halting) computation is UNDECIDABLE.

UCb

Timed Trace Language for Two-counter Machine

- Let M be a two-counter machine.
  We define L(M) to be the set of timed traces over

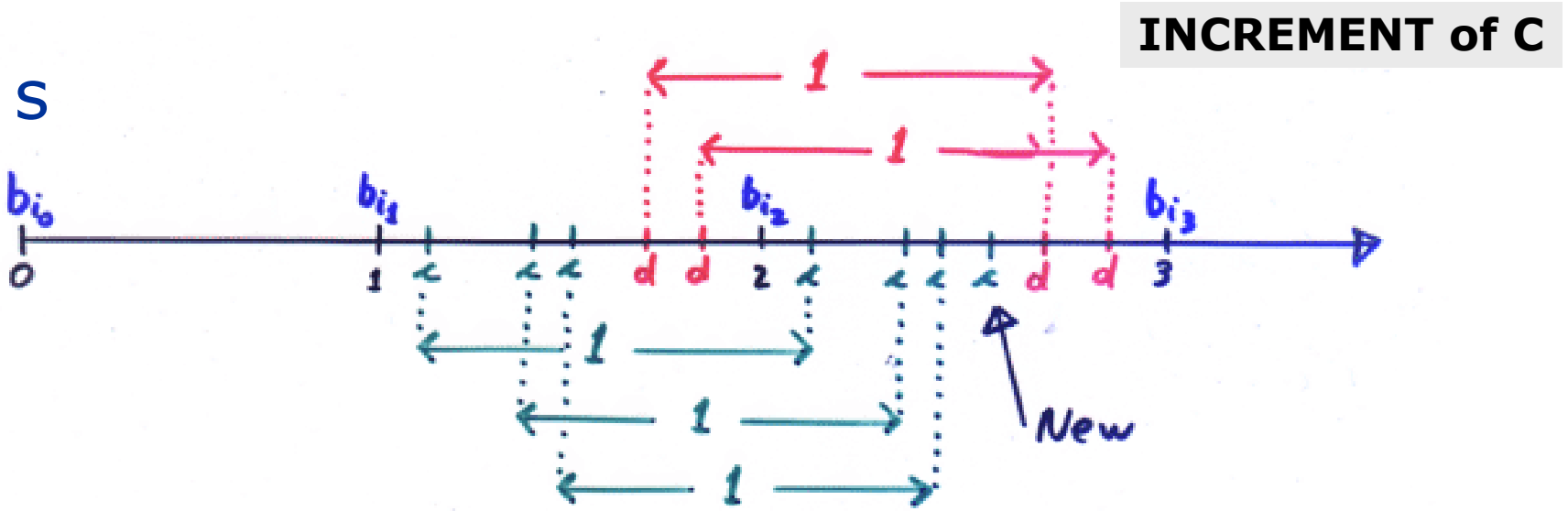
$$\Sigma=\{b_0,..,b_k,c,d\}$$

  such that whenever

$$(b_{i0},c_0,d_0)(b_{i1},c_1,d_1).....(b_{in},c_n,d_n)$$

  is a computation of M then the timed trace s is in L(M) where:
  - Untime(s)= $b_{i0}c^{c0}d^{d0}b_{i1}c^{c1}d^{d1}$ ... $b_{in}c^{cn}d^{dn}$
  - Time(b_j) = j
  - "proper matching of c's and d's"

Proper Matching

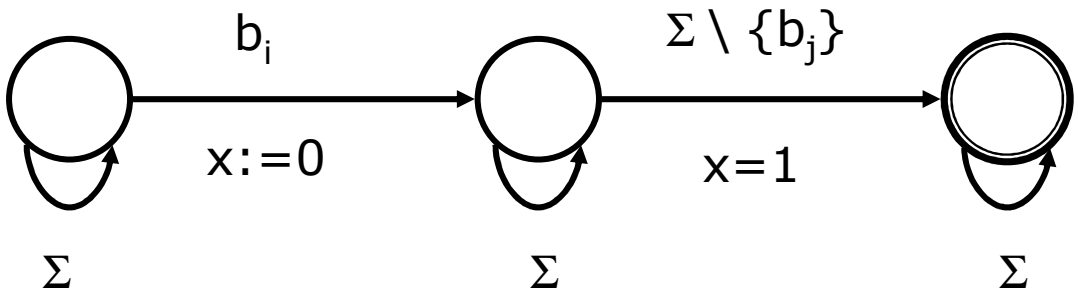


- Clearly M has a (halting) computation iff L(M)≠Ø
- One can show that $L(M)^C$ can be captured by a Timed Automaton (a union of several small ones).

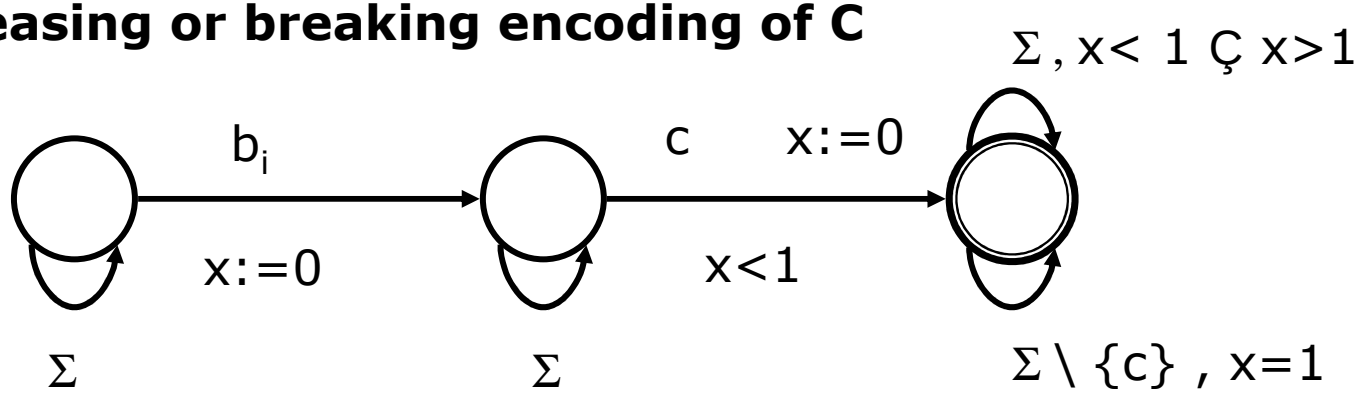
Example Automata

Violation of b_i : D:=D+1 ; goto b_j

1) Jumping to another instruction than b_j:



2) Decreasing or breaking encoding of C



UCb

The UPPAAL Verification Engine

Kim Guldstrand Larsen

AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Overview

- Train Crossing
  - Full Modeling & Specification Formalism
  - Schedulability Analysis

- UPPAAL Verification Engine
  - Symbolic On-the-fly Exploration
  - Zones & DBMs
  - CDDs

- Verification Options
  - Over- / Under Approximations
  - Storage Strategies

UCb

Timed Automata in UPPAAL
Train Crossing

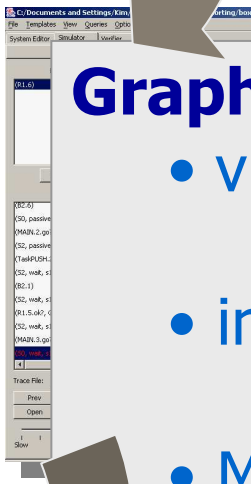
BRICS
Basic Research
in Computer Science

UPPAAL

Tool Environment for modeling, simulation, verification, optimization & synthesis of real-time systems

Graphical Design Tool

- timed automata
- clocks
- communication
- datatypes & functions
- cost variables
- uncontr. behaviour



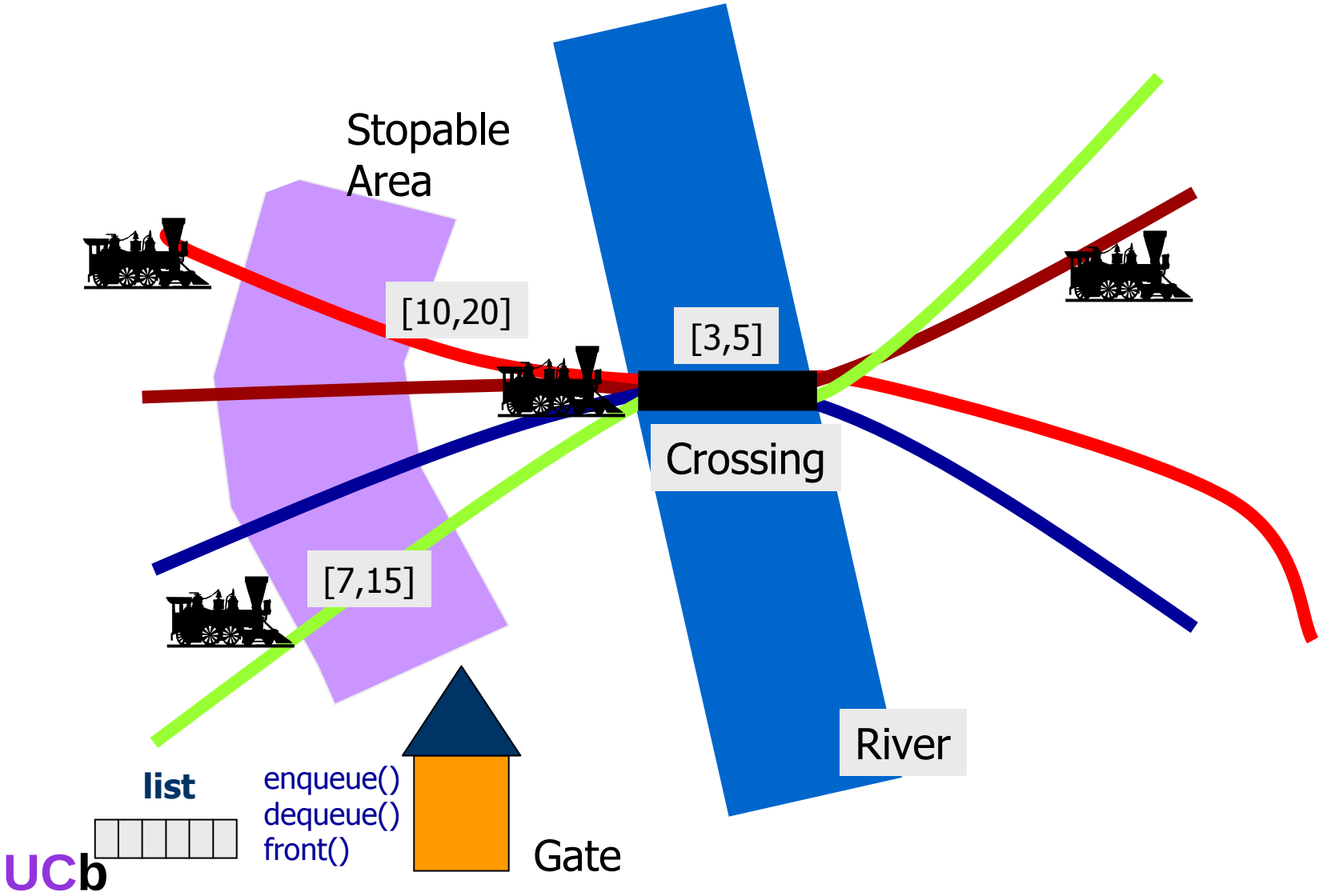
Graphical Simulator

- visualization and recording
- inexpensive fault detect.
- MSCs

Verifier

- exhaustive & automatic checking of requirements
- diagnostic traces
- optimal scheduling
- controller synthesis

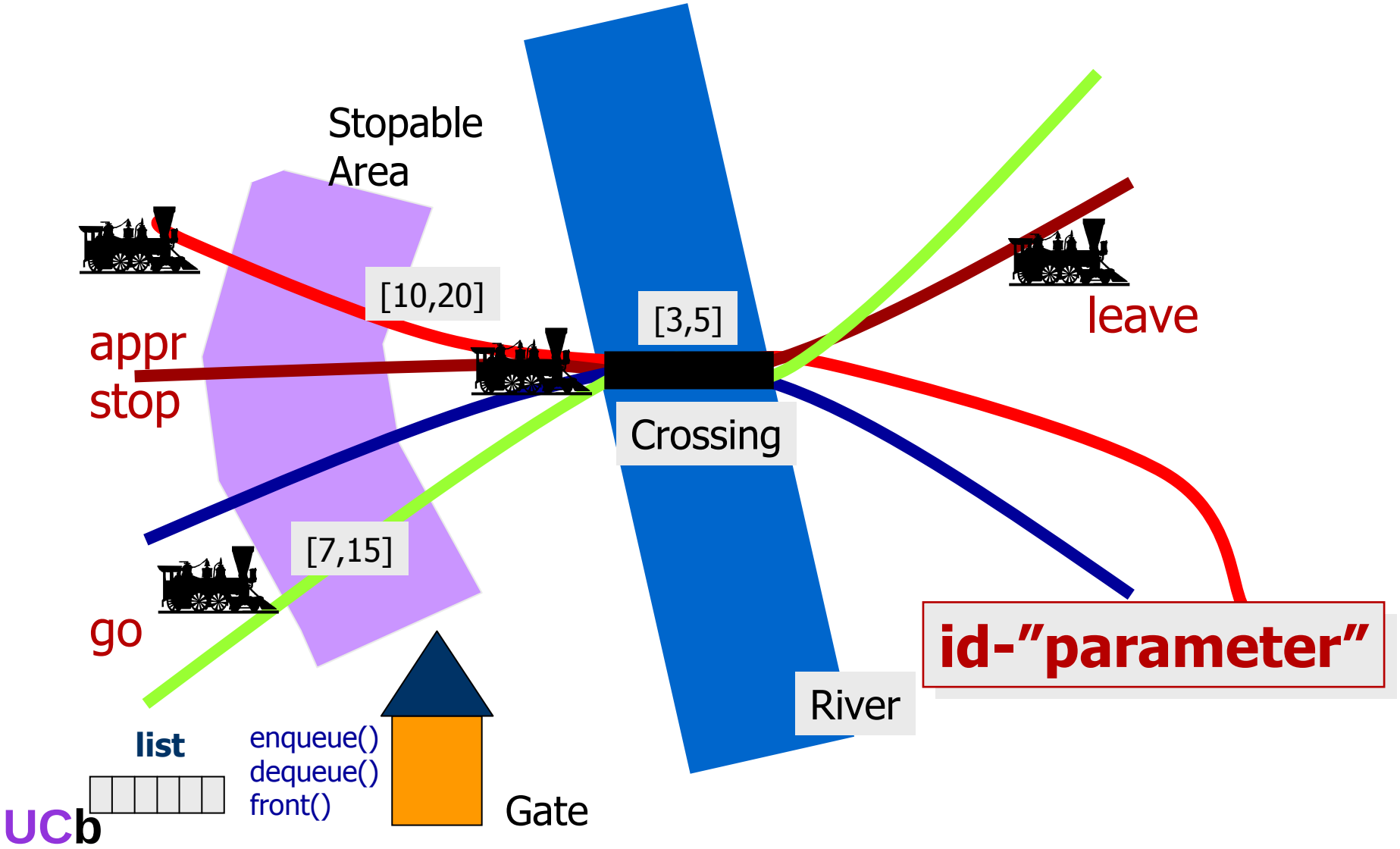
Train Crossing



UCb

Train Crossing

Communication via channels!



UCb

Queries : Specification Language

BRICS
Basic Research
in Computer Science

Logical Specifications

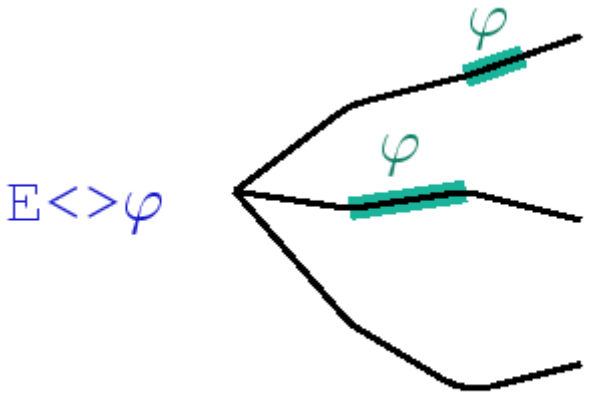
- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\leq t} Q$

The expressions *P* and *Q* must be type safe, side effect free, and evaluate to a boolean.

Only references to integer variables, constants, clocks, and locations are allowed (and arrays of these).

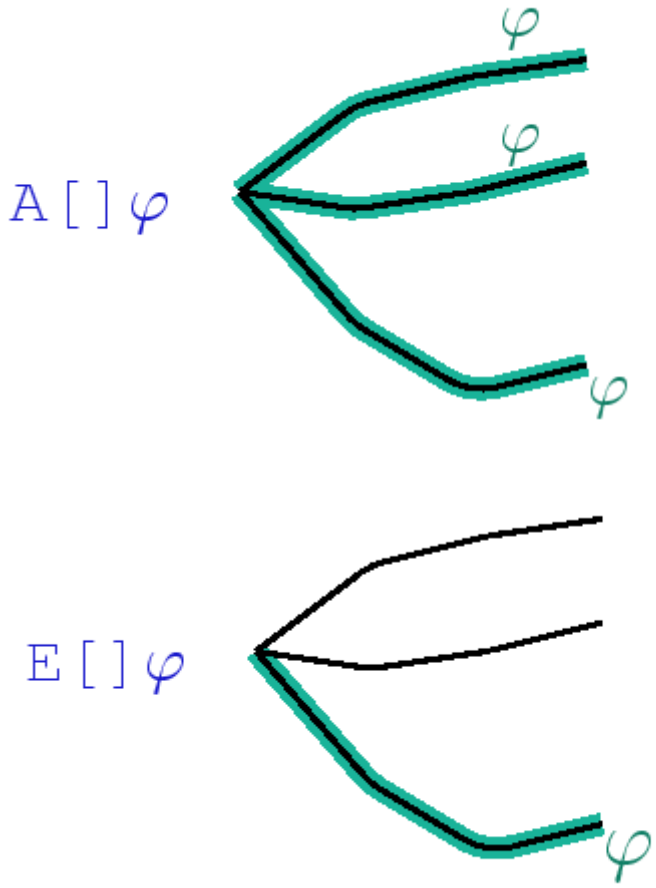
Logical Specifications

- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\leq t} Q$



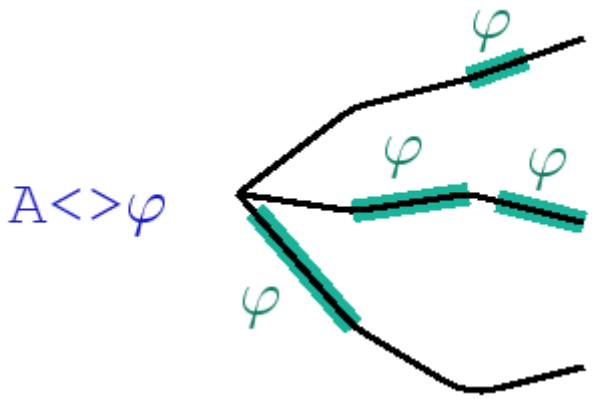
Logical Specifications

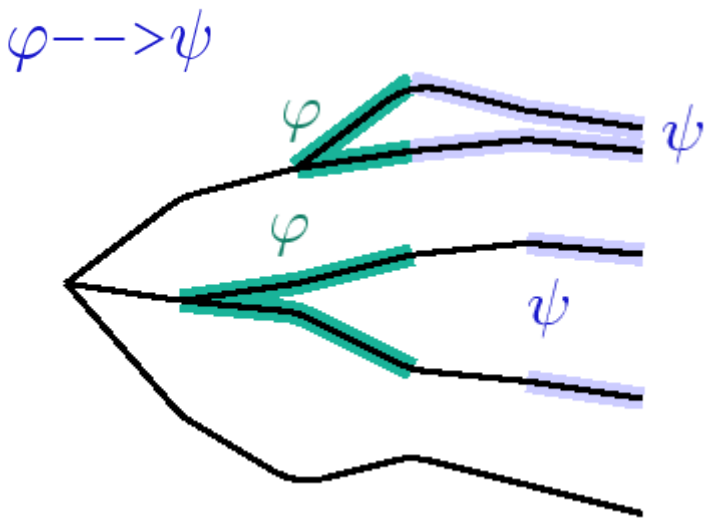
- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\leq t} Q$



Logical Specifications

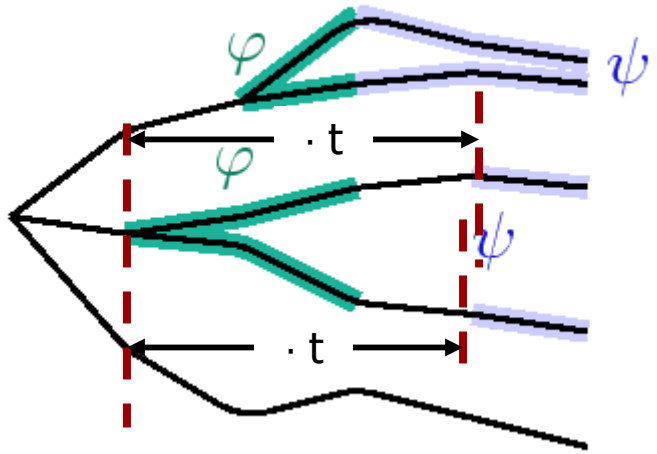
- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\leq t} Q$





Logical Specifications

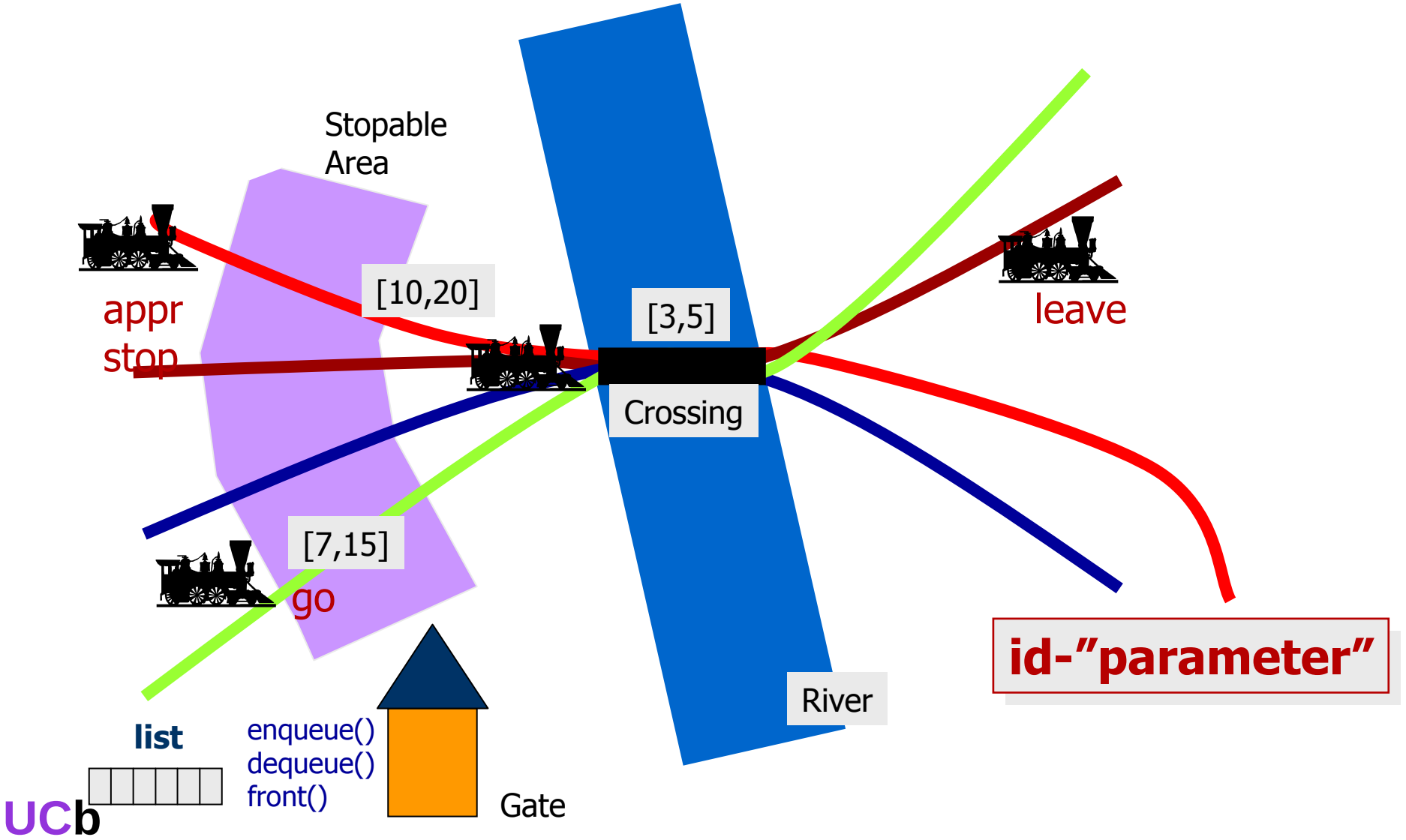
- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\cdot t} Q$



UCb

Train Crossing

Communication via channels!



UCb

Case-Studies: Controllers

- Gearbox Controller [TACAS'98]
- Bang & Olufsen Power Controller [RTPS'99,FTRTFT'2k]
- SIDMAR Steel Production Plant [RTCSA'99, DSVV'2k]
- Real-Time RCX Control-Programs [ECRTS'2k]
- Experimental Batch Plant (2000)
- RCX Production Cell (2000)
- Terma, Verification of Memory Management for Radar (2001)
- Scheduling Lacquer Production (2005)
- Memory Arbiter Synthesis and Verification for a Radar Memory Interface Card [NJC'05]

Case Studies: Protocols

- Philips Audio Protocol [HS'95, CAV'95, RTSS'95, CAV'96]
- Collision-Avoidance Protocol [SPIN'95]
- Bounded Retransmission Protocol [TACAS'97]
- Bang & Olufsen Audio/Video Protocol [RTSS'97]
- TDMA Protocol [PRFTS'97]
- Lip-Synchronization Protocol [FMICS'97]
- Multimedia Streams [DSVIS'98]
- ATM ABR Protocol [CAV'99]
- Leader Election for Mobile Ad Hoc Networks [Charme05]
- ABB Fieldbus Protocol [ECRTS'2k]
- IEEE 1394 Firewire Root Contention (2000)
- Distributed Agreement Protocol [Formats05]

UCb

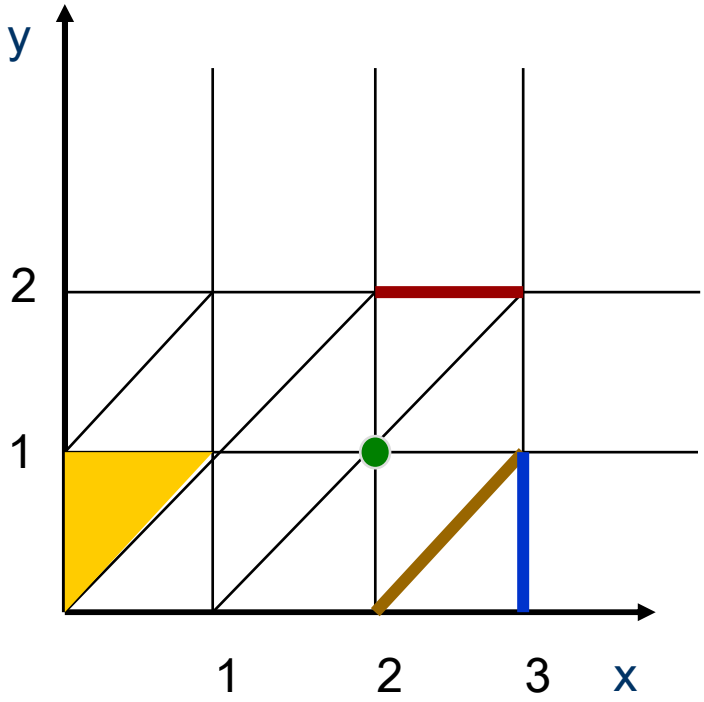
Zones & DBMs

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Regions

Finite Partitioning of State Space



Time Abstracted Bisimulation
Equivalence classes (i.e. a *region*)
in fact there is only a *finite* number of regions!!

Theorem
The number of regions is $n! \cdot 2^n \cdot \prod_{x \in C}(2c_x + 2)$.

0<x<1 Æ 0<y<1 Æ y-x>0

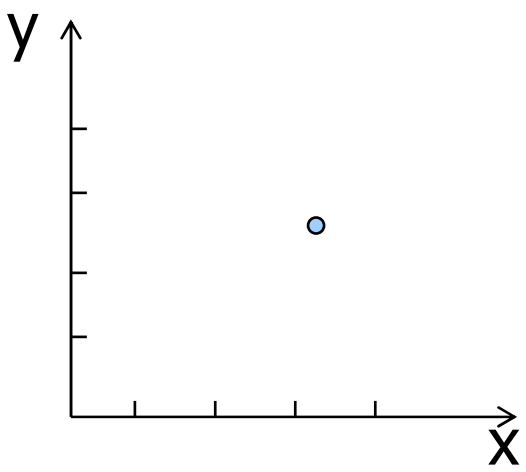
UCb

Zones

From infinite to finite

State

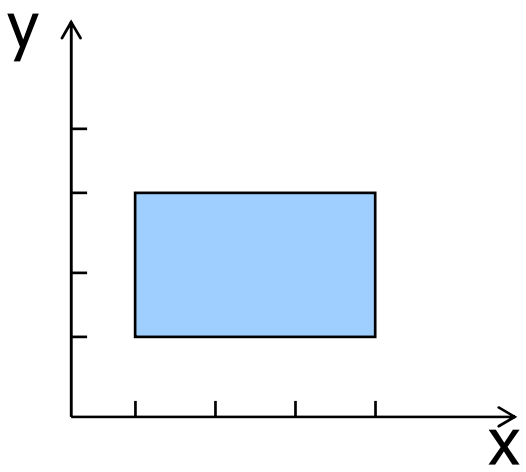
(n, x=3.2, y=2.5)



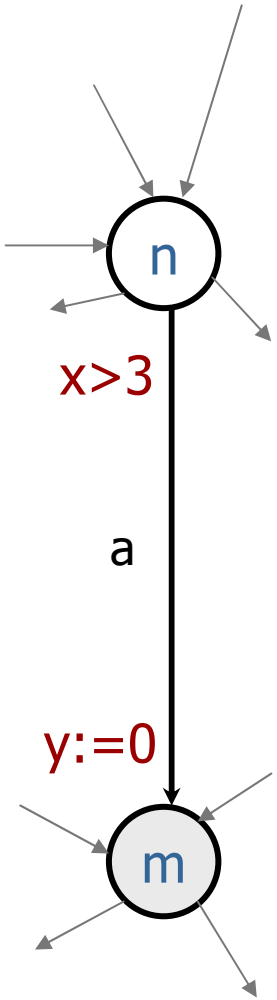
Symbolic state (set)

(n, 1·x·4, 1·y· 3)

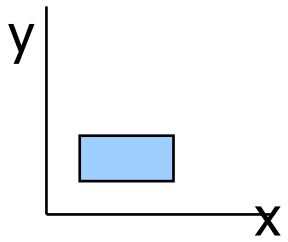
Zone:
conjunction of
x-y<=n, x<=>n



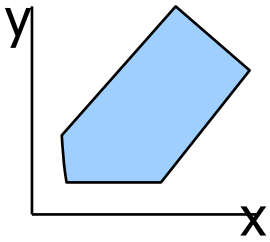
Symbolic Transitions



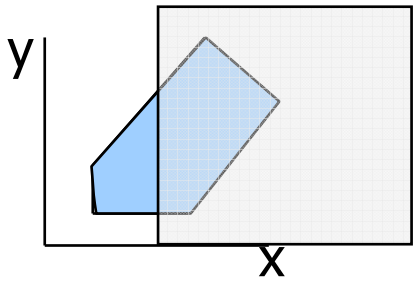
1<=x<=4
1<=y<=3



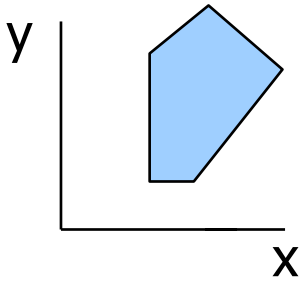
delays to



1<=x, 1<=y
-2<=x-y<=3



conjuncts to



3<x, 1<=y
-2<=x-y<=3

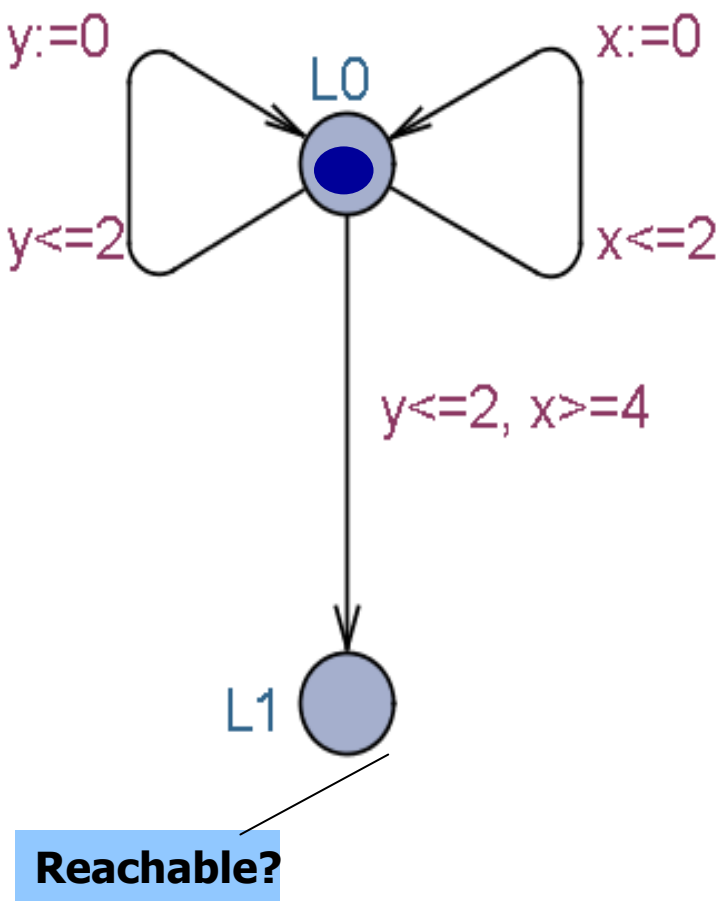
projects to

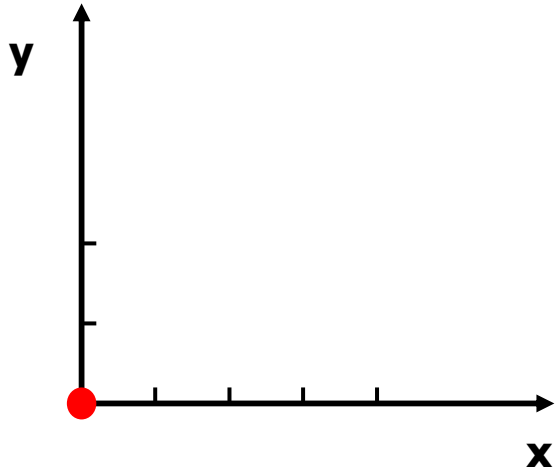
3<x, y=0

Thus (n,1<=x<=4,1<=y<=3) =a => (m,3<x, y=0)

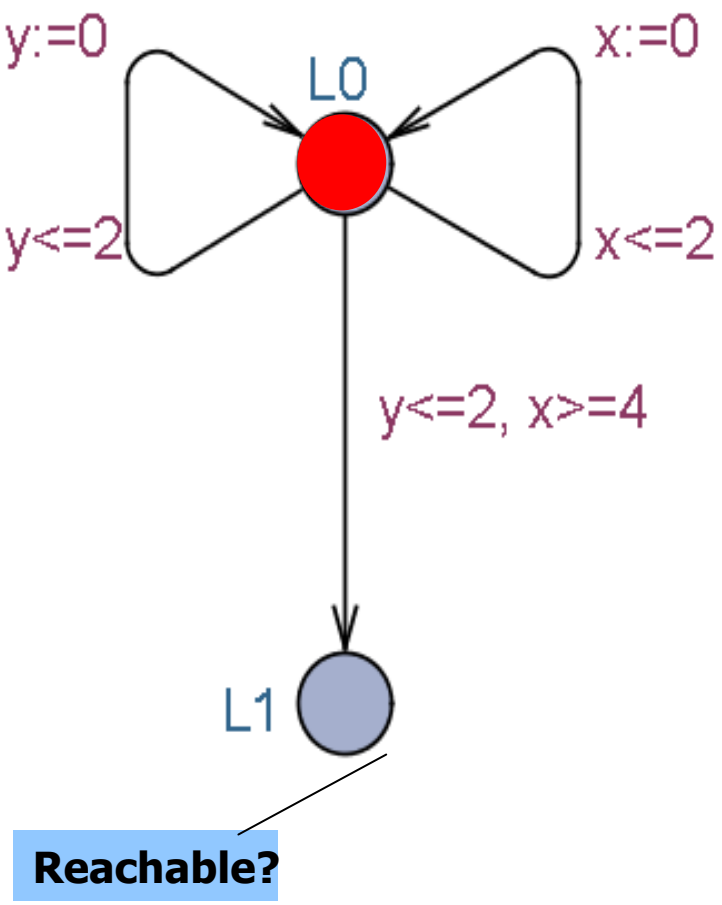
UCb

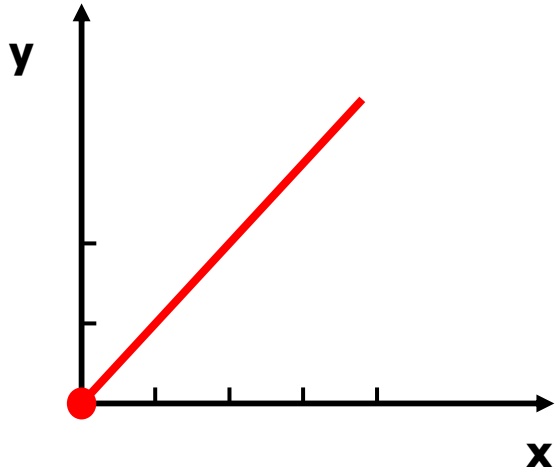
Symbolic Exploration





Symbolic Exploration

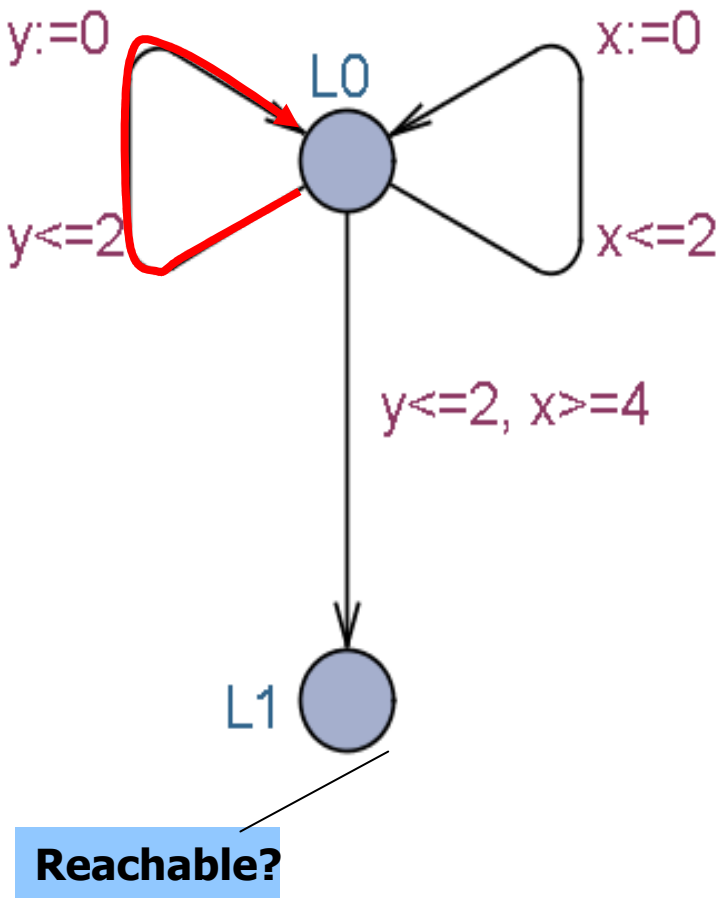


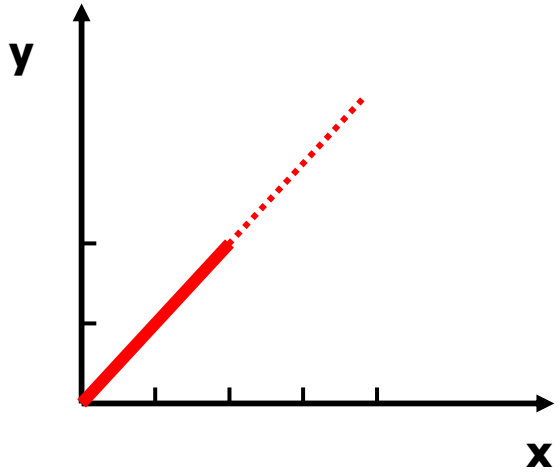


Delay

UCb

Symbolic Exploration

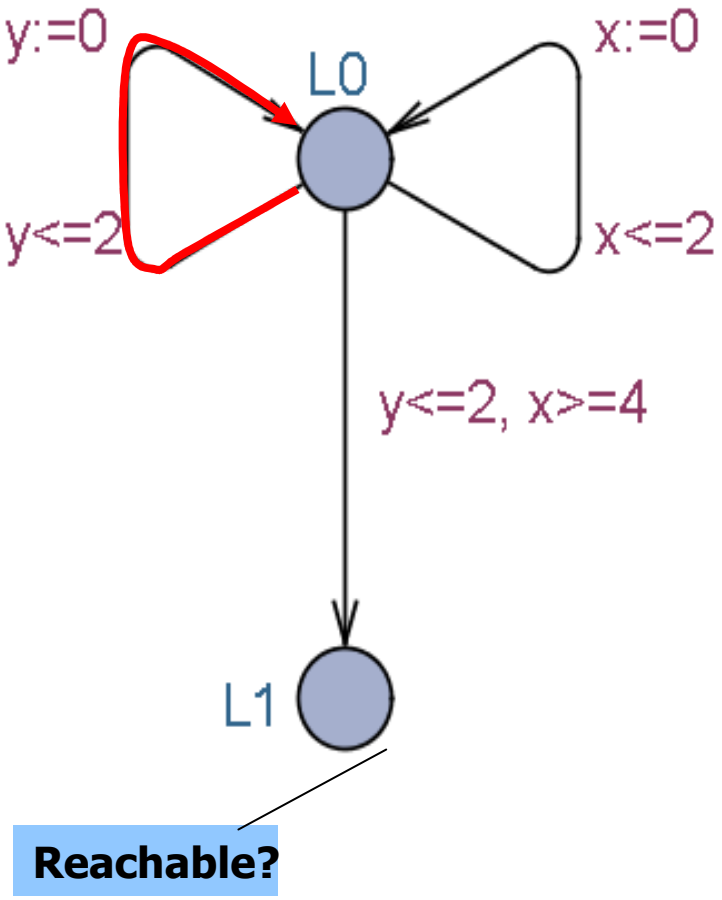


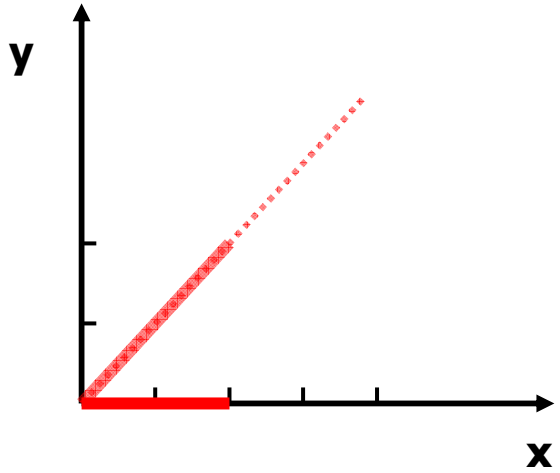


Left

UCb

Symbolic Exploration

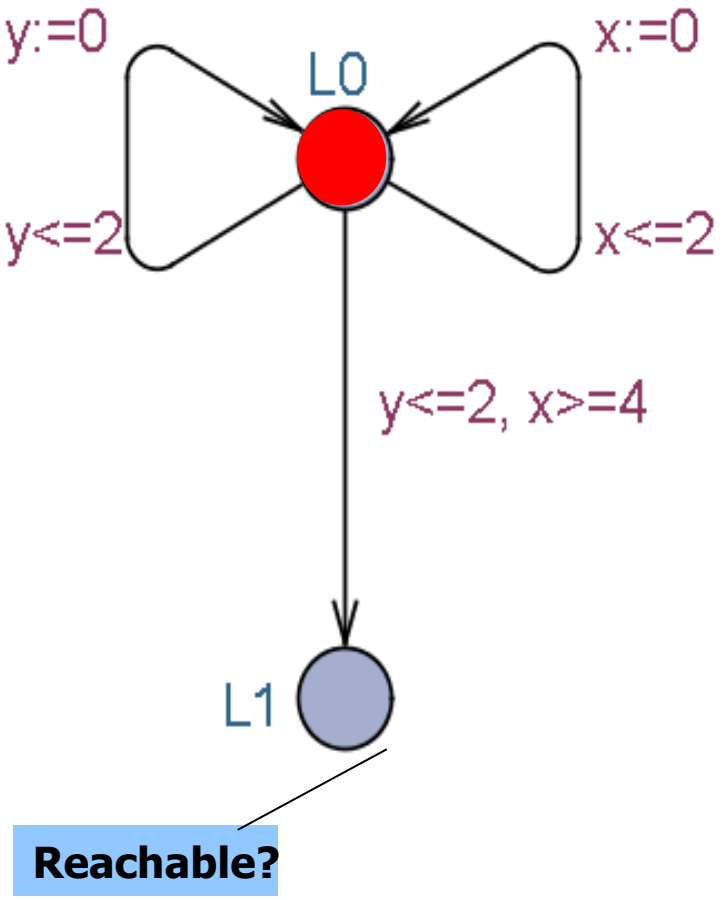


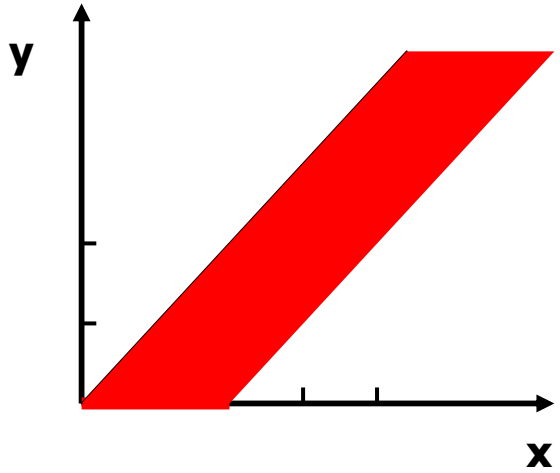


Left

UCb

Symbolic Exploration

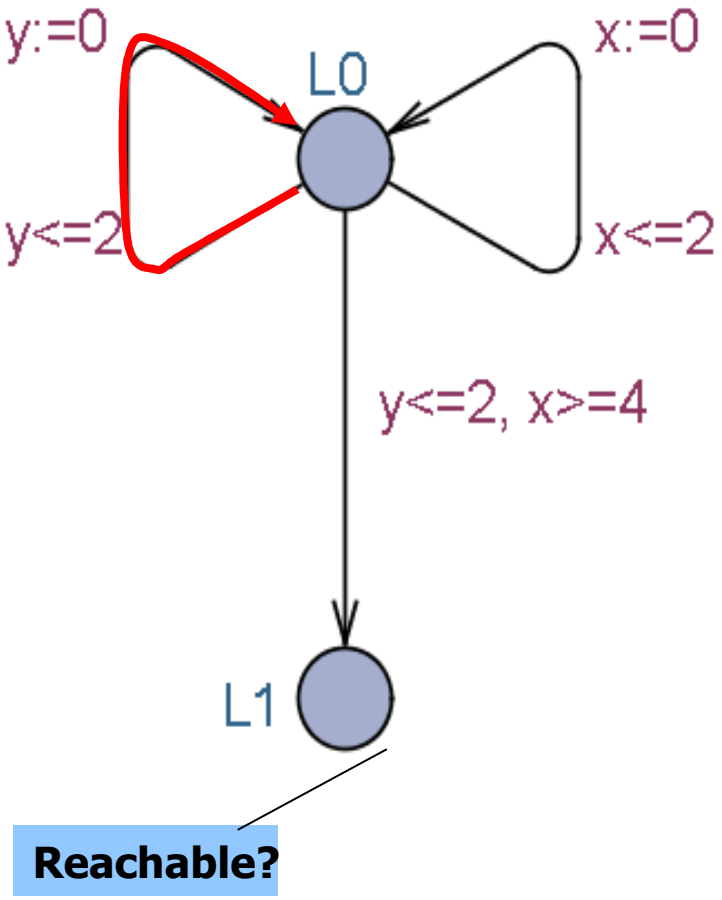


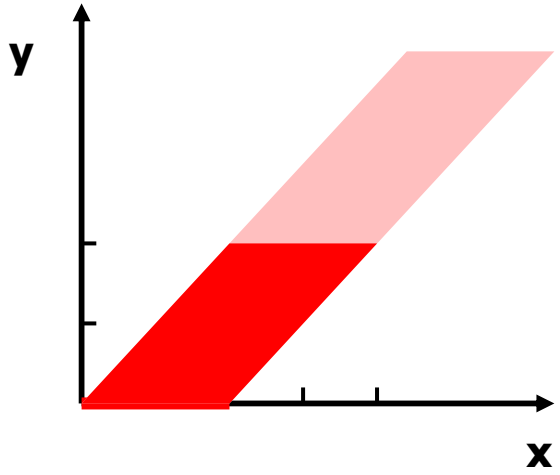


Delay

UCb

Symbolic Exploration

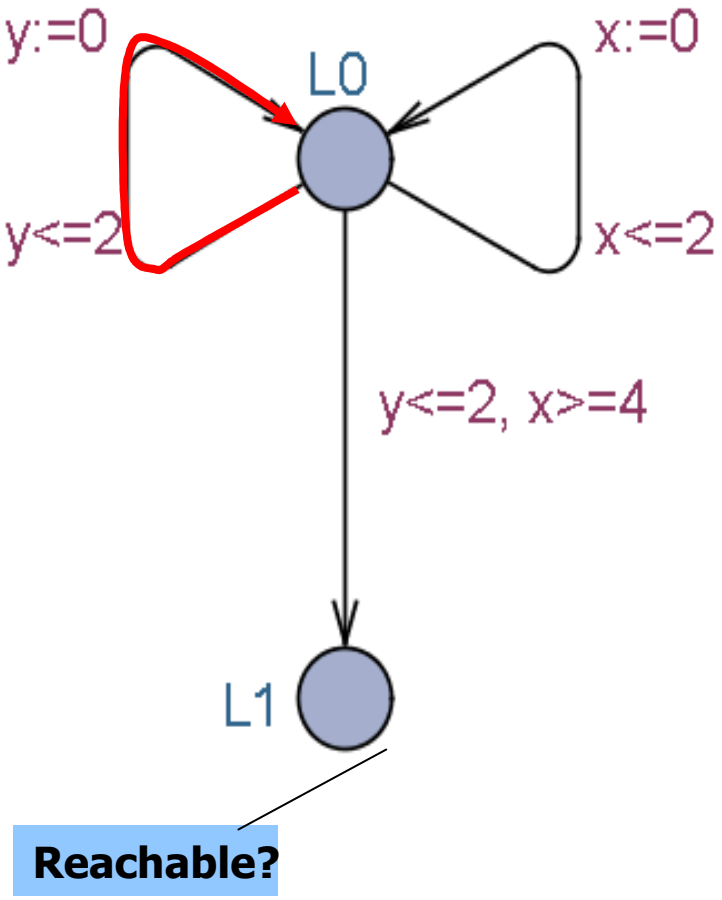


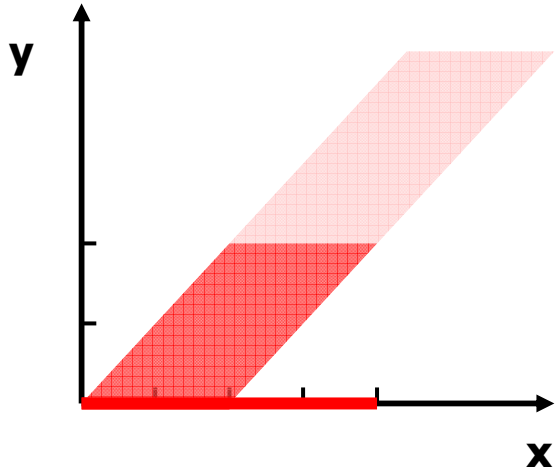


Left

UCb

Symbolic Exploration

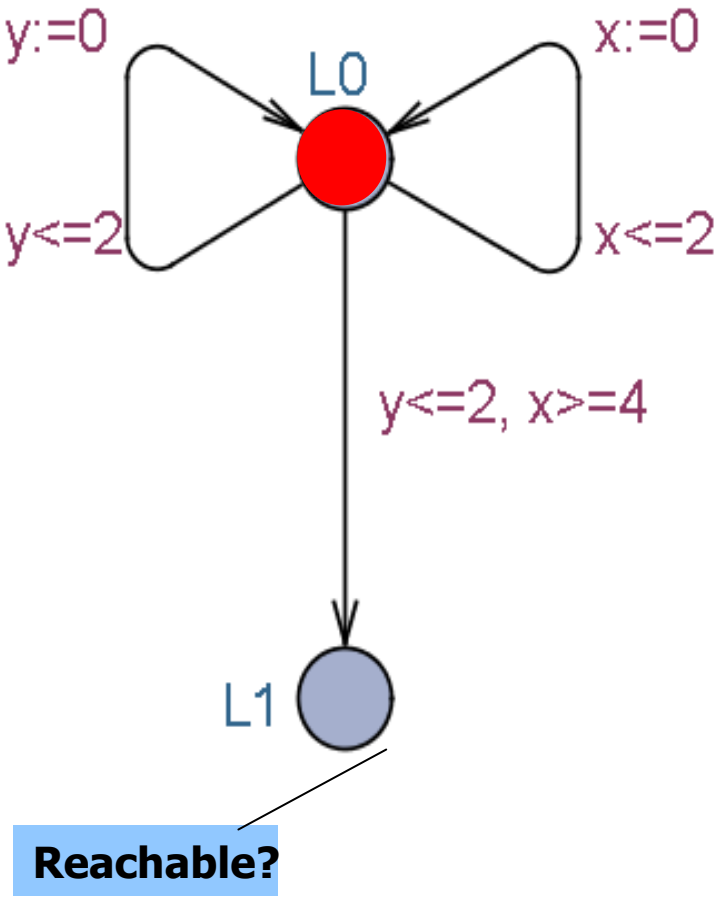


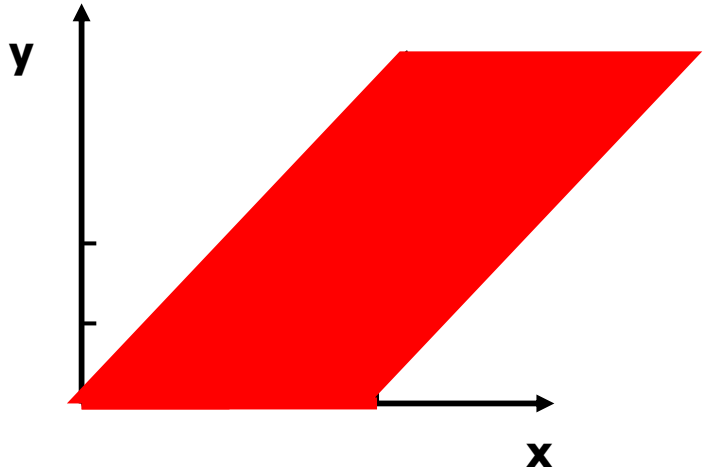


Left

UCb

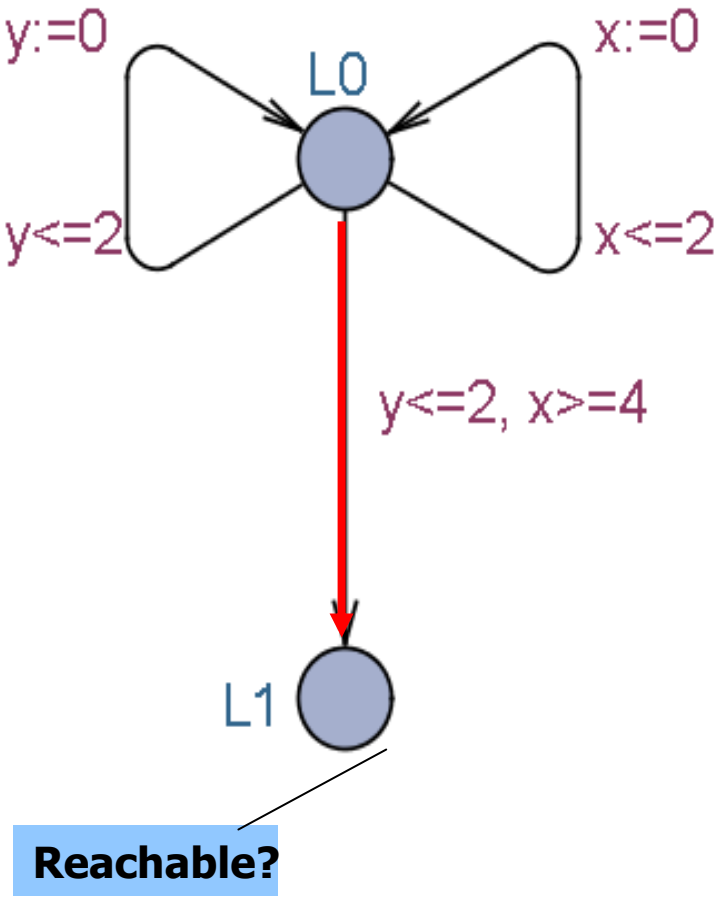
Symbolic Exploration

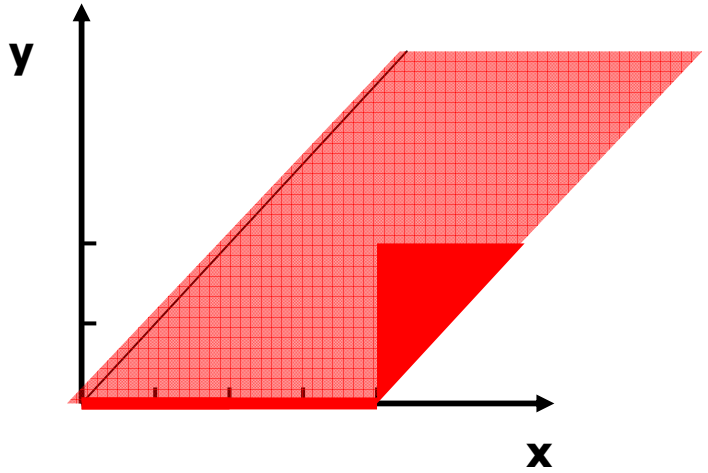




Delay

Symbolic Exploration

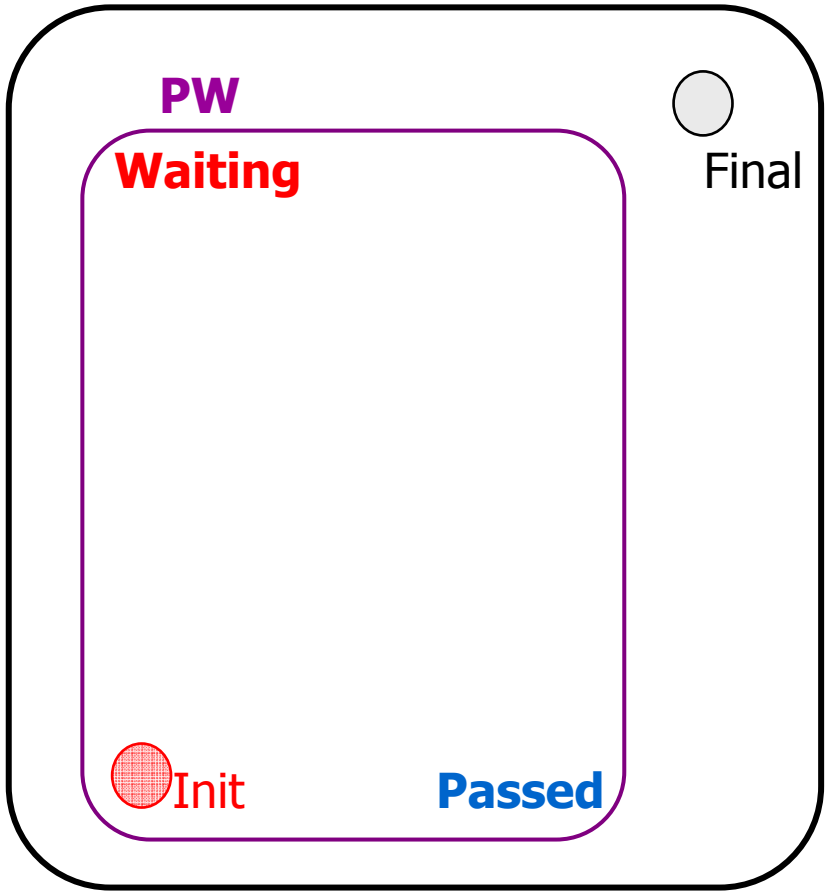




Down

Forward Reachability

Init -> Final ?



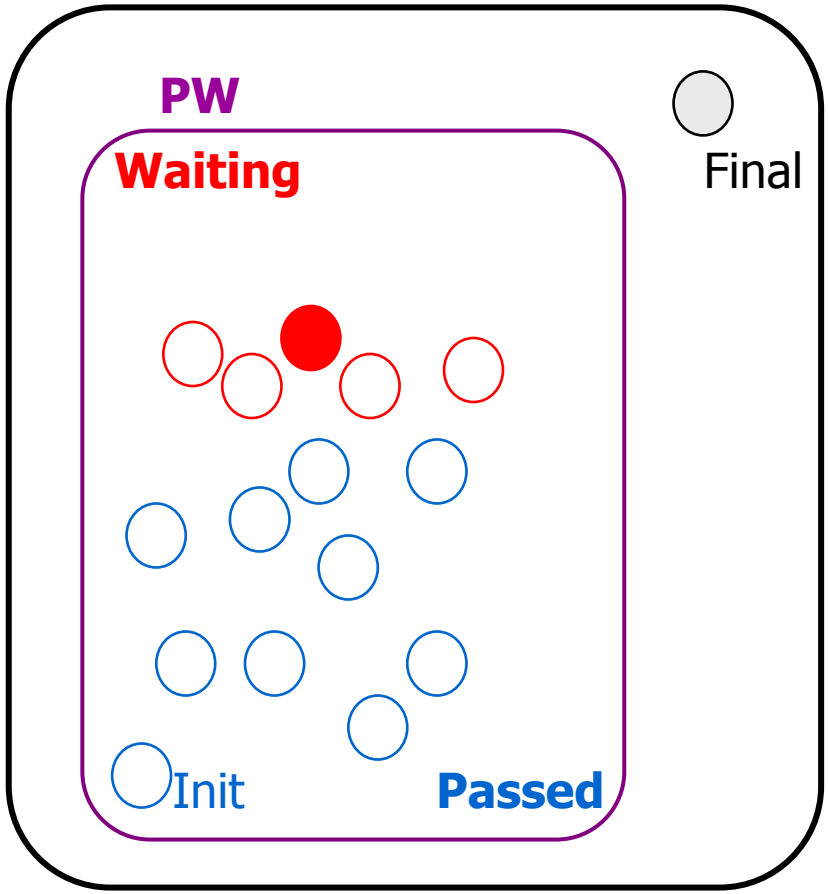
INITIAL **Passed** := Ø;
Waiting := $\{(n_0,Z_0)\}$

REPEAT

UNTIL **Waiting** = Ø
return false

Forward Reachability

Init -> Final ?



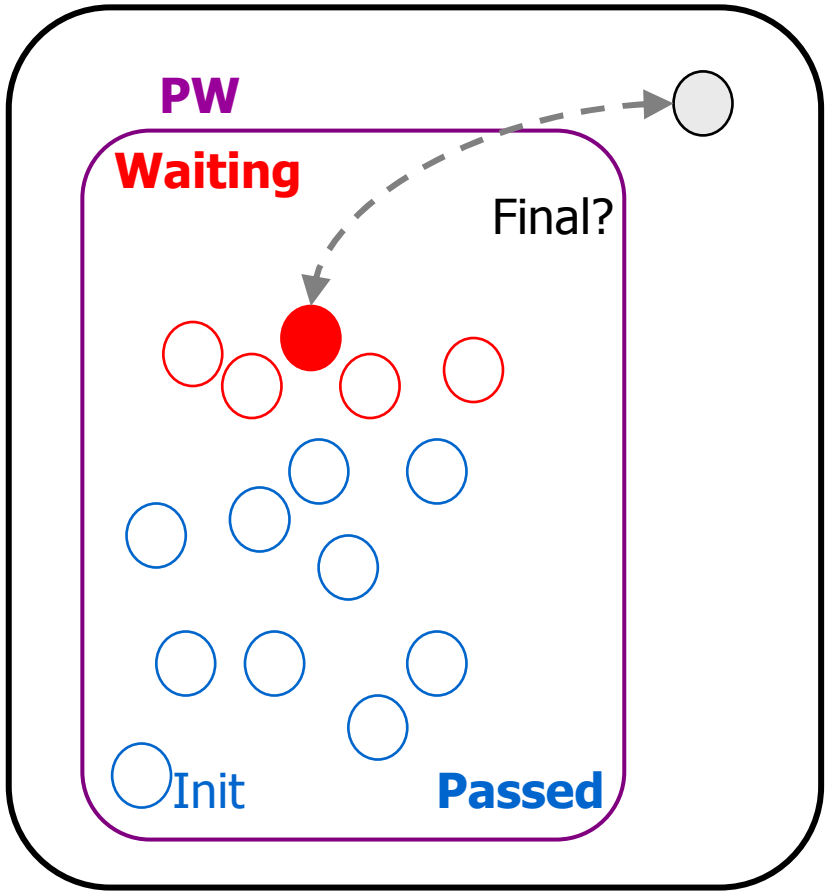
INITIAL **Passed** := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
pick **(n,Z)** in **Waiting**

UNTIL **Waiting** = Ø
return false

Forward Reachability

Init -> Final ?



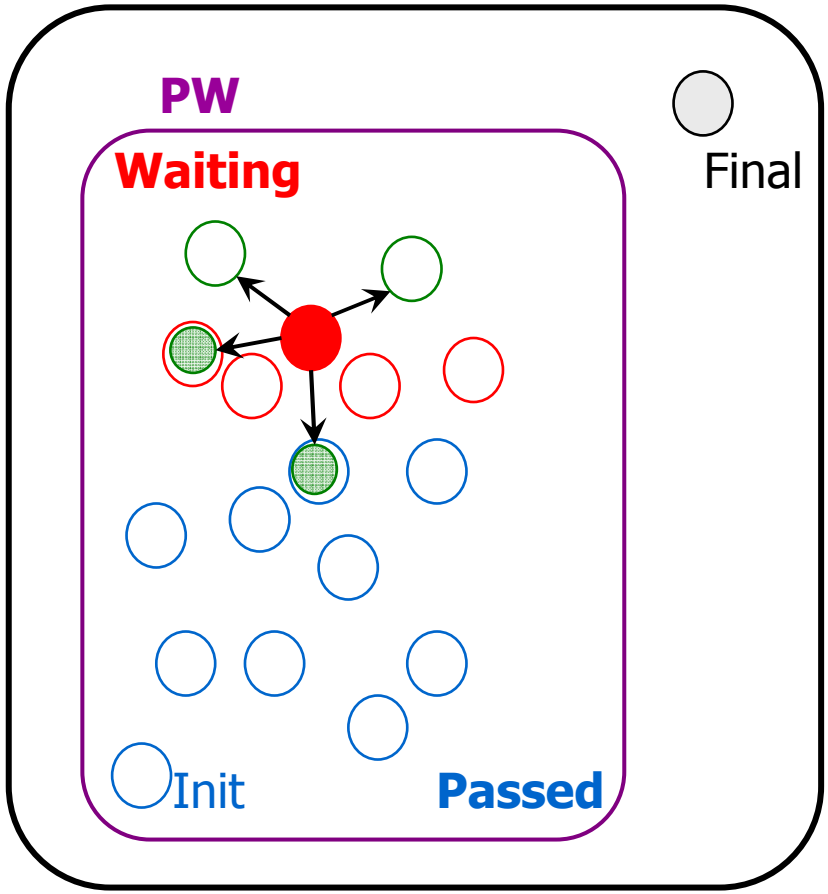
INITIAL **Passed** := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
pick **(n,Z)** in **Waiting**
if (n,Z) = Final **return true**

UNTIL **Waiting** = Ø
return false

Forward Reachability

Init -> Final ?



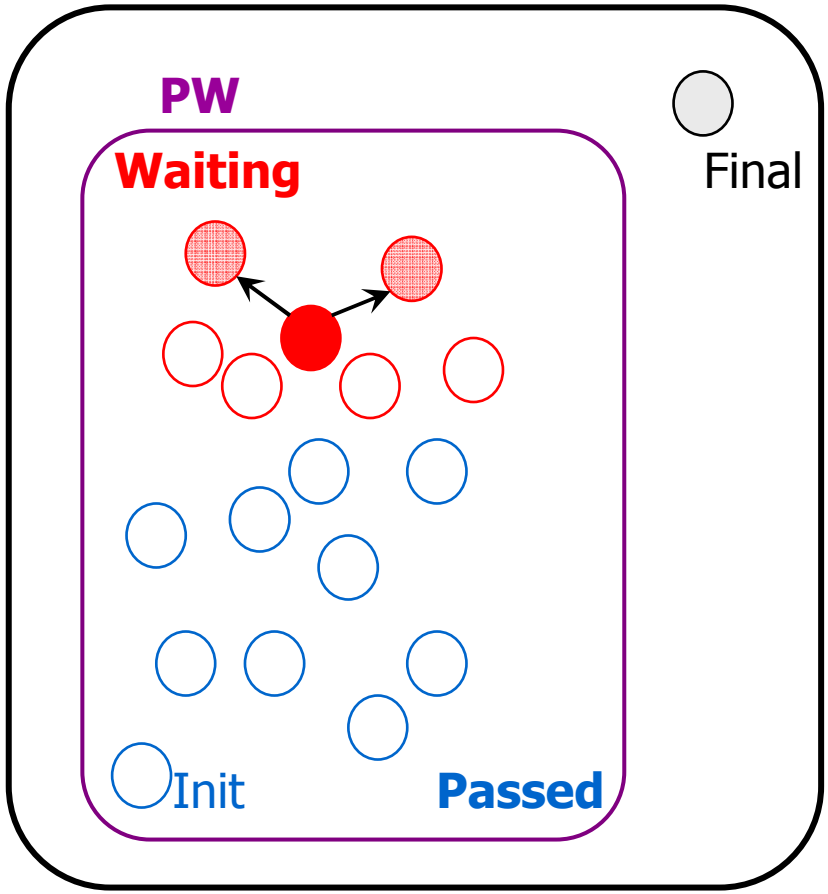
INITIAL **Passed** := Ø;
Waiting := $\{(n_0,Z_0)\}$

REPEAT
pick **(n,Z)** in **Waiting**
if (n,Z) = Final **return true**
for all (n,Z)→(**n'**,Z'):
if for some (**n'**,Z'') **Z'⊆ Z''** **continue**

UNTIL **Waiting** = Ø
return false

Forward Reachability

Init -> Final ?



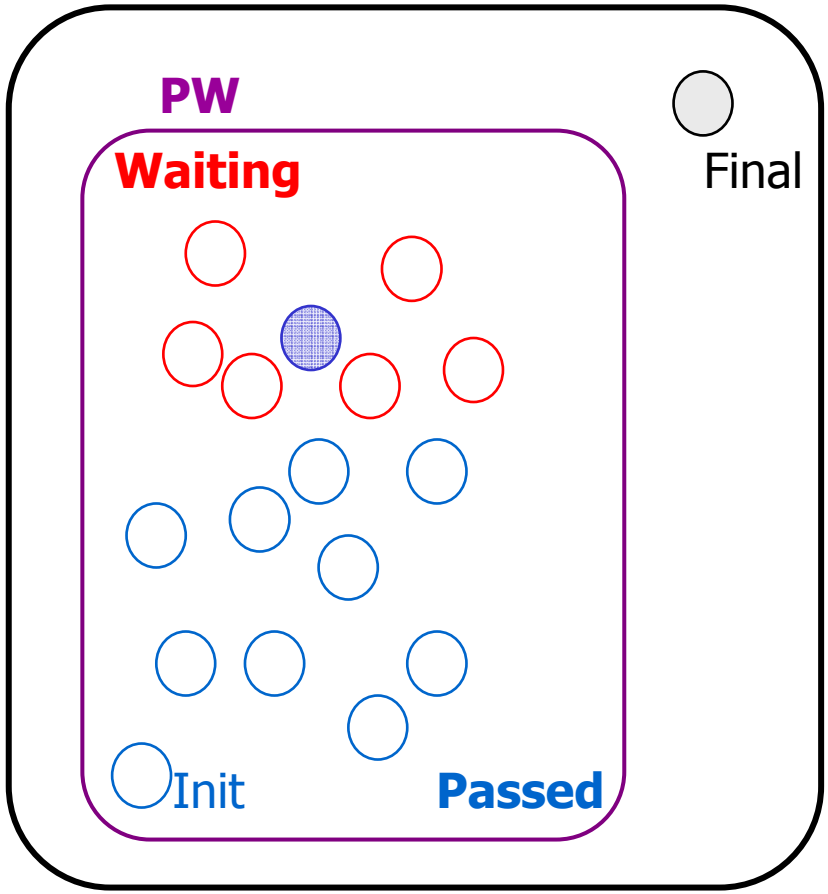
INITIAL **Passed** := Ø;
Waiting := $\{(n_0,Z_0)\}$

REPEAT
pick **(n,Z)** in **Waiting**
if (n,Z) = Final **return true**
for all (n,Z)→(**n'**,Z'):
if for some (**n'**,Z'') **Z'⊆ Z''** **continue**
else add (n',Z') to **Waiting**

UNTIL **Waiting** = Ø
return false

Forward Reachability

Init -> Final ?



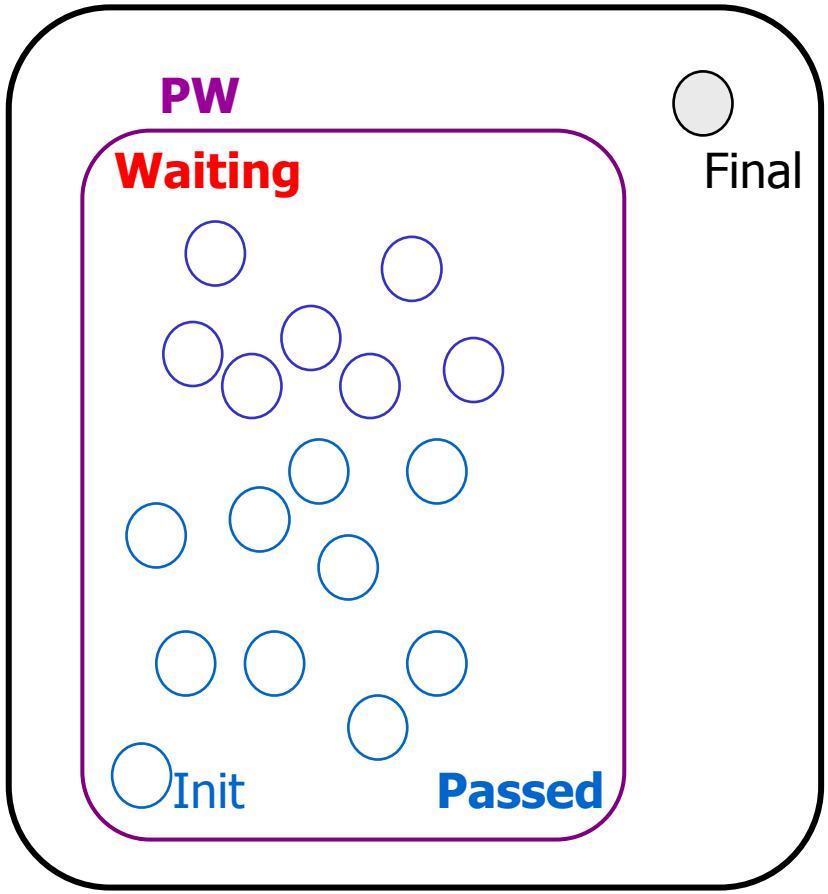
INITIAL **Passed** := Ø;
Waiting := $\{(n_0,Z_0)\}$

REPEAT
pick **(n,Z)** in **Waiting**
if (n,Z) = Final **return true**
for all (n,Z)→(**n'**,Z'):
if for some (**n'**,Z'') **Z'⊆ Z''** **continue**
else add (n',Z') to **Waiting**
move (n,Z) to **Passed**

UNTIL **Waiting** = Ø
return false

Forward Reachability

Init -> Final ?



INITIAL **Passed** := Ø;
Waiting := $\{(n_0,Z_0)\}$

REPEAT
pick **(n,Z)** in **Waiting**
if (n,Z) = Final **return true**
for all (n,Z)→(**n'**,Z'):
if for some (**n'**,Z'') **Z'⊆ Z''** **continue**
else add (n',Z') to **Waiting**
move (n,Z) to **Passed**

UNTIL **Waiting** = Ø
return false

Canonical Datastructures for Zones

Difference Bounded Matrices

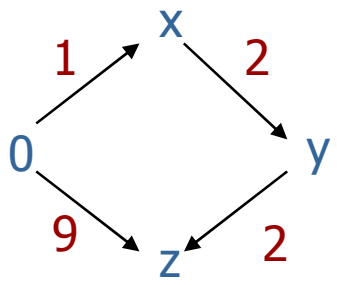
Bellman 1958, Dill 1989

Inclusion

D1

x<=1
y-x<=2
z-y<=2
z<=9

Graph

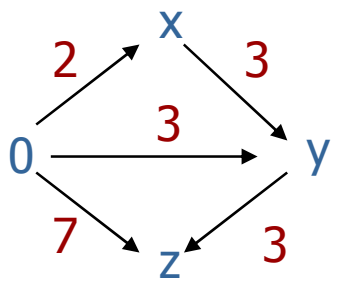


? ⊆ ?

D2

x<=2
y-x<=3
y<=3
z-y<=3
z<=7

Graph



UCb

Canonical Datastructures for Zones

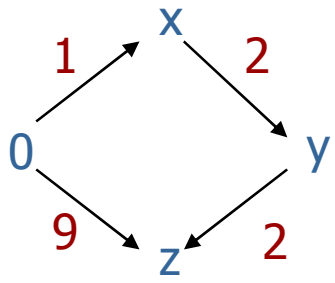
Difference Bounded Matrices

Inclusion

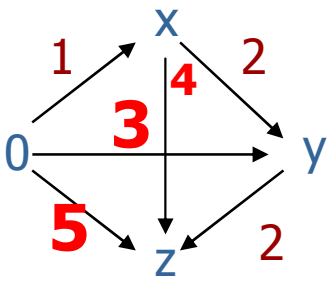
D1

x<=1
y-x<=2
z-y<=2
z<=9

Graph



Shortest Path Closure

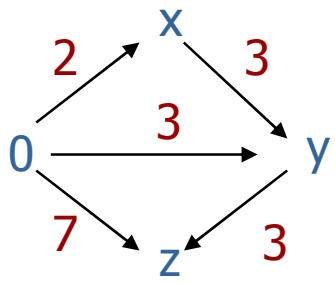


? ⊆ ?

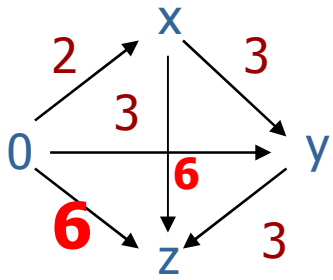
D2

x<=2
y-x<=3
y<=3
z-y<=3
z<=7

Graph



Shortest Path Closure



UCb

Canonical Datastructures for Zones

Difference Bounded Matrices

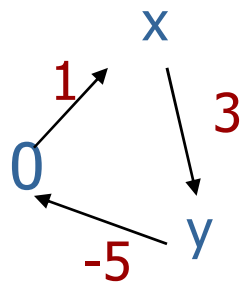
CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Emptiness

D

$x <= 1$
$y >= 5$
$y - x <= 3$

Graph



Negative Cycle
iff
empty solution set

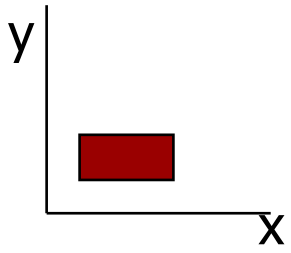
Compact

UCb

Canonical Datastructures for Zones

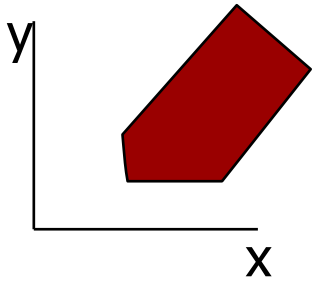
Difference Bounded Matrices

Future



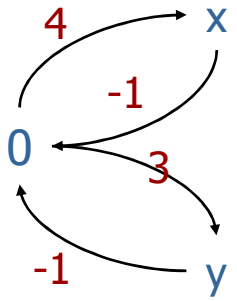
D

1<= x <=4
1<= y <=3

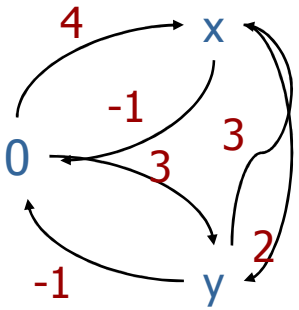


Future D

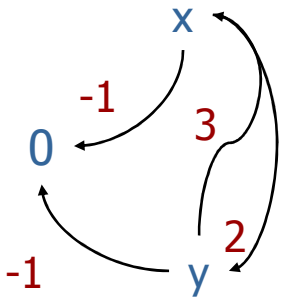
1<=x, 1<=y
-2<=x-y<=3



Shortest Path Closure



Remove upper bounds on clocks

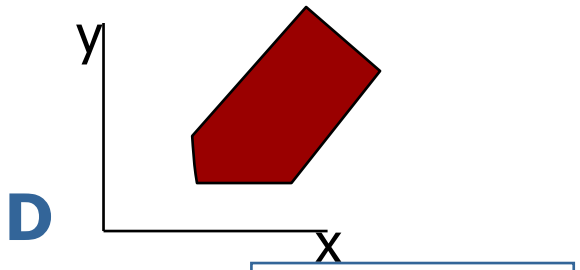


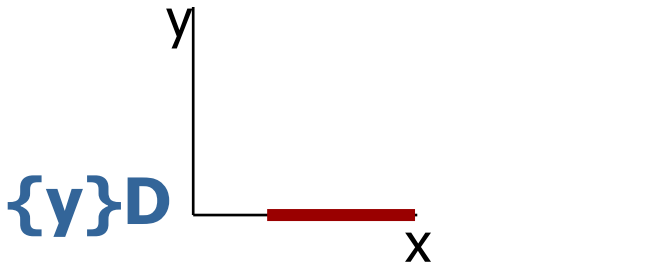
UCb

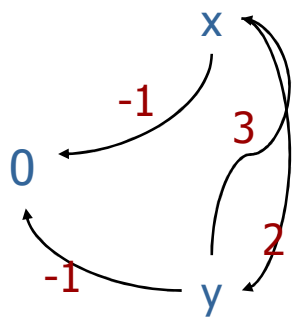
Canonical Datastructures for Zones

Difference Bounded Matrices

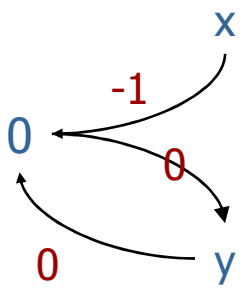
Reset







Remove all bounds involving y and set y to 0

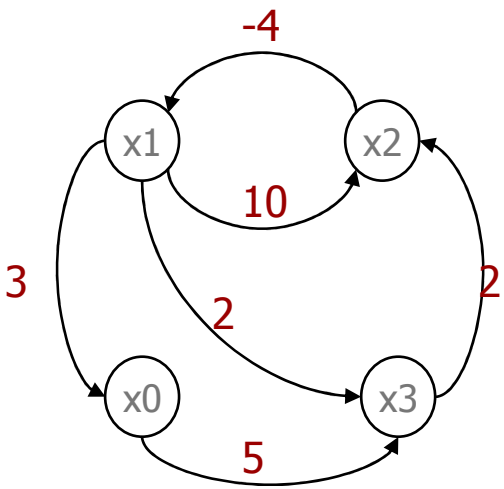


UCb

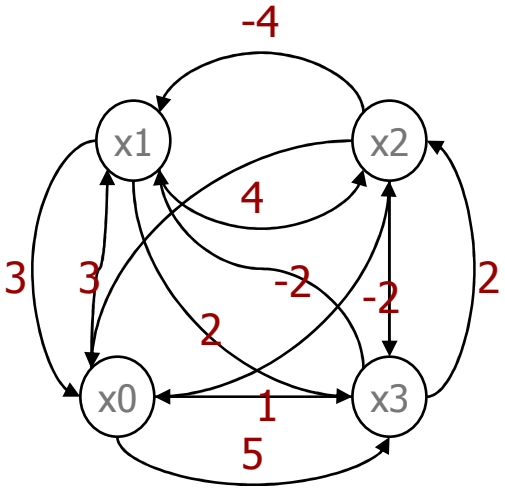
Canonical Datastructures for Zones

Difference Bounded Matrices

x1-x2<=4
x2-x1<=10
x3-x1<=2
x2-x3<=2
x0-x1<=3
x3-x0<=5



Shortest Path Closure O(n^3)



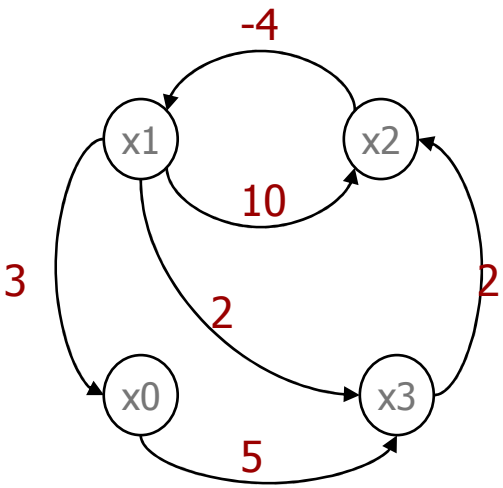
UCb

Canonical Datastructures for Zones

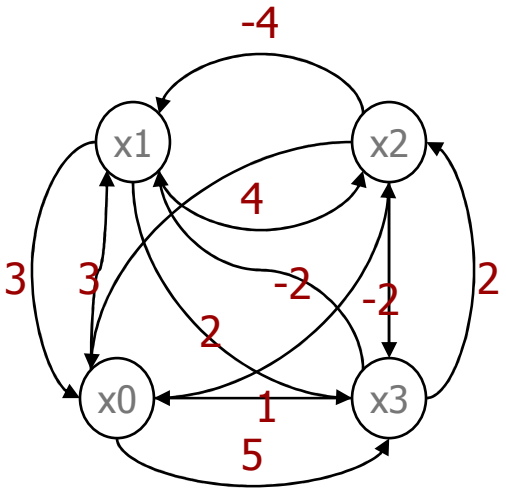
Minimal Constraint Form

RTSS 1997

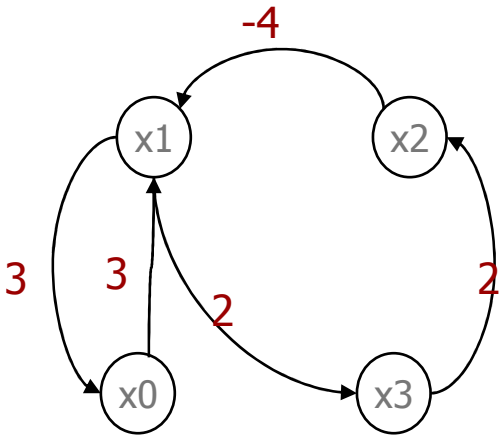
x1-x2<=4
x2-x1<=10
x3-x1<=2
x2-x3<=2
x0-x1<=3
x3-x0<=5



Shortest Path Closure O(n^3)



Shortest Path Reduction O(n^3)

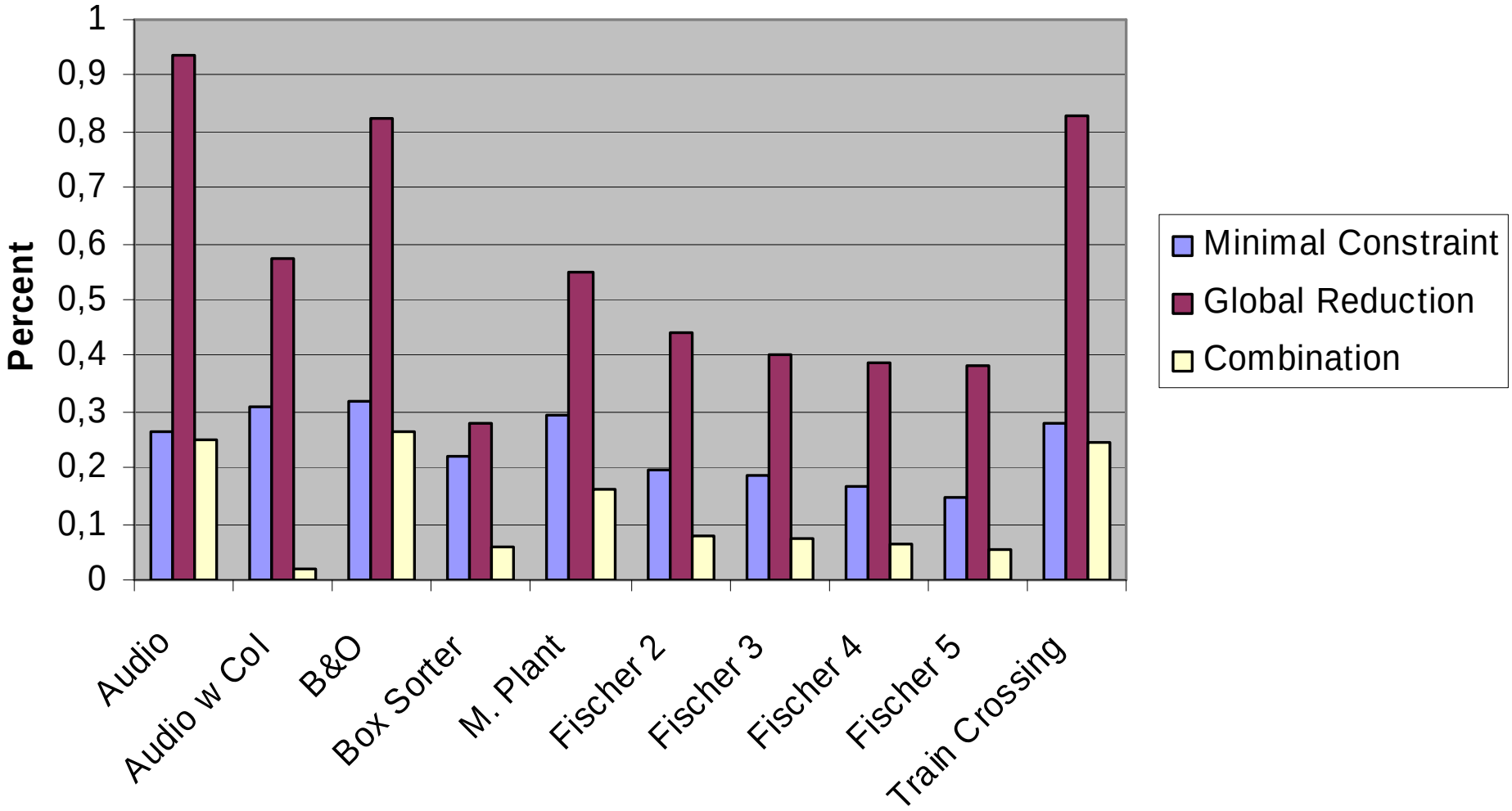


Space worst O(n^2)
practice O(n)

UCb

CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

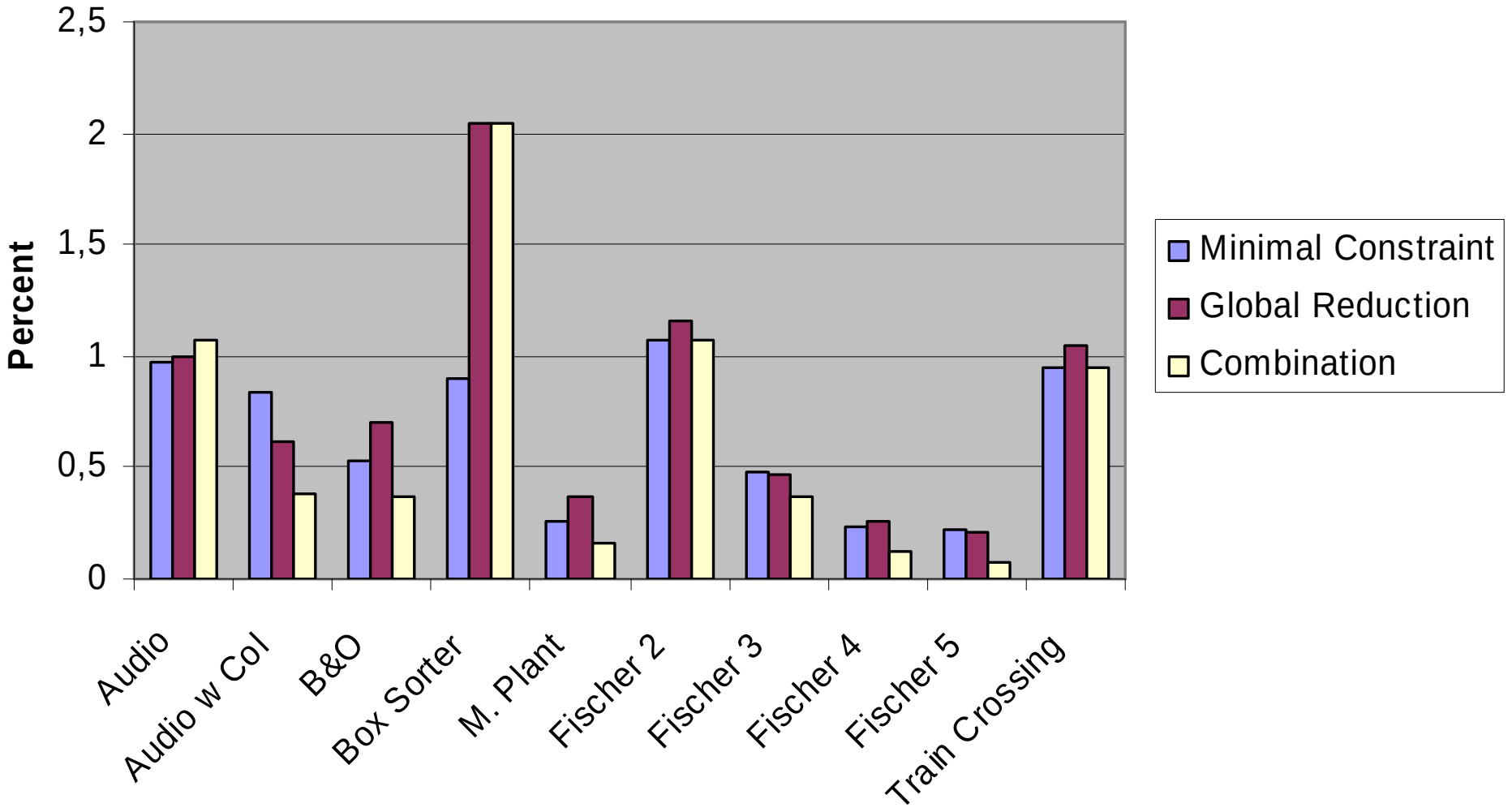
SPACE PERFORMANCE



UCb

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

TIME PERFORMANCE



UCb

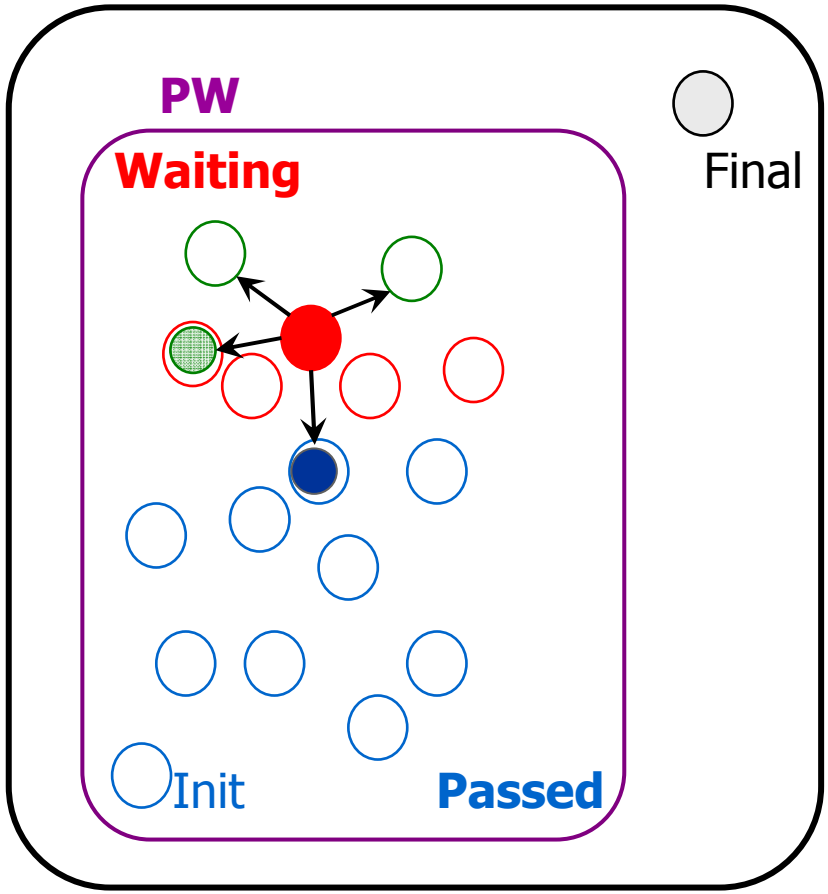
Clock Difference Diagrams

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

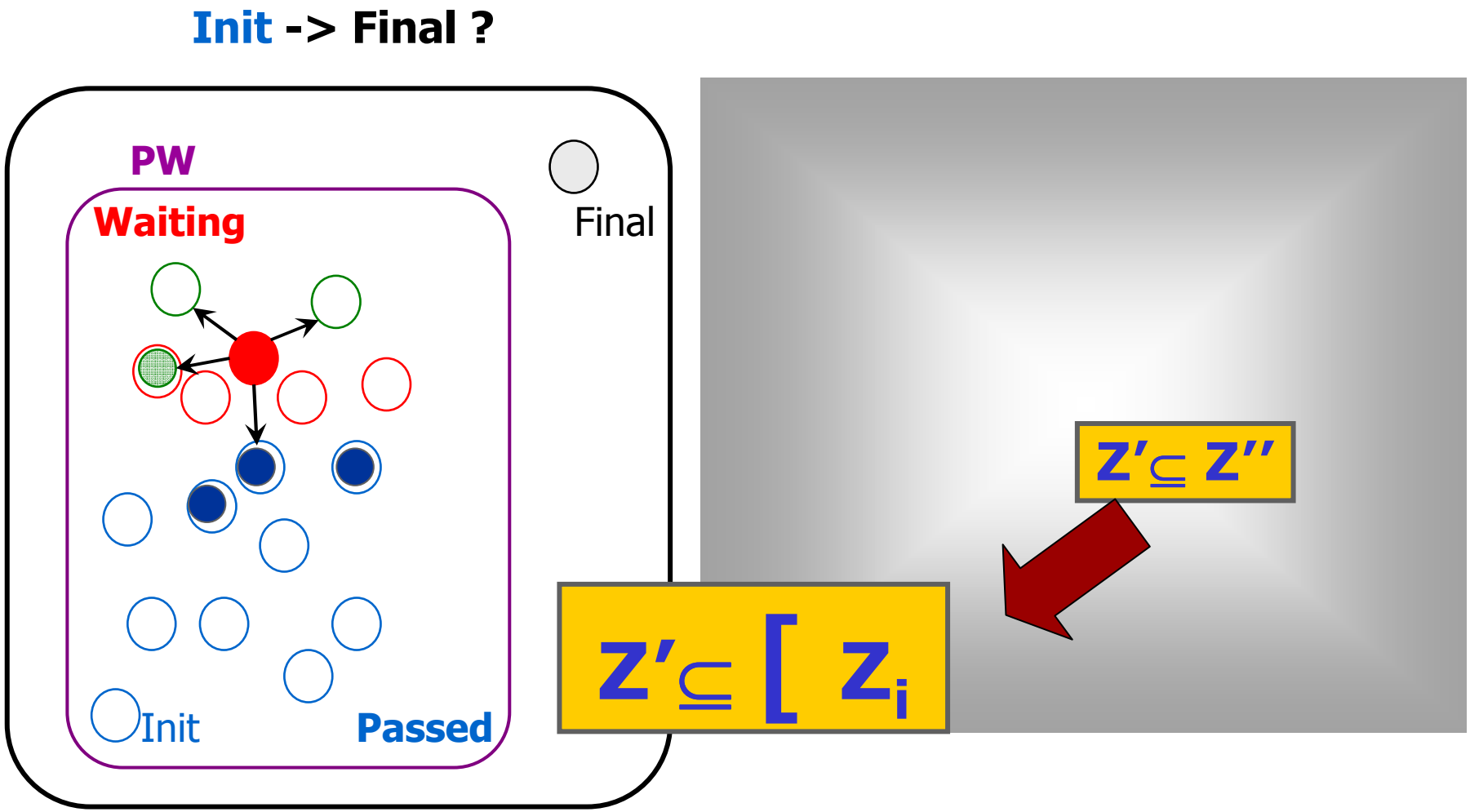
Earlier Termination

Init -> Final ?



UCb

Earlier Termination

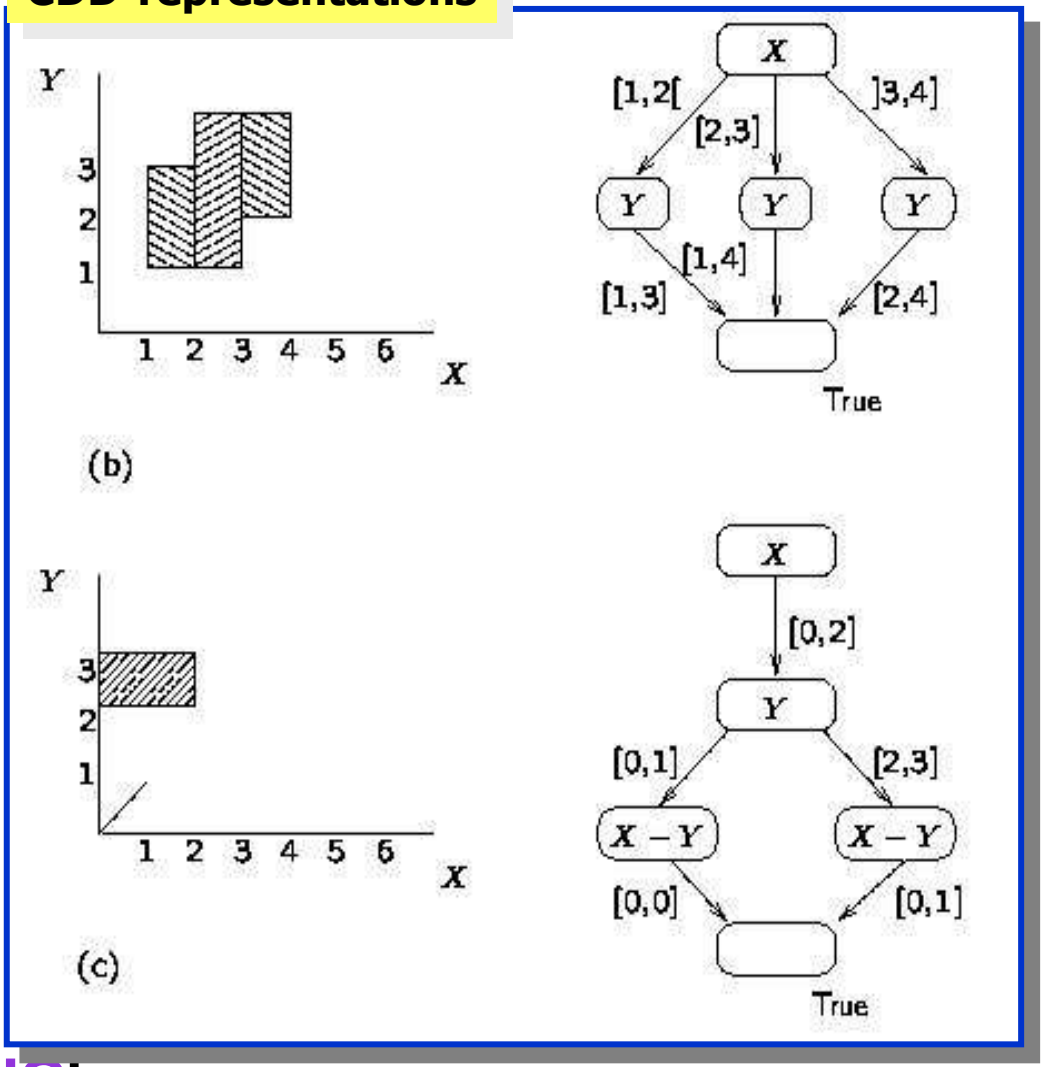


Clock Difference Diagrams

= Binary Decision Diagrams + Difference Bounded Matrices

CAV99

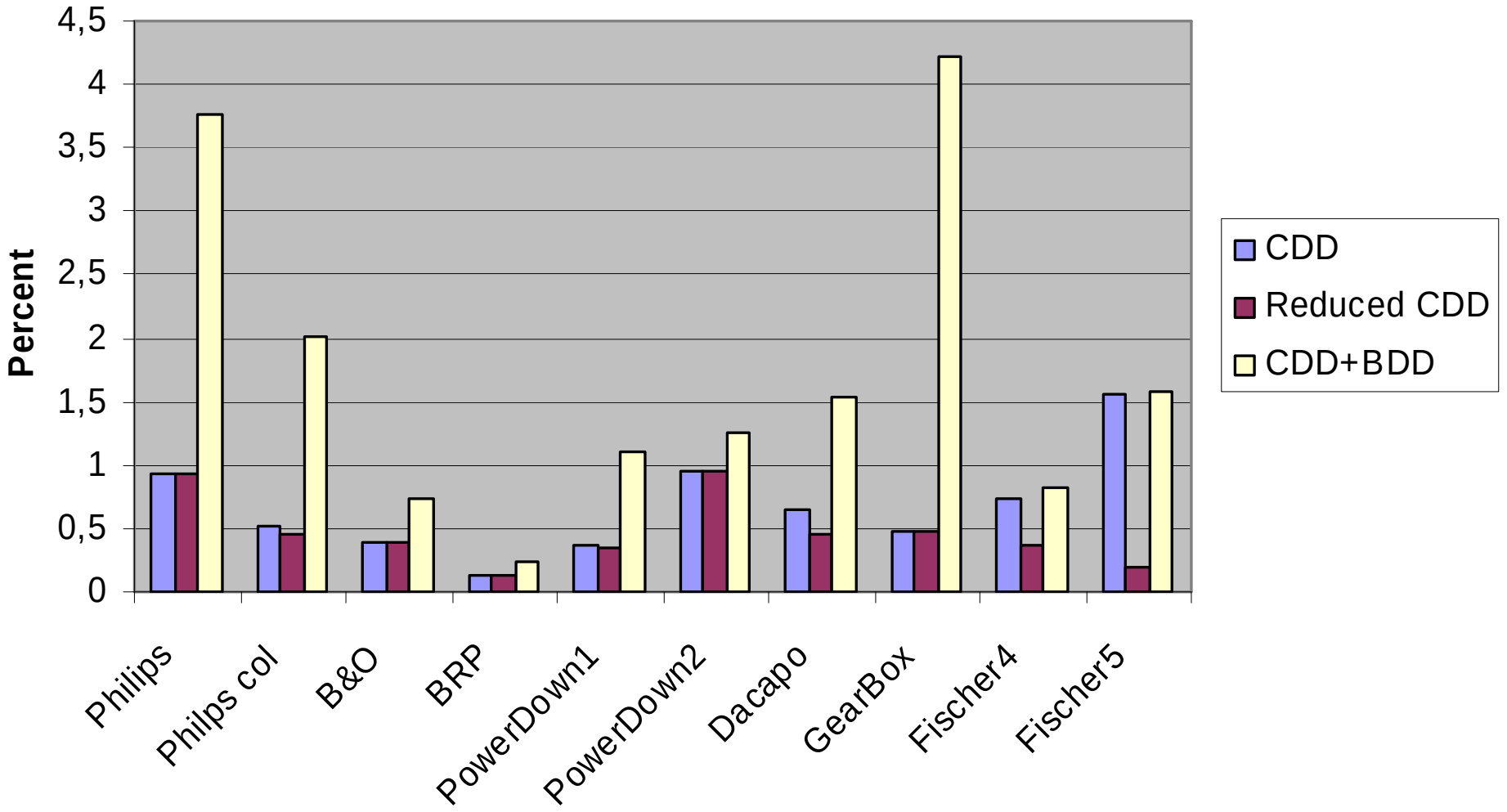
CDD-representations



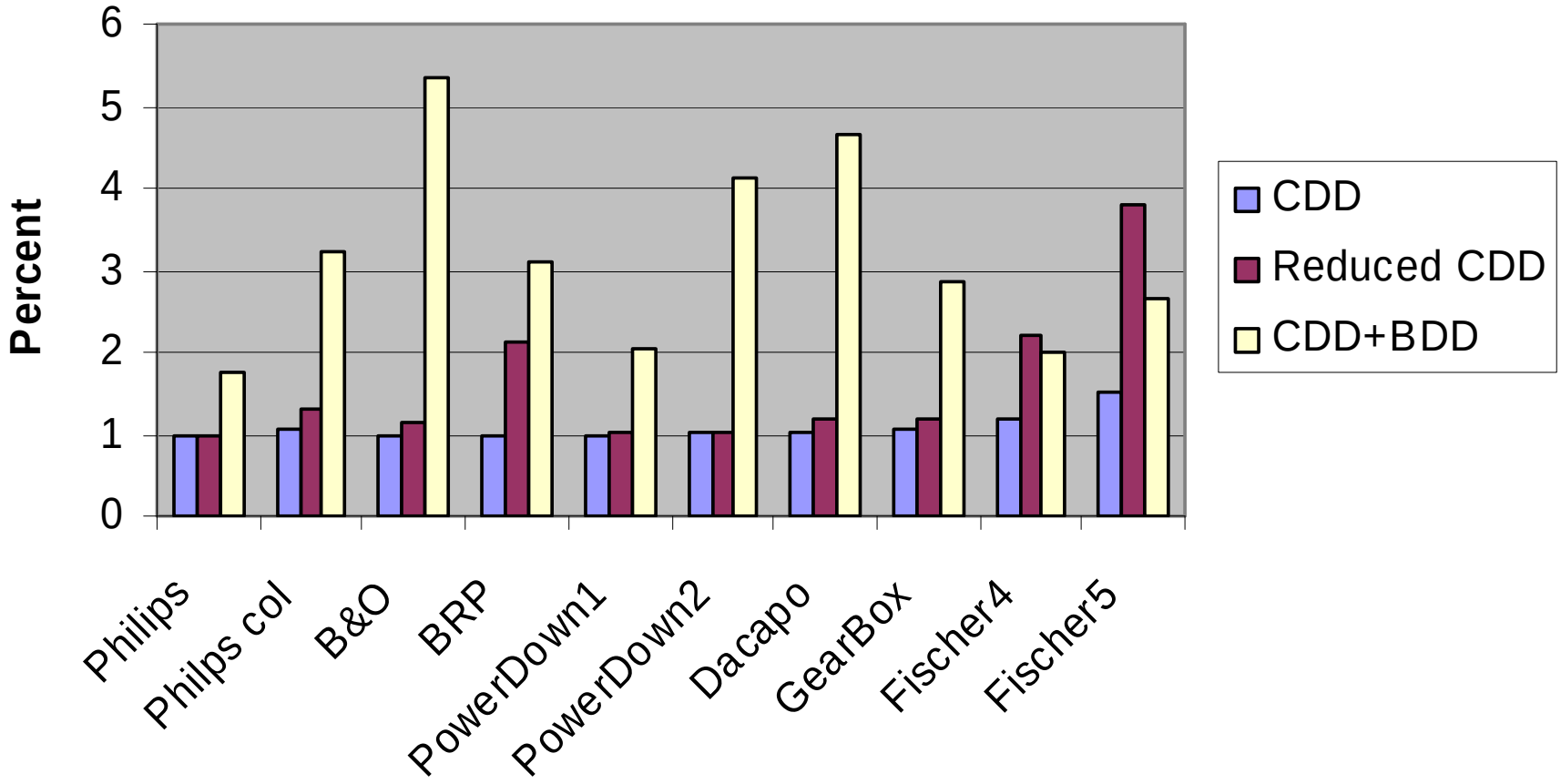
- Nodes labeled with differences
- Maximal sharing of substructures (also across different CDDs)
- Maximal intervals
- Linear-time algorithms for set-theoretic operations.

UCb

SPACE PERFORMANCE



TIME PERFORMANCE



Related & Future Work

- DDD: Andersen et al.
- NDD: Asarin, Bozga, Kerbrat, Maler, Pnueli, Rasse.
- IDD: Strehl, Thiele.

- No efficient algorithm for FUTURE and RESET operation on CDD.
- No canonical form.

- An efficient, fully symbolic engine for TA is still missing!!

UCb

Additional "secrets"

- Sharing among symbolic states
  - location vector / discrete values / zones
- Distributed implementation of UPPAAL
- Symmetry Reduction
- Sweep Line Method
- Guiding wrt Heuristic Value
  - User-supplied / Auto-generated
- Slicing wrt "C" Code

Open Problems

- Fully symbolic exploration of TA (both discrete and continuous part) ?
- Canonical form for CDD's ?
- Partial Order Reduction ?
- Compositional Backwards Reachability ?
- Bounded Model Checking for TA ?
- Exploitation of multi-core processors ?
- …

Datastructures for Zones

- Difference Bounded
-
-
-

Alexandre David

Elegant RUBY bindings for easy implementations

UPPAAL DBM Library

The library used to manipulate DBMs in UPPAAL

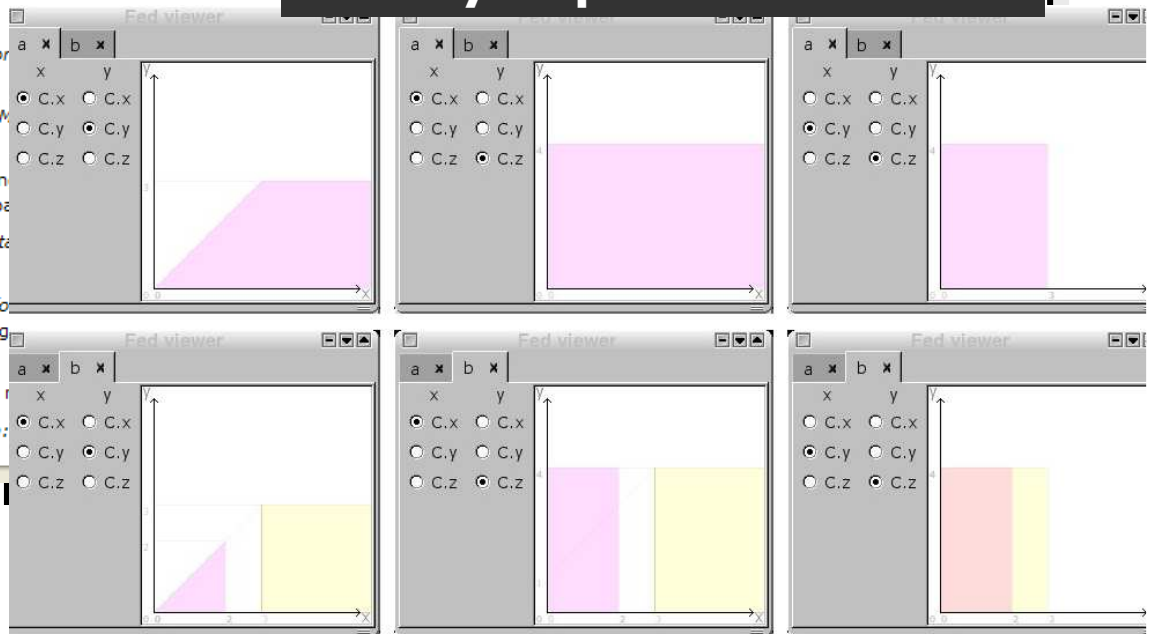
Main Page | Download | Ruby Binding | Help | Contact us

Welcome!

DBMs [dill89, rokicki93, lpw:fct95, bengtsson02] are efficient data structures to represent clock constrain in timed automata [ad90]. They are used in UPPAAL [lpy97, by04, bdl04] as the core data structure represent time. The library features all the common operations such as up (delay, or future), down (past), general updates, different extrapolation functions, etc.. on DBMs and federations. The lib subtractions. The API is in C and C++. The C++ part uses active clocks and hides memory management.

References

- [dill89] David L. Dill. *Timing Assumptions and Verificatior* Springer Berlin 1989, pp 197-212.
- [rokicki93] Tomas Gerhard Rokicki. *Representing and M* University 1993.
- [lpw:fct95] Kim G. Larsen, Paul Pettersson, and Wan Fundamentals of Computation Theory 1995, LNCS 965 pa
- [bengtsson02] Johan Bengtsson. *Clocks, DBM, and Sta* University 2002.
- [ad90] Rajeev Alur and David L. Dill. *Automata fo* Colloquium on Algorithms, Languages, and Programming
- [lpy97] Kim G. Larsen, Paul Pettersson, and Wang Yi. Software Tools for Technology Transfer , October 1997,
- [by04] Johan Bengtsson and Wang Yi. *Timed Automata:*



[SPIN03]

UCb

CORA

AMETIST
advanced methods for timed systems

Optimal & Real Time Scheduling

using

Model Checking Technology

AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Overview

- Timed Automata and Scheduling

- Priced Timed Automata
- Optimal Scheduling
- Optimal Scheduling wrt Multiple Objectives
- Optimal Infinite Scheduling

AMETIST
advanced methods for timed systems

April 2002 – June 2005 IST-2001-35304

- **Academic partners:**
  - Nijmegen
  - Aalborg
  - Dortmund
  - Grenoble
  - Marseille
  - Twente
  - Weizmann
- **Industrial Partners**
  - Axxom
  - Bosch
  - Cybernetix
  - Terma

UCb

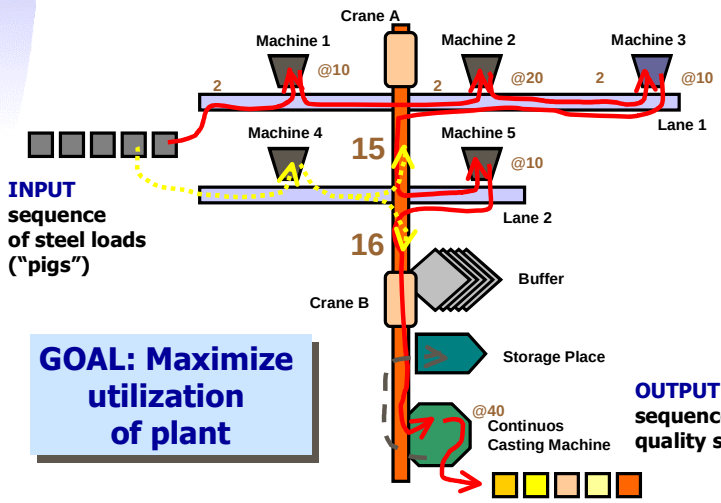
AMETIST
advanced methods for timed systems

OBJECTIVES

- powerful, unifying mathematical modelling
- efficient computerized problem-solving tools
- distributed real-time systems
- time-dependent behaviour and dynamic resource allocation
- **TIMED AUTOMATA**

UCb

SIDMAR Overview



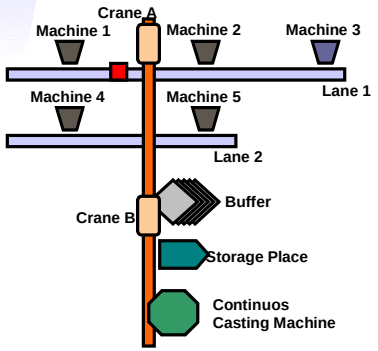
AALBORG UNIVERSITY

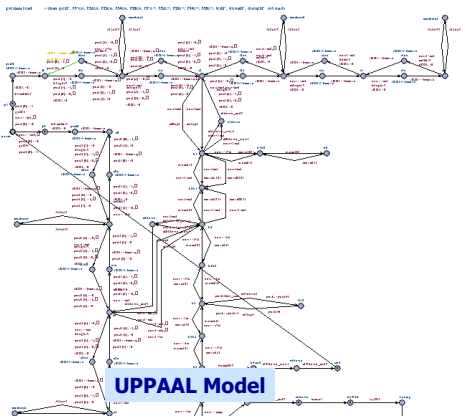
CSS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

LTR Project VHS (Verification of Hybrid systems)

SIDMAR Modelling

A Single Load





UPPAAL Model

AALBORG UNIVERSITY

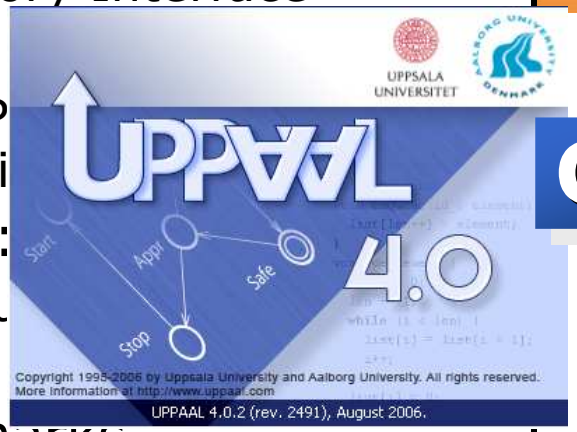
CSS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

AMETIST
advanced methods for timed systems

Case S

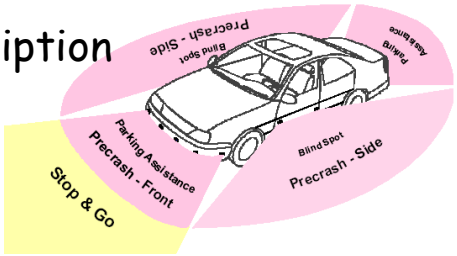
- Cyberne
  - Smar
    Perso
- Terma:
  - Memory Interface
- Bosch:
  - Car P
    Sensi
- AXXOM:
  - Lacqu
- Benchmarks

CLASSIC



CORA

CPS: Informal description

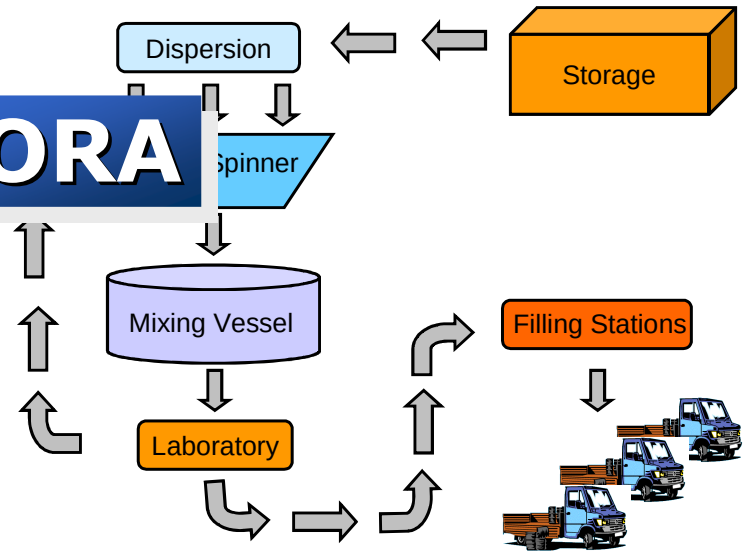


- CPS obtains and makes available for other systems information about environment of a car. This information may be used for:
  - Parking assistance
  - Stop & go
  - Etc
- Based on Short Range Radar (SRR) technology
- The CPS considered in this case study
  - One sensor group only (currently 2 sensors)
  - Only the front sensors and corresponding controllers
  - Application: pre-crash detection, parking assistance, stop & go

Brinksma — Car Periphery Supervision System: Case Study 3 — 2

Product flow of a Product

axxom

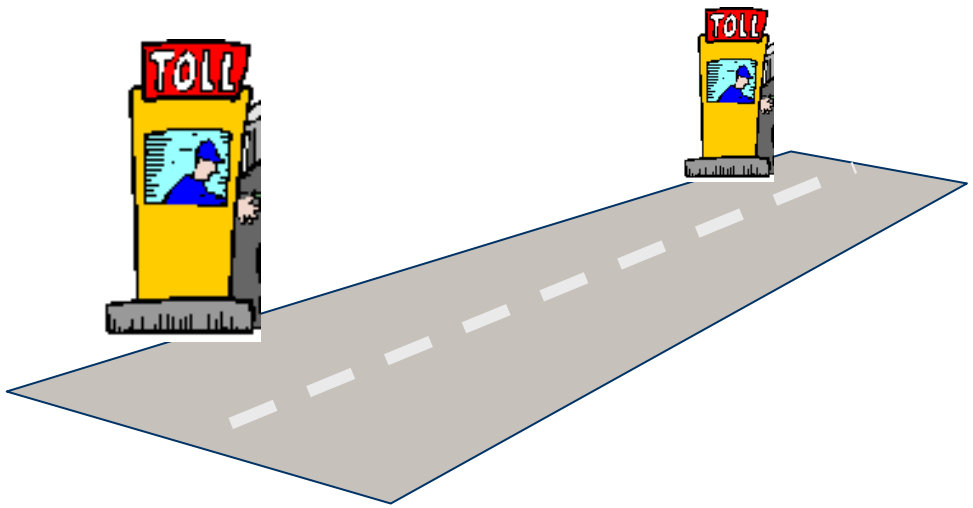


05.06.2005 — Axxom Software AG — Seite: 3

UCb

Real Time Scheduling

- Only 1 "Pass"
- Cheat is possible
  (drive close to car with "Pass")



UNSAFE

Crossing Times


5


10


Pass


20


25

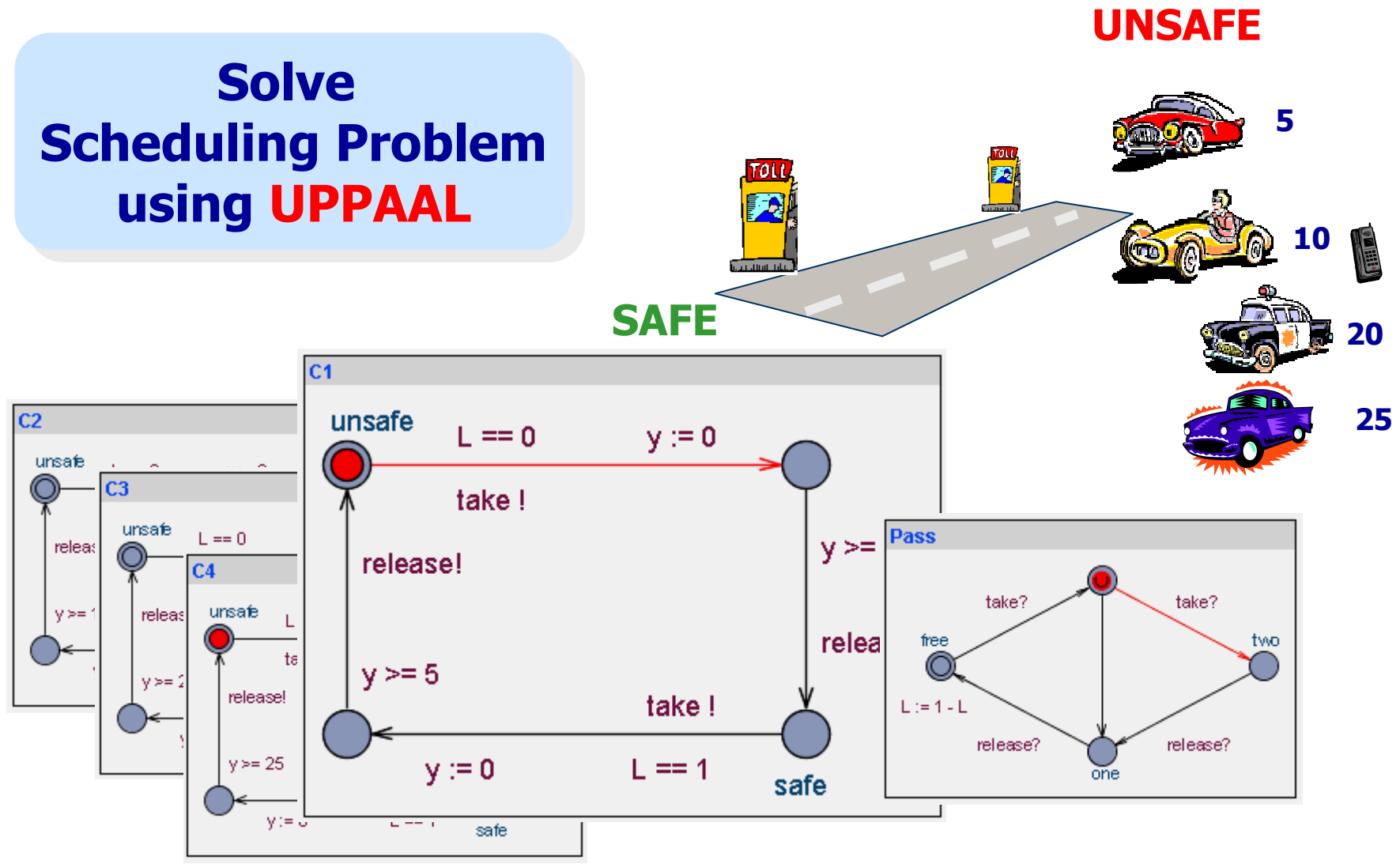
SAFE

CAN THEY MAKE IT TO SAFE WITHIN 70 MINUTES ???

UCb

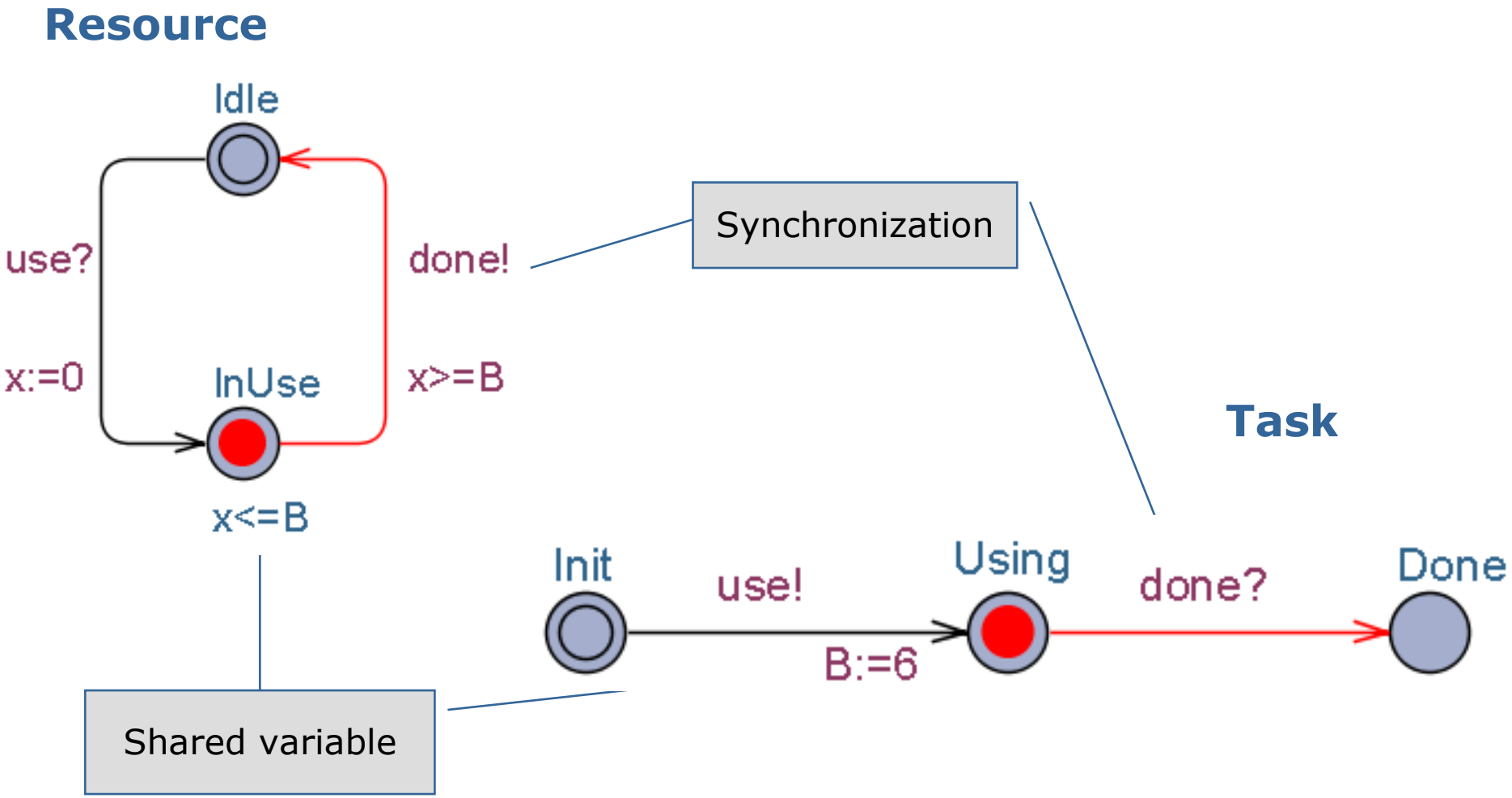
Real Time Scheduling

Solve Scheduling Problem using UPPAAL



UCb

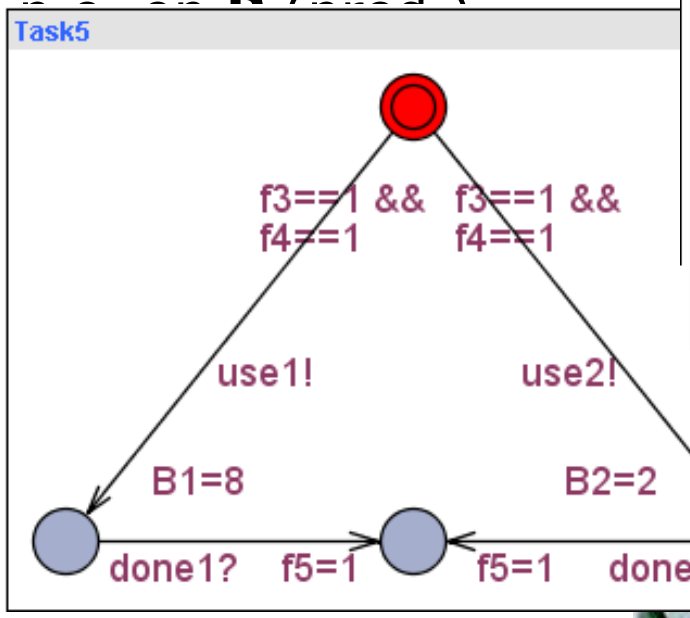
Resources & Tasks

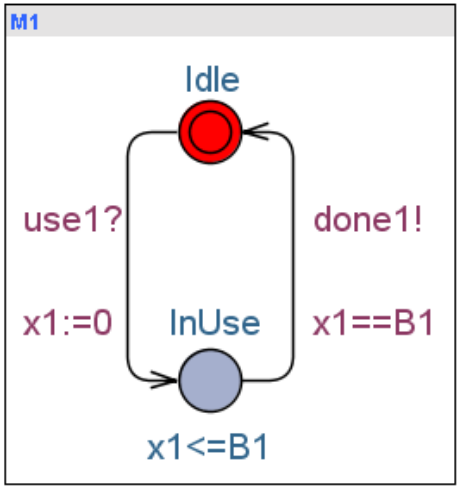


Task Graph Scheduling

Optimal Static Task Scheduling

- Task **P**={P_1,.., P_m}
- Machines **M**={M_1,..,M_n}
- Duration Δ : **(P£M) ! N_1**
- < :
- A t
  if a
  cor
- Eac
  at
- Tas
- Compute schedule with minimum completion-time!



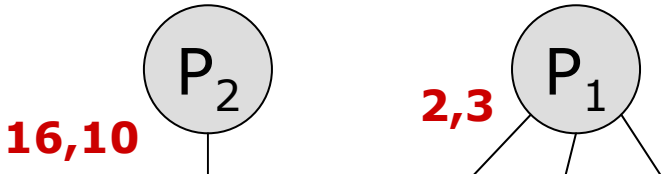


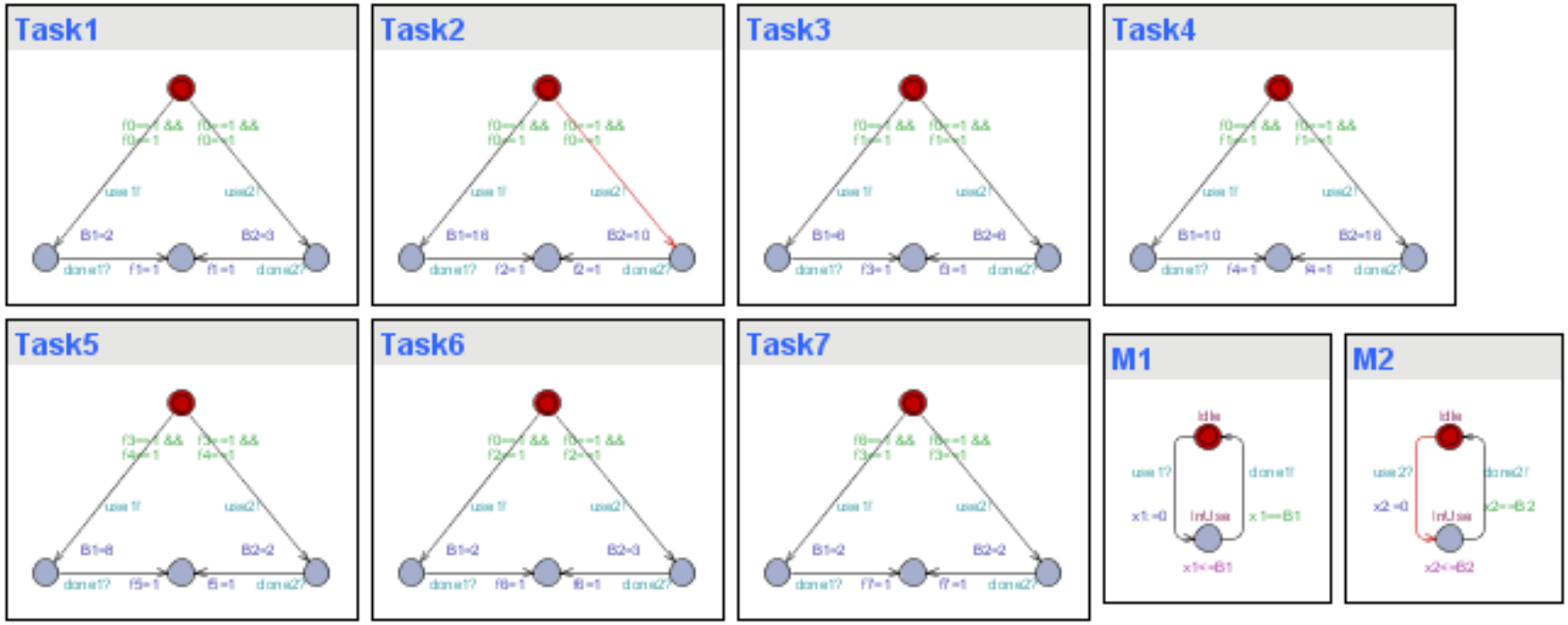
UCb

Task Graph Scheduling

Optimal Static Task Scheduling

- Task **P**={P_1,.., P_m}
- Machines **M**={M_1,...,M_n}





E<> (Task1.End and … and Task7.End)

Experimental Results

name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.	1150
233	1980	207	19	1118	1121
294	2014	141	17	1257	1261
295	2168	965	18	1318	1322
292	2333	318	3	8009	8009
298	2399	303	10	2471	2473

Symbolic A*
Brand-&-Bound
60 sec

Abdeddaïm, Kerbaa, Maler

Optimality

Priced Timed Automata

with Paul Pettersson, Thomas Hune, Judi Romijn, Ansgar Fehnker, Ed Brinksma, Frits Vaandrager, Patricia Bouyer, Franck Cassez, Henning Dierks Emmanuel Fleury, Jacob Rasmussen,..

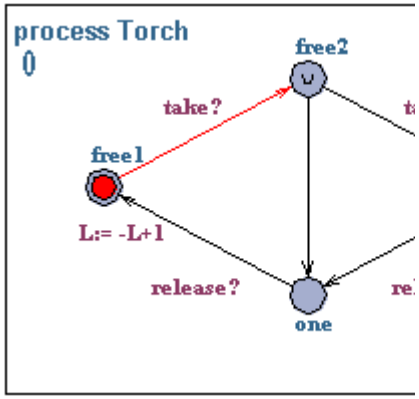
EXAMPLE: Optimal rescue plan for cars with different subscription rates for city driving !

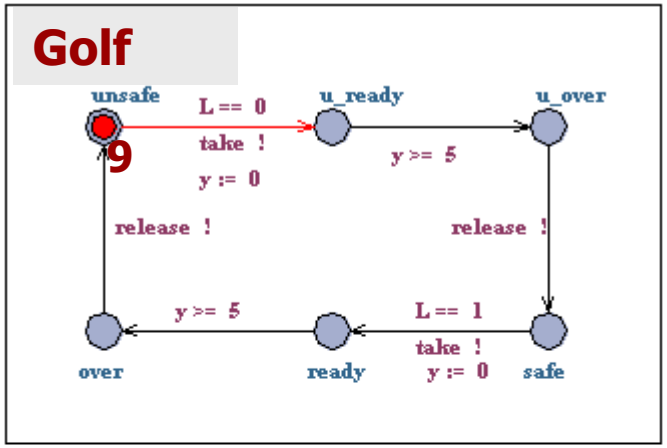

5

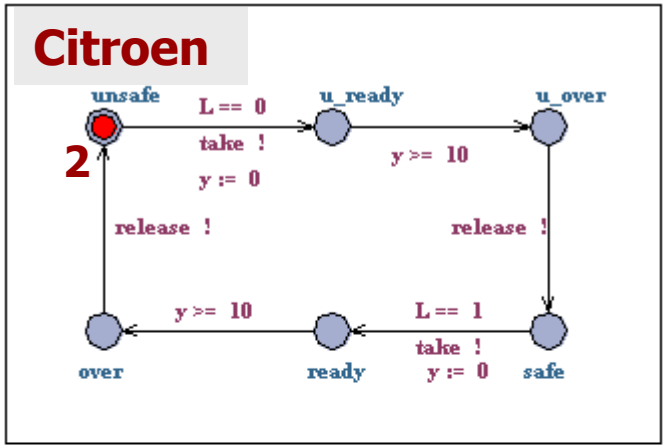

10

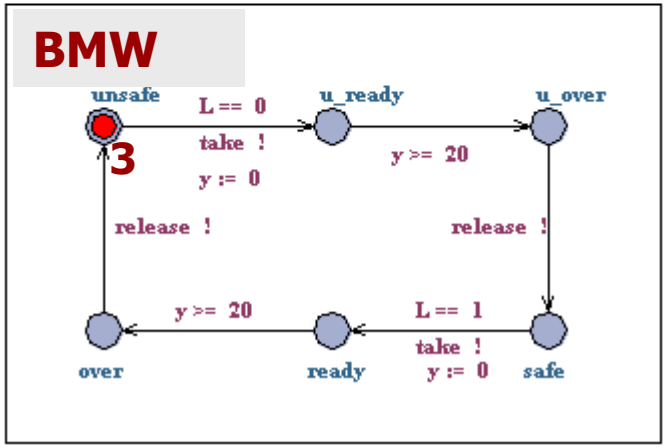

20

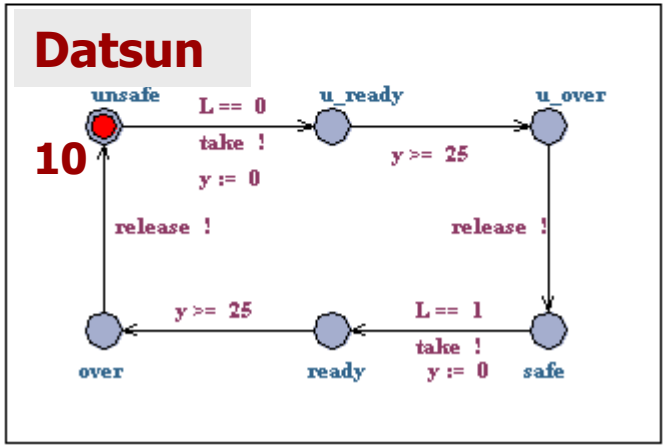

25











OPTIMAL PLAN HAS ACCUMULATED COST=195 and TOTAL TIME=65!

UCb

Experiments

COST-rates				SCHEDULE	COST	TIME	#Expl	#Pop'd
G	C	B	D					
Min Time				CG> G< BD> C< CG>		60	1762 1538	2638
1	1	1	1	CG> G< BG> G< GD>	55	65	252	378
9	2	3	10	GD> G< CG> G< BG>	195	65	149	233
1	2	3	4	CG> G< BD> C< CG>	140	60	232	350
1	2	3	10	CD> C< CB> C< CG>	170	65	263	408
1	20	30	40	BD> B< CB> C< CG>	975 1085	85 time<85	-	-
0	0	0	0		0	-	406	447

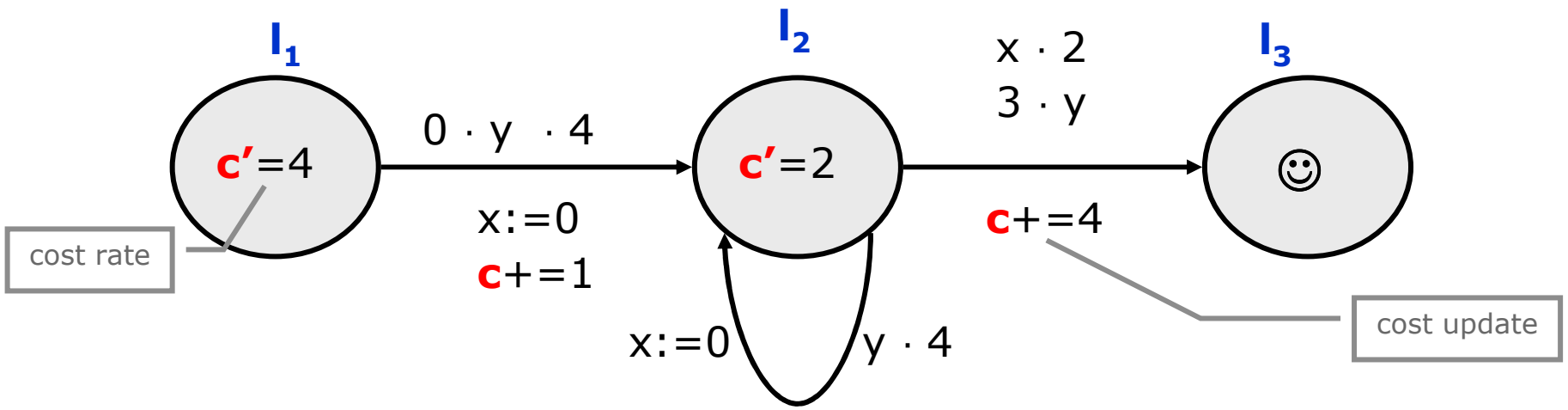
UCb

Priced Timed Automata

Timed Automata + **COST** variable

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)



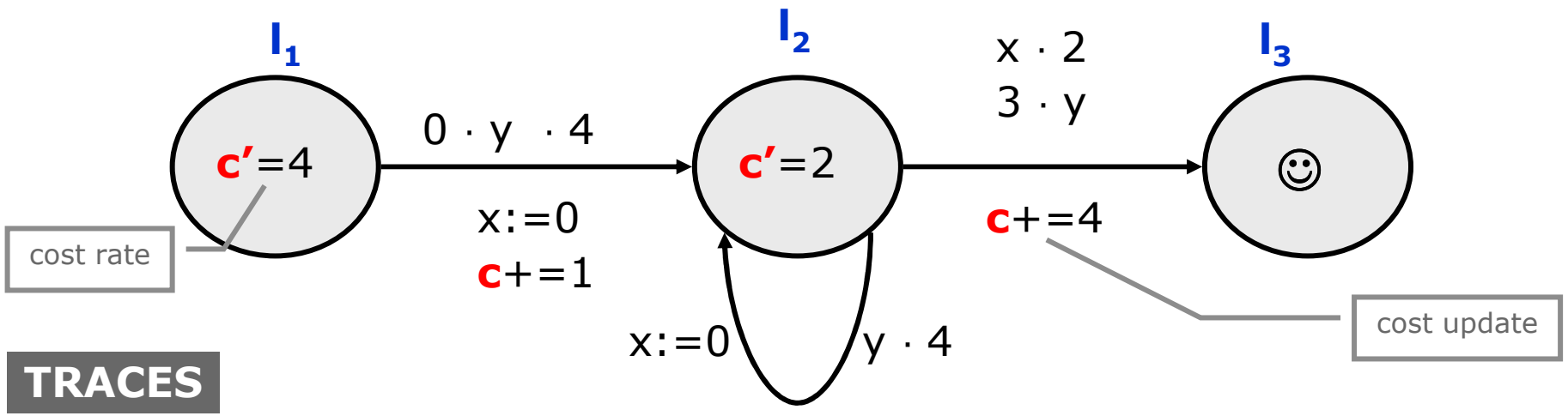
UCb

Priced Timed Automata

Timed Automata + **COST** variable

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)



TRACES

$(l_1, x=y=0) \xrightarrow[12]{\varepsilon(3)} (l_1, x=y=3) \xrightarrow{1} (l_2, x=0, y=3) \xrightarrow{4} (l_3, \_, \_)$

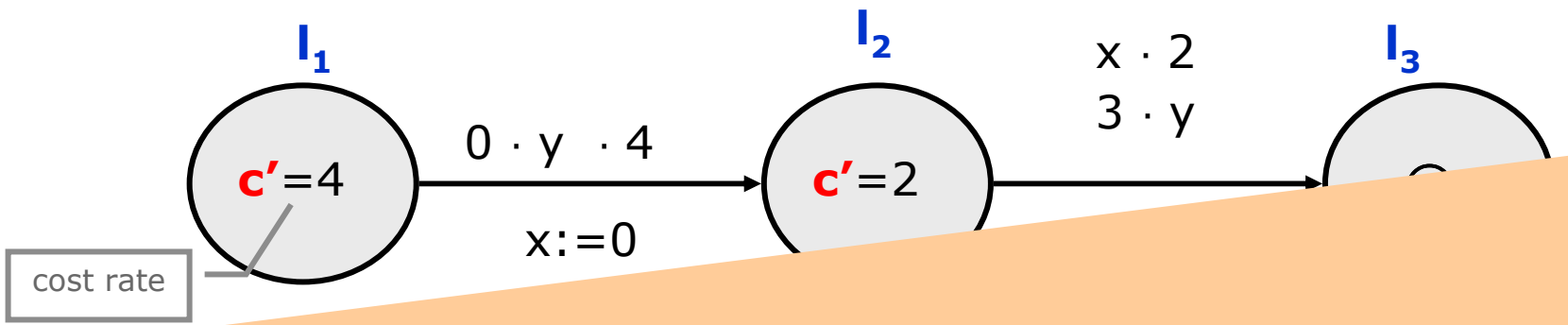
Σ c=17

Priced Timed Automata

Timed Automata + **COST** variable

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)



Problem :

Find the minimum cost of reaching location l_3

Efficient Implementation:
CAV'01 and TACAS'04

$(l_1, x=y=0) \xrightarrow{1} (l_2, x=0, y=0) \xrightarrow{\varepsilon(3)\ 6} (l_2, x=3, y=3) \xrightarrow{0} (l_2, x=0, y=3) \xrightarrow{4} (l_3, \_, \_)$

Σ c=11

$(l_3, \_, \_)$

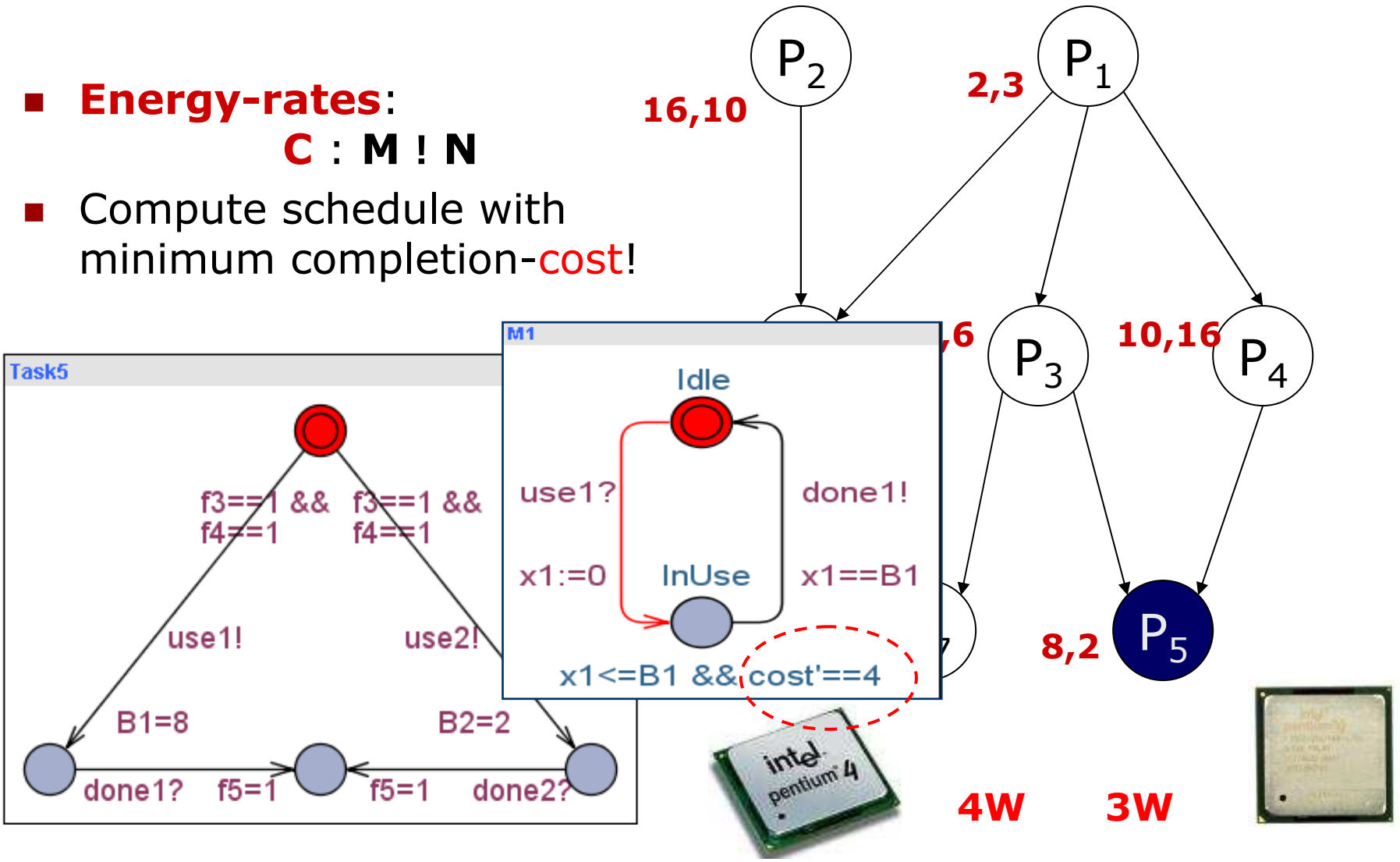
Σ c=16

UCb

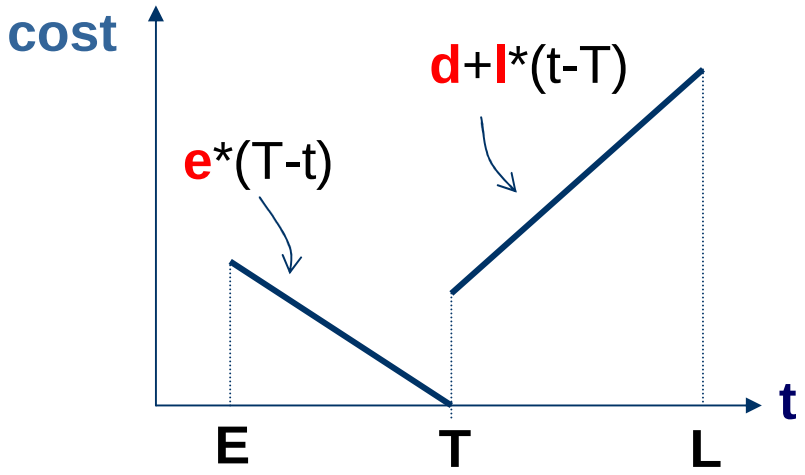
Optimal Task Graph Scheduling

Power-Optimality

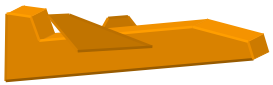
- **Energy-rates**:
  C : M ! N
- Compute schedule with minimum completion-cost!



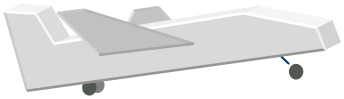
Aircraft Landing Problem

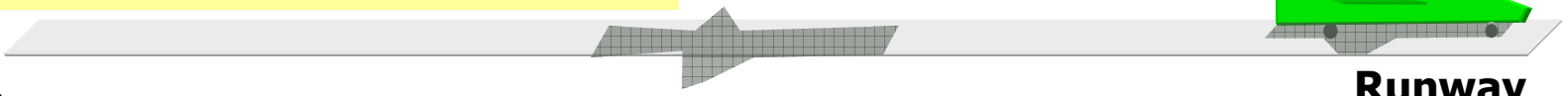


E earliest landing time
T target time
L latest time
e cost rate for being early
l cost rate for being late
d fixed cost for being late



Planes have to keep separation distance to avoid turbulences caused by preceding planes

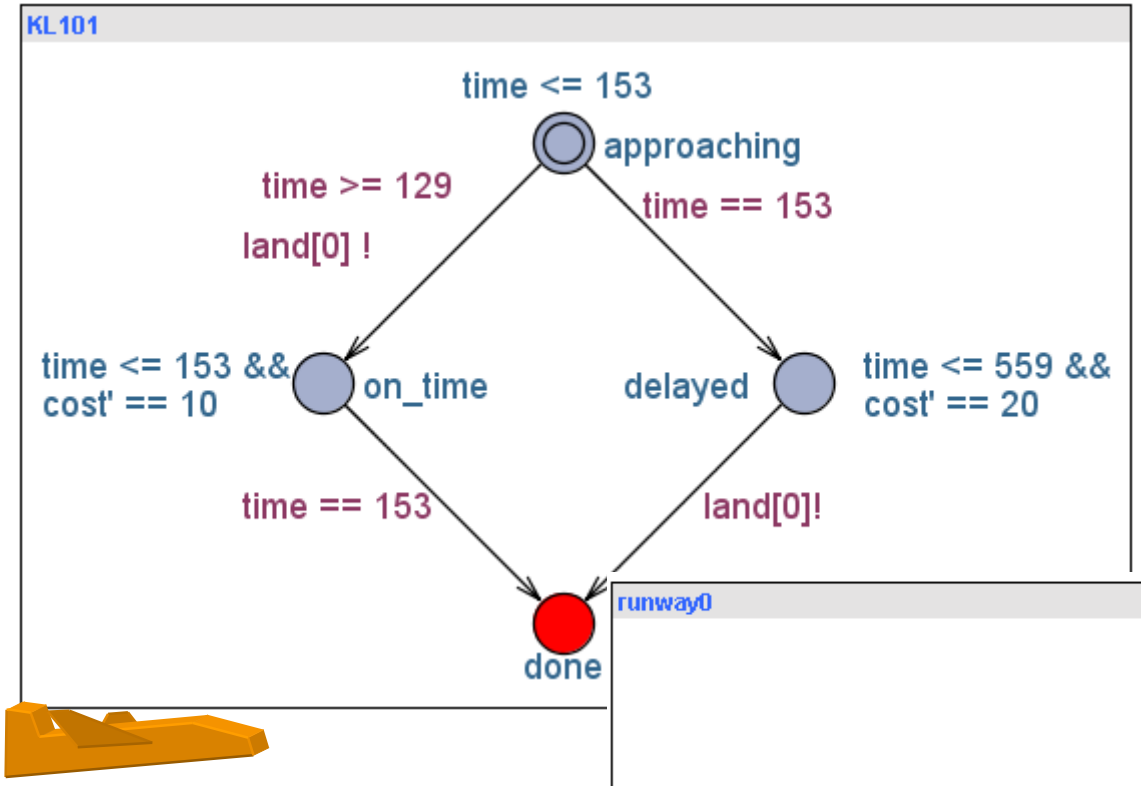




Runway

UCb

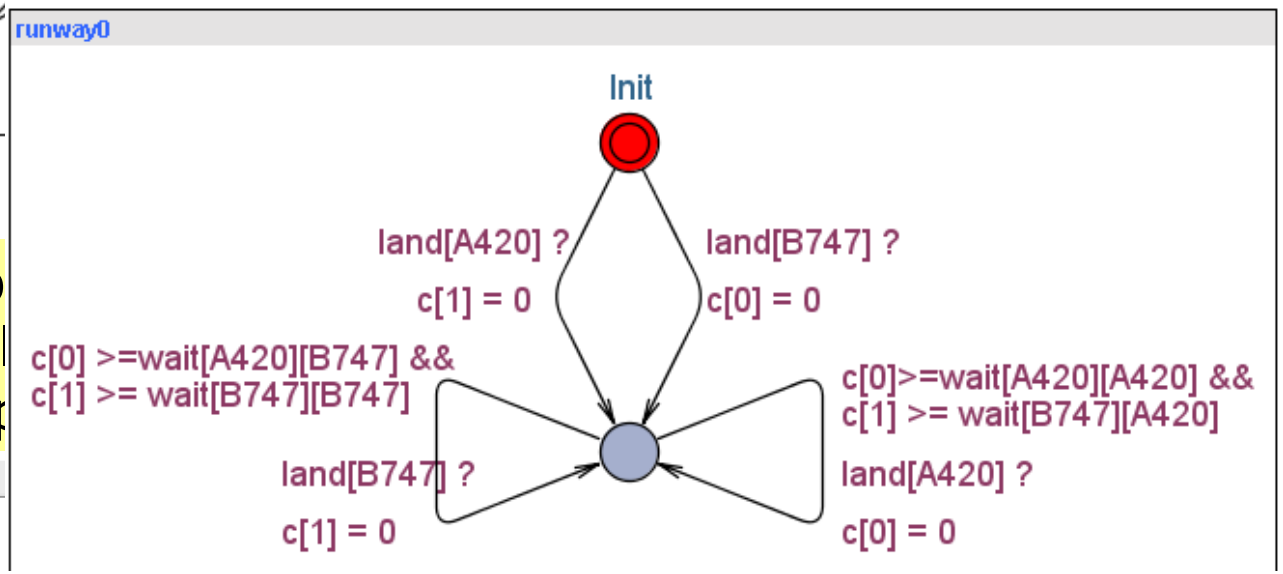
Modeling ALP with PTA



129: Earliest landing time
153: Target landing time
559: Latest landing time
10: Cost rate for early
20: Cost rate for late

Runway handles 2 types of planes

Planes have to keep sep
distance to avoid turbul
caused by preceding p



UCb

Aircraft Landing

CAV'01

Source of examples:
Baesley et al'2000

	problem instance	1	2	3	4	5	6	7
	number of planes	10	15	20	20	20	30	44
	number of types	2	2	2	2	2	4	2
1	optimal value	700	1480	820	2520	3100	24442	1550
	explored states	481	2149	920	5693	15069	122	662
	cputime (secs)	4.19	25.30	11.05	87.67	220.22	0.60	4.27
2	optimal value	90	210	60	640	650	554	0
	explored states	1218	1797	669	28821	47993	9035	92
	cputime (secs)	17.87	39.92	11.02	755.84	1085.08	123.72	1.06
3	optimal value	0	0	0	130	170	0	
	explored states	24	46	84	207715	189602	62	N/A
	cputime (secs)	0.36	0.70	1.71	14786.19	12461.47	0.68	
4	optimal value				0	0		
	explored states	N/A	N/A	N/A	65	64	N/A	N/A
	cputime (secs)				1.97	1.53		

UCb

Symbolic ”A*”

Zones

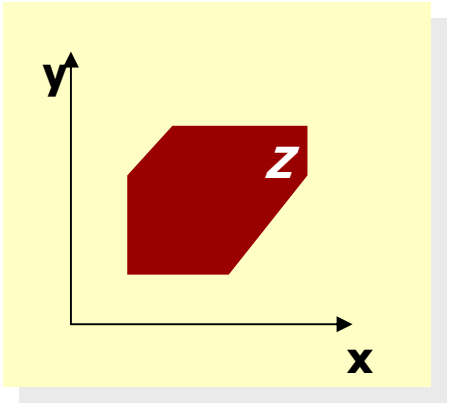
Definition

A *zone* Z over a set of clocks C is a finite conjunction of simple constraints of the forms:

$$x \geq l \qquad x \leq u \qquad x - y \geq l' \qquad x - y \leq u'$$

where $x, y \in C$, $l, u \in \mathbb{N}$ and $l', u' \in \mathbb{Z}$.

For $u \in \mathbb{R}^C$ and Z a zone we write $u \models Z$ if u satisfies all constraints of Z.



Operations

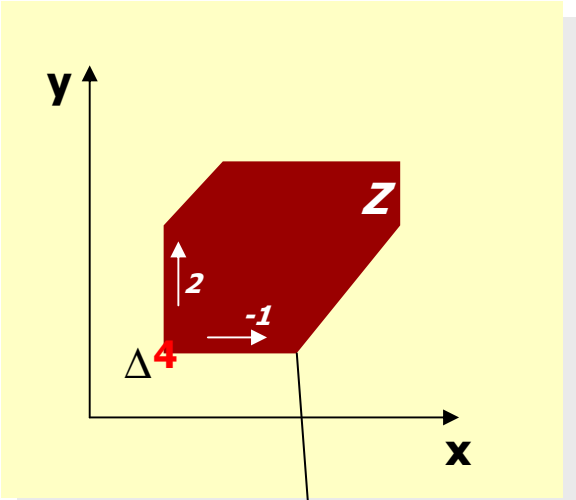
Reset: $\{x\}Z = \{u[0/x] \mid u \models Z\}$

Delay: $Z^{\uparrow} = \{u + d \mid u \models Z\}$

Offset: $\Delta_Z \models Z$ such that $\forall u \models Z . \forall x \in C . \Delta_Z(x) \leq u(x)$.

Priced Zone

CAV'01



Definition

A priced zone P is a tuple (Z, c, r), where:

- Z is a zone
- $c \in \mathbb{N}$ describes the cost of Δ_Z
- $r : C \to \mathbb{Z}$ gives a *rate* for any clock $x \in C$.

We write $u \models P$ whenever $u \models Z$. For $u \models P$ we define $\mathrm{Cost}(u, P)$ as follows:

$$\mathrm{Cost}(u, P) = c + \sum_{x \in C} r(x) \cdot (u(x) - \Delta_Z(x))$$

$Cost(x,y) = 2y - x + 2$

UCb

Branch & Bound Algorithm

Cost := ∞
Passed := $\emptyset$
Waiting := $\{(l_0, Z_0)\}$
while Waiting $\neq \emptyset$ **do**
    select (l, Z) from Waiting
    if $l = l_g$ and minCost(Z) < Cost **then**
        Cost := minCost(Z)
    if minCost(Z) + $\text{Rem}_{(l,Z)} \geq$ Cost **then** break
    if for all (l, Z') in Passed: $Z' \not\leq Z$ **then**
        add (l, Z) to Passed
        add all (l', Z') with $(l, Z) \rightarrow (l', Z')$ to Waiting
return Cost

Optimization
with Multi Objectives

with Jacob I. Rasmussen

EXAMPLE: Optimal rescue plan for cars with different subscription rates for city driving !

CONDITIONAL

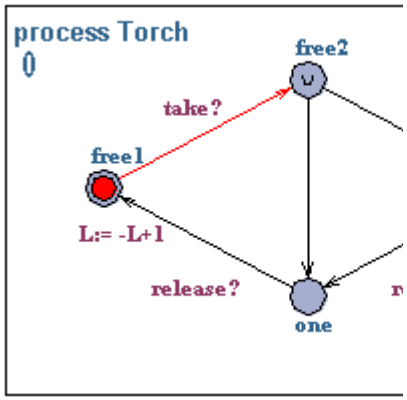
UNSAFE

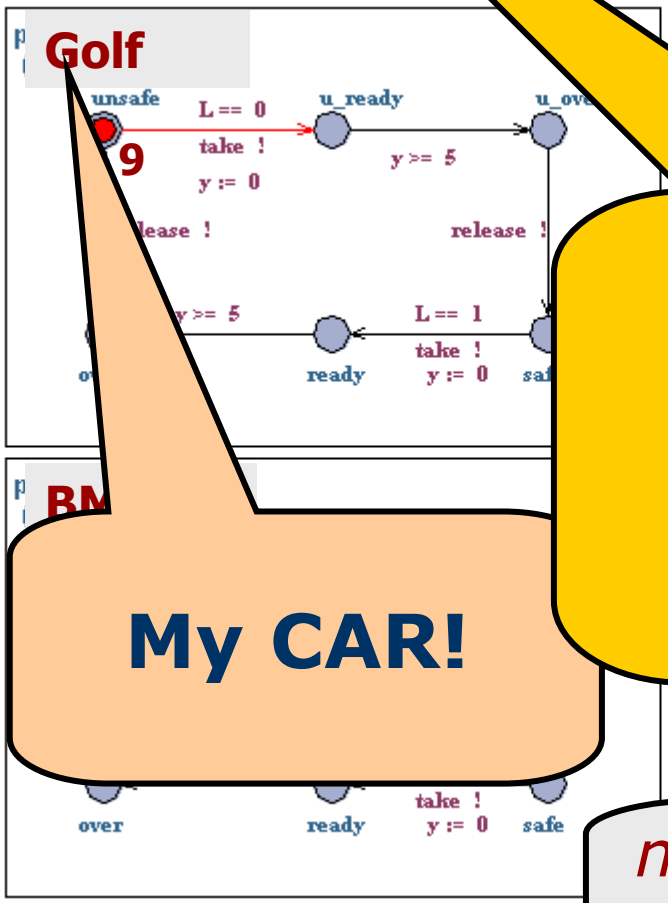

5

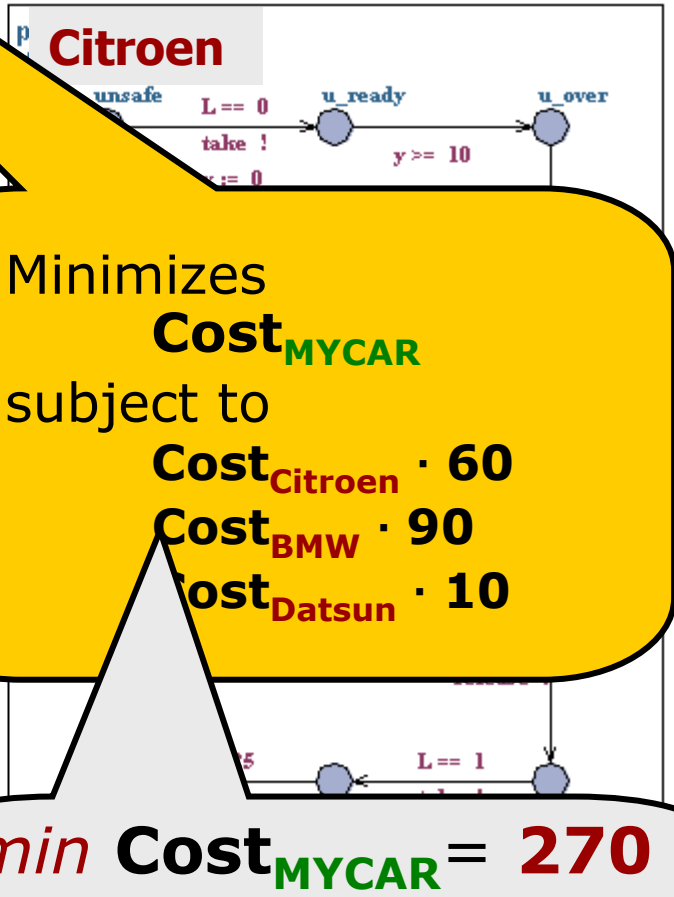

10


20


25







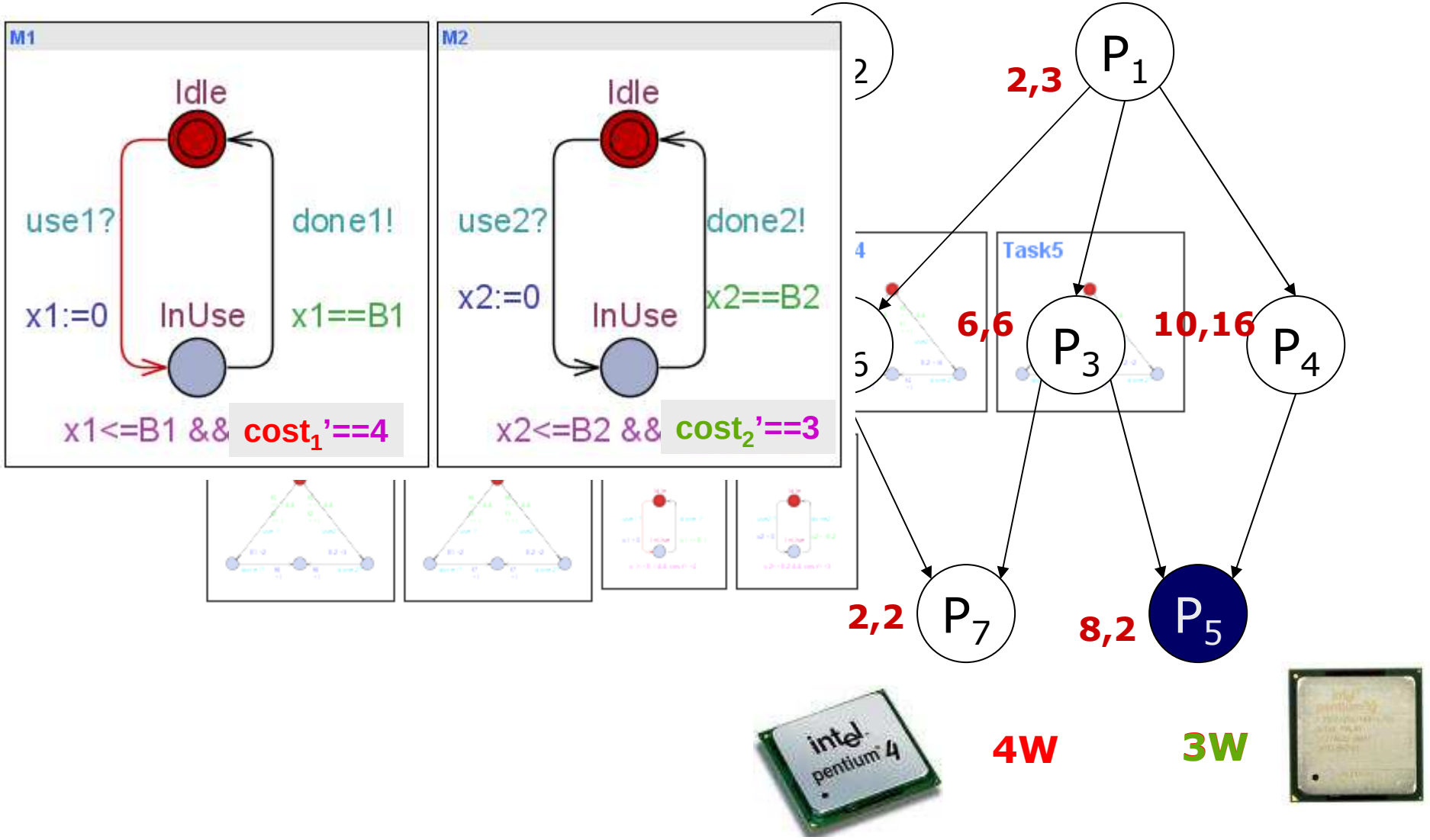
My CAR!

Minimizes
$Cost_{MYCAR}$
subject to
$Cost_{Citroen} \cdot 60$
$Cost_{BMW} \cdot 90$
$Cost_{Datsun} \cdot 10$

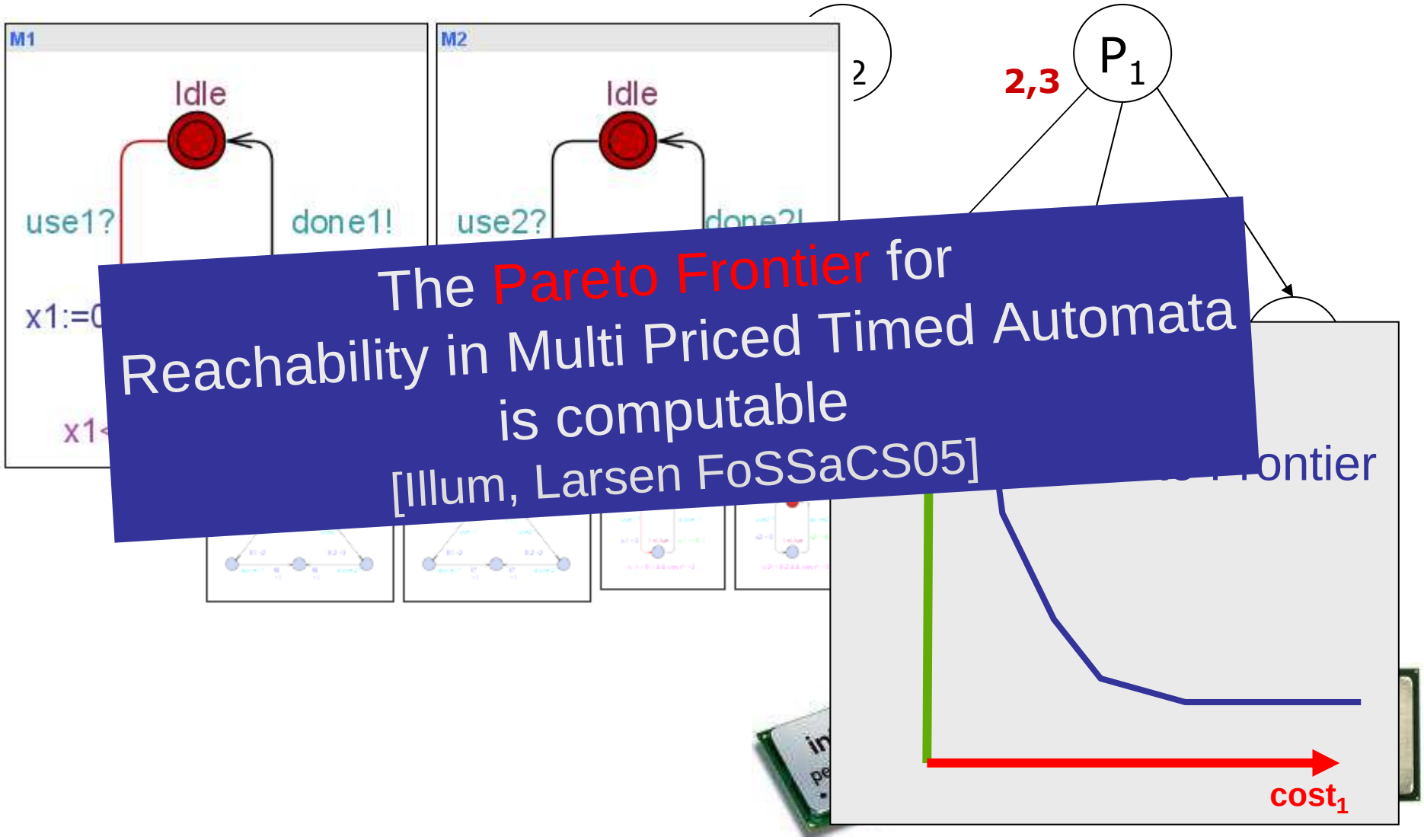
min $Cost_{MYCAR}$ = **270**
time = 70

UCb

Multiple Objective Scheduling



Multiple Objective Scheduling



UCb

Optimal Infinite Scheduling

with Ed Brinksma
Patricia Bouyer
Arne Skou
Ulrich Fahrenberg

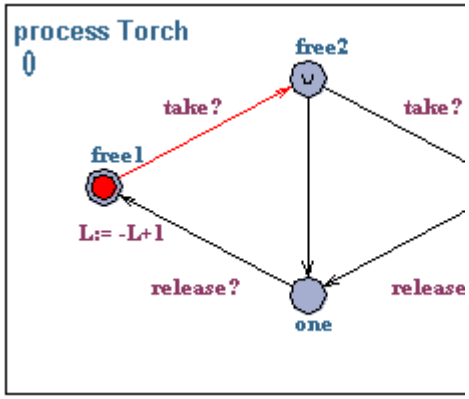
EXAMPLE: **Optimal WORK plan for cars with different subscription rates for city driving !**

5

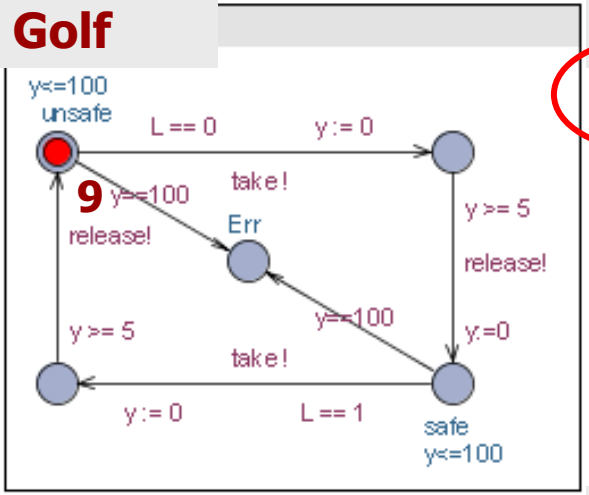
10

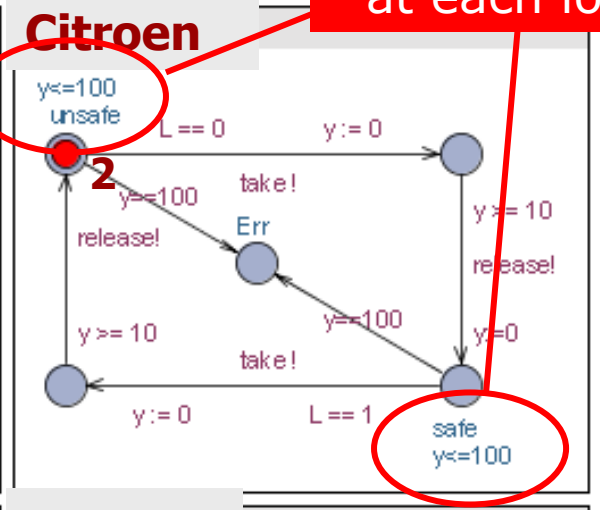
20

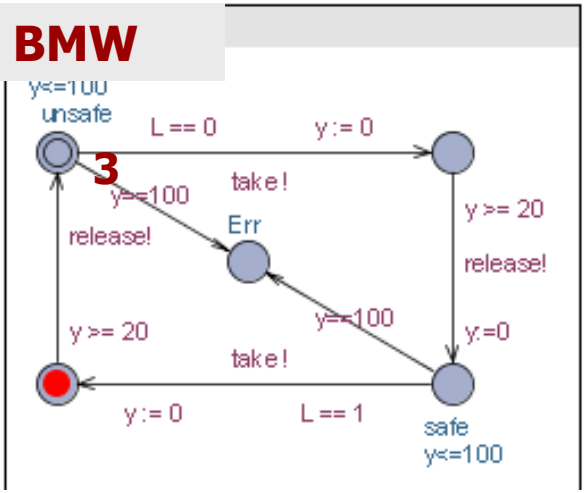
25

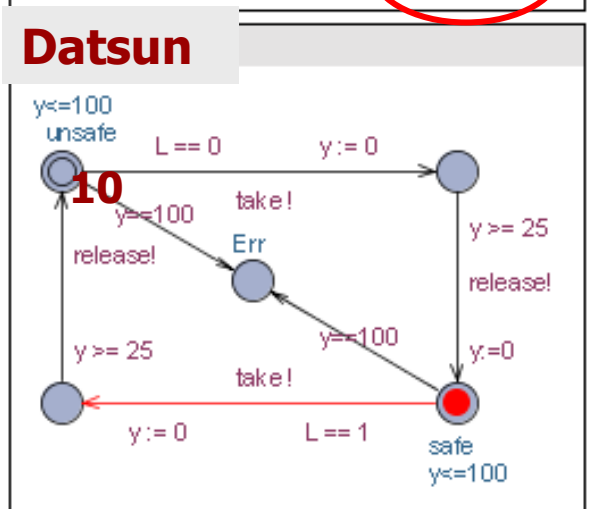


maximal 100 min. at each location



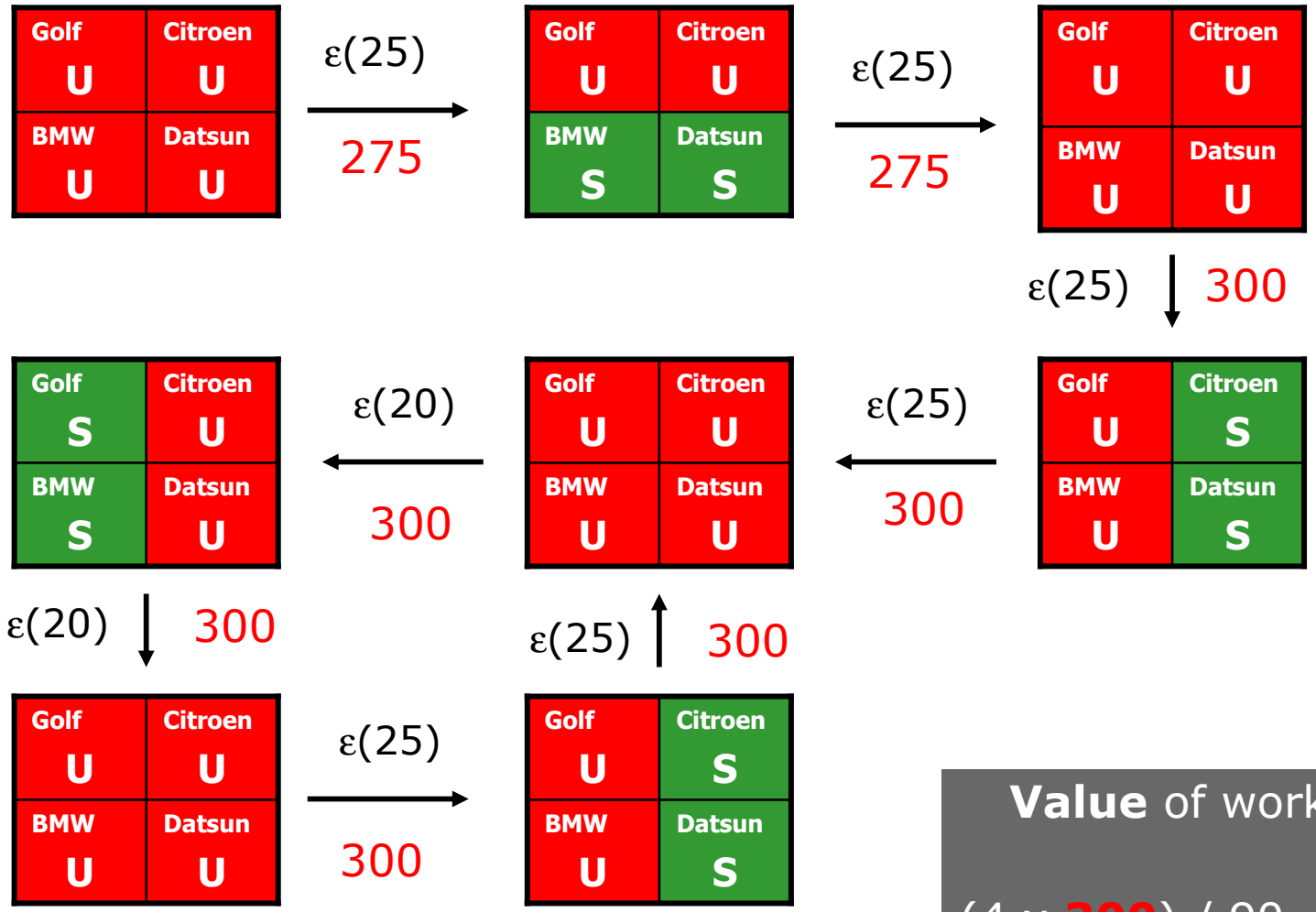






UCb

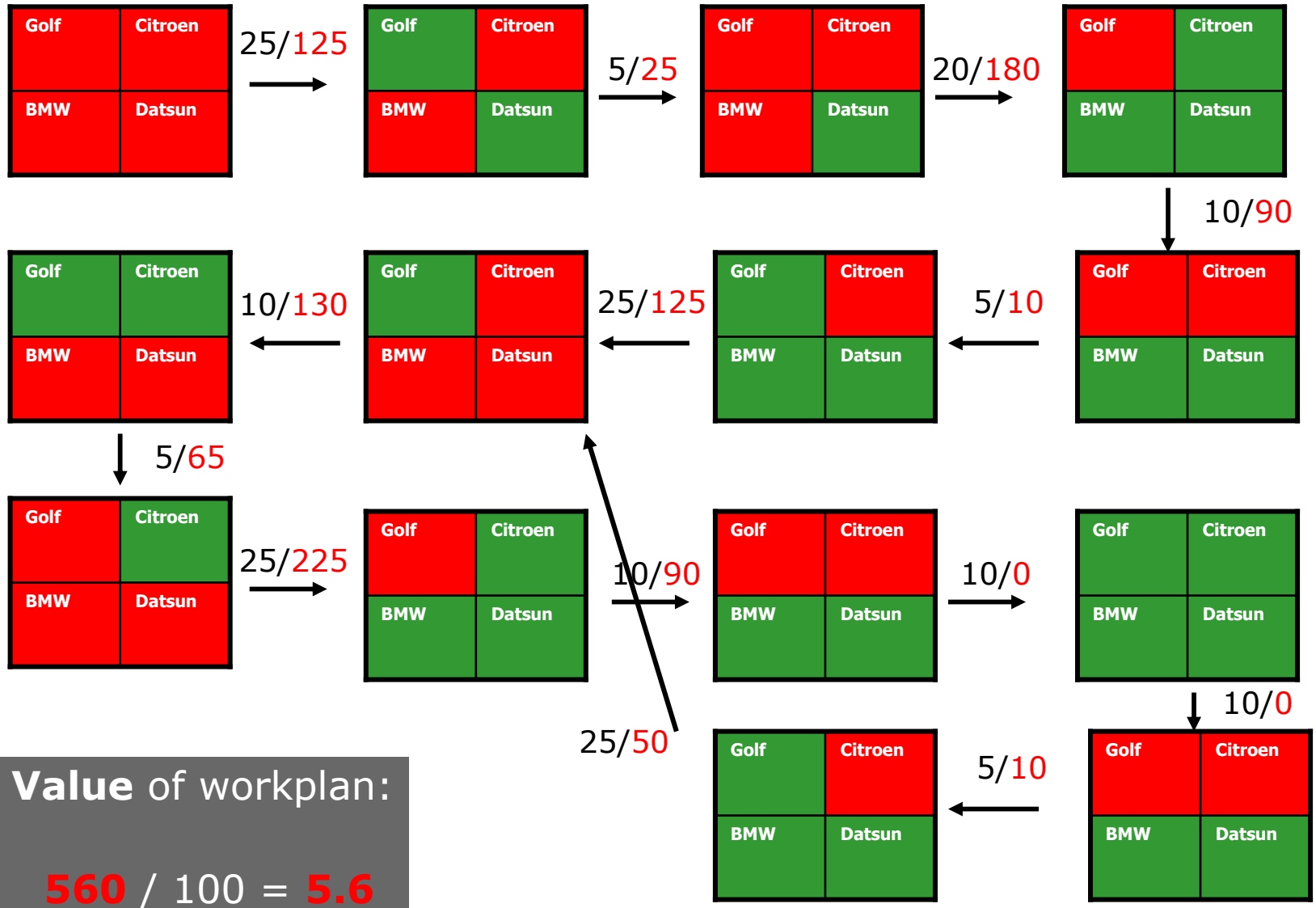
Workplan I



Value of workplan:

(4 x **300**) / 90 = **13.33**

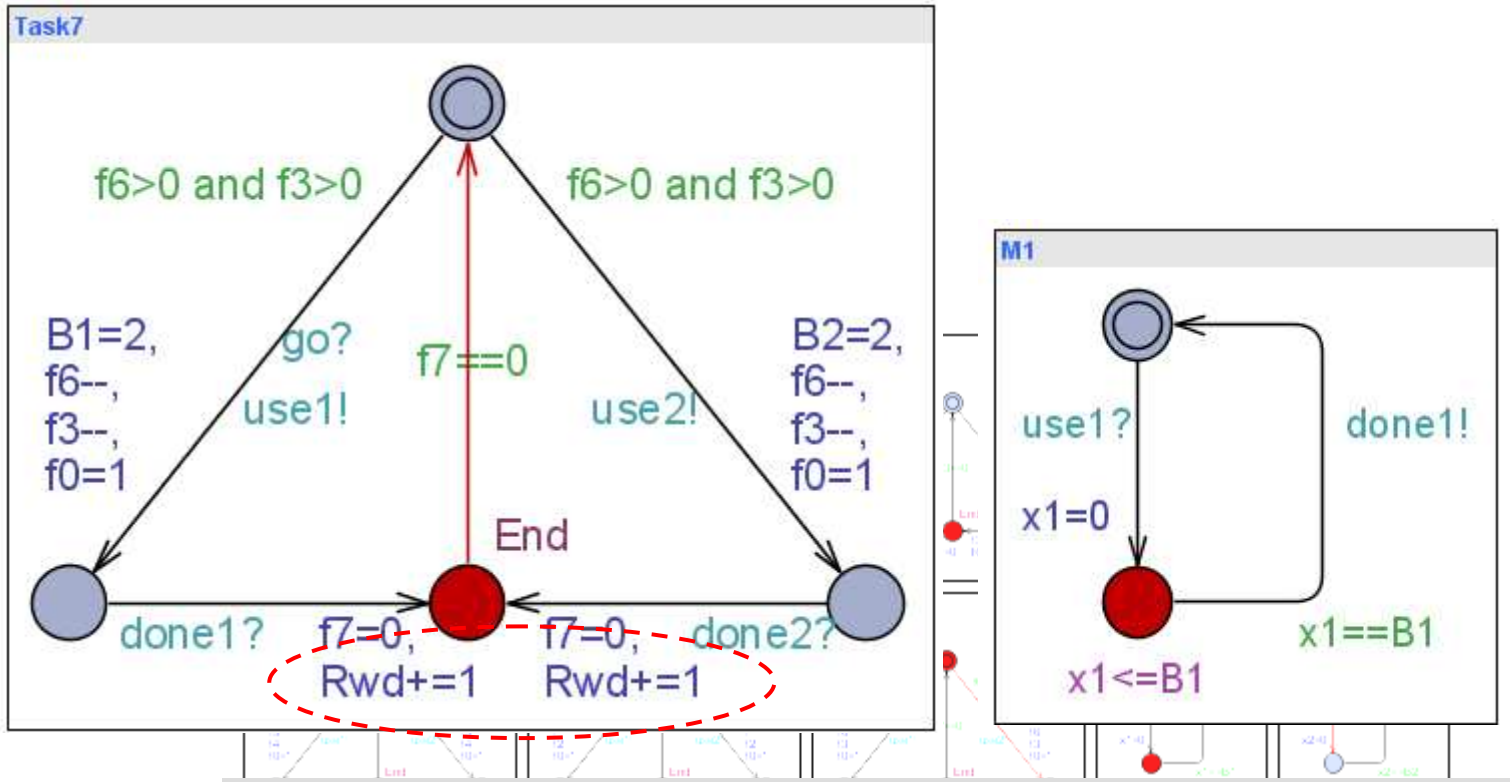
Workplan II



Value of workplan:

560 / 100 = **5.6**

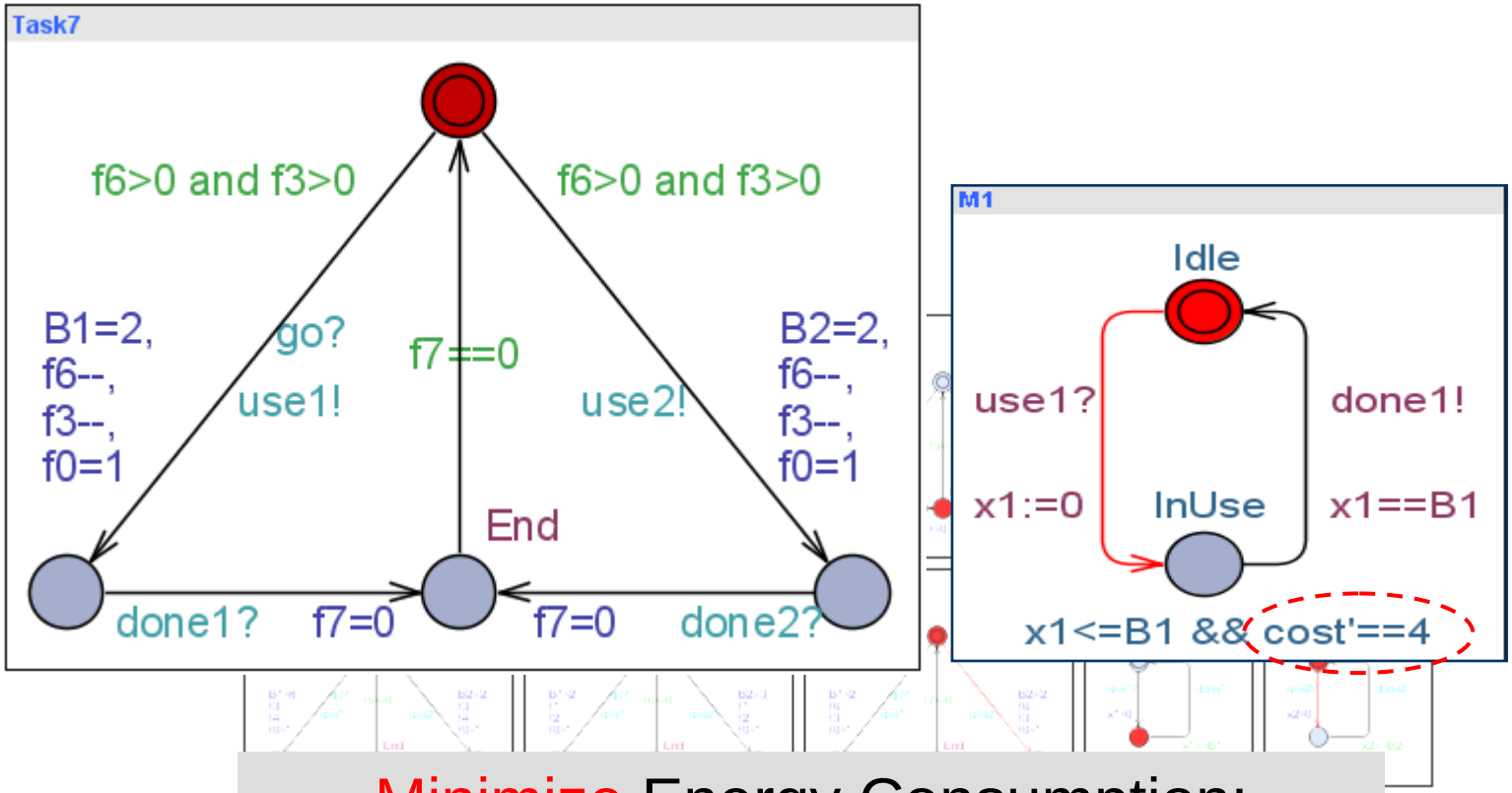
Optimal Infinite Scheduling



Maximize throughput:
i.e. maximize Reward / Time in the long run!

UCb

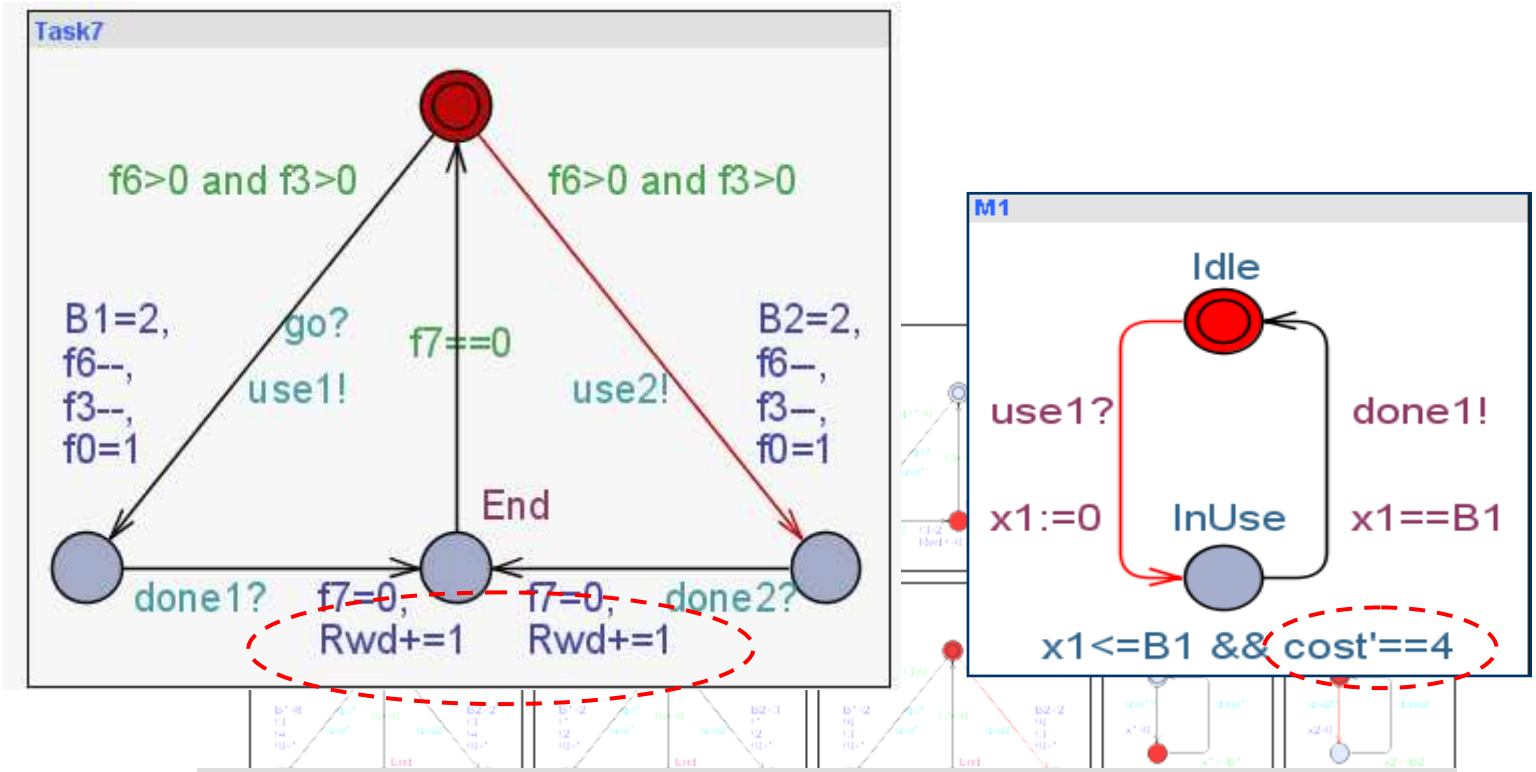
Optimal Infinite Scheduling



Minimize Energy Consumption:
i.e. minimize Cost / Time in the long run

UCb

Optimal Infinite Scheduling

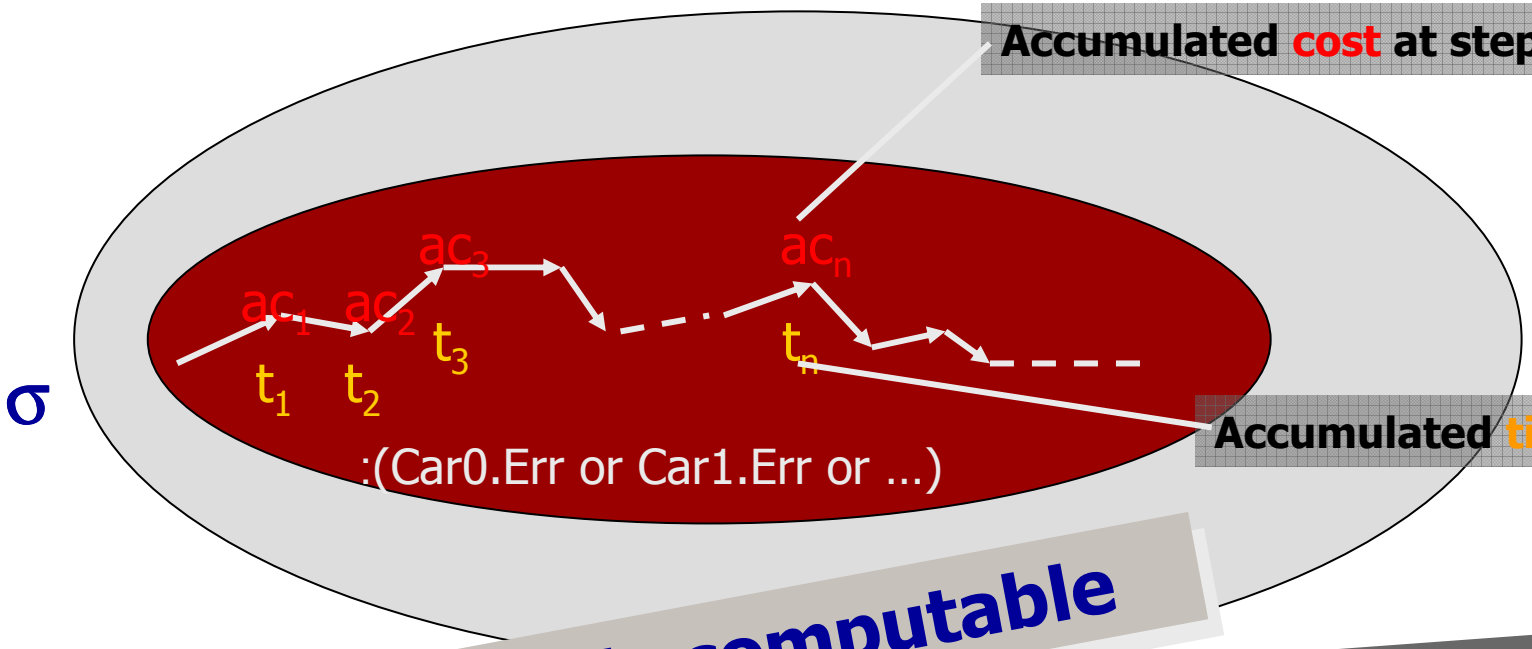


Maximize throughput:
i.e. maximize Reward / Cost in the long run

UCb

Optimal Infinite Scheduling I

Limit Ratio (Pay-off)



THEOREM: σ^* is computable

Bouyer, Brinksma, Larsen HSCC'04, FMSD'08

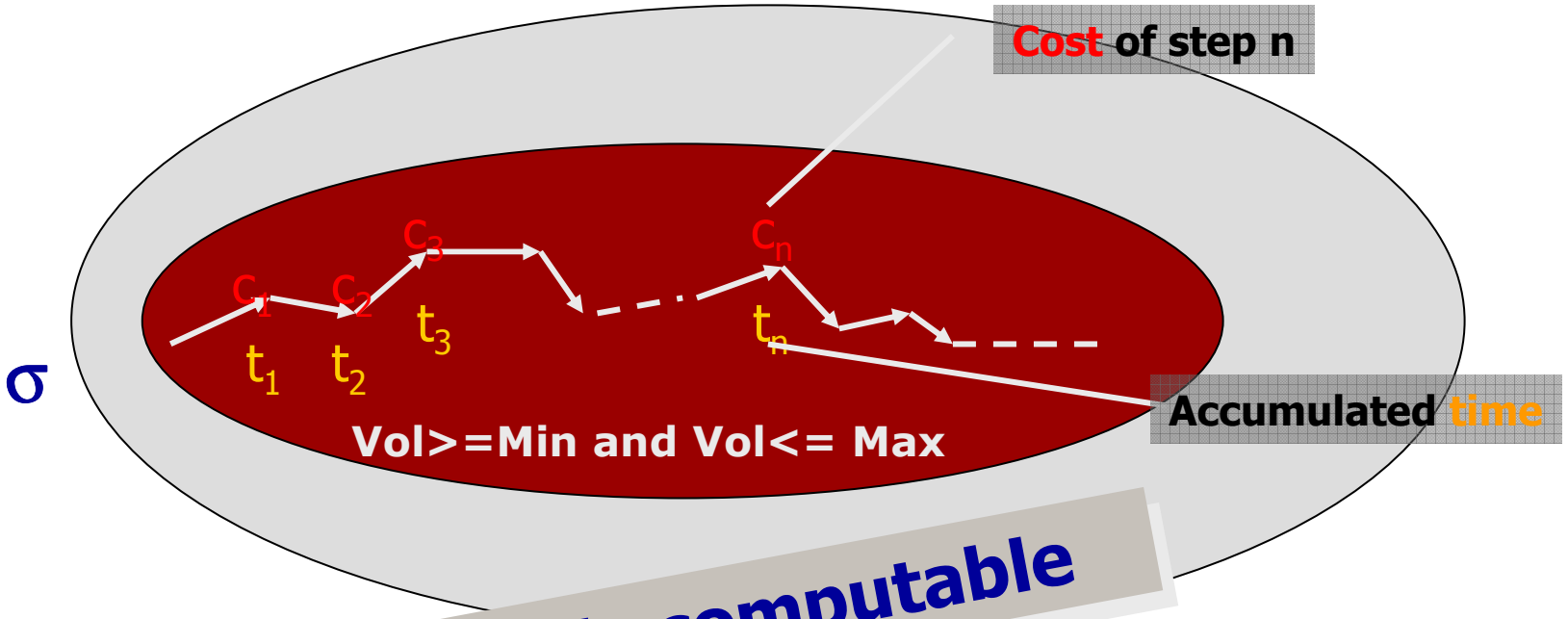
path σ: val(σ) = $\lim_{n!1} ac_n/t_n$

Optimal Schedule σ^*: val(σ^*) = $\inf_\sigma$ val(σ)

Optimal Infinite Scheduling I

Discounting

$\lambda < 1$: discounting factor



THEOREM: σ^* is computable

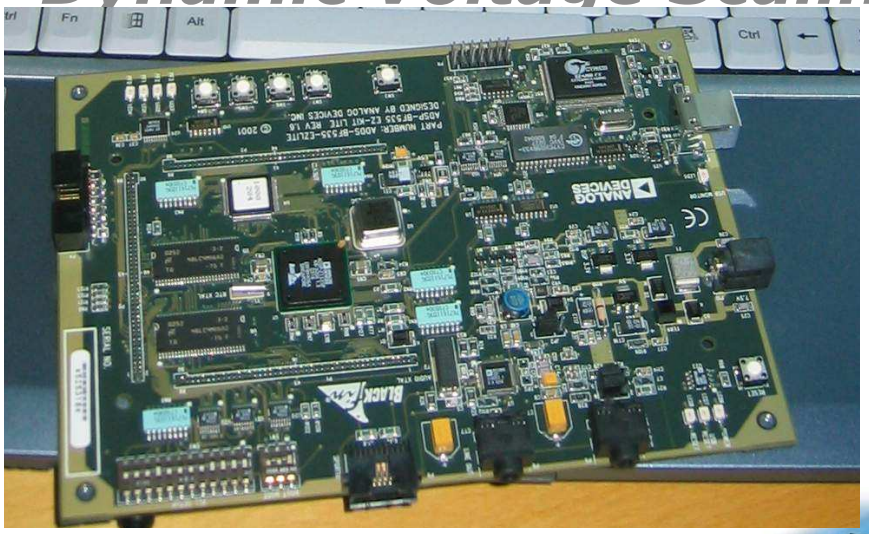
Fahrenberg, Larsen: INFINITY'08

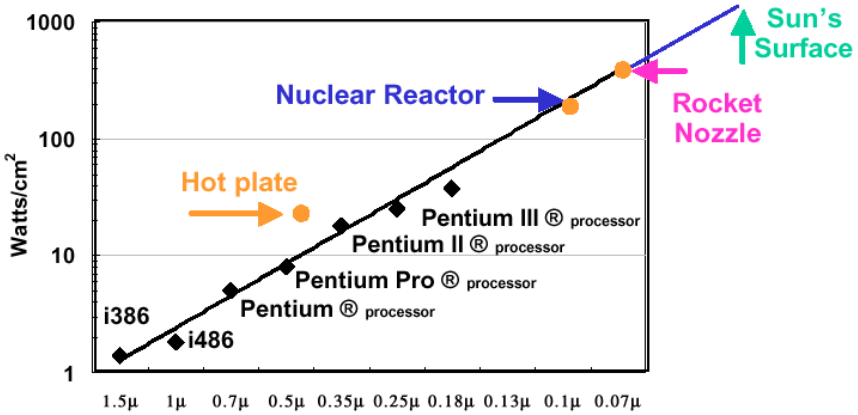
path σ: $\text{val}(\sigma) = \sum_{i=1}^{i=\infty} c_i \cdot \lambda^{t_i}$

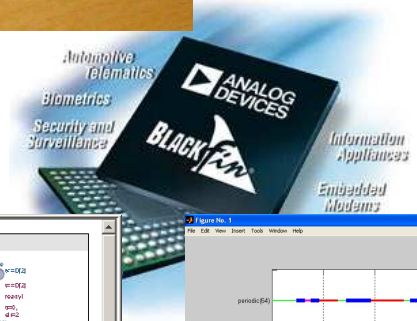
Optimal Schedule σ^*: $\text{val}(\sigma^*) = \inf_\sigma \text{val}(\sigma)$

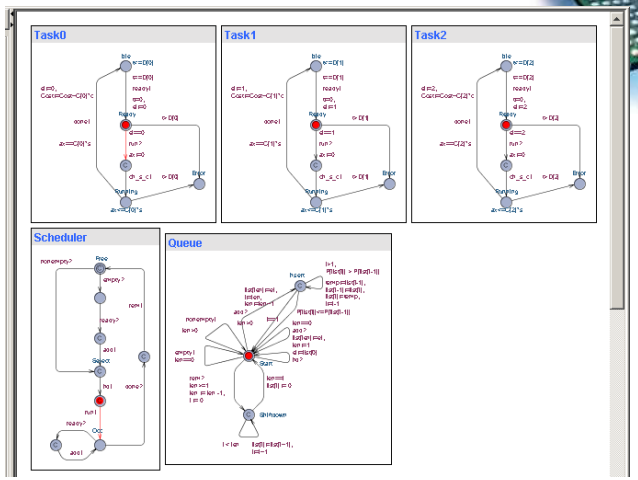
Application

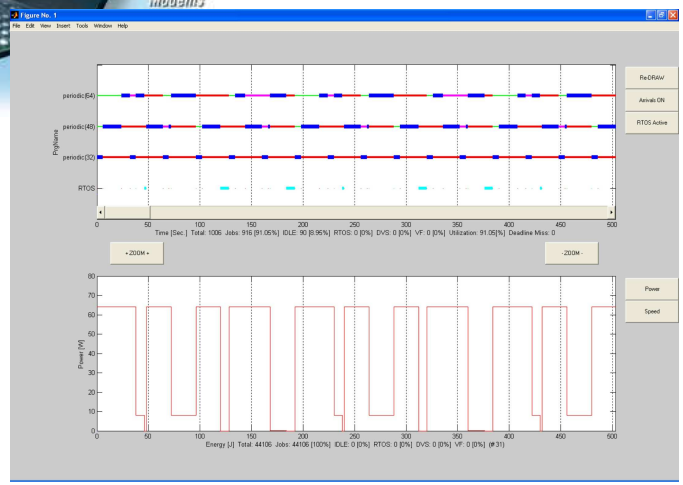
Dynamic Voltage Scaling

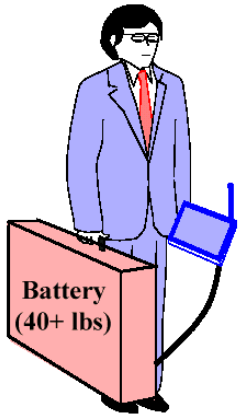










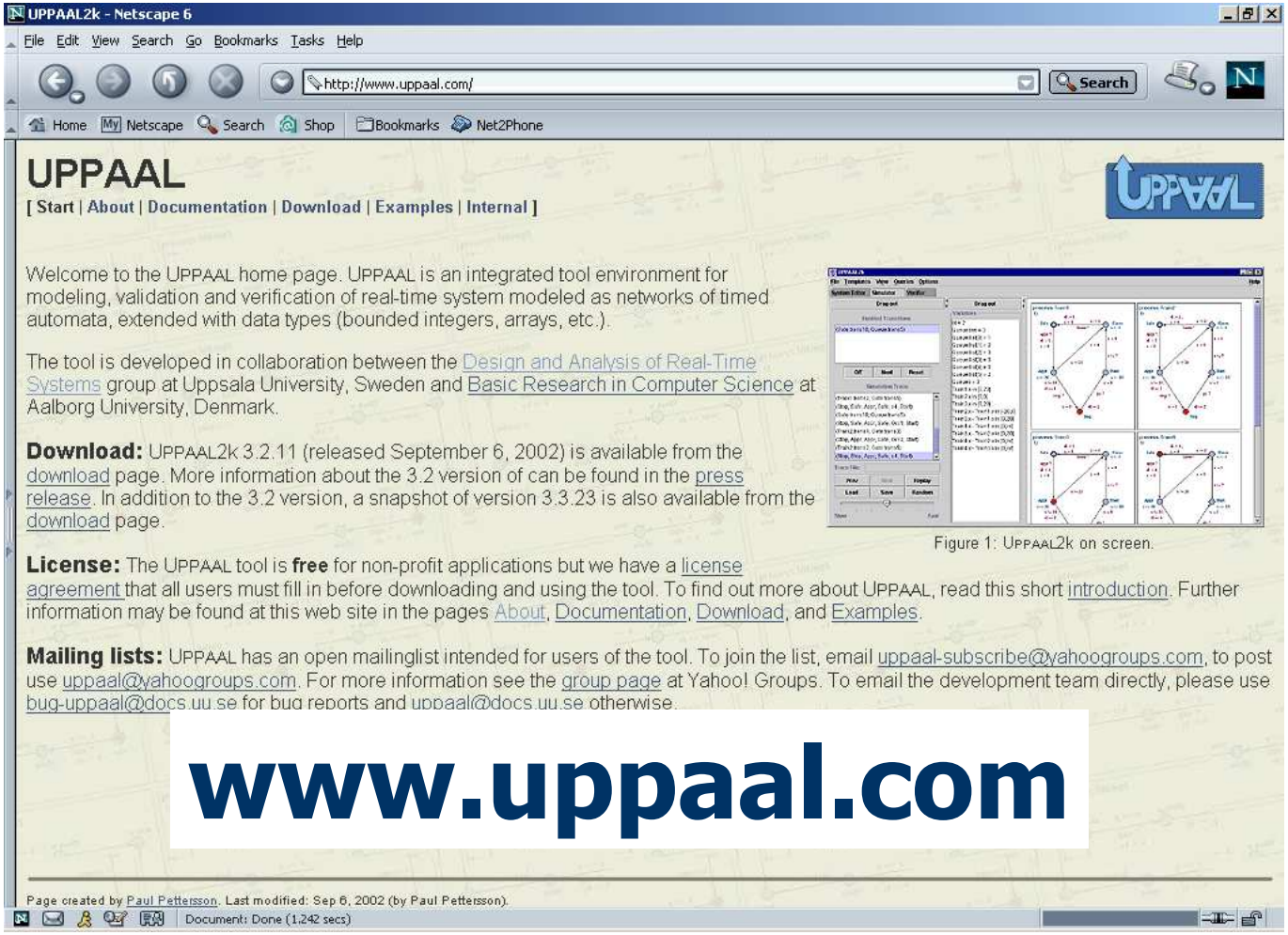


UCb

Future Work & Challenges

- 😐 Optimal Constrained Infinite Strategies for Multi-Priced TA
- ☹ / ☺ Model Checking wrt PTCTL
  [Raskin et al, FORMATS'04]
  [Bouyer, Larsen, Markey FoSSACS07]
- ☹ Efficient implementations of Optimal Infinite Scheduling
- 😐 Negative Cost Rates → Cost Bounded Infinite Runs
  [Bouyer, Fahrenberg,Larsen, Markey, Srba FORMATS08]

Further Information



UCb

TIGA

Real Time Controller Synthesis

with

Gerd Behrmann, Patricia Bouyer,
Franck Cassez, Agnes Counard, Alexandre David
Emmanuel Fleury, Didier Lime, Nicolas Markey,
Jean-Francois Raskin

AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

CSS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

RELATED SITES: UPPAAL

UPPAAL TIGA

UPPAAL for Timed Games

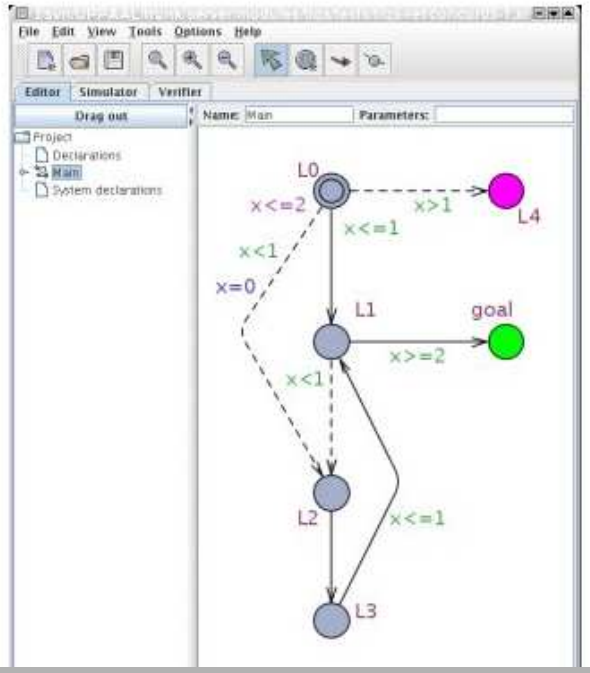
www.uppaal.com

Main Page | Download | Contact us

Welcome!

UPPAAL TIGA (Fig. 1) is an extension of UPPAAL [BDL04] and it implements the first efficient on-the-fly algorithm for solving games based on timed game automata with respect to reachability and safety properties. Though timed games for long have been known to be decidable there has until now been a lack of efficient and truly on-the-fly algorithms for their analysis.

The algorithm we propose [CDFLL05] is a symbolic extension of the on-the-fly algorithm suggested by Liu & Smolka [LS98] for linear-time model-checking of finite-state systems. Being on-the-fly, the symbolic algorithm may terminate long before having explored the entire state-space. Also the individual steps of the algorithm are carried out efficiently by the use of so-called zones as the underlying data structure. Our tool implements various optimizations of the basic sy… algorithm, as well as methods for ob… time-optimal winning strategies



Latest News

Versions 0.10 and 0.11 released.

7 July 2007

Versions 0.10 and 0.11 are released today. Version 0.11 contains a new concrete simulator that allows the user to play strategies from the GUI. Both versions fix the following bugs: maximal constants in the formula are now taken into account, the command line simulator is new and works better, delay when no clock was used, better user feedback, end-of-game detection fixed, other bugs involving delays in the strategy, precision problems in the simulator, and leak in the DBM library. These new versions have also the following new features: options to control the type of strategy output, better control on the search ordering (forward and backward), cooperative strategies, and time optimal strategies. The manual has been updated to

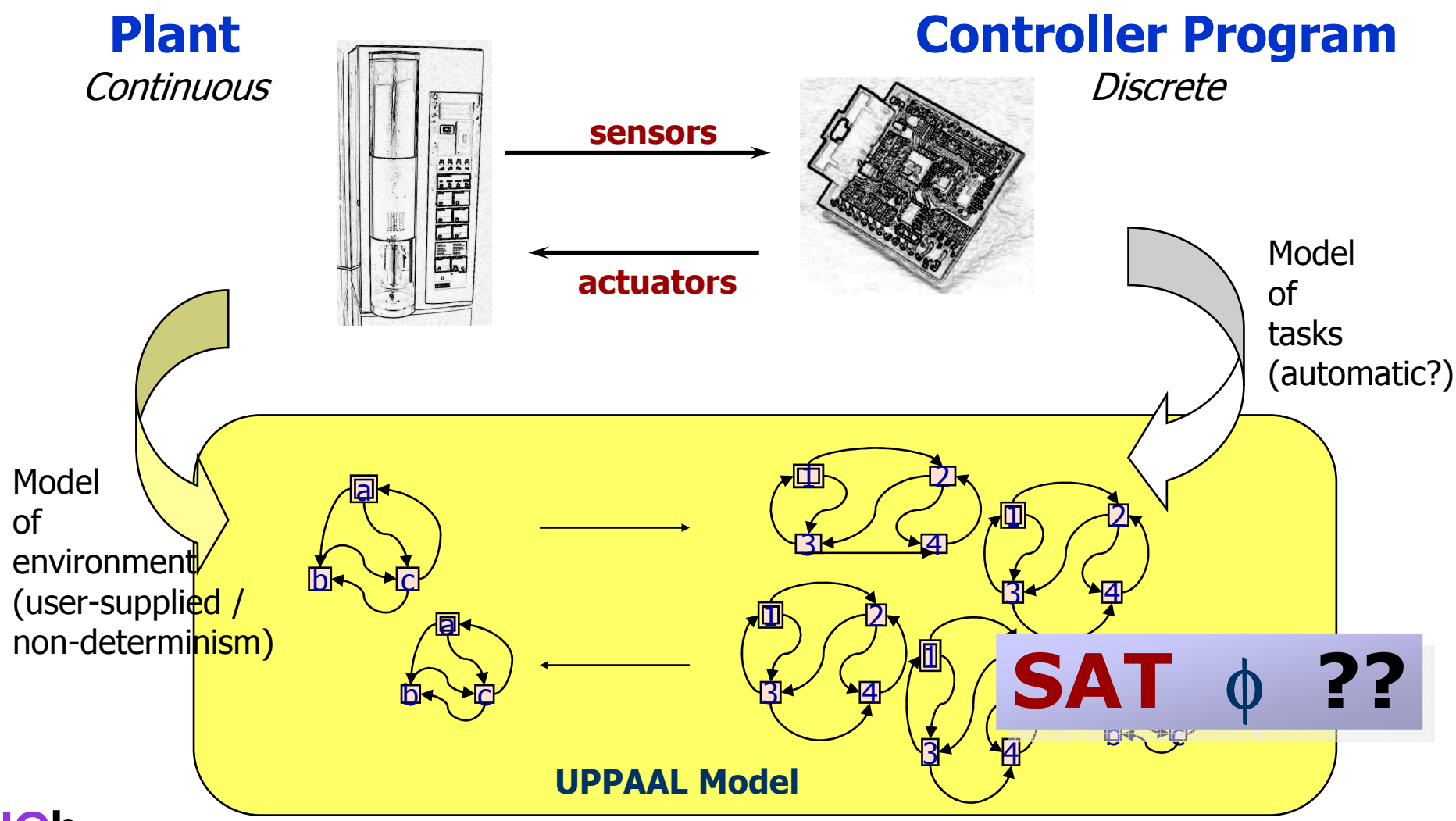
CONCUR 2005, TACAS07, CAV07

UPPSALA UNIVERSITET

AALBORG UNIVERSITY

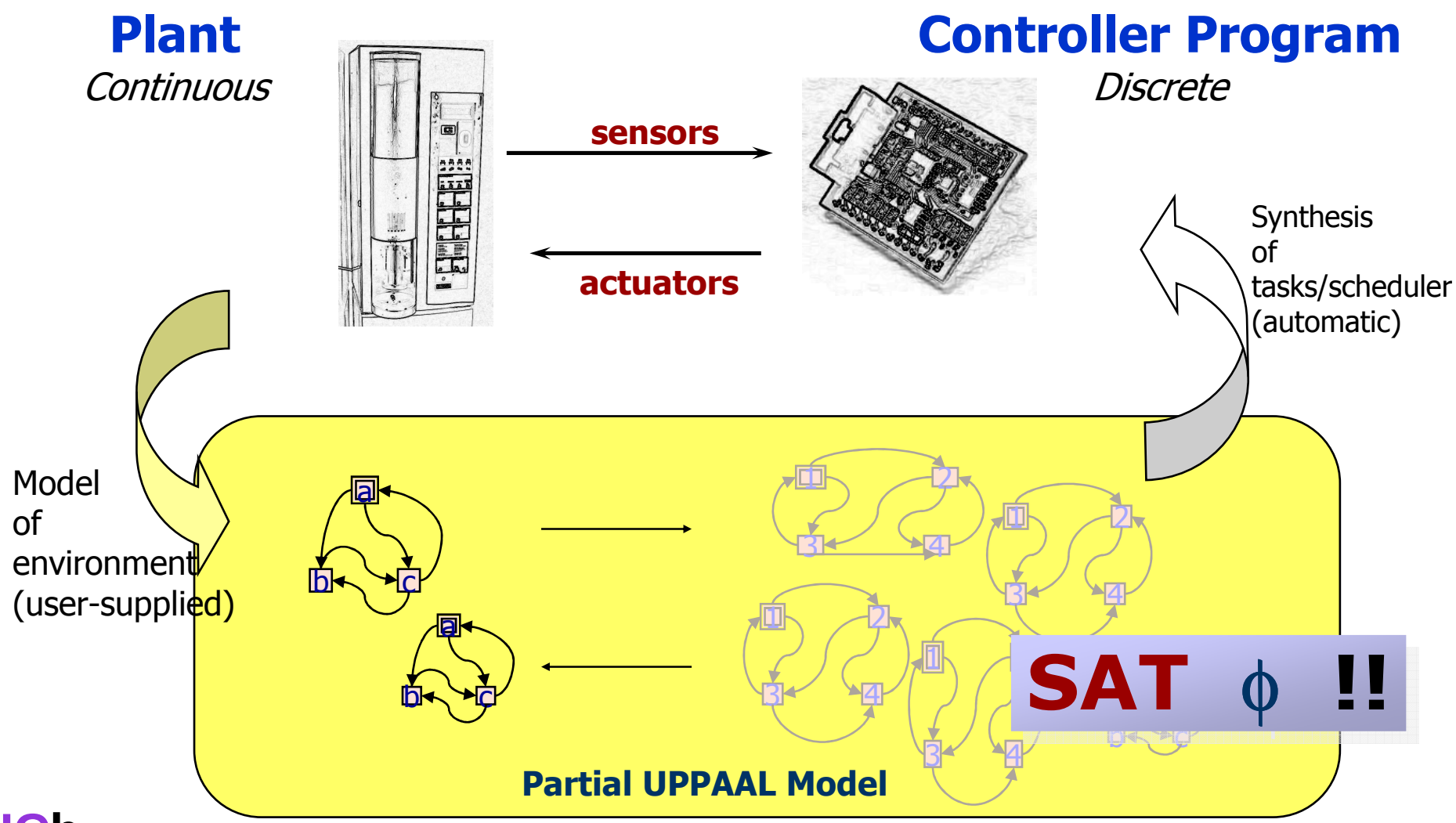
UCb

Model Checking



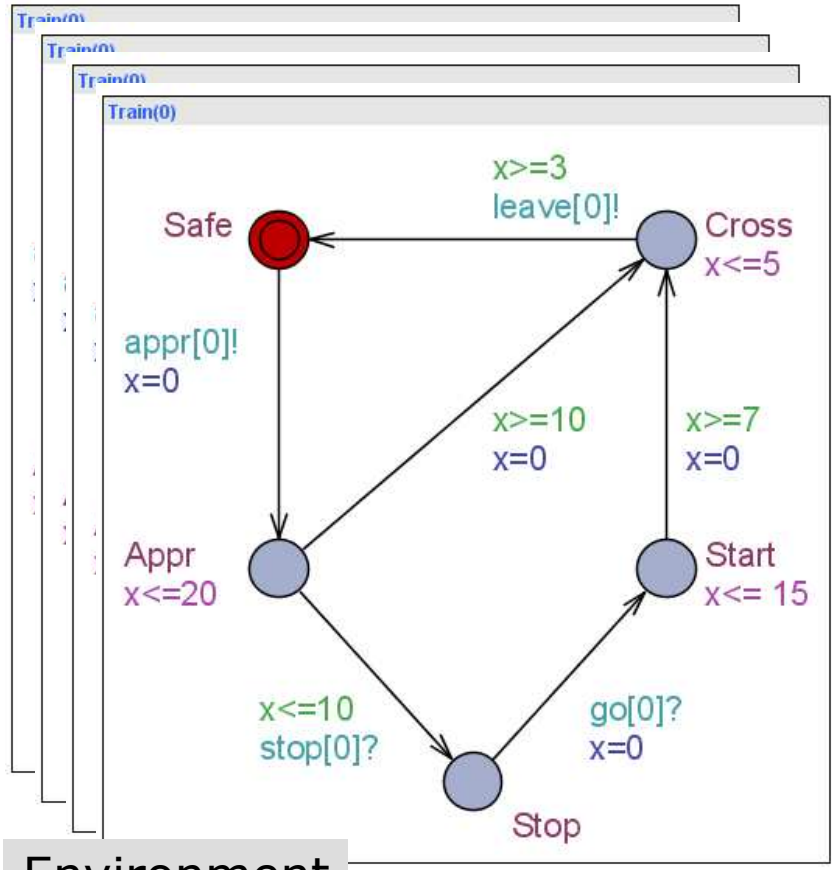
UCb

Synthesis

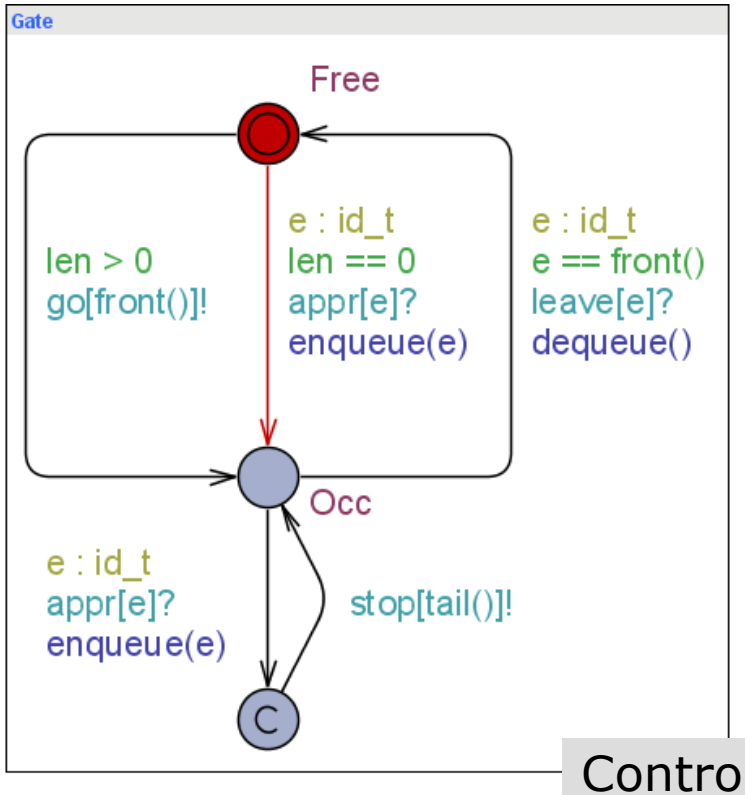


UCb

Model Checking



Environment

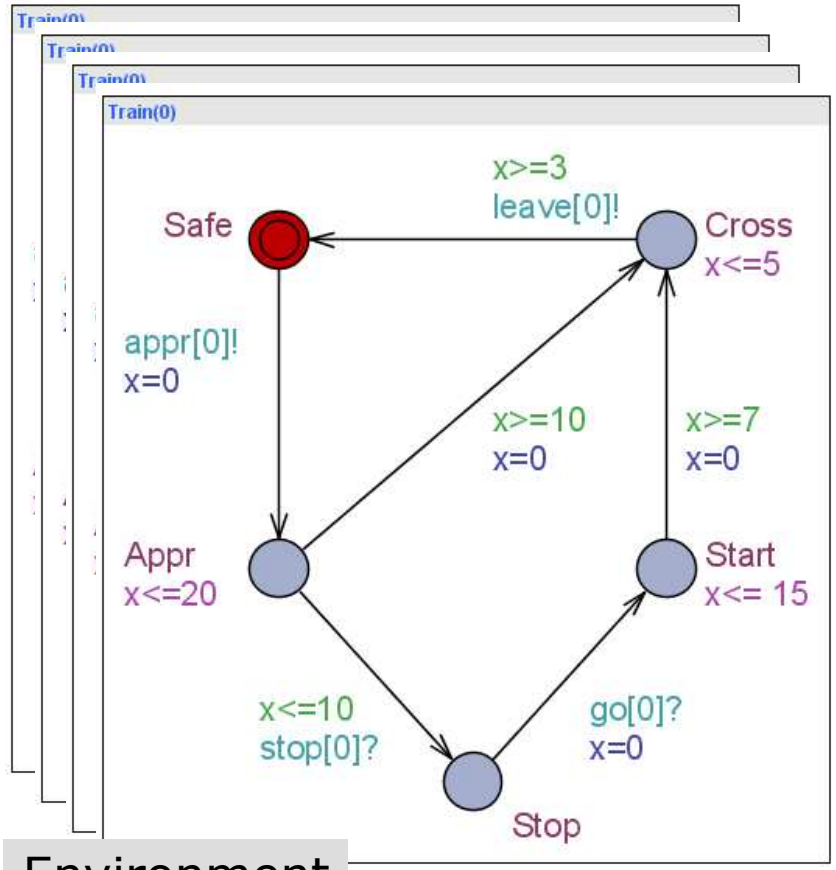


Controller

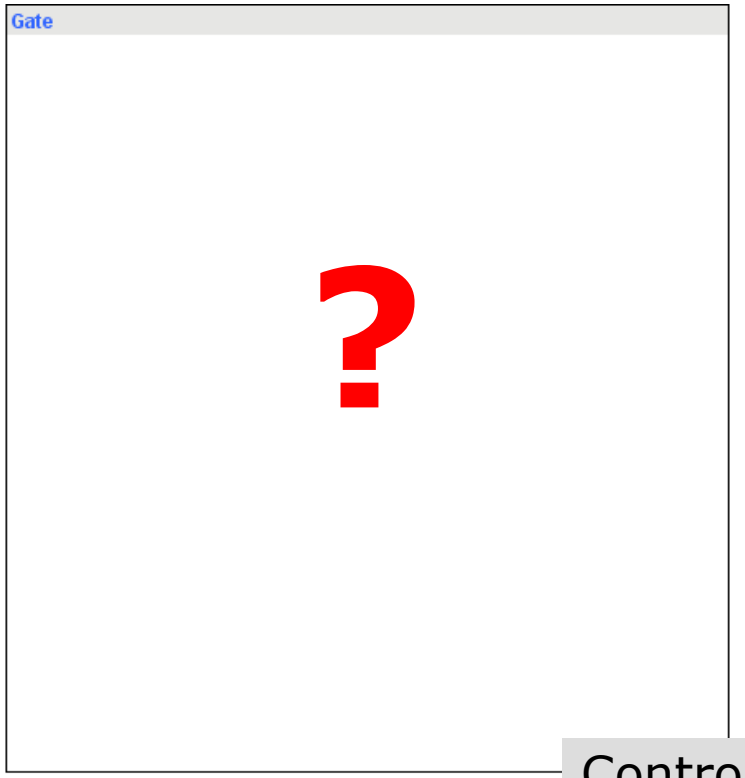
ϕ: Never two trains at the crossing at the same time

UCb

Synthesis



Environment

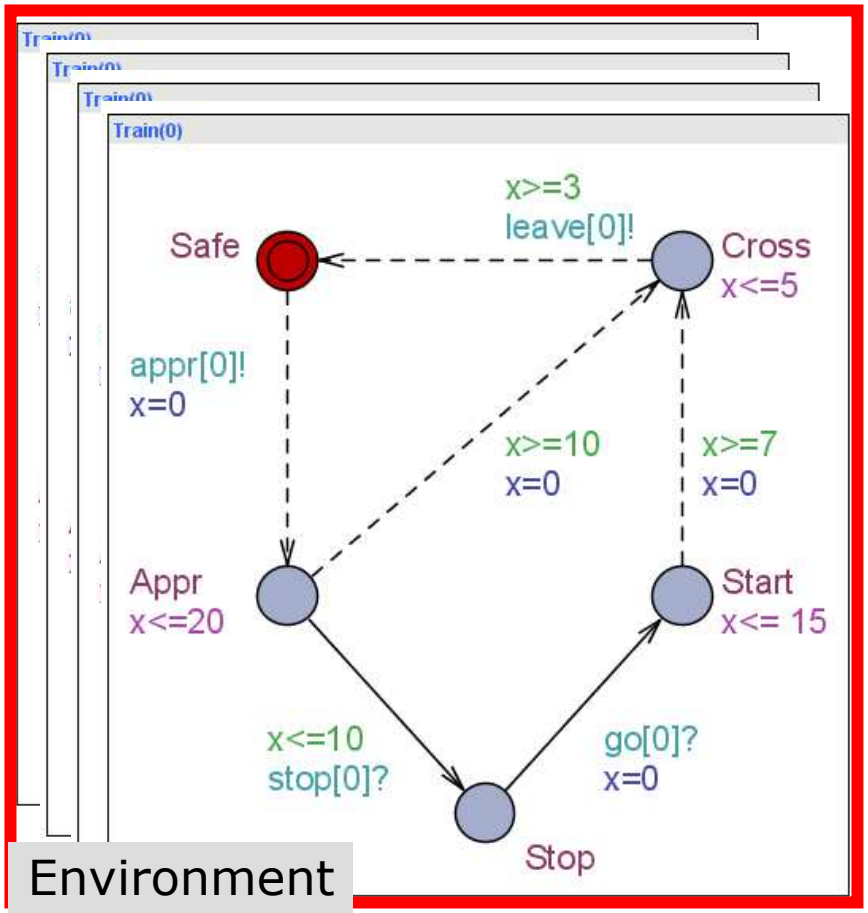


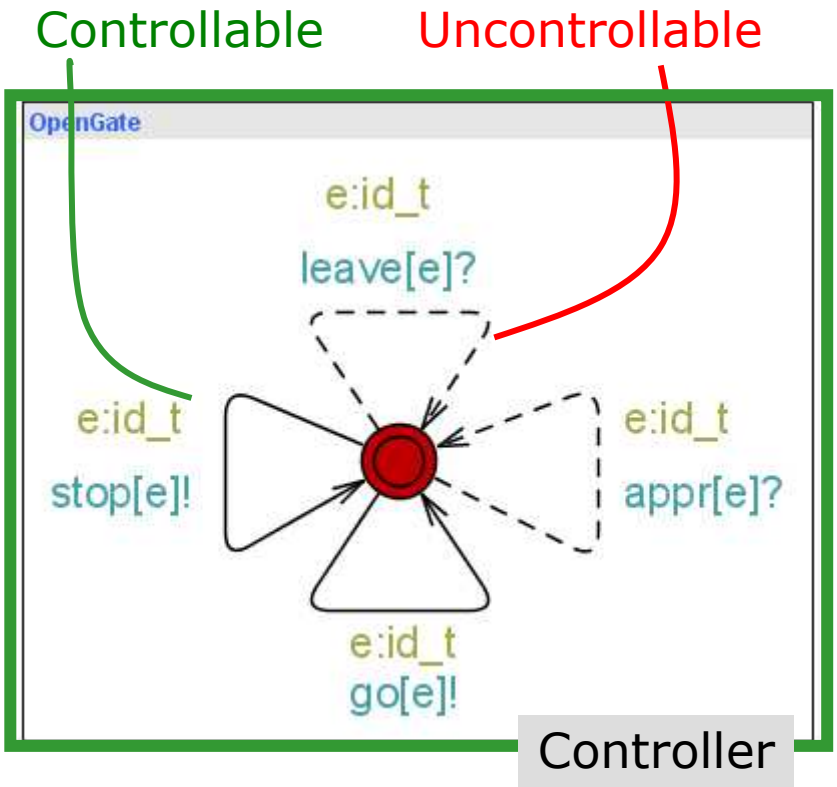
Controller

ϕ: Never two trains at the crossing at the same time

Synthesis

Two Player Game





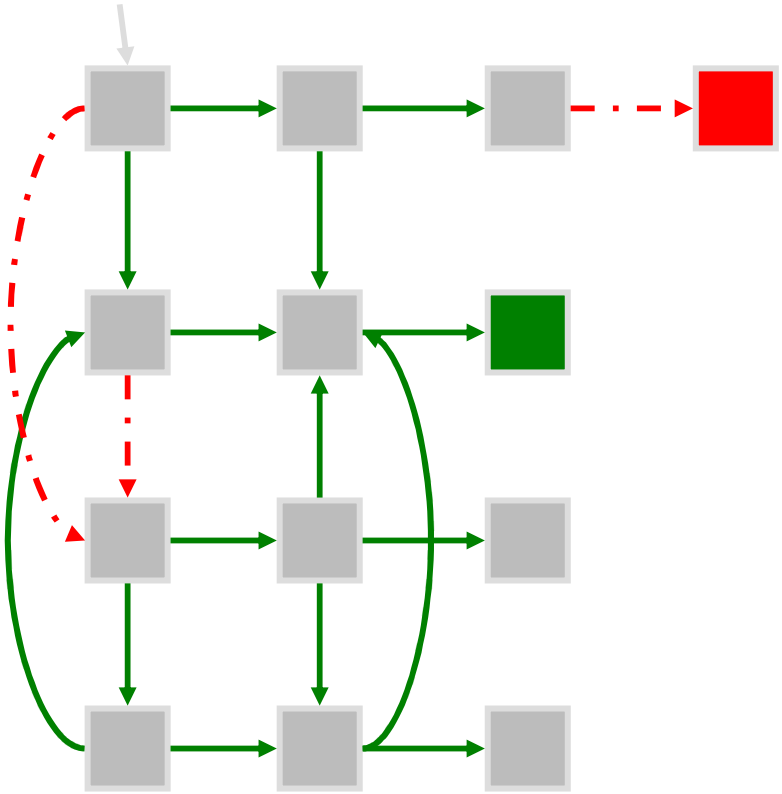
Find strategy for controllable actions st behaviour satisfies ϕ

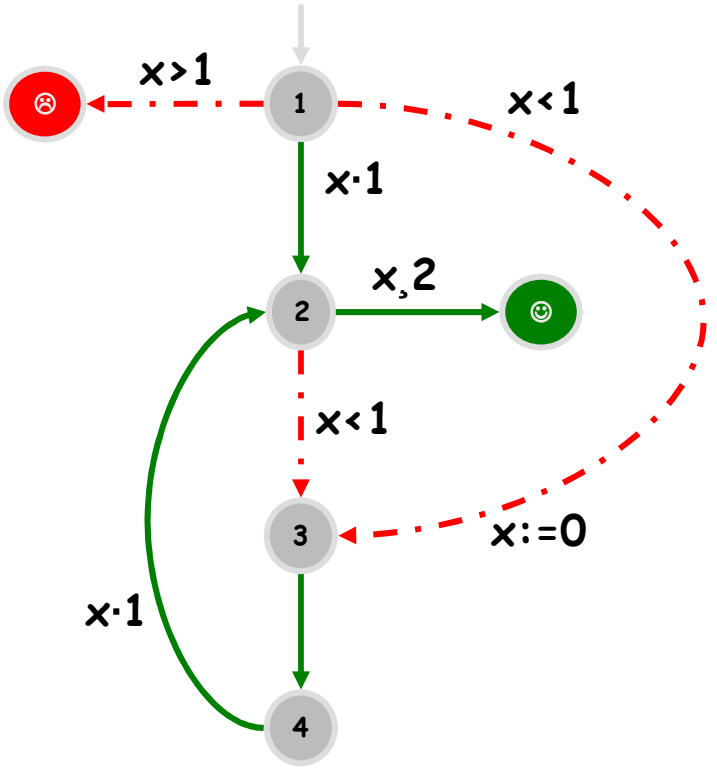
ϕ: Never two trains at the crossing at the same time

UCb

Untimed and Timed Games

Reachability / Safety Games



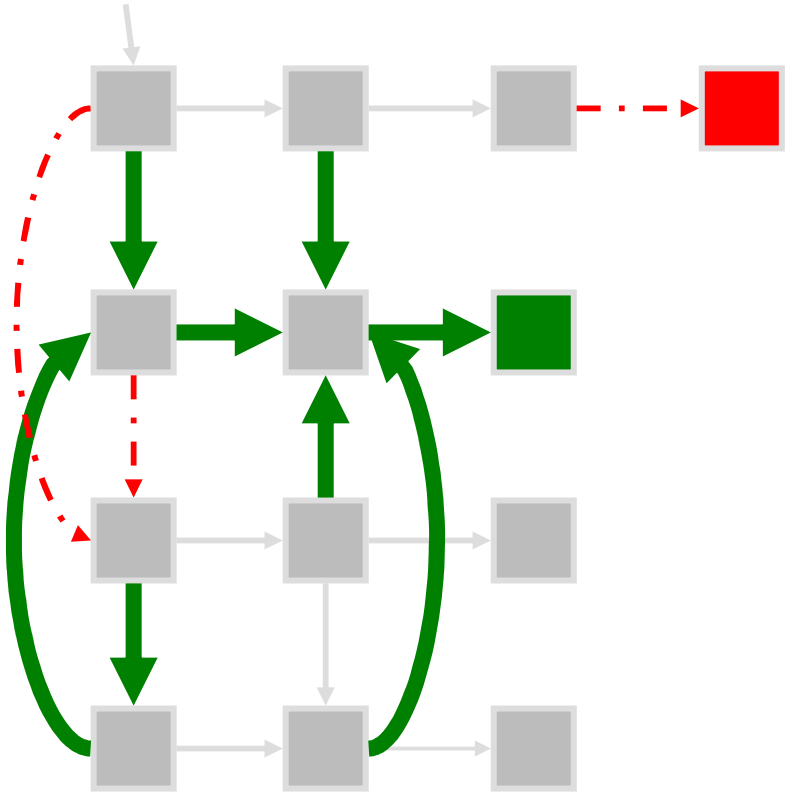


Uncontrollable

Controllable

UCb

Untimed Games



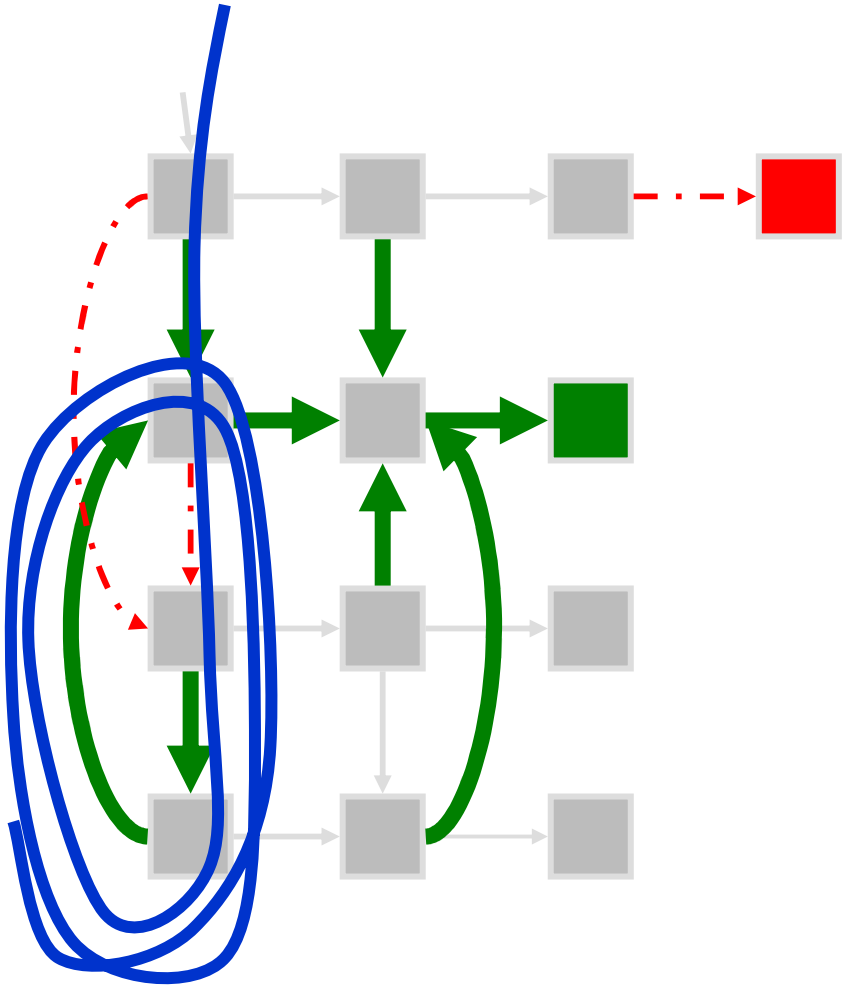
Memoryless Strategy:
$F : Q \rightarrow E_c$

Uncontrollable

Controllable

UCb

Untimed Games



Memoryless Strategy:

$F : Q \rightarrow E_c$

Winning Run ρ:

States(ρ) Å $G \neq \emptyset$

Winning Strategy:

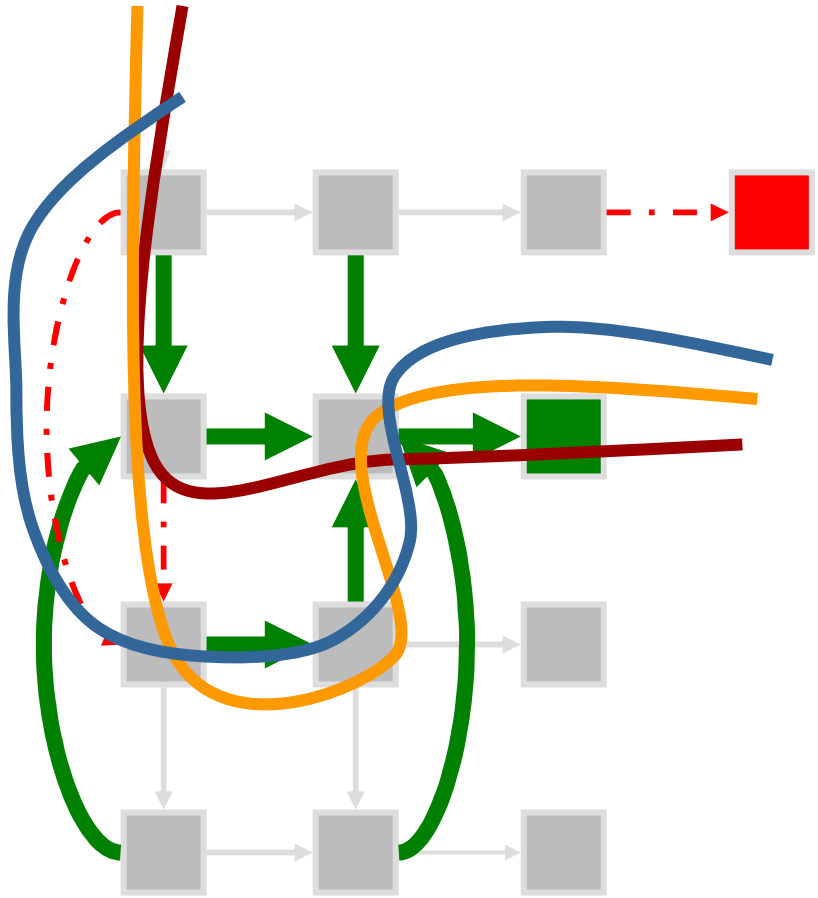
Runs(F) µ WinRuns

Uncontrollable

Controllable

UCb

Untimed Games



Memoryless Strategy:

$F : Q \rightarrow E_c$

Winning Run ρ :

States(ρ) Å $G \neq \emptyset$

Winning Strategy:

Runs(F) µ WinRuns

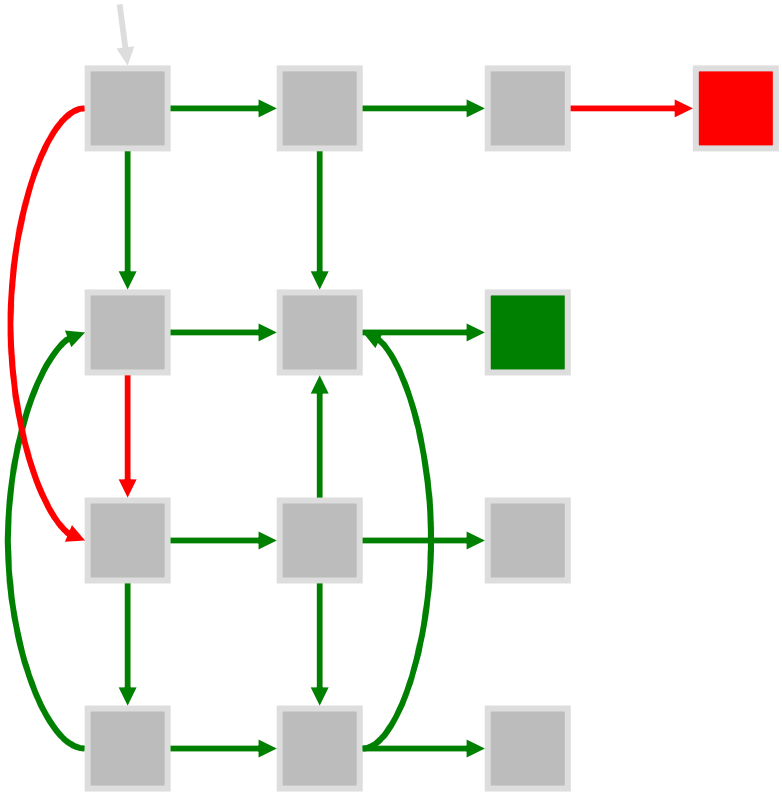
Uncontrollable

Controllable

UCb

Untimed Games

Backwards Fixed-Point Computation



$cPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_c q' \}$

$uPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_u q' \}$

$\pi(X) = cPred(X) \setminus uPred(X^C)\ ]$

Theorem:
The set of winning states is obtained as the least fixpoint of the function:

$$X \mapsto \pi(X)\ [\ Goal$$

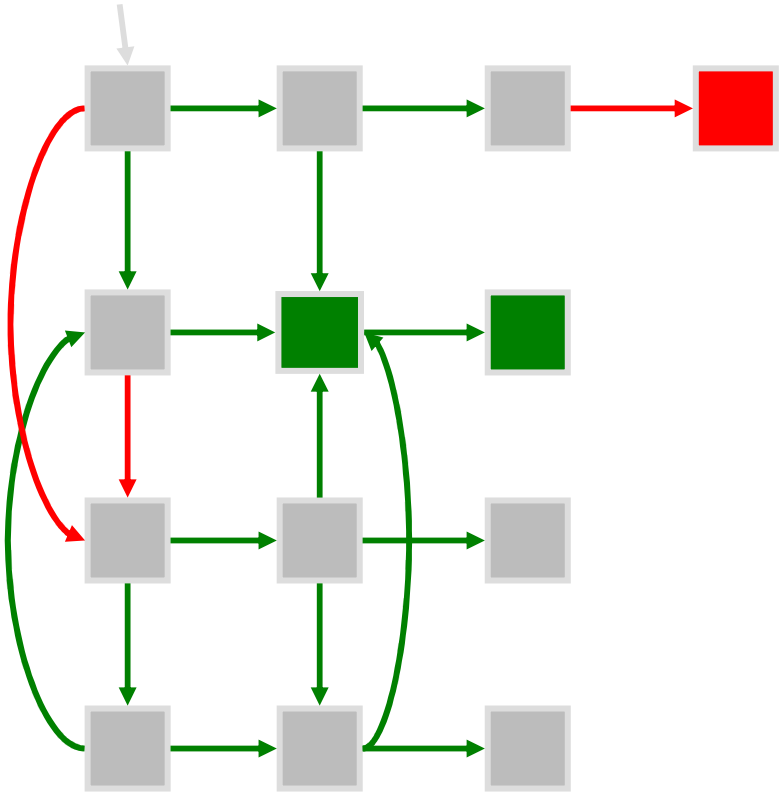
→ Uncontrollable

→ Controllable

UCb

Untimed Games

Backwards Fixed-Point Computation



$cPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_c q' \}$

$uPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_u q' \}$

$\pi(X) = cPred(X) \setminus uPred(X^C)\ ]$

Theorem:
The set of winning states is obtained as the least fixpoint of the function:

$$X \mapsto \pi(X)\ [\ Goal$$

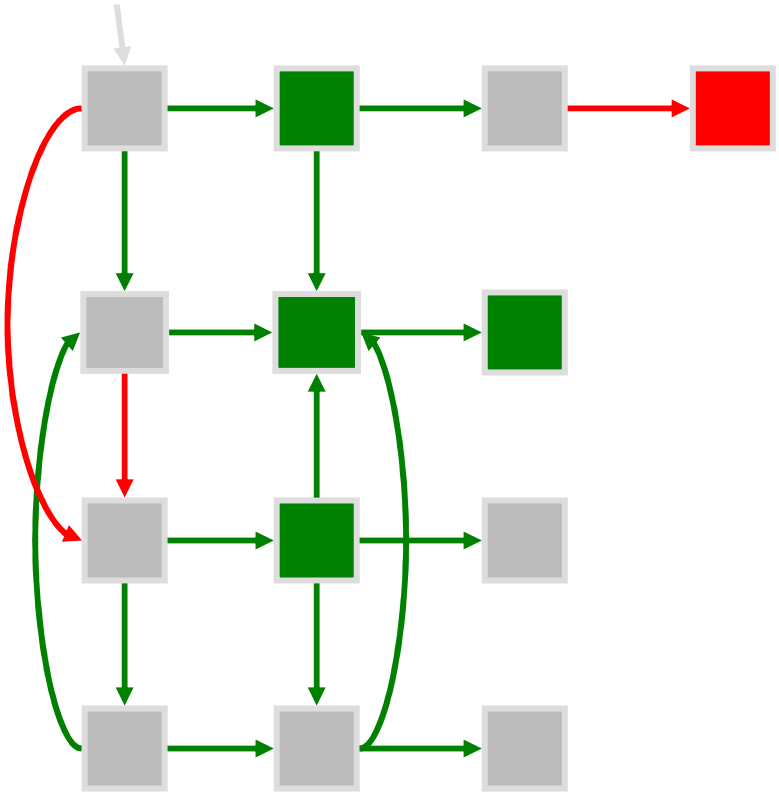
→ Uncontrollable

→ Controllable

UCb

Untimed Games

Backwards Fixed-Point Computation



$cPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_c q' \}$

$uPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_u q' \}$

$\pi(X) = cPred(X) \setminus uPred(X^C)\ ]$

Theorem:
The set of winning states is obtained as the least fixpoint of the function:

$$X \mapsto \pi(X)\ [\ Goal$$

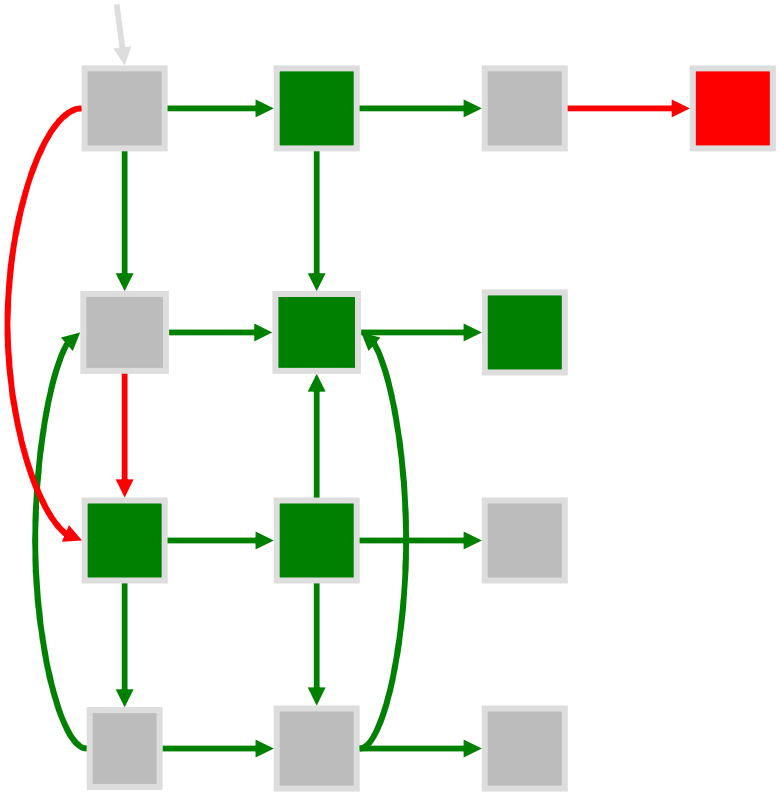
→ Uncontrollable

→ Controllable

UCb

Untimed Games

Backwards Fixed-Point Computation



$cPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_c q' \}$

$uPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_u q' \}$

$\pi(X) = cPred(X) \setminus uPred(X^C)\ ]$

Theorem:
The set of winning states is obtained as the least fixpoint of the function:

$$X \mapsto \pi(X)\ [\ Goal$$

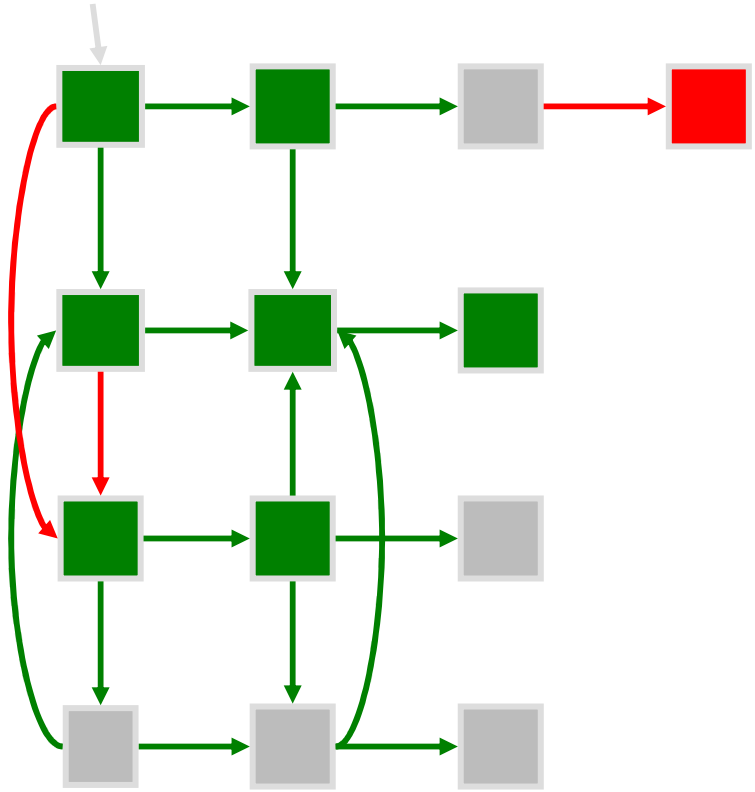
→ Uncontrollable

→ Controllable

UCb

Untimed Games

Backwards Fixed-Point Computation



$cPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_c q' \}$

$uPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_u q' \}$

$\pi(X) = cPred(X) \setminus uPred(X^C)\ ]$

Theorem:
The set of winning states is obtained as the least fixpoint of the function:

$X \mapsto \pi(X)\ [\ Goal$

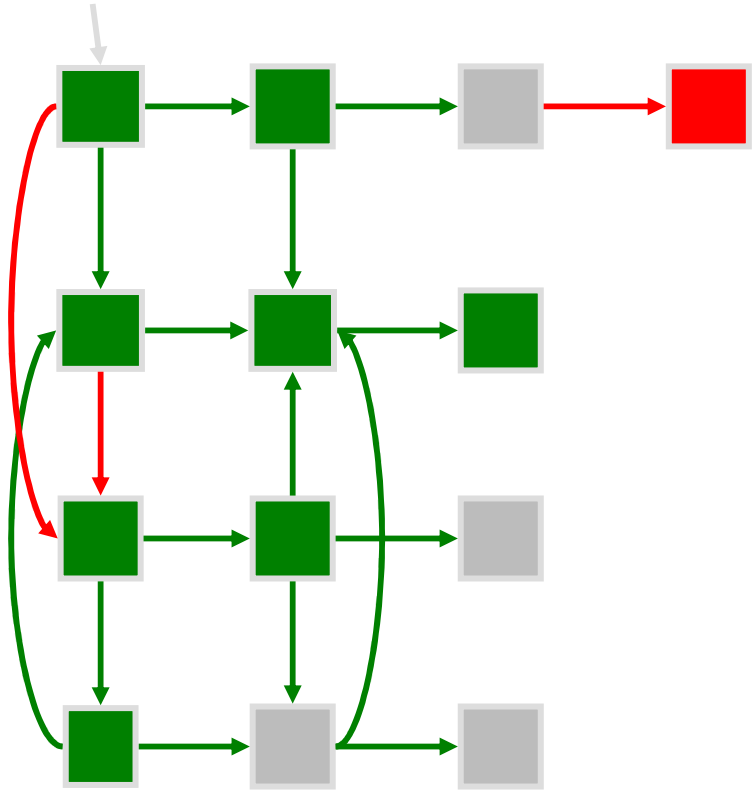
→ Uncontrollable

→ Controllable

UCb

Untimed Games

Backwards Fixed-Point Computation



$cPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_c q' \}$

$uPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_u q' \}$

$\pi(X) = cPred(X) \setminus uPred(X^C)\ ]$

Theorem:
The set of winning states is obtained as the least fixpoint of the function:

$$X \mapsto \pi(X)\ [\ Goal$$

→ Uncontrollable

→ Controllable

UCb

Timed Games

Memoryless Strategy:

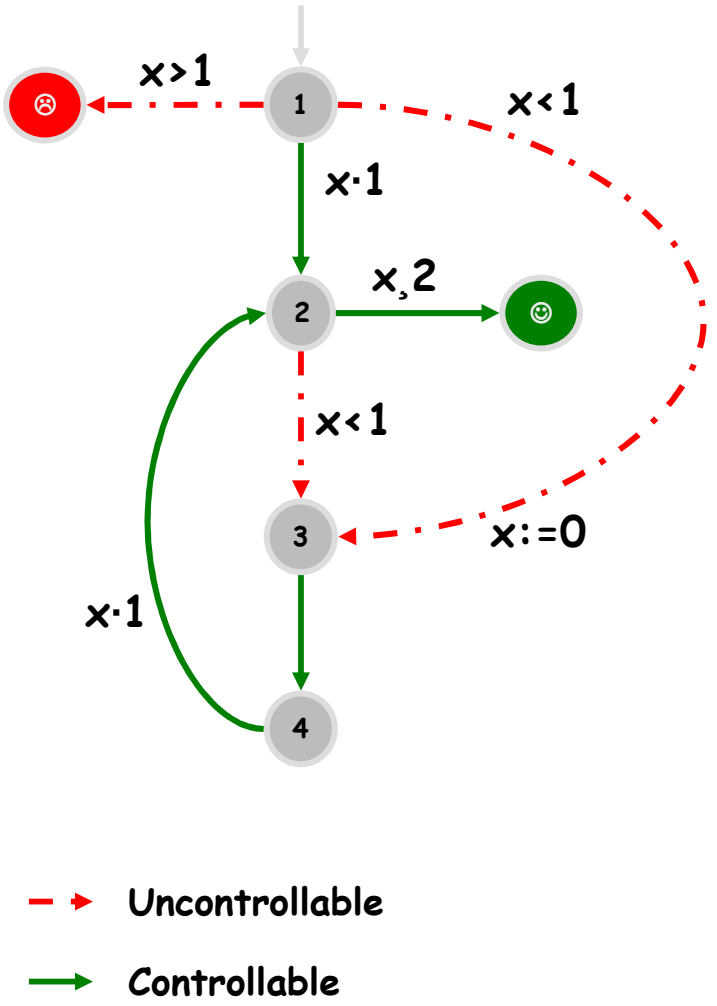
$F : Q \rightarrow E_c [\lambda$

Winning Run:

States(ρ) Å $G \neq \emptyset$

Winning Strategy:

Runs(F) µ WinRuns



Timed Games

Memoryless Strategy:

$F : Q \rightarrow E_c [\ \lambda$

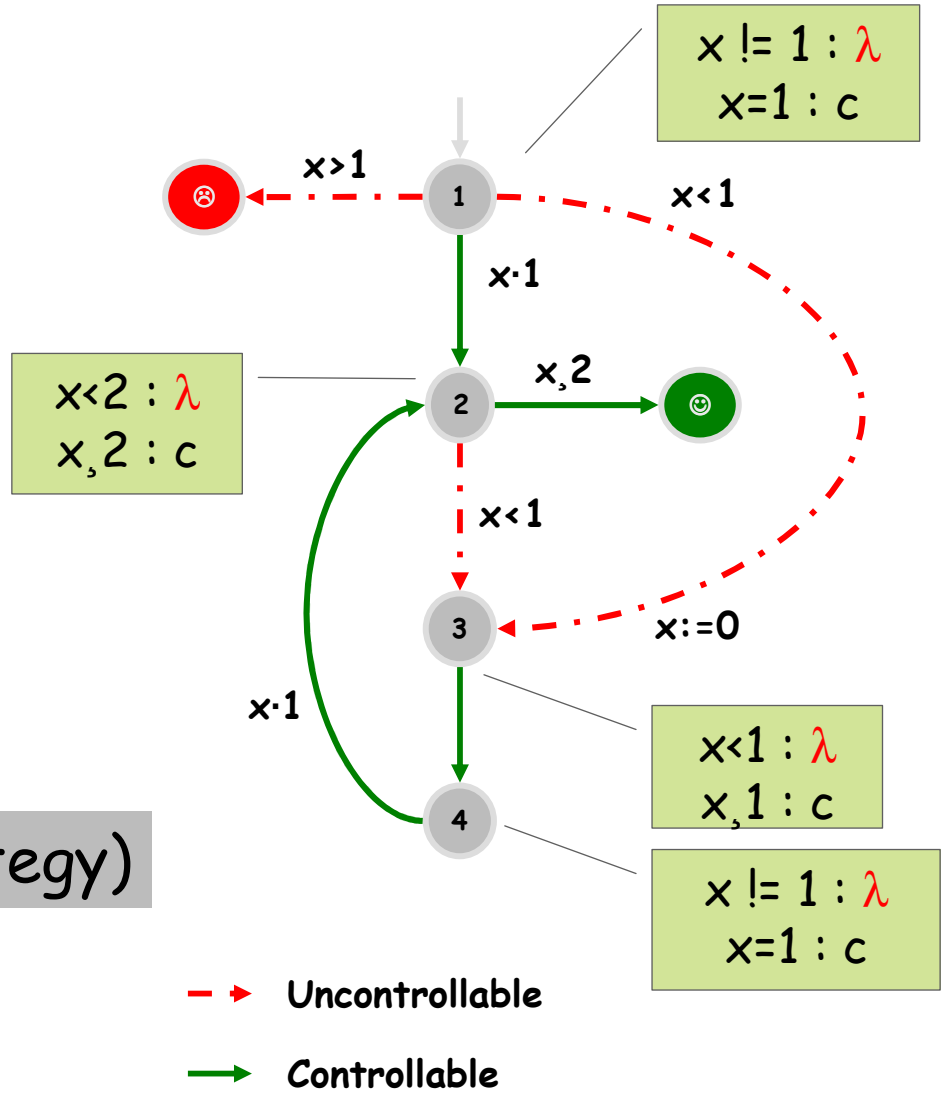
Winning Run:

States(ρ) Å $G \neq \emptyset$

Winning Strategy:

Runs(F) µ WinRuns

Winning (memoryless) strategy)



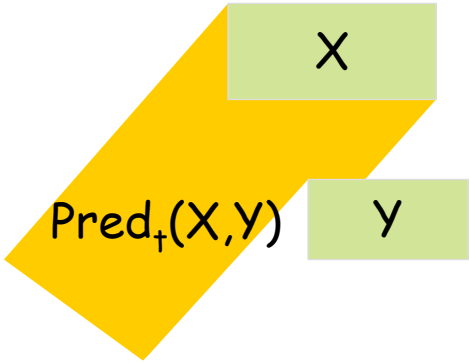
Timed Games – State-of-the-Art

Backwards Fixed-Point Computation

Definitions

$$cPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_c q' \}$$
$$uPred(X) = \{ q \in Q \mid \exists q' \in X.\ q \rightarrow_u q' \}$$
$$Pred_t(X,Y) = \{ q \in Q \mid \exists t.\ q^t \in X \text{ and } \forall s \cdot t.\ q^s \in Y^C \}$$

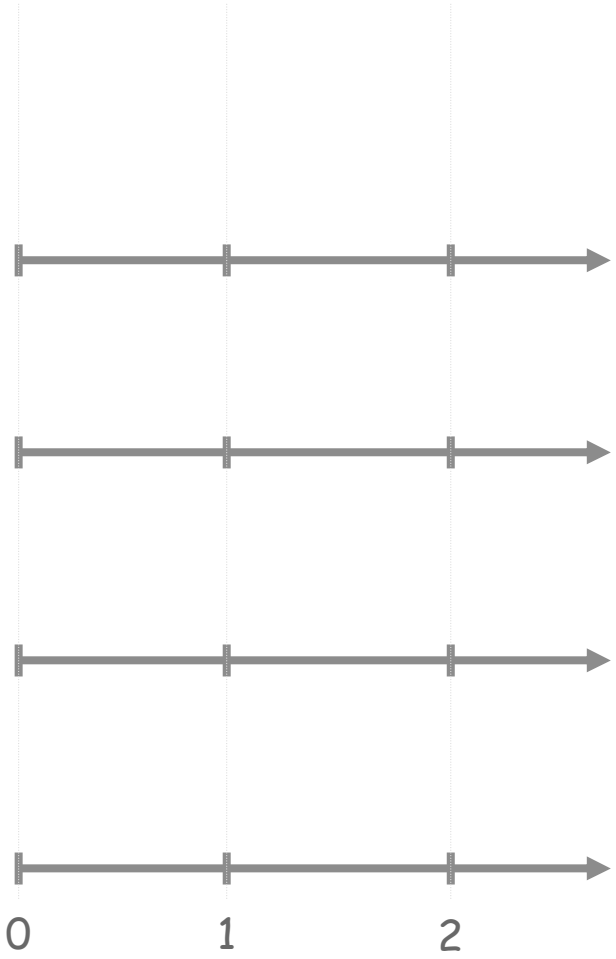
$$\pi(X) = Pred_t[\ X\ [\ cPred(X)\ ,\ uPred(X^C)\ ]$$

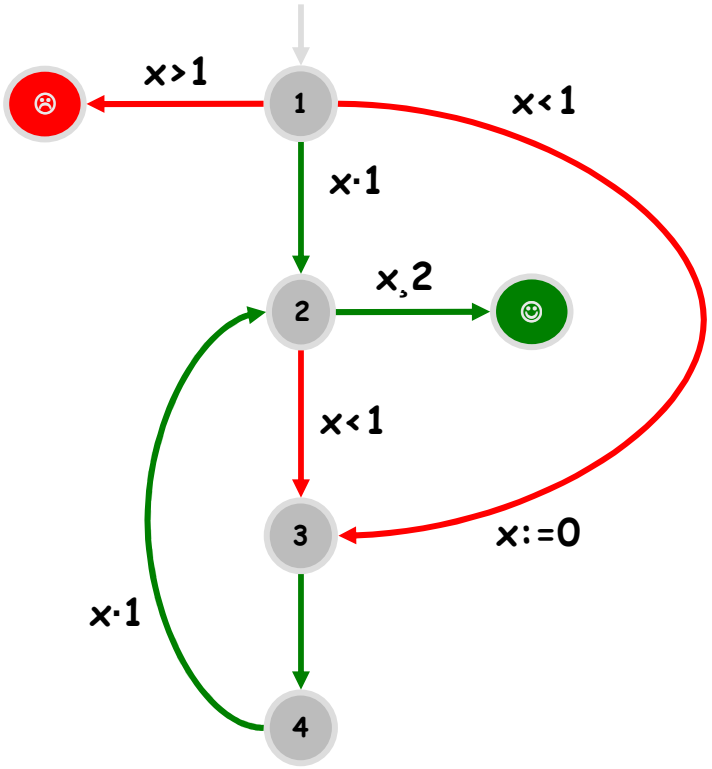


Theorem:
The set of winning states is obtained as the least fixpoint of the function: $X \mapsto \pi(X)\ [\ Goal$

Timed Games – State-of-the-Art

Backwards Fixed-Point Computation

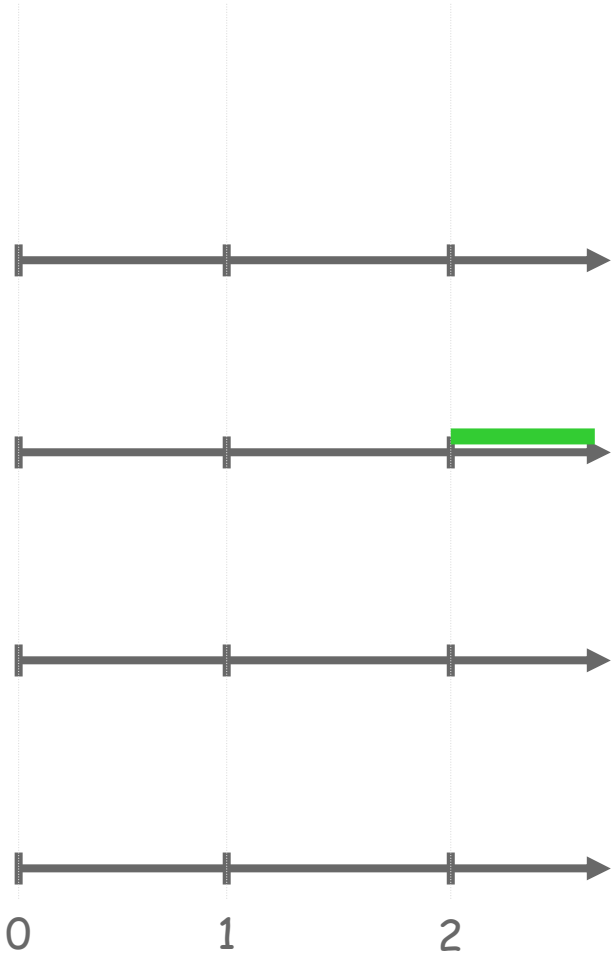


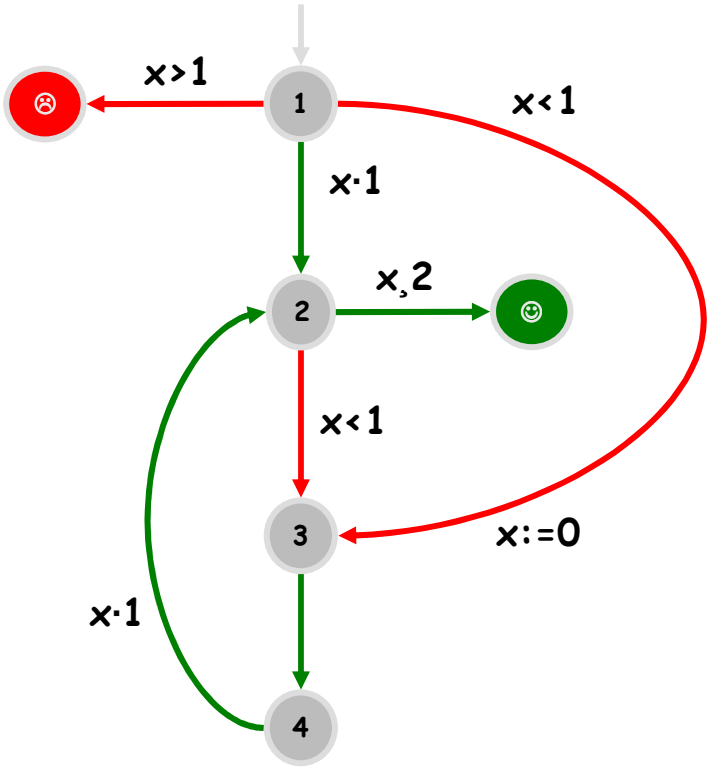


UCb

Timed Games – State-of-the-Art

Backwards Fixed-Point Computation

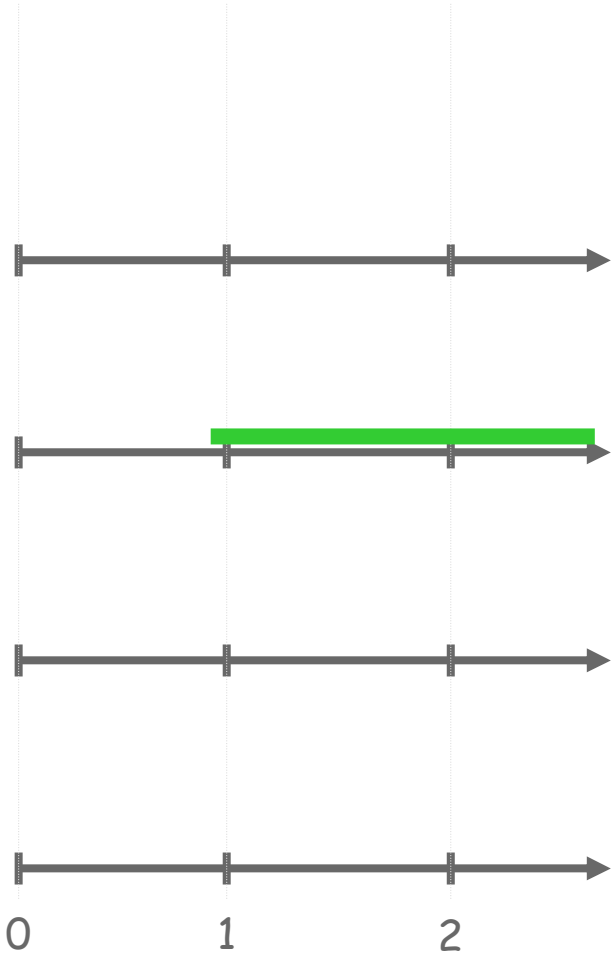


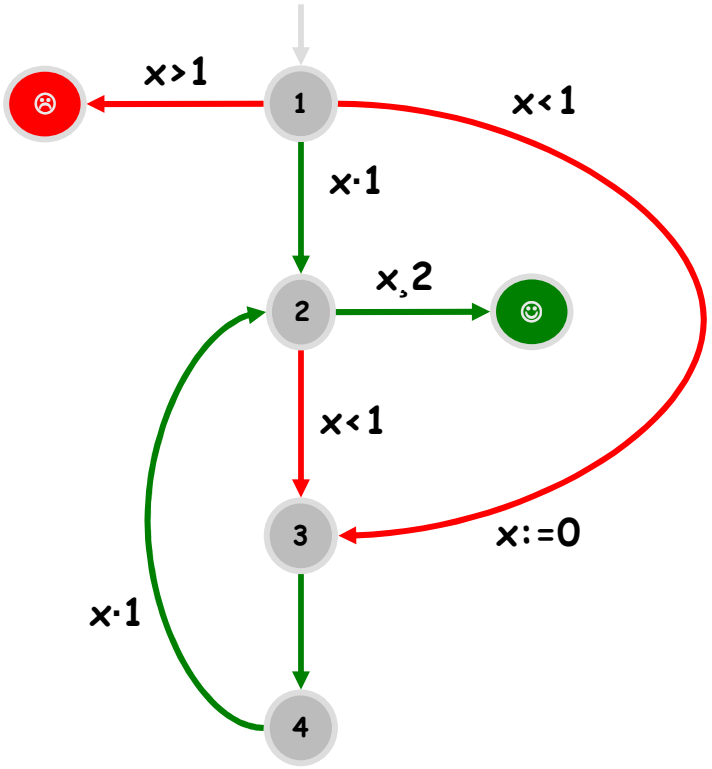


UCb

Timed Games – State-of-the-Art

Backwards Fixed-Point Computation

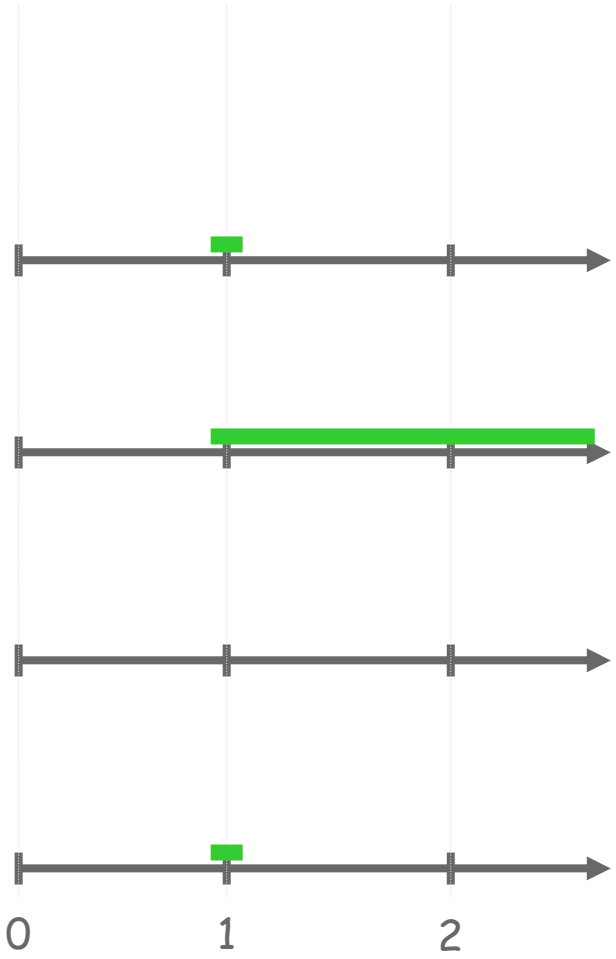


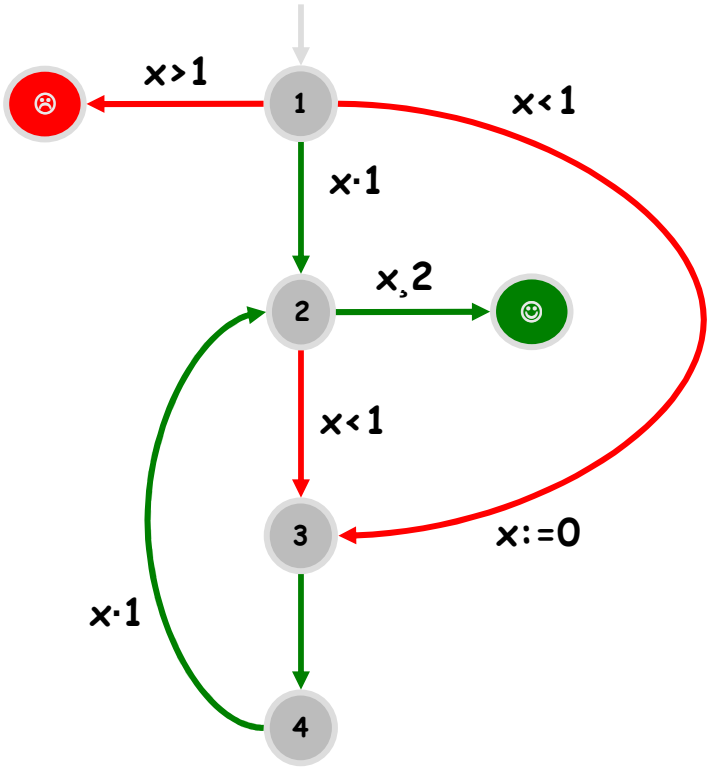


UCb

Timed Games – State-of-the-Art

Backwards Fixed-Point Computation

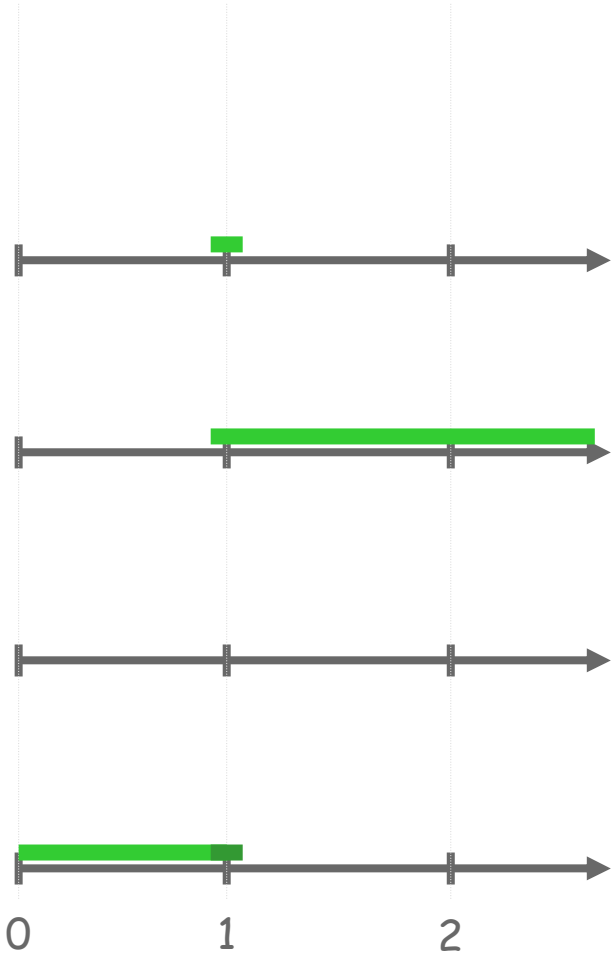


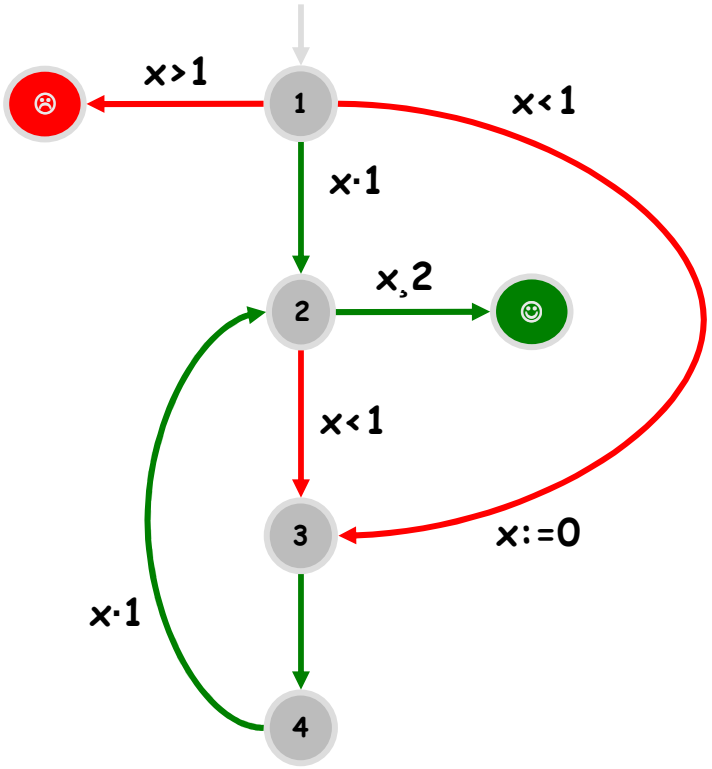


UCb

Timed Games – State-of-the-Art

Backwards Fixed-Point Computation

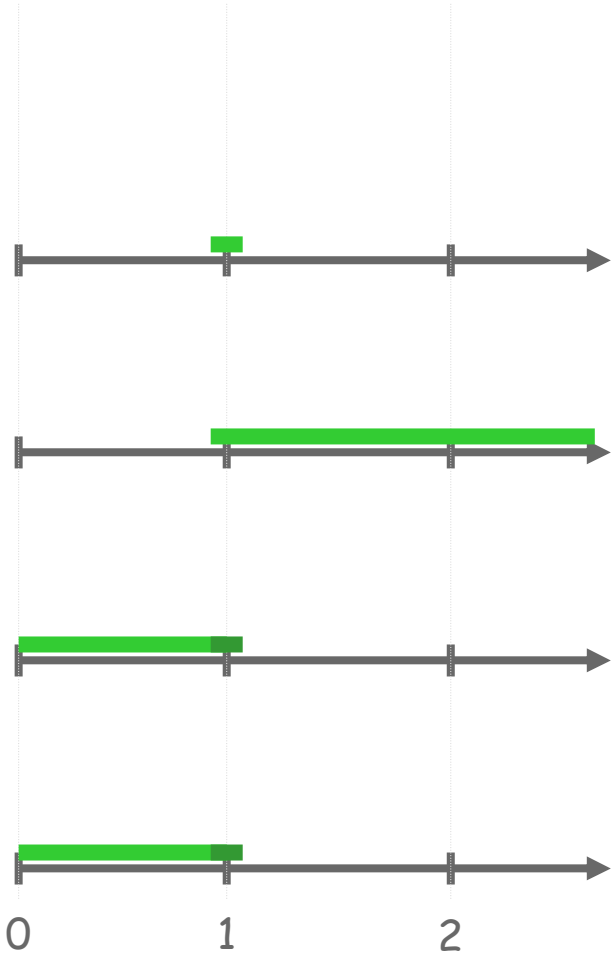


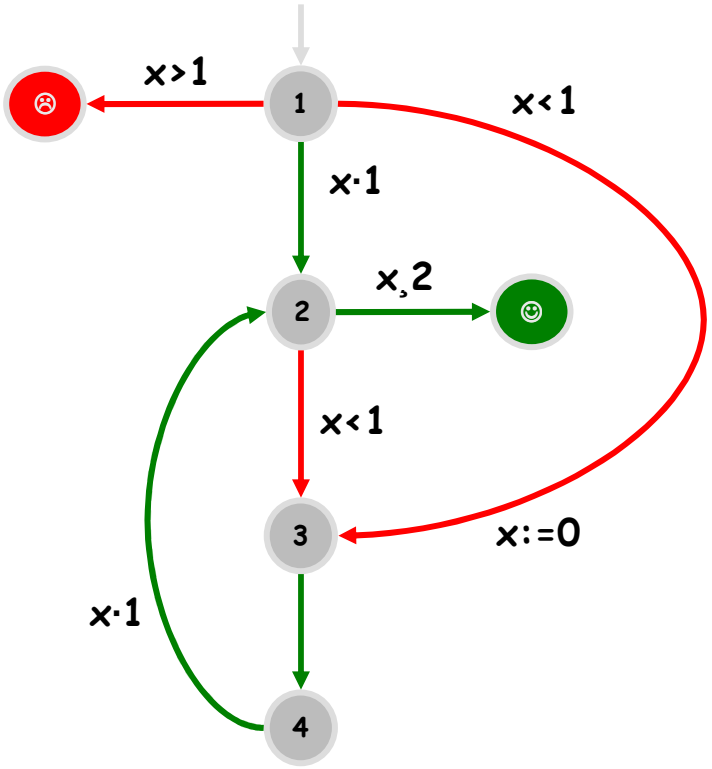


UCb

Timed Games – State-of-the-Art

Backwards Fixed-Point Computation

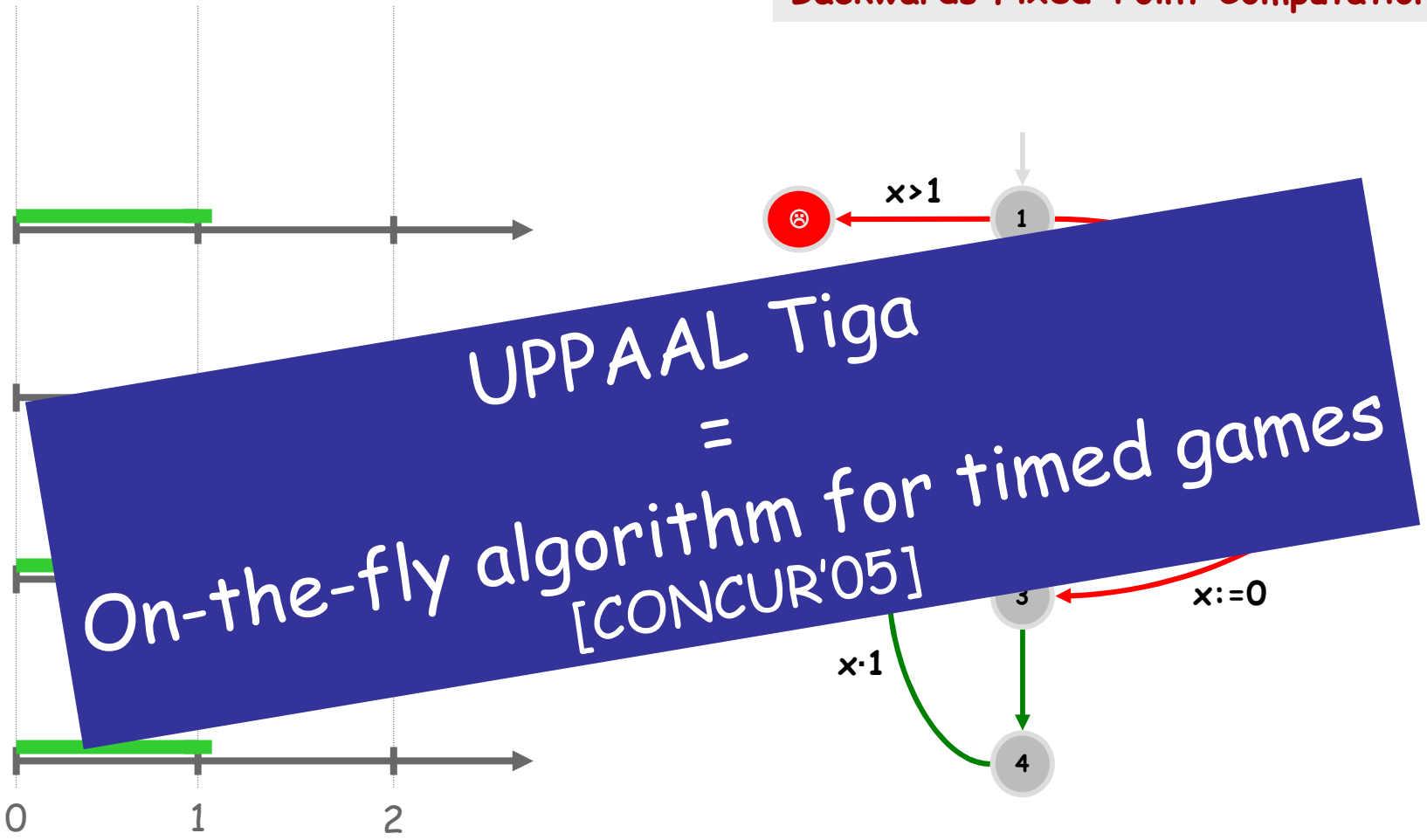




UCb

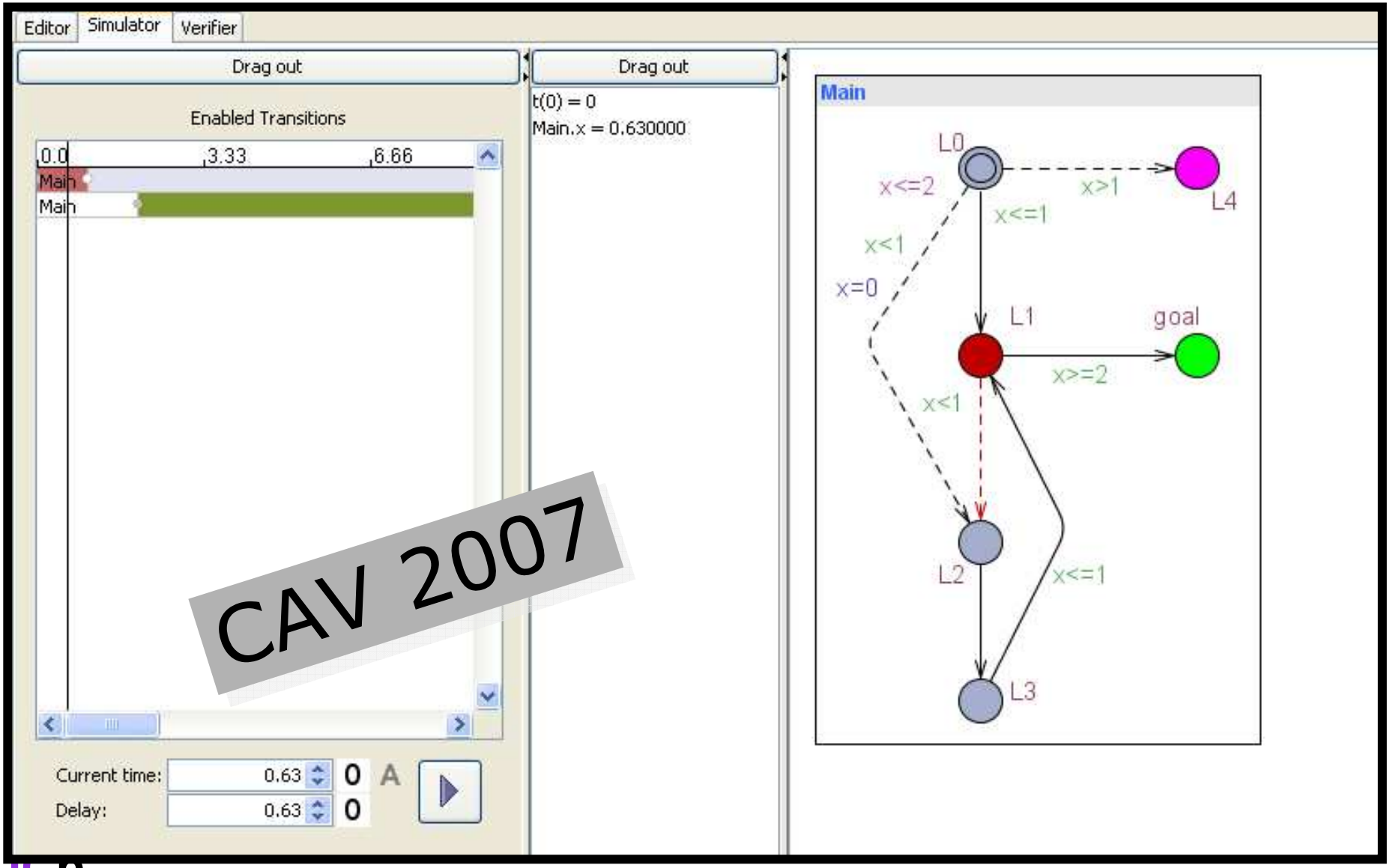
Timed Games – State-of-the-Art

Backwards Fixed-Point Computation



UPPAAL Tiga
=
On-the-fly algorithm for timed games
[CONCUR'05]

UPPAAL Tiga


Editor
Simulator
Verifier
Drag out
Drag out
Enabled Transitions
0.0
3.33
6.66
Main
Main
t(0) = 0
Main.x = 0.630000
Main
L0
L4
L1
goal
L2
L3
x<=2
x>1
x<=1
x<1
x=0
x>=2
x<1
x<=1
Current time:
0.63
0
A
Delay:
0.63
0
CAV 2007

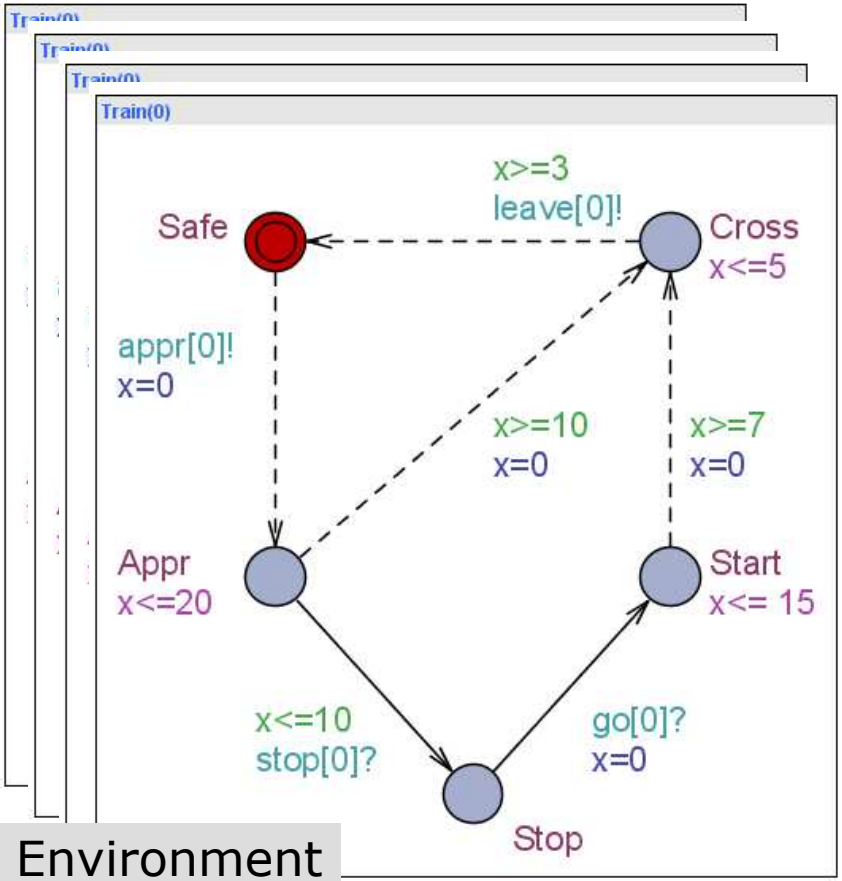
UPPAAL Tiga

CTL Control Objectives

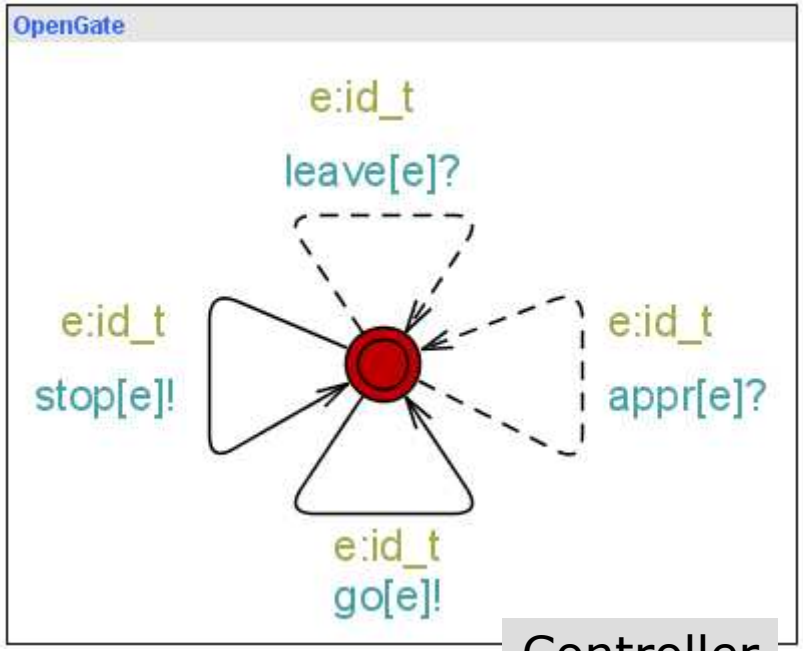
- Reachability properties:
  - control: A[p U q] *until*
  - control: A<> q ⇔ control: A[true U q]
- Safety properties:
  - control: A[p W q] *weak until*
  - control: A[] p ⇔ control: A[p W false]
- Time-optimality :
  - control_t*(u,g): A[p U q]
    - u is an upper-bound to prune the search
    - g is the time to the goal from the current state

UCb

Train Crossing



Environment

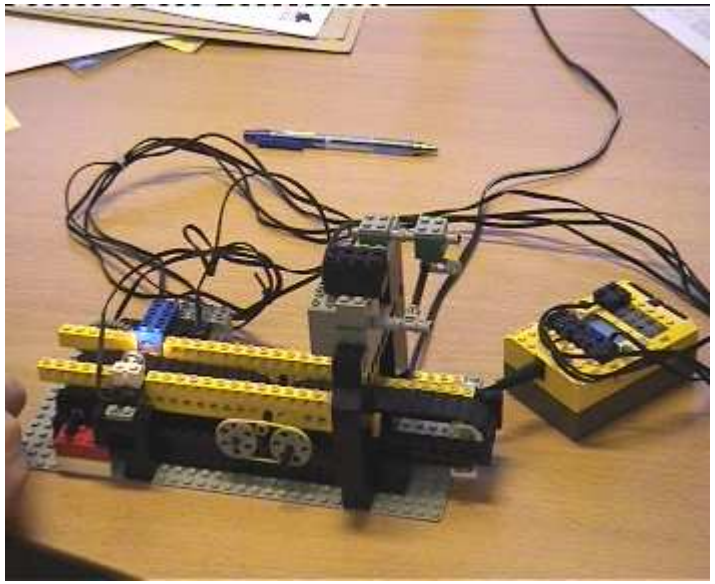


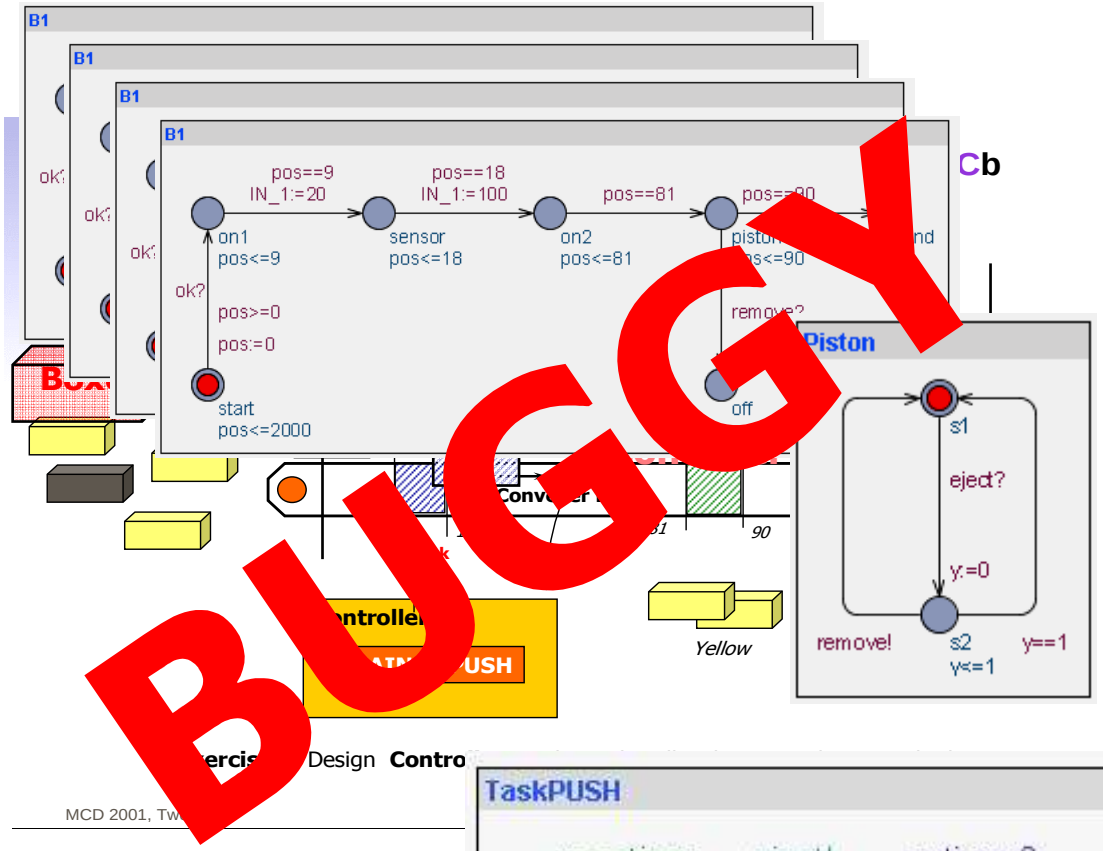
Controller

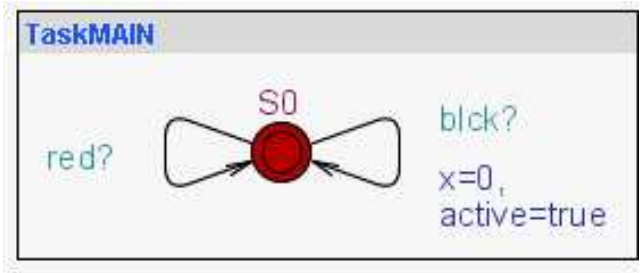
ϕ: Never two trains at the crossing at the same time

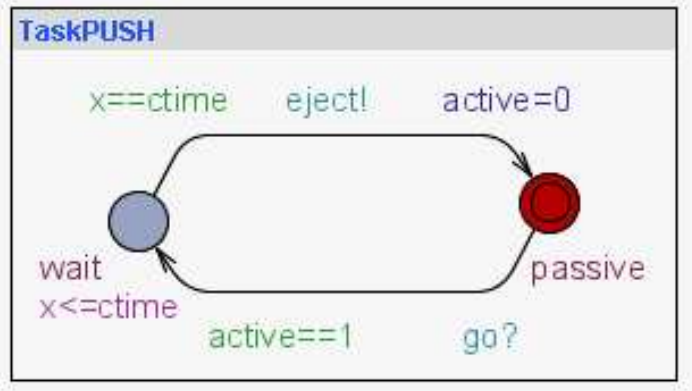
UCb

A Buggy Brick Sorting Program

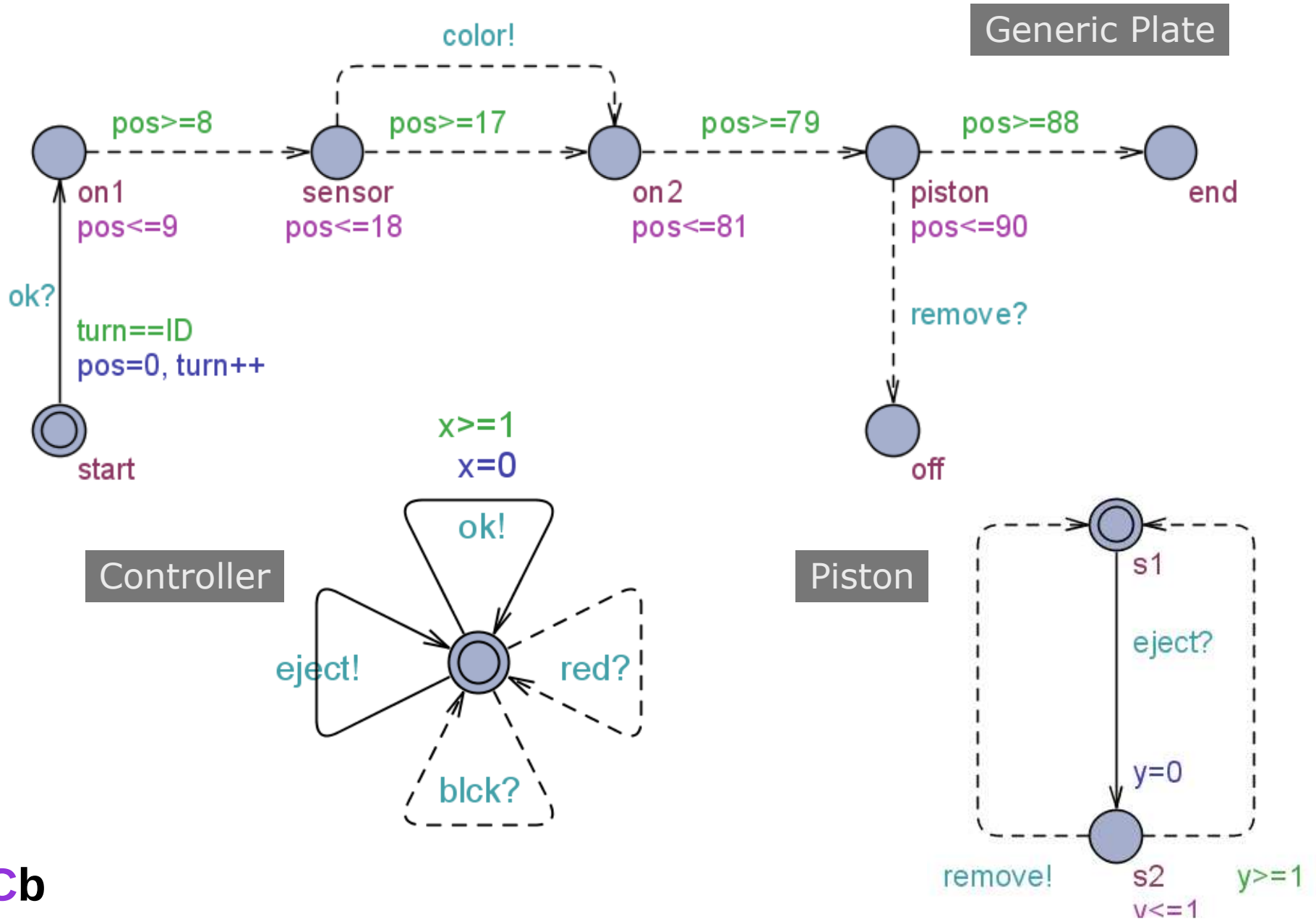






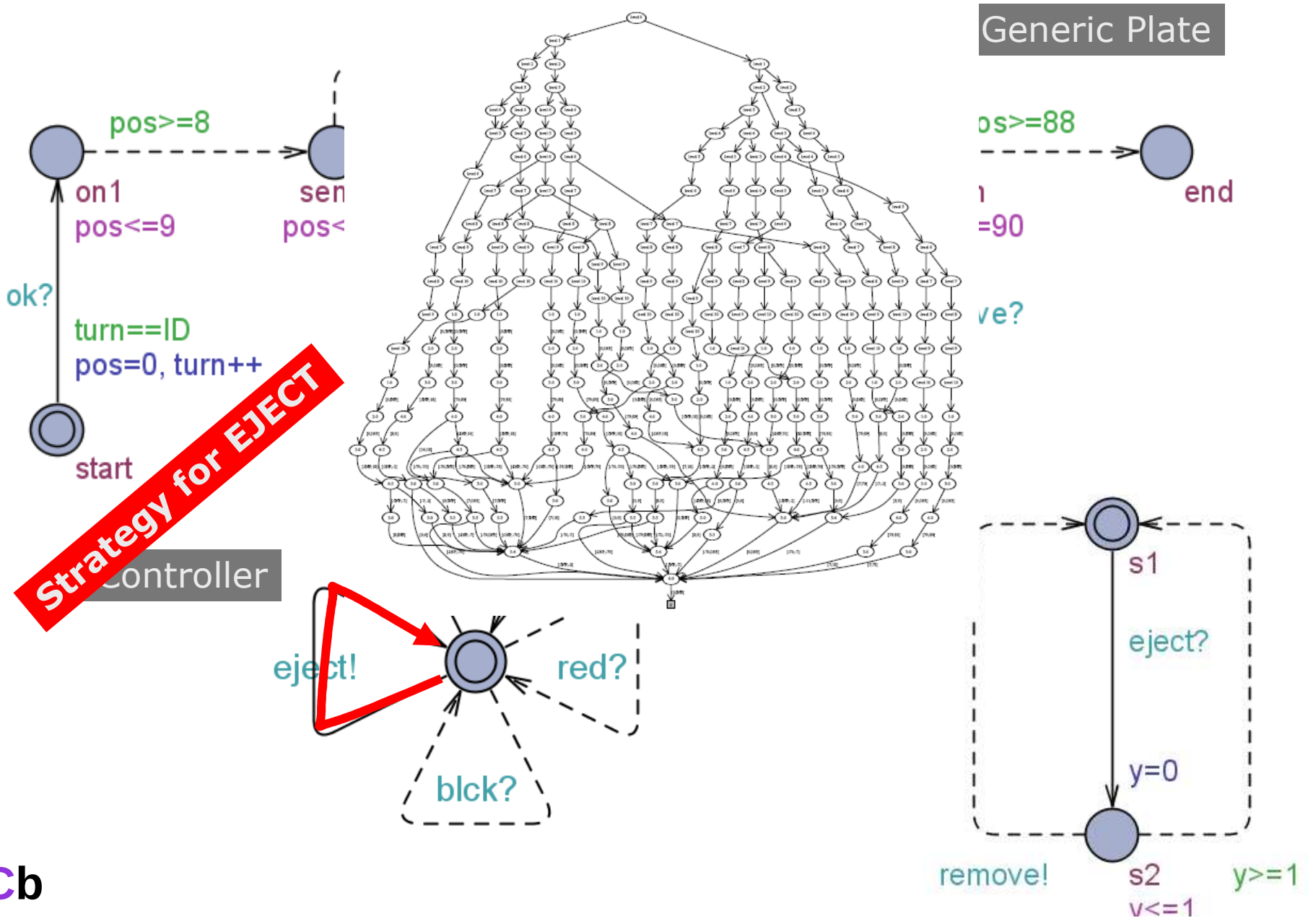


Brick Sorting



UCb

Brick Sorting



UCb

Production Cell

Plates		Basic		Basic +inc		Basic +inc +pruning		Basic+lose +inc +pruning		Basic+lose +inc +topt	
		time	mem	time	mem	time	mem	time	mem	time	mem
2	win	0.0s	1M	0.0s	1M	0.0s	1M	0.0s	1M	0.04s	1M
	lose	0.0s	1M	0.0s	1M	0.0s	1M	0.0s	1M	n/a	n/a
3	win	0.5s	19M	0.0s	1M	0.0s	1M	0.1s	1M	0.27s	4M
	lose	1.1s	45M	0.1s	1M	0.0s	1M	0.2s	3M	n/a	n/a
4	win	33.9s	1395M	0.2s	8M	0.1s	6M	0.4s	5M	1.88s	13M
	lose	-	-	0.5s	11M	0.4s	10M	0.9s	9M	n/a	n/a
5	win	-	-	3.0s	31M	1.5s	22M	2.0s	16M	13.35s	59M
	lose	-	-	11.1s	61M	5.9s	46M	7.0s	41M	n/a	n/a
6	win	-	-	89.1s	179M	38.9s	121M	12.0s	63M	220.3s	369M
	lose	-	-	699s	480M	317s	346M	135.1s	273M	n/a	n/a
7	win	-	-	3256s	1183M	1181s	786M	124s	319M	6188s	2457M
	lose	-	-	-	-	16791s	2981M	4075s	2090M	n/a	n/a

Model		3		6		12		50		100	
Old	c	0.1s	1M	12s	63M	-	-	-	-	-	-
	u	0.2s	3M	235s	273M	-	-	-	-	-	-
New	c	0.05s	3.5M	0.05s	3.5M	0.14s	55M	2.79s	104M	18.5s	426M
	u	0.02s	3.5M	0.04s	3.5M	0.12s	55M	2.32s	94M	15.6s	340M

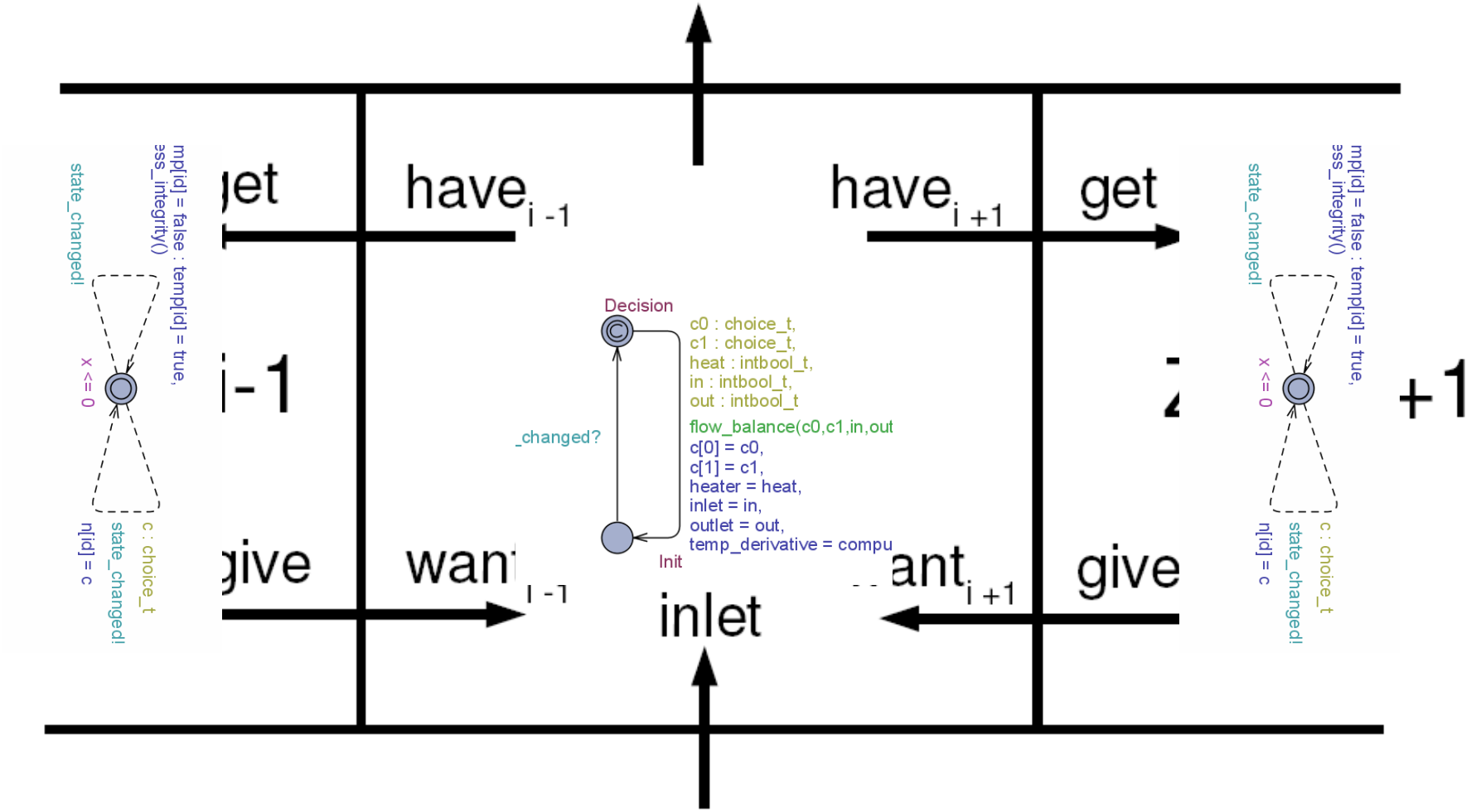
UCb

Climate Control

With Jan J. Jessen
Jacob I. Rasmussen

UC

Climate Control



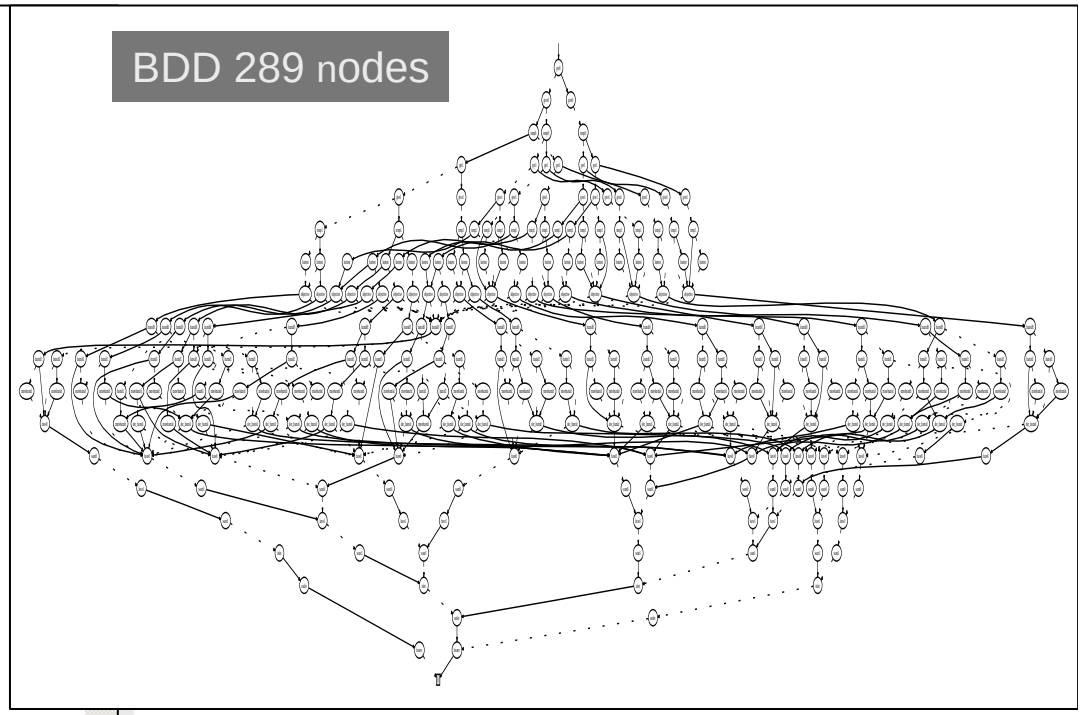
UCb

Obtaining executable code

```
-------------------
Strategy for state:
   Zone i-1: (Temp. lower/equal, wants flow)
   Zone i+1: (Temp. lower/equal, no interaction)
   Hottest neighbor: i-1
   Objective: heat
is:
   Wants flow from i-1
   Wants flow from i+1
   inlet closed
   outlet off
   heater on
-------------------
Strategy for state:
   Zone i-1: (Temp. greater, offers flow)
   Zone i+1: (Temp. greater, offers flow)
   Hottest neighbor: i+1
   Objective: cool
is:
   Has flow for i-1
   Has flow for i+1
   inlet open
   outlet on
   heater off
-------------------
Strategy for state:
   Zone i-1: (Temp. lower/equal, no interaction)
   Zone i+1: (Temp. greater, no interaction)
   Hottest neighbor: i+1
   Objective: cool
```

Stragegy

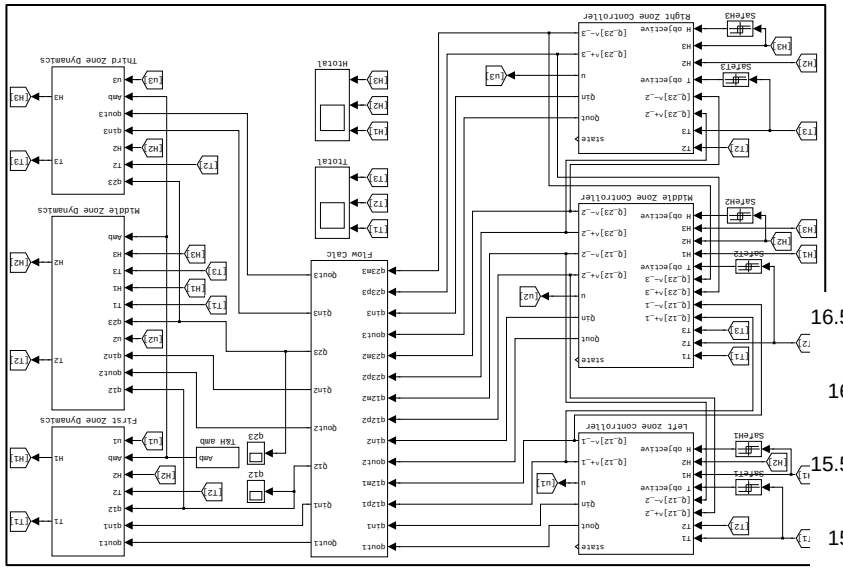
1296 cases

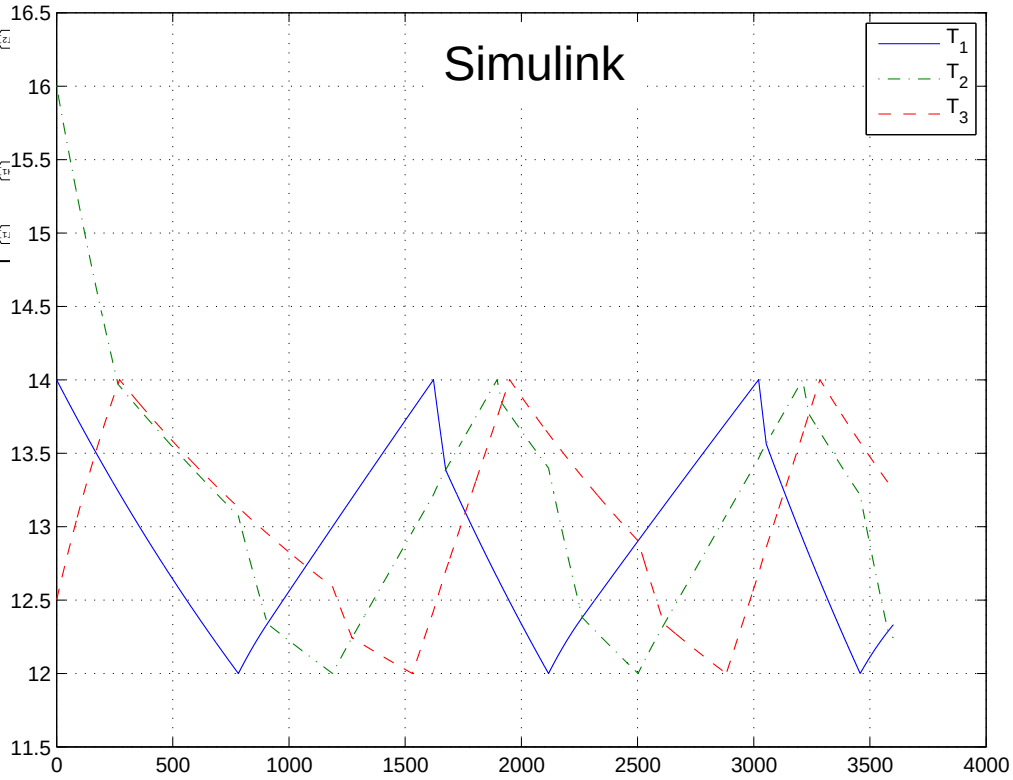


control : A[]
((ZC.Init && objective) imply temp_derivative > 0) &&
((ZC.Init && !objective) imply temp_derivative < 0)

UCb

Obtaining executable code





Open Problems

Decidability

- Priced Timed Games
  - Reachability & Model Checking
  - Safety
  - Negative cost rates
- Probabilistic Pric Timed Automata
- Timed Automata
  - Fully Symbolic
  - Static Analysis & Slicing of C-code
- Timed Games
  - Alternating Logics
  - Partial Observability
  - CEGAR

Efficiency

- Priced Timed Games

Thanks for your attention!

kgl@cs.aau.dk

www.cs.aau.dk

- APPLICATIONS
- Live Sequence Charts
- Gantt Chart
- Code Generation
  - From TA models of controllers
  - From stategies

Usability

UCb